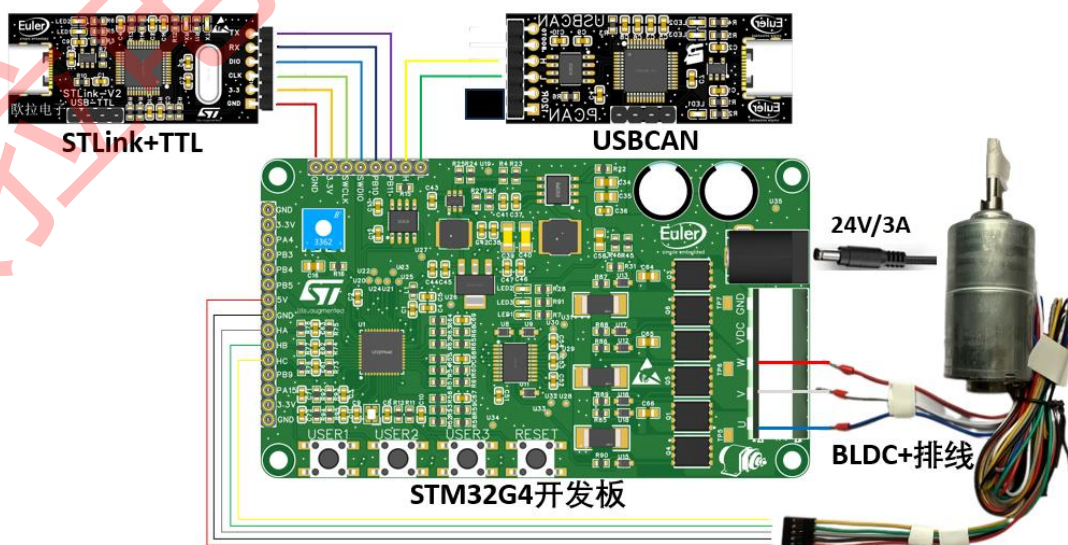


# STM32G4 Simulink FOC 开发 套件用户手册 V12



B 站 UP 主：欧拉电子 编著

STM32G4 底层外设初始化  
Simulink 建模及代码生成



## 致读者的一封信

尊敬的读者：

欢迎您阅读本技术文档，我们深感荣幸能与您分享我们的知识和成果。本文档是我们团队经过长时间研究和实践的智慧结晶，旨在提供最准确、最前沿的电机控制技术和解决方案。

在您开始阅读之前，我们希望强调几点重要的事项。首先，本文档的所有内容，包括但不限于文本、图表、代码、Simulink 模型和设计，均为我们的知识产权，受到国家版权法的严格保护。未经我们的明确书面许可，任何个人或组织不得以任何形式复制、传播、修改或出售本文档的全部或部分内容。

**作品著作权登记信息如下：**

**登记号：国作登字-2024-J-\*1042693**

**作品名称：STM32G4 Simulink FOC 开发套件用户手册**

我们特别提醒，二次售卖或传播本文档的行为属于侵犯版权的违法行为。我们对此持零容忍态度，并保留追究一切法律责任的权利。我们一定通过法律手段维护我们的合法权益，同时也保护消费者免受盗版带来的潜在风险。

## 关于资料侵权事情声明

尊敬的各位客户：

我代表欧拉电子工作室，针对 2024 年 3 月发现的 STM32G4 Simulink FOC 开发套件资料侵权事件，特此发表正式声明。

### 一、事件概述

2024 年 3 月，我方注意到李某在某鱼平台未经授权售卖我方拥有著作权的 STM32G4 Simulink FOC 开发套件资料，该行为侵犯了我方的合法权益。

### 二、已采取措施

我方在第一时间向李某发出警告，并劝导其立即停止侵权行为并下架相关侵权资料，但未得到任何回应。

我方随后委托上海兰迪律师事务所向某鱼平台发送律师函，要求提供李某个人信息，并采取措施制止侵权行为。

在获取李某个人信息后，我方通过上海兰迪律师事务所向其发送律师函，正式要求其承担侵权责任。

### 三、法律责任

李某的侵权行为违反了《中华人民共和国著作权法》及相关法律法规，我方已向法院提起诉讼，要求李某赔偿因侵权行为给我方造成的损失。

经过法庭调解，双方于 2024 年 8 月达成协议，李某同意支付赔偿金，并承担全部诉讼费用。

### 四、学校处分

李某所在的大学对此事高度重视，认为李某的行为严重违反了学校的学术诚信规范。

经过学校的调查和审议，李某受到了严厉的处分，包括学术警告和一段时间内禁止发表学术论文。

### 五、法律责任和处罚说明

我方希望通过此事件警示公众，侵犯知识产权的行为将受到法律的严厉制裁，不仅会带来经济上的损失，还会对个人的声誉和未来发展造成严重影响。

## 六、郑重声明

我方对李某的侵权行为表示强烈谴责，并将继续通过法律途径维护自身合法权益。我方呼吁社会各界尊重知识产权，共同维护一个公平、有序的学术和市场环境。

## 七、举报有奖政策

为进一步打击侵权行为，保护知识产权，我方决定实施举报有奖政策。任何个人或机构若发现侵犯我方著作权的行为，均可向我方或相关执法机关进行举报。一经查实，我方将根据举报信息的重要性和准确性，给予举报人一定数额的奖励。我方承诺对举报人的个人信息严格保密，并保护举报人的合法权益。

## 八、联系方式

举报联系方式：欧拉电子淘宝店铺客服，Bilibili 私信 UP，QQ 技术支持。

此致

敬礼收起



## 版本更新说明

| 版本号 | 更新说明                                                                                                                                                       |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| V11 | <ol style="list-style-type: none"><li>1. 增加 SVPWM 模型说明；</li><li>2. ADC 采样电位器的引脚更正；</li><li>3. 非线性磁链观测器中的 PLL 计算公式更正。</li><li>4. 增加电机参数修改说明（第十二章）</li></ol> |
| V12 | <ol style="list-style-type: none"><li>1. 增加第二十四章：HALL 传感器 0 速闭环正反转</li></ol>                                                                               |

## 目录

|                                        |    |
|----------------------------------------|----|
| STM32G4 Simulink FOC 开发套件用户手册 V12..... | 1  |
| 致读者的一封信.....                           | 2  |
| 关于资料侵权事情声明.....                        | 3  |
| 基础入门篇.....                             | 10 |
| 第一章：STM32G4 FOC 开发套件简介.....            | 10 |
| 1. STM32G4 电机开发板简介.....                | 10 |
| 1.1 开发板输入电压电流范围.....                   | 10 |
| 1.2 板载接口介绍：.....                       | 10 |
| 1.3 测试点介绍：.....                        | 11 |
| 1.4 STM32G431CBU6 简介.....              | 12 |
| 2. 配套 BLDC 电机简介.....                   | 13 |
| 3. USBCAN 模块.....                      | 13 |
| 4. STlinkV2+TTL 模块.....                | 14 |
| 第二章：开发板上电测试.....                       | 14 |
| 1. STM32G4 开发套件测试.....                 | 14 |
| 第三章：软件安装.....                          | 16 |
| 1. STM32CUBEMX 6.3.0 安装.....           | 16 |
| 2. Keil 安装.....                        | 17 |
| 3. ST MCSDK_v5.Y.4 安装.....             | 17 |
| 4. MATLAB 2021b 安装.....                | 17 |
| 第四章：ST MCSDK 快速启动电机.....               | 18 |
| 1. BLDC 霍尔 MCSDK 配置及代码生成.....          | 18 |
| 1.1 ST MCSDK 配置.....                   | 18 |
| 1.2 实验现象.....                          | 26 |
| 2. BLDC 无感 MCSDK 配置及代码生成.....          | 29 |
| 3. 使用电位器调节 BLDC 转速.....                | 31 |
| 3.2 实验现象.....                          | 31 |
| 4. 使用 CAN 上位机调节 BLDC 转速.....           | 33 |
| 4.2 实验现象.....                          | 37 |

|                                 |    |
|---------------------------------|----|
| STM32G4 电机外设篇.....              | 40 |
| 第五章：MCU 基础配置及 GPIO、EXTI 外设..... | 40 |
| 1. STM32CUBEMX 配置及 Keil 代码..... | 40 |
| 2. 实验现象 .....                   | 46 |
| 第六章：UART 外设及 VOFA+ 上位机通讯.....   | 47 |
| 1. STM32CUBEMX 配置及 Keil 代码..... | 47 |
| 2. 实验现象 .....                   | 49 |
| 2.1 UART 重定向 printf 打印数据.....   | 49 |
| 2.2 UART DMA 上传数据.....          | 51 |
| 2.3 UART 中断接收数据.....            | 53 |
| 3. VOFA 上位机显示波形 .....           | 55 |
| 第七章：ADC 采样电位器电压 .....           | 57 |
| 1. STM32CUBEMX 配置及 Keil 代码..... | 57 |
| 2. 实验现象 .....                   | 59 |
| 第八章：MCU 内部 OPAMP-电流采样.....      | 60 |
| 1. STM32CUBEMX 配置及 Keil 代码..... | 62 |
| 2. 实验现象 .....                   | 65 |
| 第九章：TIM1 生成互补 PWM .....         | 66 |
| 1. STM32CUBEMX 配置及 Keil 代码..... | 67 |
| 2. 实验现象 .....                   | 71 |
| 第十章：TIM1 ADC COMP DAC 级联使用..... | 73 |
| 1. STM32CUBEMX 配置及 Keil 代码..... | 75 |
| 2. 实验现象 .....                   | 81 |
| 第十一章：DAC 输出电流波形 .....           | 82 |
| 1. STM32CUBEMX 配置及 Keil 代码..... | 83 |
| 2. 实验现象 .....                   | 84 |
| 第十二章：CAN/CANFD 通讯.....          | 84 |
| 1. STM32CUBEMX 配置及 Keil 代码..... | 85 |
| 2. 实验现象 .....                   | 88 |
| MATLAB 控制算法篇.....               | 90 |

|                                          |     |
|------------------------------------------|-----|
| 第十三章: Clark 变换及代码生成.....                 | 91  |
| 1. Clark 变换建模.....                       | 91  |
| 2. 实验现象 .....                            | 96  |
| 第十四章: Park 及反 Park 变换 .....              | 110 |
| 1. Park 及反 Park 变换建模 .....               | 110 |
| 2. 实验现象 .....                            | 113 |
| 第十五章: SVPWM,VF 开环强托电机 .....              | 116 |
| 1. SVPWM 建模.....                         | 121 |
| 2. VF 模式建模.....                          | 125 |
| 3. VF 启动电机实验现象 .....                     | 129 |
| 第十六章: 电流环闭环, IF 模式及启动状态机(Stateflow)..... | 136 |
| 1. 电流环闭环建模.....                          | 137 |
| 2. IF 模式建模.....                          | 146 |
| 3. IF 启动电机实验现象 .....                     | 155 |
| 第十七章: 速度环闭环 .....                        | 160 |
| 1. 速度环闭环建模.....                          | 160 |
| 2. HALL 有感运行 .....                       | 169 |
| 第十八章: 龙伯格观测器 .....                       | 173 |
| 1. 龙伯格观测器及 PLL 建模 .....                  | 174 |
| 2. 龙伯格观测器模块内部及数据字典截图 .....               | 185 |
| 3. 无感龙伯格观测器实验现象 .....                    | 188 |
| 第十九章: 滑膜观测器 .....                        | 189 |
| 1. 滑膜观测器及 PLL 建模 .....                   | 189 |
| 2. 滑膜观测器模块内部及数据字典截图 .....                | 201 |
| 3. 无感滑膜观测器实验现象 .....                     | 204 |
| 第二十章: 拓展卡尔曼观测器 .....                     | 205 |
| 1. 卡尔曼观测器理论引入 .....                      | 205 |
| 2. 拓展卡尔曼观测器模块内部及数据字典截图 .....             | 222 |
| 3. 无感拓展卡尔曼实验现象 .....                     | 236 |
| 第二十一章: Simulink 上位机 .....                | 237 |

|                               |     |
|-------------------------------|-----|
| 第二十二章：电机参数转换说明.....           | 241 |
| 第二十三章：非线性磁链观测器.....           | 242 |
| 1. 非线性磁链观测器理论引入.....          | 242 |
| 2. 非线性磁链观测器模块内部及数据字典截图.....   | 248 |
| 第二十四章：HALL 传感器 0 速闭环正反转 ..... | 250 |
| 1. HALL 传感器 0 速角度判断 .....     | 251 |
| 2. HALL 传感器正反转角度计算.....       | 252 |
| 3. HALL 传感器偏置角度测量计算 .....     | 256 |
| 4. 速度环模型修改.....               | 257 |
| 5. HALL 0 速闭环正反转实验现象 .....    | 258 |
| 第二十五章：VOFA+ 控制参数下发 .....      | 260 |
| 1. 建立上位机控件及协议.....            | 260 |
| 2. STM32G4 接收并解析协议 .....      | 264 |
| 3. VOFA+上位机控制电机实验现象.....      | 268 |

## 基础入门篇

## 第一章：STM32G4 FOC 开发套件简介

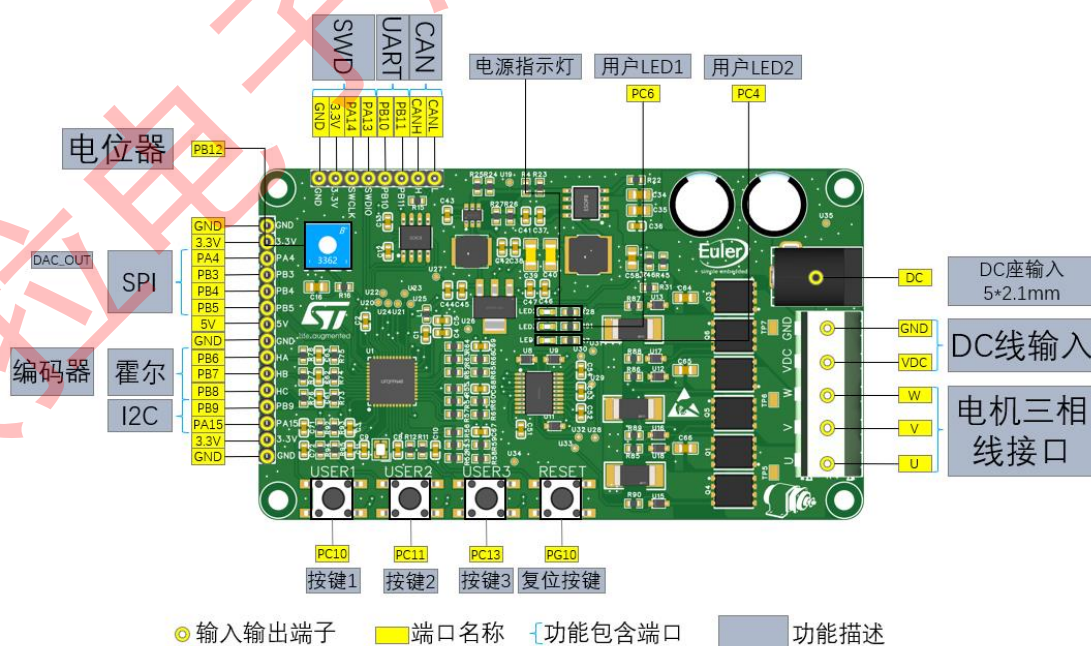
## 1. STM32G4 电机开发板简介

STM32G4 电机控制开发板包含控制系统及驱动系统的开发板。控制系统使用 STM32G431CBU6, 驱动系统使用 FD6288T 为 PMSM 电机提供三相电源。MOSFET 使用英飞凌 BSC0702LS,  $V_{DS}$  60V,  $R_{DS}$  2.7m $\Omega$ ,  $I_D$  100A。

### 1.1 开发板输入电压电流范围

| 参数  | 电压     | 电流    |
|-----|--------|-------|
| VDC | 18-38V | 0-10A |

## 1.2 板载接口介绍:





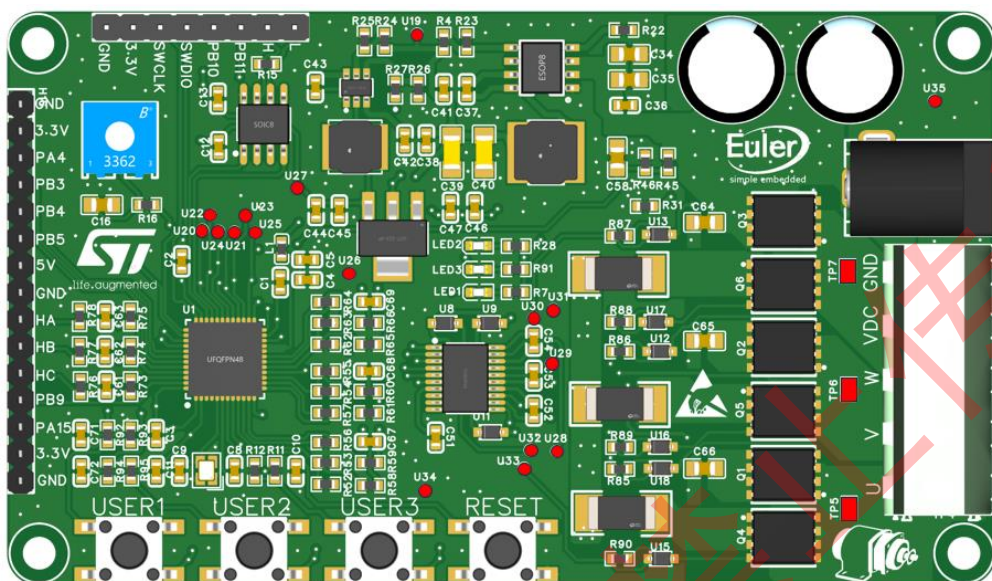
| 板载资源  | 功能           | GPIO                                    |
|-------|--------------|-----------------------------------------|
| 调试接口  | SWD          | PA13(SWDIO), PA14(SWCLK)                |
| 通讯接口  | UART         | PB11(RX), PB10(TX)                      |
|       | CAN/CANFD    | PA12(TX), PA11(RX)                      |
|       | I2C          | PB9(SDA), PA15(SCK)                     |
|       | SPI          | PA4(CS), PB3(SCK), PB4(MISO), PB5(MOSI) |
| DAC接口 | DAC_OUT      | PA4                                     |
| 电位器   | 采样电压并调速      | PB12(ADC_IN11)                          |
| 按键    | USER1        | PC10                                    |
|       | USER2        | PC11                                    |
|       | USER3        | PC13                                    |
|       | RESET        | PG10                                    |
| LED灯  | 电源LED        | 3.3V电源正常，电源LED亮                         |
|       | 用户LED1       | PC6                                     |
|       | 用户LED2       | PC4                                     |
| DC接口  | DC线输入        | <b>务必注意VDC与GND对应的端子，反接会烧芯片</b>          |
|       | DC005座       | 尺寸：5*2.1mm，建议使用该接口给开发板供电                |
| 编码器接口 | 霍尔传感器        | PB6(HA), PB7(HB), PB8(HC)               |
|       | 增量式编码器       | PB6(A), PB7(B), PB8(Z)                  |
|       | 磁编码器         | 根据编码器接口选择对应SPI/I2C通讯                    |
| 电机类型  | PMSM/BLDC/DC | 支持三相永磁同步电机、无刷电机、有刷电机                    |

### 注意事项：

1. UART 接口与 USB 转 UART 模块连接时，开发板 TX 与 USB 转 UART 模块 RX 连接，开发板 RX 与 USB 转 UART 模块 TX 连接；
2. 使用 DC 线输入时，VDC 与 GND 不可接反，若客户接反导致开发板无法使用，由客户承担损坏后果，商家可拒绝售后或者退款请求。

### 1.3 测试点介绍：

| 测试点 | U19     | U20      | U21       | U22      | U23       | U24      | U25       | U26  | U27 | U34  |
|-----|---------|----------|-----------|----------|-----------|----------|-----------|------|-----|------|
| 网络  | 12V     | TIM1_CH3 | TIM1_CH3N | TIM1_CH2 | TIM1_CH2N | TIM1_CH1 | TIM1_CH1N | 3.3V | 5V  | AGND |
| 测试点 | U28     | U26      | U30       | U31      | U32       | U33      | TP4       | TP5  | TP6 | U35  |
| 网络  | Vgs(Q1) | Vgs(Q2)  | Vgs(Q3)   | Vgs(Q6)  | Vgs(Q5)   | Vgs(Q4)  | U         | V    | W   | GND  |



## 1.4 STM32G431CBU6 简介

STM32G431CBU6 基于高性能 Arm®Cortex®-M4 32 位 RISC 内核。工作频率高达 170MHz, 是意法半导体定位于电机控制系列的 MCU, 片内包含 3\*op-amps (PGA), 硬件 CORDIC 及滤波器 FMAC 等外设。

| Connectivity                            |                                                | Timers                                  |
|-----------------------------------------|------------------------------------------------|-----------------------------------------|
| 3x SPI, 3x I <sup>2</sup> C, 4x UxART   | Arm® Cortex®-M4<br>Up to 170 MHz               | 5x 16-bit timers                        |
| 1x USB 2.0 FS,<br>1x USB-C PD3.0 (+PHY) | Floating Point Unit                            | 2x 16-bit basic timer                   |
| 1x CAN-FD                               | Memory Protection Unit                         | 2x 16-bit advanced motor control timers |
| 2x I <sup>2</sup> S half duplex, SAI    | Embedded Trace Macrocell                       | 1x 32-bit timer                         |
| Accelerators                            | 12-channel DMA + MUX                           | 1x 16-bit LP timer                      |
| ART Accelerator™                        | Up to 128-Kbyte Flash memory / ECC Single Bank | Analog                                  |
| 10-Kbyte CCM-SRAM                       | 22-Kbyte SRAM                                  | 2x 12-bit ADC                           |
| Math Accelerators                       |                                                | 4x Comparators                          |
| Cordic (trigo...) Filtering             |                                                | 4x DAC (2x buff + 2x non-buff)          |
|                                         |                                                | 3x op-amps (PGA)                        |
|                                         |                                                | 1x temperature sensor                   |
|                                         |                                                | Internal voltage reference              |

## 2. 配套 BLDC 电机简介

- BLDC 电机

电压: 24 V

额定转速: 1300 (RPM)

Hall 传感器: 120° 安装

极对数: 4 对极

相电阻:  $6.97\ \Omega$  (+-10%)

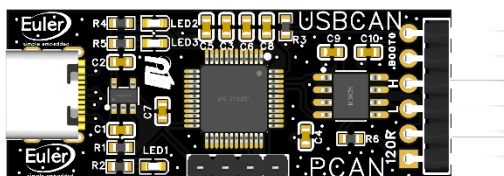
相电感: 5.35mH(+ -20%)

反电动势系数: 12.25 V/krpm



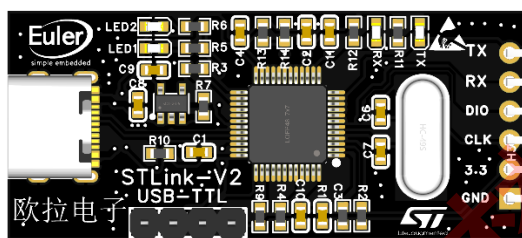
## 3. USBCAN 模块

在实际电机开发项目中由于电磁干扰等因素，通常会使 CAN/CANFD 与其他设备通讯，掌握 CAN/CANFD 通讯非常重要。资料在 q 群中，请按照 USBCAN 用户手册使用模块。



## 4. STlinkV2+TTL 模块

STlinkV2+TTL 可以 Debug 和下载 STM32G4 应用程序，同时 TTL 可以实现 FOC 工程中的数字信号量波形上传至 VOFA 上位机显示。STlinkV2+TTL Pin 脚与 STM32G4 开发板 Pin 脚一一对应，接线非常简单。

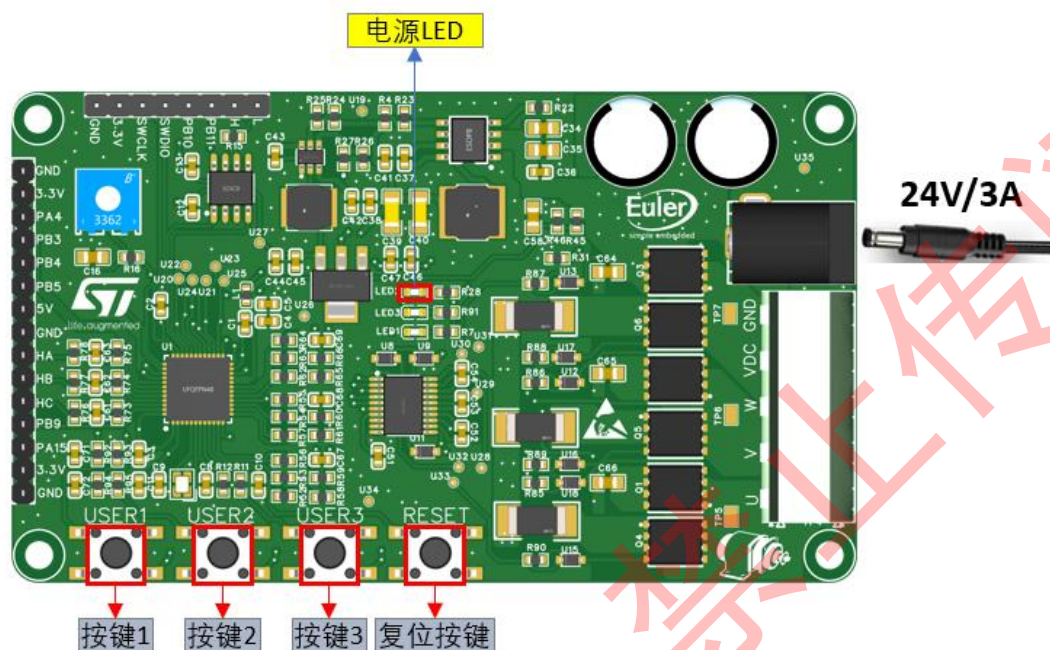


## 第二章：开发板上电测试

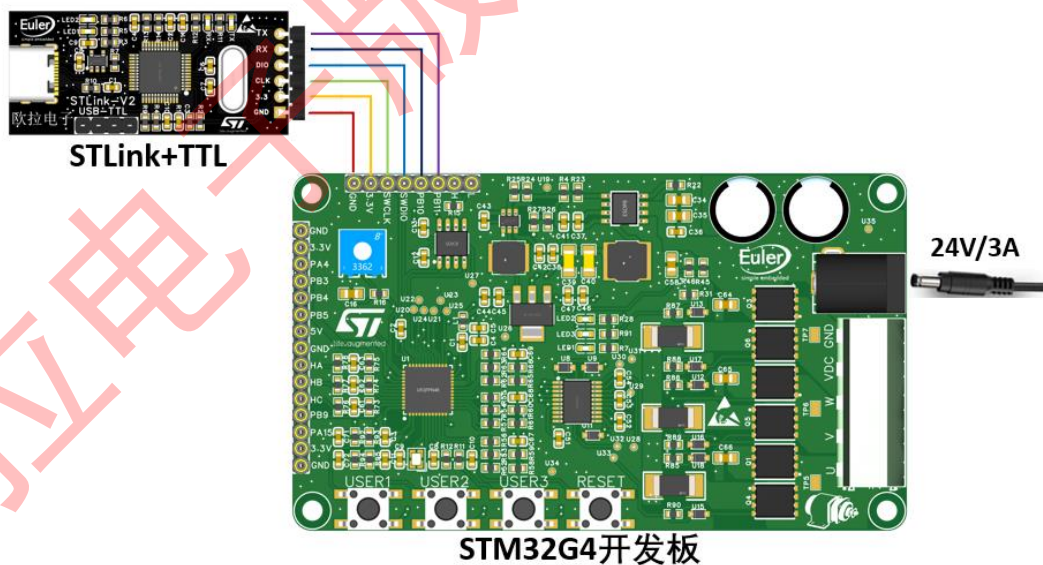
### 1. STM32G4 开发套件测试

- ① 强烈建议使用 24V 电源适配器给开发板供电，电源 LED 正常亮起，代表电源供电正常；若客户使用接线端子端供电，请确保电源的正负极与开发板 VDC 及 GND 对应，接反会导致电源相关芯片烧毁由客户自行承担后果；

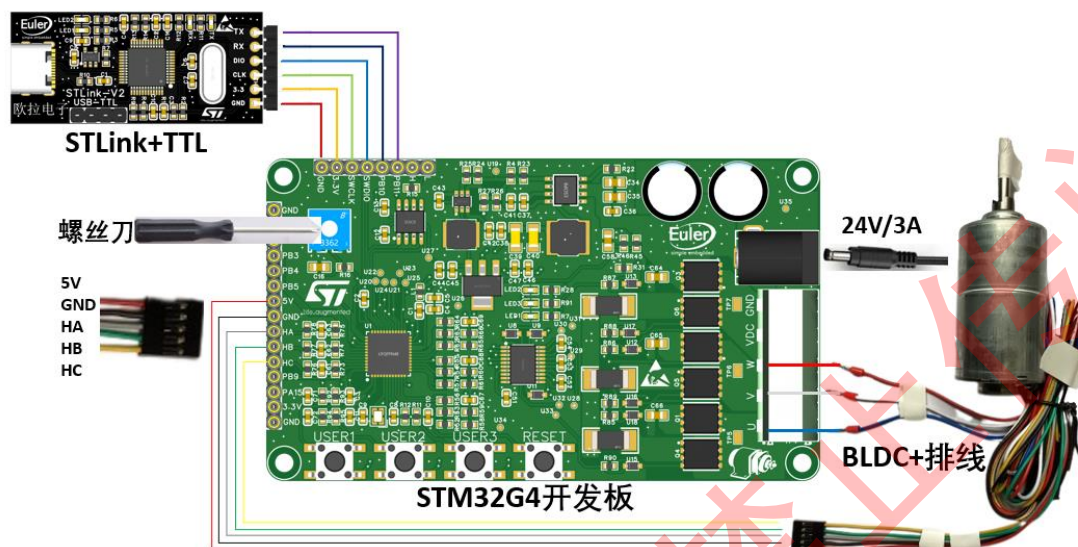




- ② 连接 stlink 工具，可下载电机外设篇的任何一章节的代码，与实验现象进行对比；



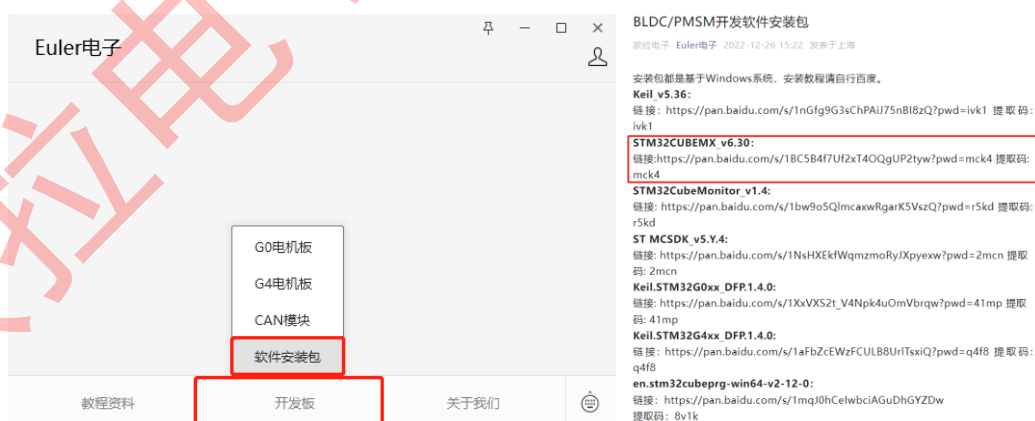
- ③ 对于由购买 BLDC 及 USBCAN 模块的客户，可下载第四章 ST MCSDK 快速启动电机相关程序并查看相关实验现象；



## 第三章：软件安装

### 1. STM32CUBEMX 6.3.0 安装

① 后续教程全部基于 6.3.0 版本，用户需要安装此版本，打开 Euler 电子微信公众号->开发板->软件安装包：



② 从百度网盘链接下载 STM32CUBEMX 6.3.0 安装包，安装流程默认下一步直到安装结束；

## 2. Keil 安装

① Keil 安装版本并没有限制，用户如果电脑上已经安装，可以直接使用；  
若没有安装可以在微信公众号软件安装包里面下载安装包；

### Keil\_v5.36:

链接: <https://pan.baidu.com/s/1nGfg9G3sChPAiJ75nBl8zQ?pwd=ivk1> 提取码: ivk1

② Keil 注册机使用可以参考以下网址:

[Keil uvision5 MDKv518 安装教程 \(qq.com\)](#)

③安装 Keil 成功之后，需要安装 STM32G4 Keil 支持包，在微信公众号软件安装包下载安装，安装流程默认下一步直到安装结束；

### Keil.STM32G4xx\_DFP.1.4.0:

链接: <https://pan.baidu.com/s/1aFbZcEWzFCULB8UrlTsxiQ?pwd=q4f8> 提取码: q4f8

## 3. ST MCSDK\_v5.Y.4 安装

在微信公众号软件安装包下载安装，安装流程默认下一步直到安装结束；

### ST MCSDK\_v5.Y.4:

链接: <https://pan.baidu.com/s/1NsHXEkfWqmzmRyJXpyexw?pwd=2mcn> 提取码: 2mcn

## 4. MATLAB 2021b 安装

安装包下载及安装流程如以下网址:

[MATLAB R2021b 安装教程 \(qq.com\)](#)



## 第四章：ST MCSDK 快速启动电机

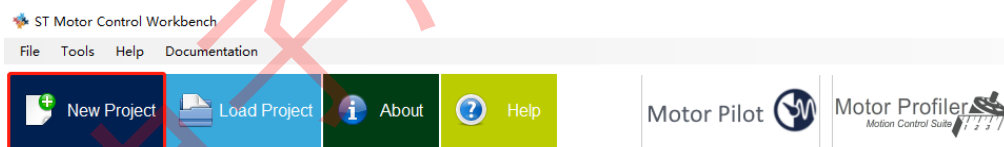
### 1. BLDC 霍尔 MCSDK 配置及代码生成

#### 1.1 ST MCSDK 配置

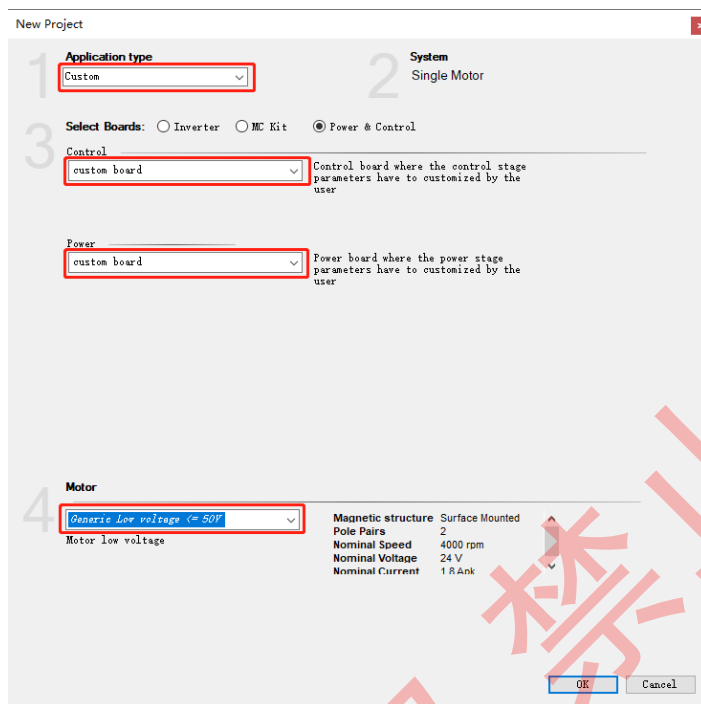
- ① 双击打开 MotorControl Workbench 5.Y.4 ：



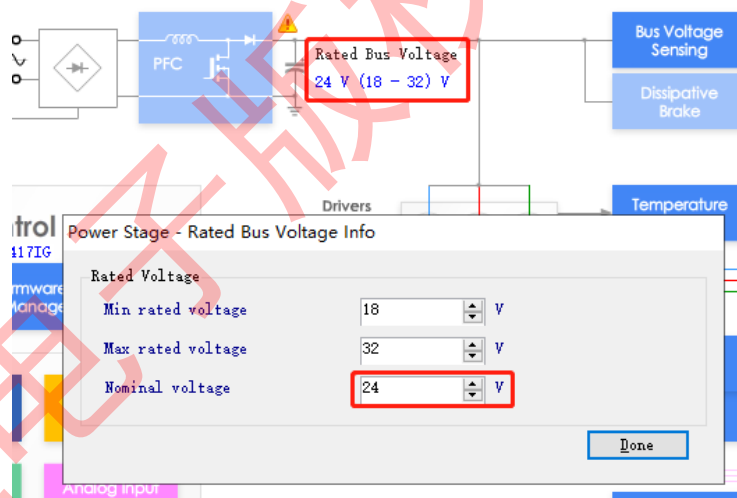
- ② 新建工程：



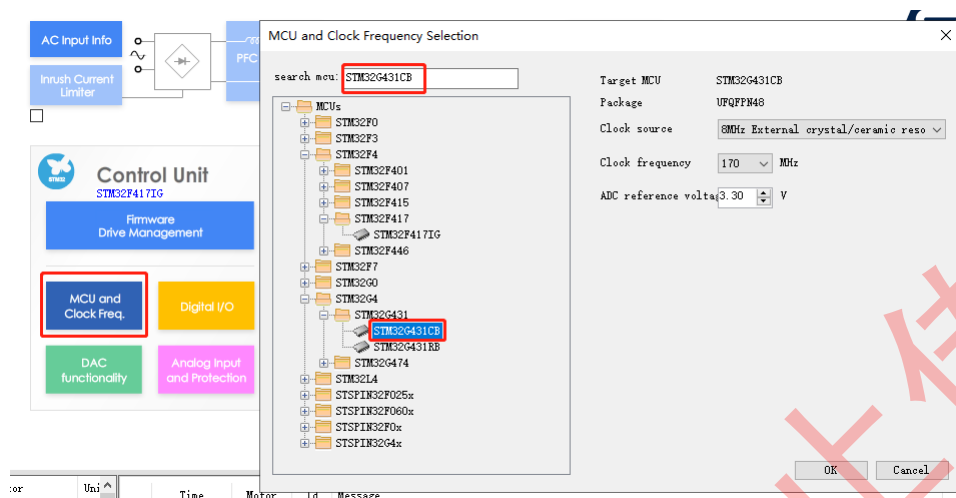
- ③ 选择用户应用，用户板子，电压小于 50V，点击 OK：



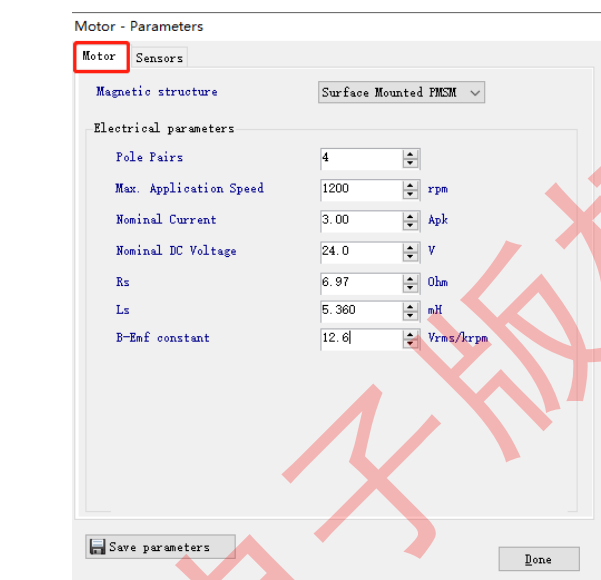
④ 正常工作电压修改为 24V，我们使用 24V 电源适配器供电：



⑤ 选择 MCU 型号及时钟频率，先选择外部 8M 时钟，后续在 CUBEMX 再修改：



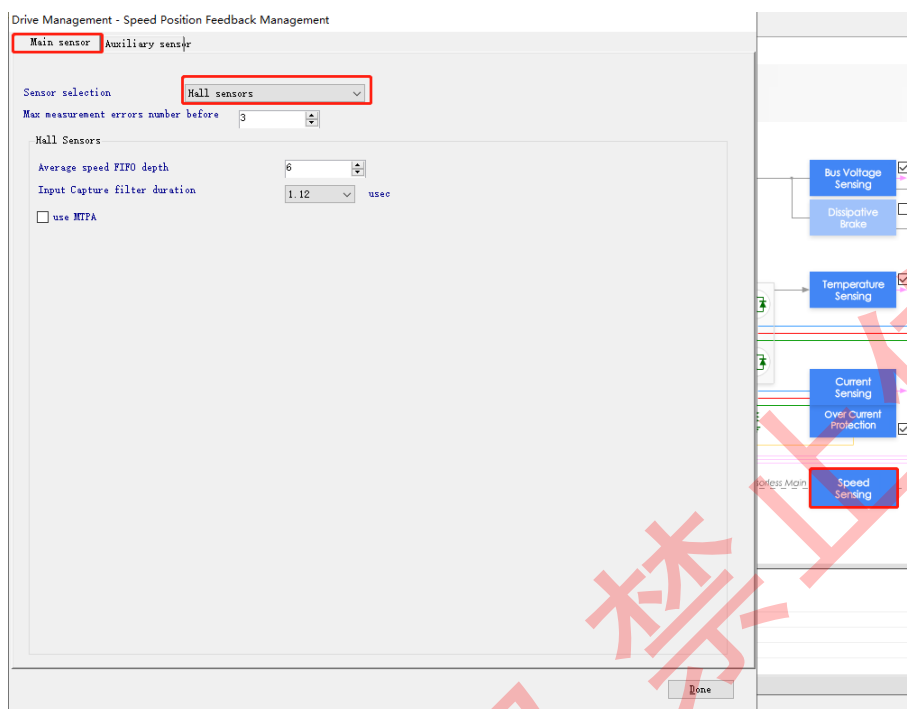
⑥ 配置电机参数及霍尔传感器参数：



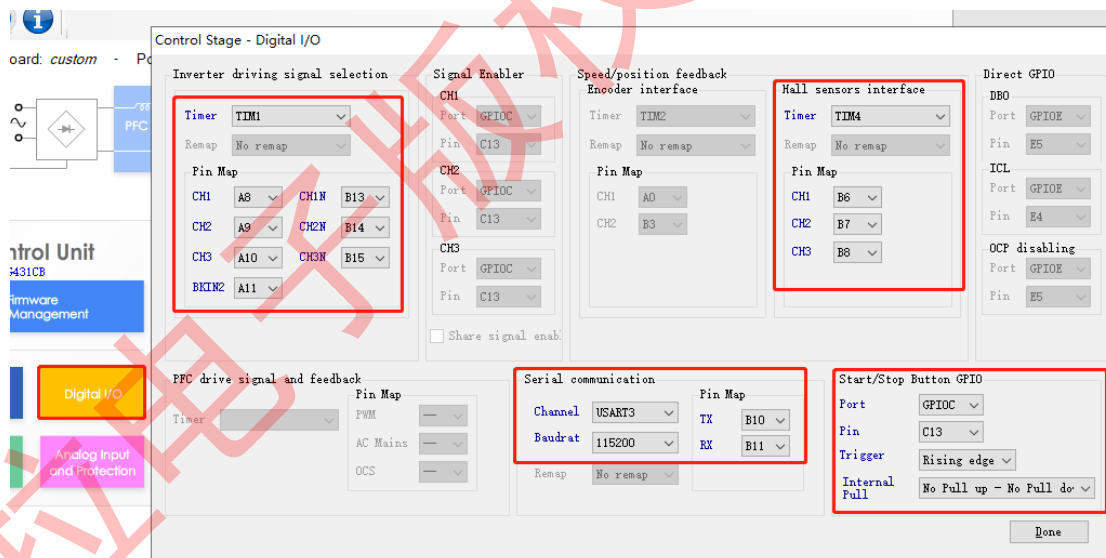
⑦ 速度传感器选择 HALL 传感器：

版本：V13

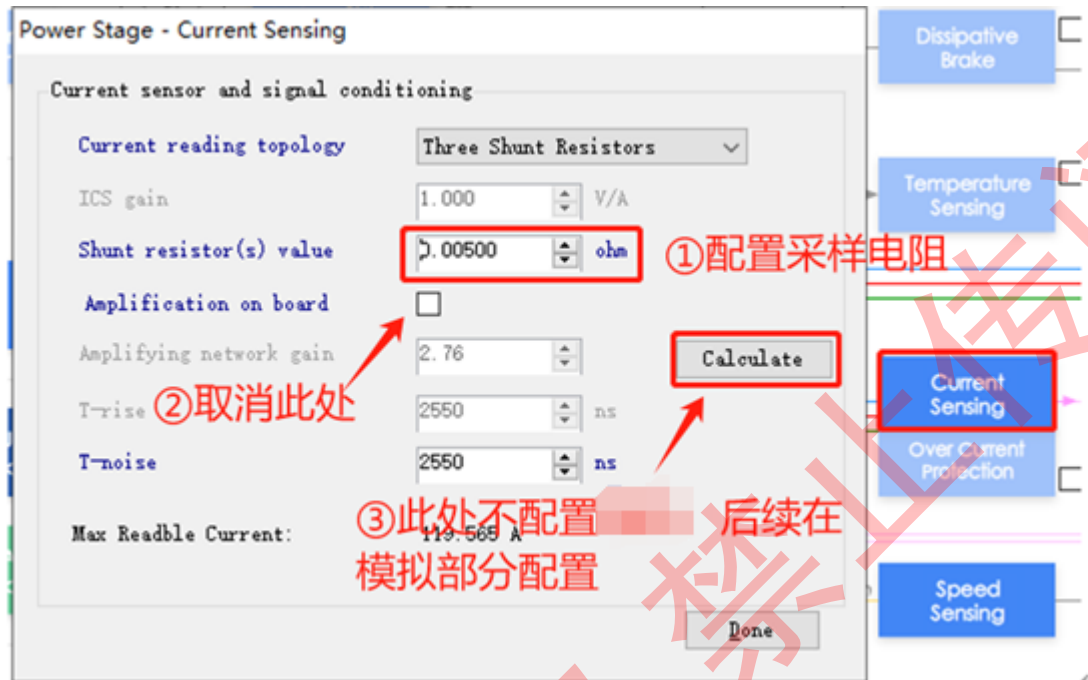
B 站 up 主：欧拉电子



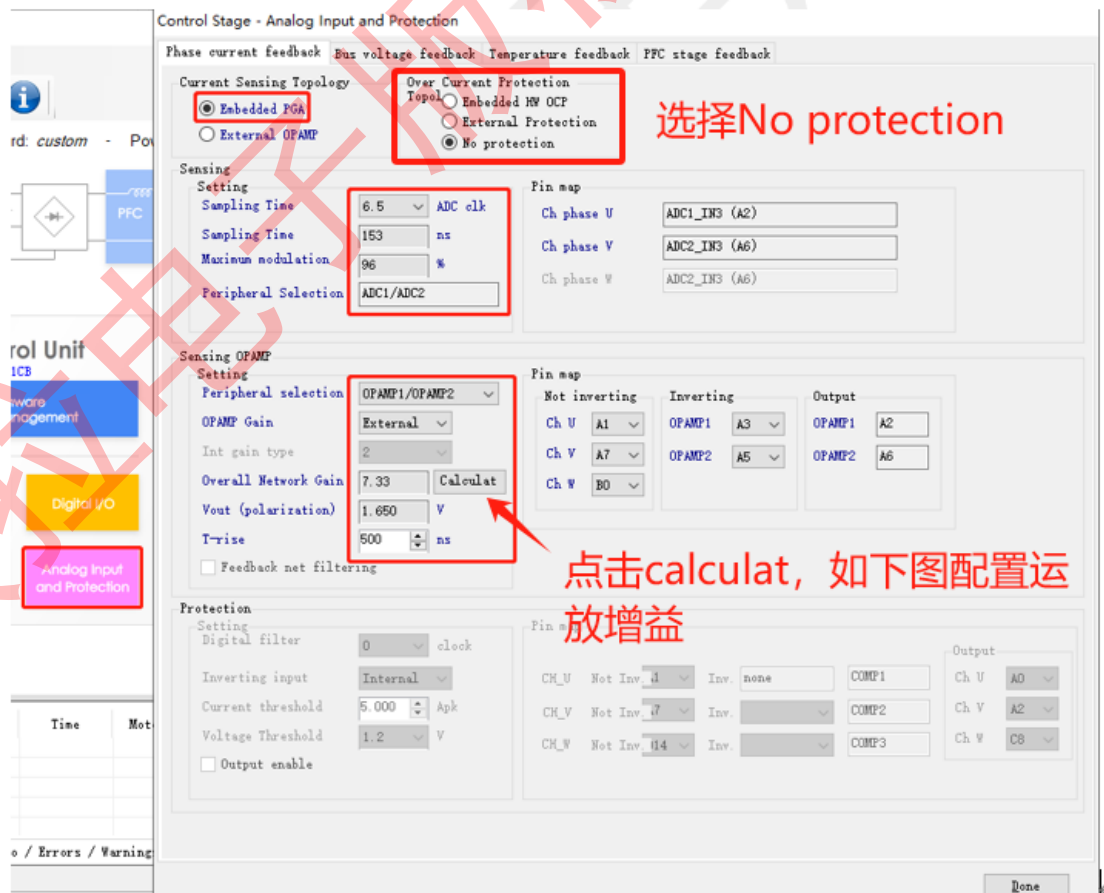
⑧ 根据原理图来配置各个数字 IO:

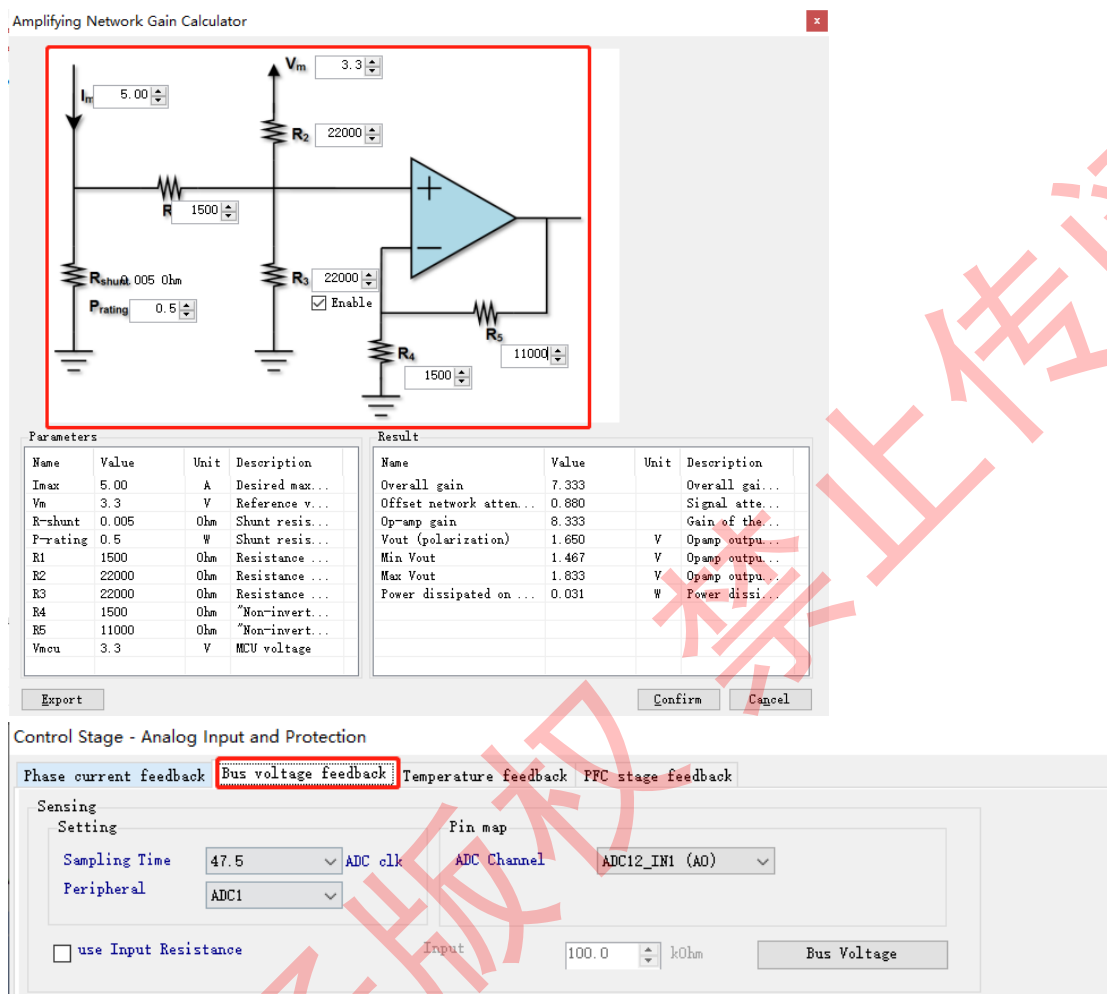


⑨ 配置电流采样部分，三电阻采样，修改采样电阻大小，取消 amp on board，此处不配置运放增益，在后续 analog 部分配置：



10 配置 MCU 模拟部分输入:

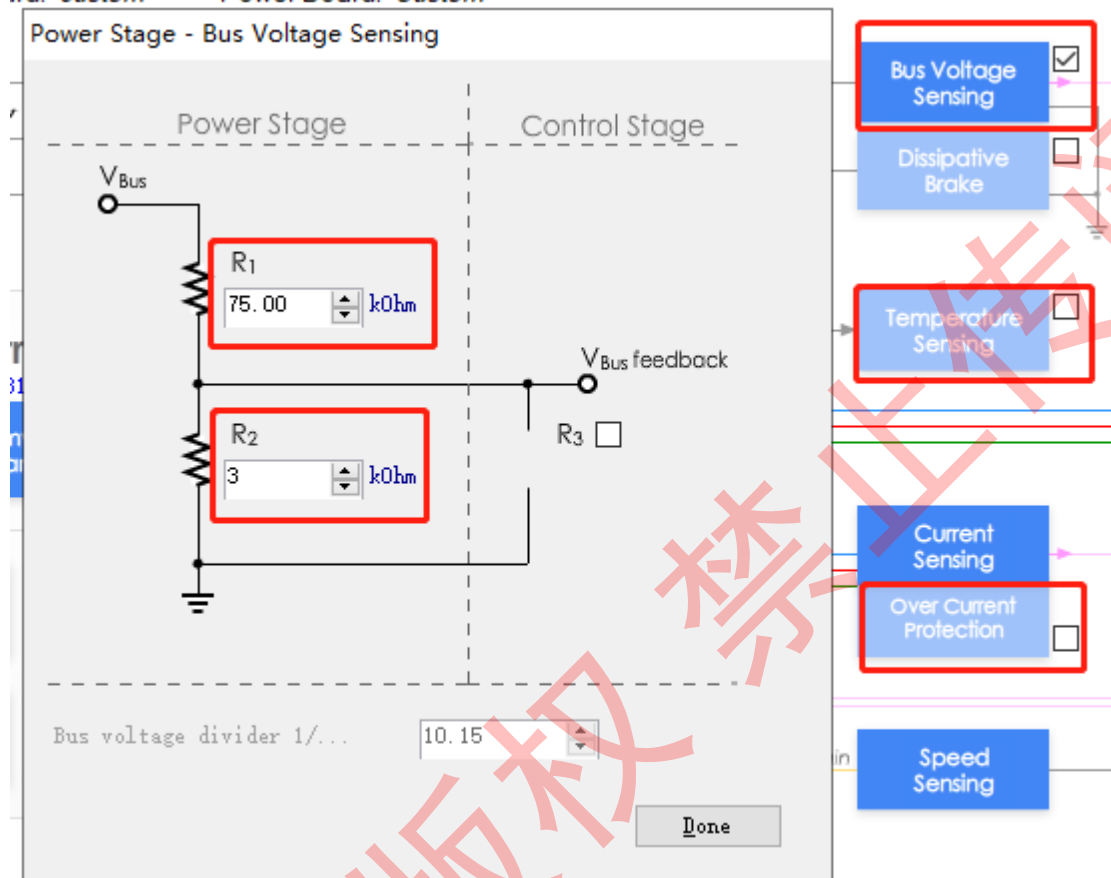




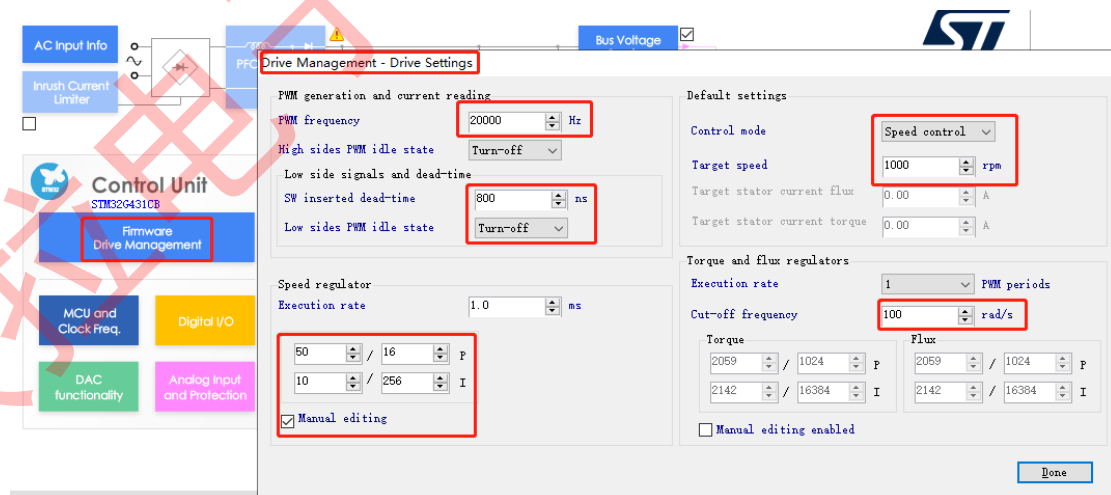
11 基于原理图配置输入电压检测，取消温度传感器和过流保护，过流保护由软件保护：



ard: custom - Power Board: Custom



### ⑬ 配置电流环及速度环 KP/KI 参数:

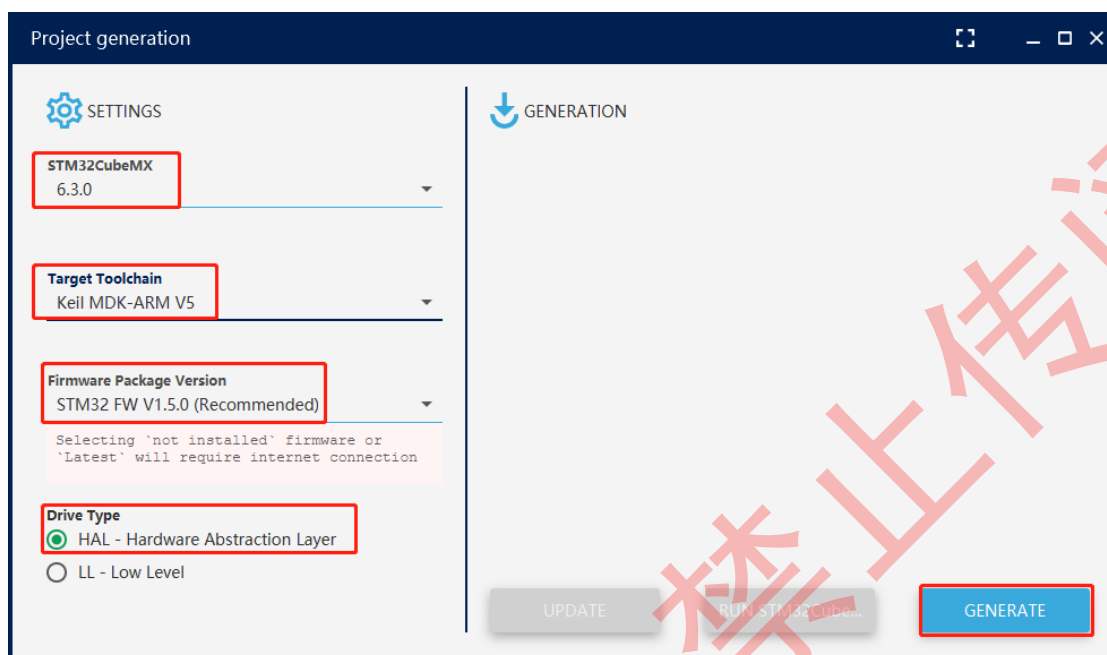


### ⑭ 点击生成代码



, 将工程保存后点击

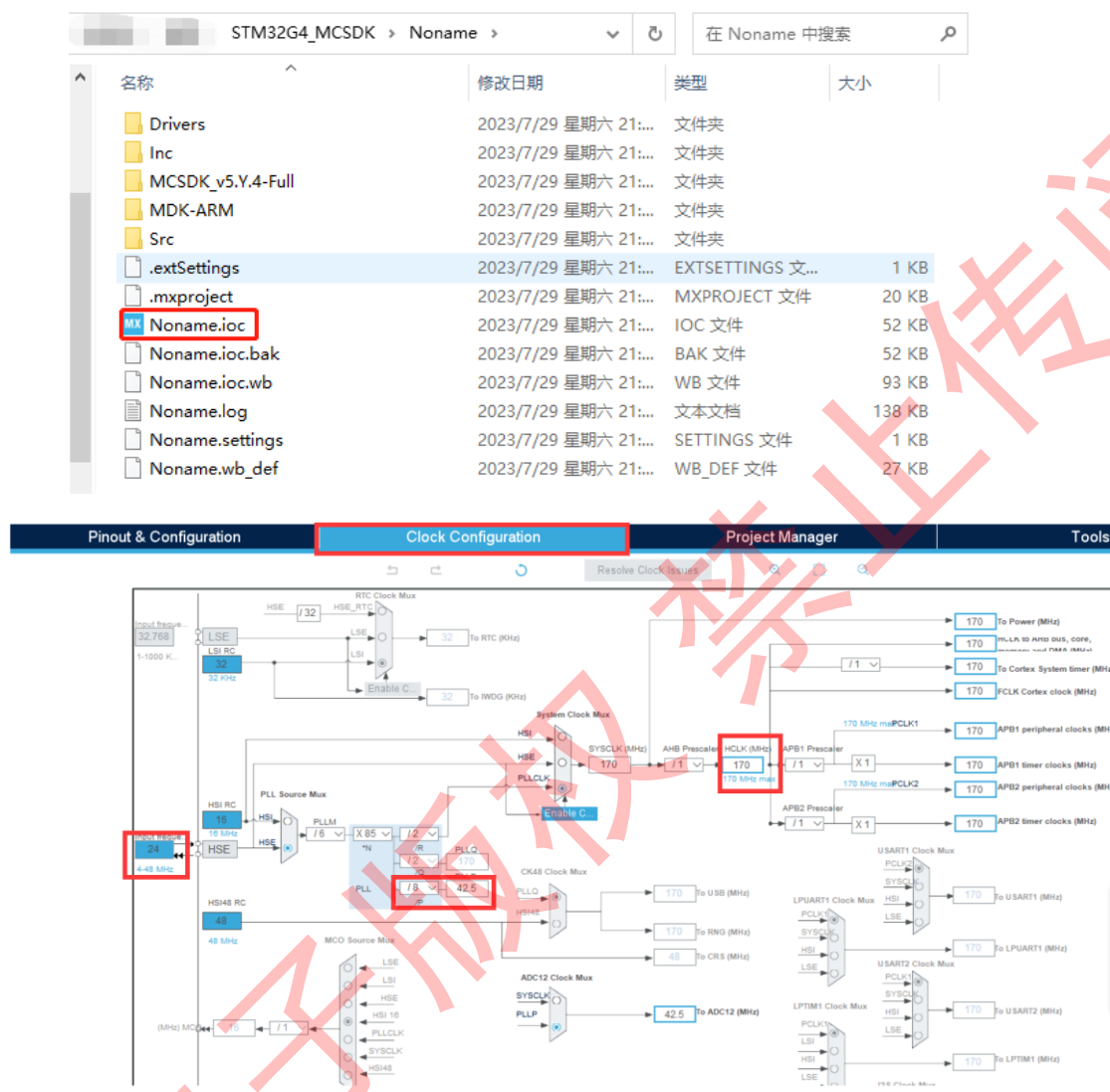
GENERATE



⑮ 生成代码过程中，此类报错可忽略，原因时 CUBEMX 版本过低：

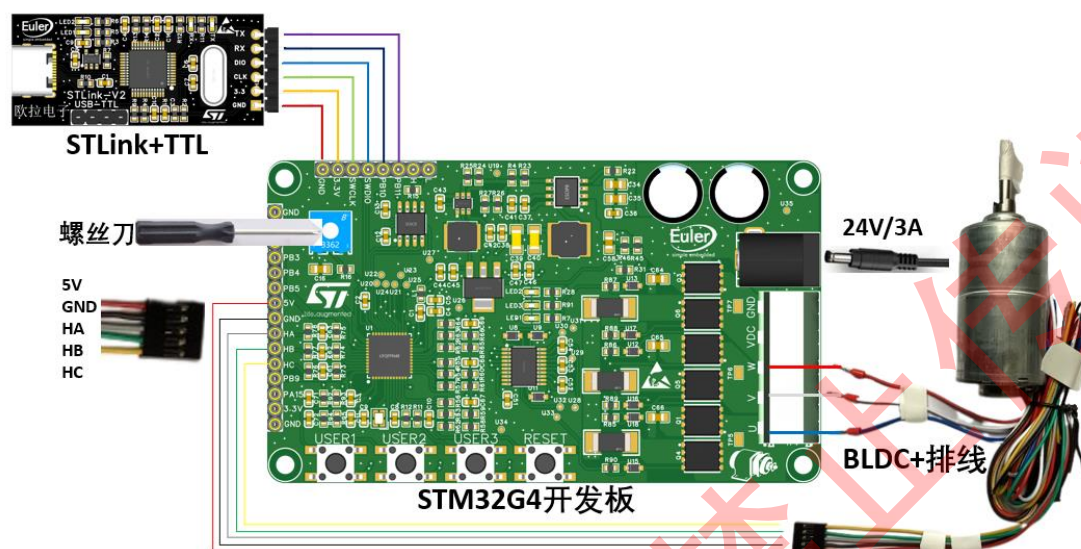
```
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.X-CUBE-SMBUS.2.1.0.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.FP-SNS-MOTENV1.4.3.0.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.FP-SNS-MOTENV1.4.3.2.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.FP-SNS-FLIGHT1.5.0.0.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.FP-SNS-MOTENV1.4.3.1.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.FP-SNS-FLIGHT1.5.0.1.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.FP-SNS-FLIGHT1.5.0.2.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.X-CUBE-DISPLAY.3.0.0.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.X-CUBE-FREERTOS.1.0.0.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.X-CUBE-FREERTOS.1.0.1.pdsc: PDSC version is not supported
C:\Users\Administrator\.stm32cubemx\thirdparties\pdsc\STMicroelectronics.X-CUBE-TOP1.3.3.1.pdsc: PDSC version is not supported
```

⑯ ⑯ 生成代码，打开 IOC 文件修改时钟，然后点击生成代码：

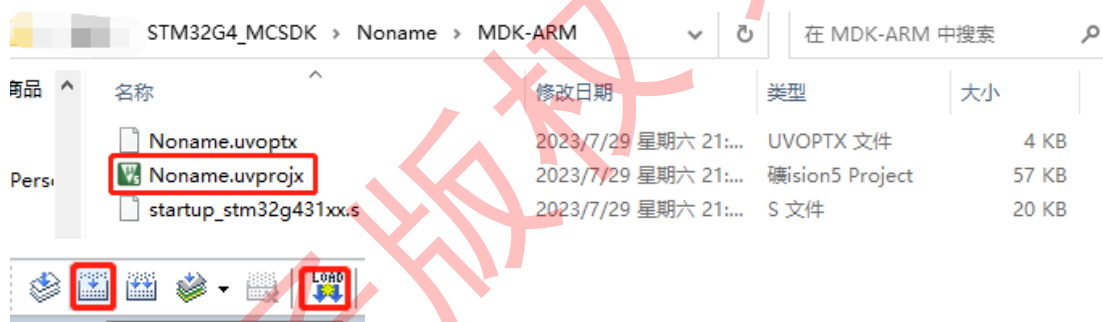


## 1.2 实验现象

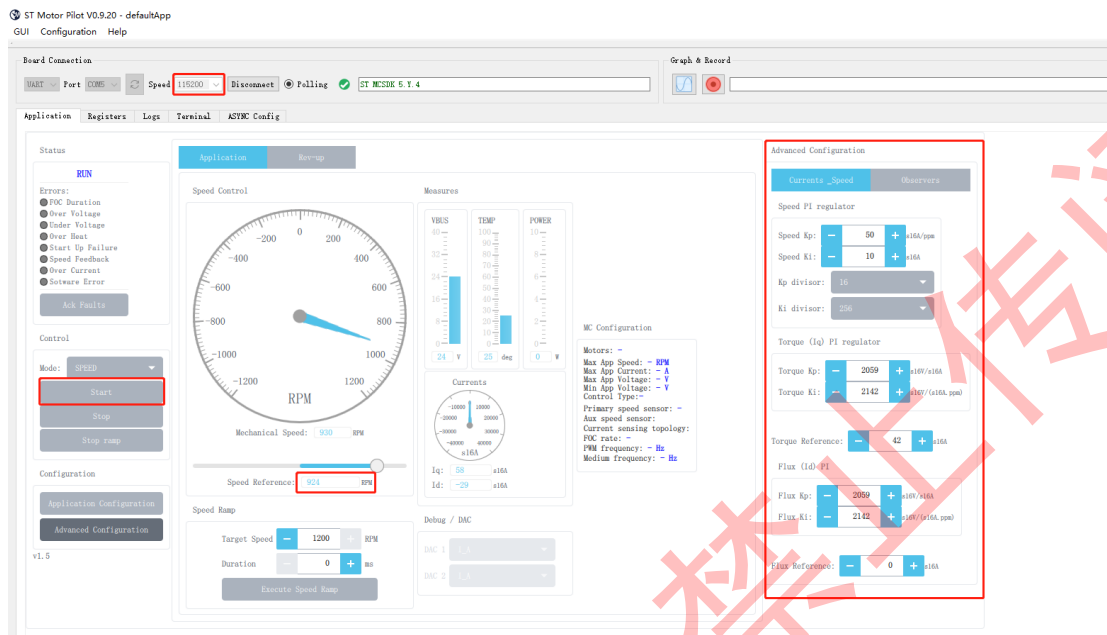
按以下图片接线，并将 USB 转 UART 模块连接至电脑：



①打开 KEIL 工程，使用 stlink 连接开发板 SWD 及 UART，编译并下载代码；

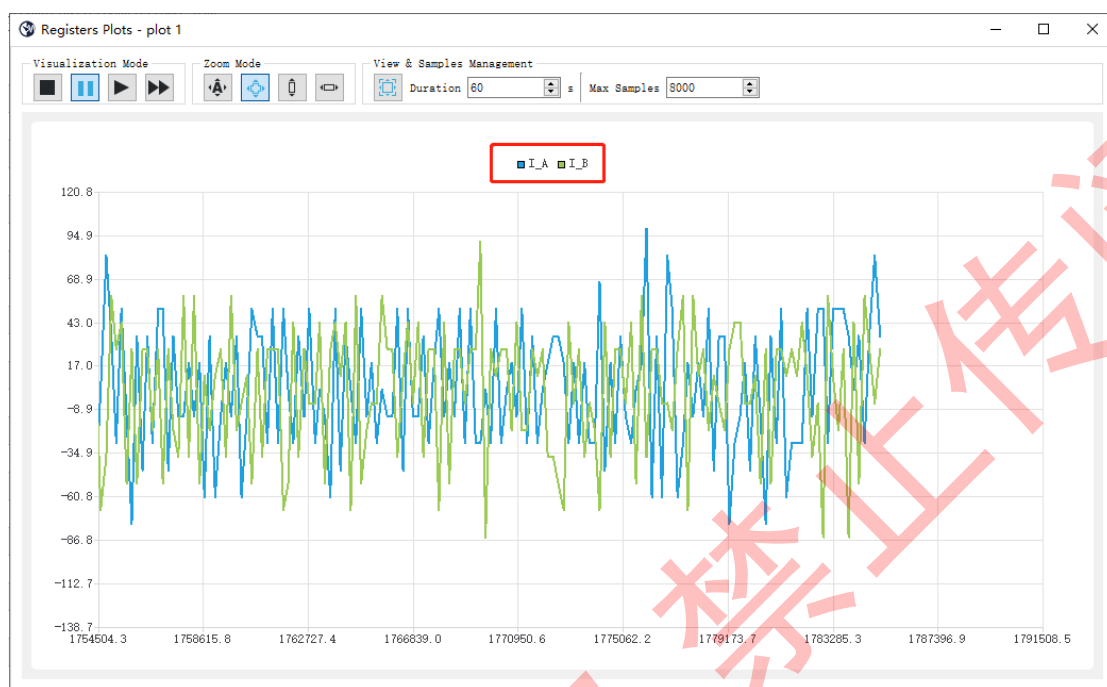


②代码下载成功后，按下开发板 RESET 按键，打开 SDK 上位机  软件，  
打开 UART，波特率设置 115200，点击连接；点击开始，即可启动电机，或按下  
用户按键 3 也可启动电机；

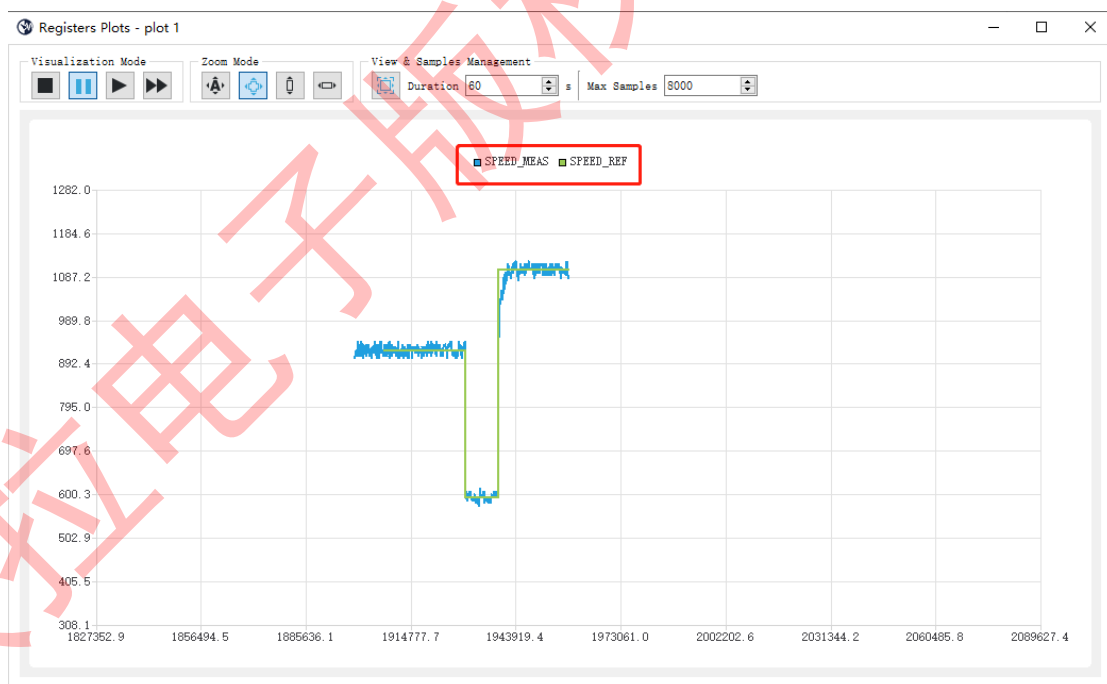


③使用上位机的画图工具可以看到相应数据的波形：

| Pol                                 | Id   | Name            | Value | Access | Unit | Type | Plot      | Status | Description                                            |
|-------------------------------------|------|-----------------|-------|--------|------|------|-----------|--------|--------------------------------------------------------|
| <input checked="" type="checkbox"/> | 1489 | HEATS_TEMP      | 25    | R      | deg  | U16  | Configure | OK     | Heatsink temperature                                   |
| <input checked="" type="checkbox"/> | 1553 | MOTOR_POWER     | 0     | R      | W    | U16  | Configure | OK     | Motor power                                            |
| <input checked="" type="checkbox"/> | 1616 | DAC_OUT1        | 2001  | RW     | N/A  | U16  | Configure | Init   | DAC Out 1                                              |
| <input checked="" type="checkbox"/> | 1680 | DAC_OUT2        | 2001  | RW     | N/A  | U16  | Configure | Init   | DAC Out 2                                              |
| <input checked="" type="checkbox"/> | 1937 | FLUXWK_BUS_MEAS | 0     | R      | %    | U16  | Configure | Init   | Flux weakening BUS Voltage measured percentage         |
| <input checked="" type="checkbox"/> | 2001 | I_A             | 35    | R      | s16A | S16  | plot 1    | OK     | I_a phase current measurement                          |
| <input checked="" type="checkbox"/> | 2065 | I_B             | -85   | R      | s16A | S16  | plot 1    | OK     | I_b phase current measurement                          |
| <input type="checkbox"/>            | 2129 | I_ALPHA_MEAS    | 0     | R      | s16A | S16  | Configure | Init   | Ia current measurement                                 |
| <input type="checkbox"/>            | 2193 | I_BETA_MEAS     | 0     | R      | s16A | S16  | Configure | Init   | Ib current measurement                                 |
| <input checked="" type="checkbox"/> | 2257 | I_Q_MEAS        | 54    | R      | s16A | S16  | Configure | OK     | I_q current measurement (synonymous to Torque measure) |
| <input checked="" type="checkbox"/> | 2321 | I_D_MEAS        | -65   | R      | s16A | S16  | Configure | OK     | I_d current measurement (synonymous to Flux measure)   |
| <input checked="" type="checkbox"/> | 2385 | I_Q_REF         | 39    | RW     | s16A | S16  | Configure | OK     | I_q current reference (synonymous to Torque reference) |
| <input checked="" type="checkbox"/> | 2449 | I_D_REF         | 0     | RW     | s16A | S16  | Configure | OK     | I_d current reference (synonymous to Flux reference)   |
| <input type="checkbox"/>            | 2513 | V_Q             | 0     | R      | s16V | S16  | Configure | Init   | V_q voltage                                            |

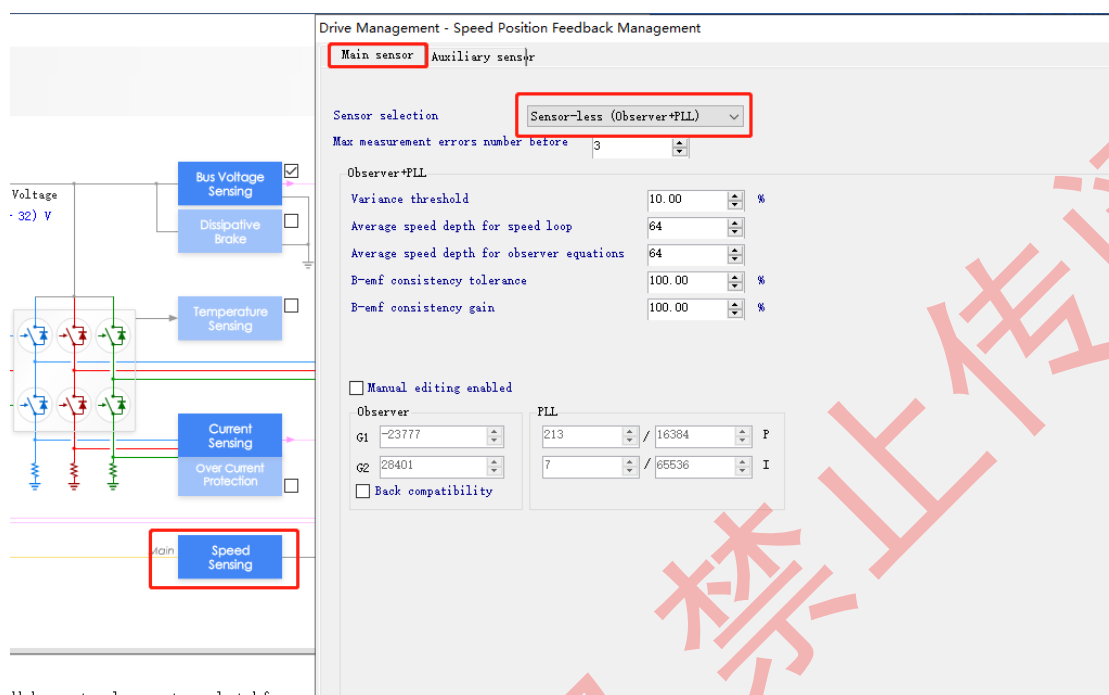


|                                     |     |            |     |   |     |     |        |    |                                  |
|-------------------------------------|-----|------------|-----|---|-----|-----|--------|----|----------------------------------|
| <input checked="" type="checkbox"/> | 89  | SPEED_MEAS | 930 | R | RPM | S32 | plot 1 | OK | Average Mechanical Speed measure |
| <input checked="" type="checkbox"/> | 153 | SPEED_REF  | 924 | R | RPM | S32 | plot 1 | OK | Mechanical speed reference       |

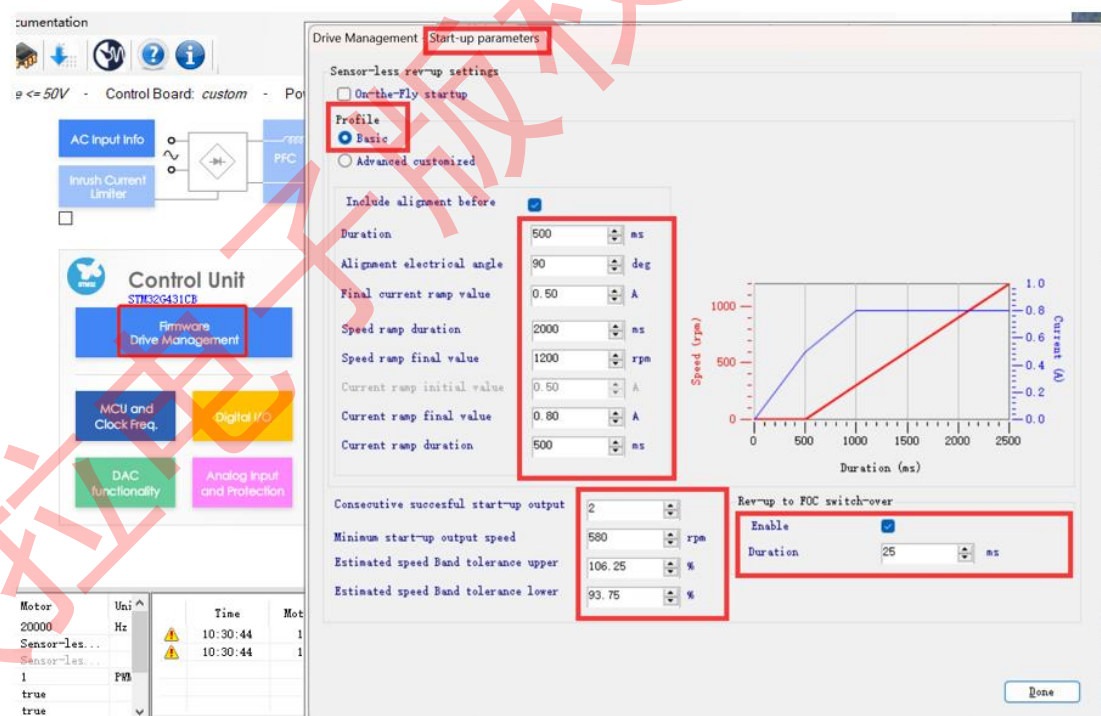


## 2. BLDC 无感 MCSDK 配置及代码生成

① 无感配置需要将速度传感器配置无传感器模式：



# ① 配置无感启动参数：



# ② 其他参数无需修改，点击生成代码，后续流程与霍尔传感器配置流程完全一样；



### 3. 使用电位器调节 BLDC 转速

基于第一小节的 BLDC 霍尔有感代码的基础上增加电位器的 ADC 采样；

- ① 增加电位器 ADC 采样相关变量；

```
58  /* USER CODE BEGIN PV */
59  RegConv_t Poten;
60  uint8_t Poten_handle;
61  uint16_t adc_vaule;
62  int16_t finalspeed;
63  /* USER CODE END PV */
```

- ② 设置采样通道及时间，注册规则组 ADC 采样；

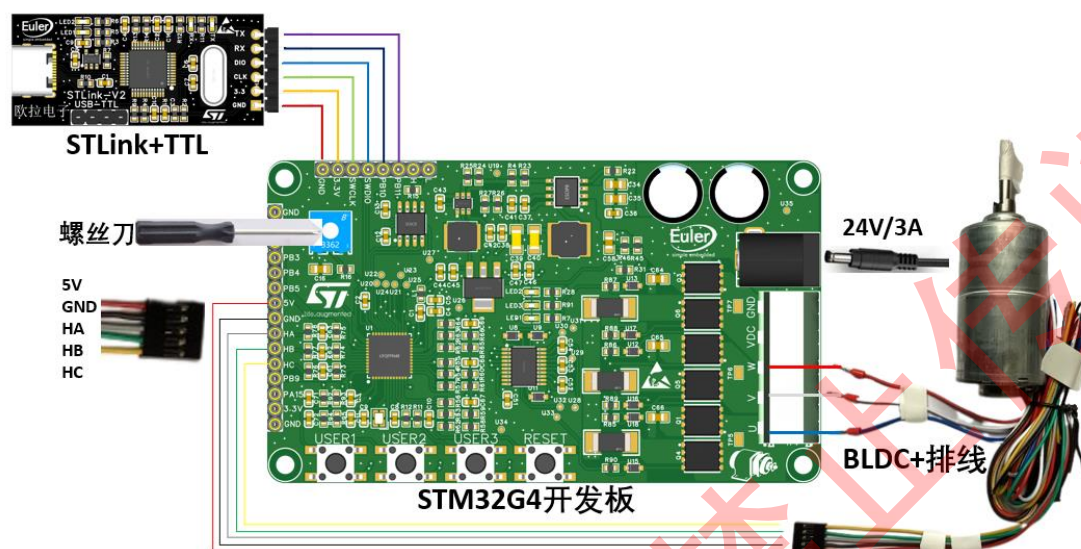
```
128  /* USER CODE BEGIN 2 */
129  Poten.regADC = ADC1;
130  Poten.channel = 11;
131  Poten.samplingTime = ADC_SAMPLETIME_6CYCLES_5;
132  Poten_handle = RCM_RegisterRegConv(&Poten);
133  /* USER CODE END 2 */
```

- ③ 在 while 循环中请求并执行 ADC 转换；采样值 0-128 转速对应-1100~-400；  
采样值 128-256 对应转速 400~1100；

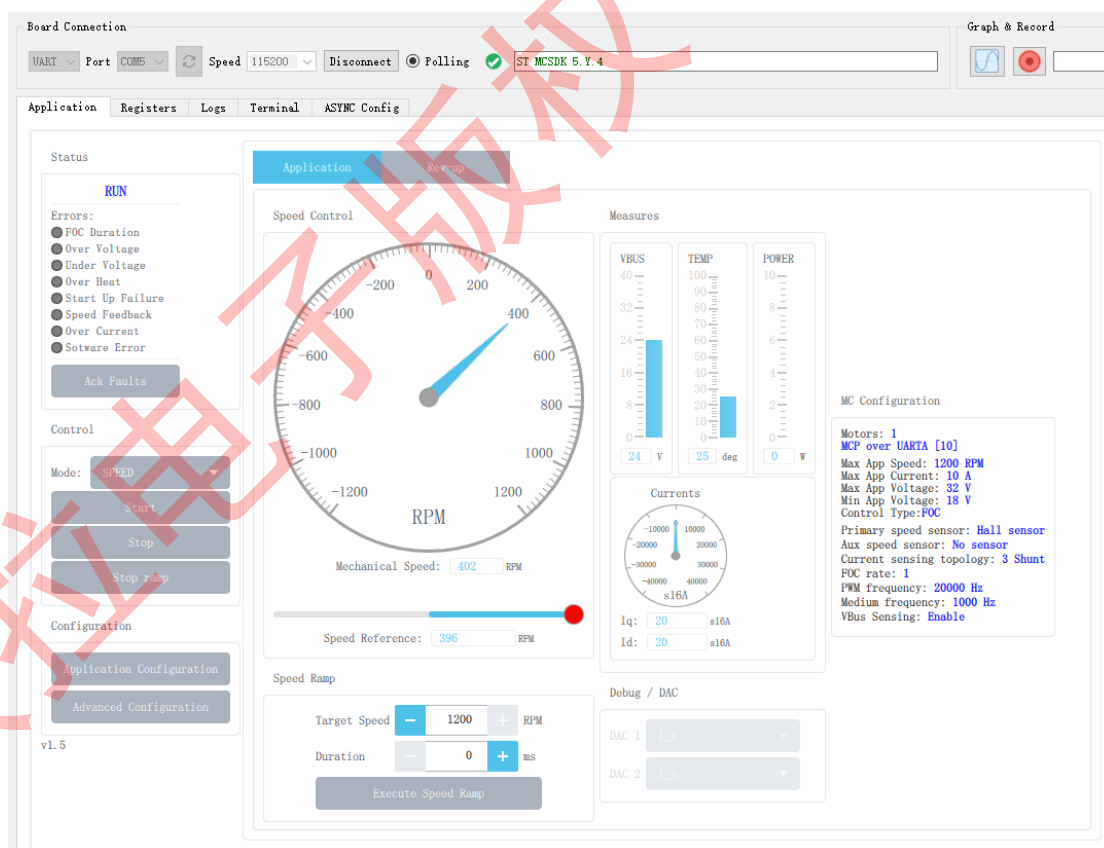
```
137  while (1)
138  {
139      RCM_RequestUserConv(Poten_handle);
140      RCM_ExecRegularConv(Poten_handle);
141      if(RCM_GetUserConvState() == RCM_USERCONV_EOC)
142      {
143          adc_vaule = RCM_GetUserConv();
144      }
145      finalspeed = ((adc_vaule>>8) - 128) * 8;
146      if(finalspeed>0 && finalspeed<400)
147      {
148          finalspeed = 400;
149      }
150      if(finalspeed<0 && finalspeed>-400)
151      {
152          finalspeed = -400;
153      }
154      MC_ProgramSpeedRampMotor1(finalspeed/6,100);
155      HAL_Delay(10);
156  /* USER CODE END WHILE */
```

#### 3.2 实验现象

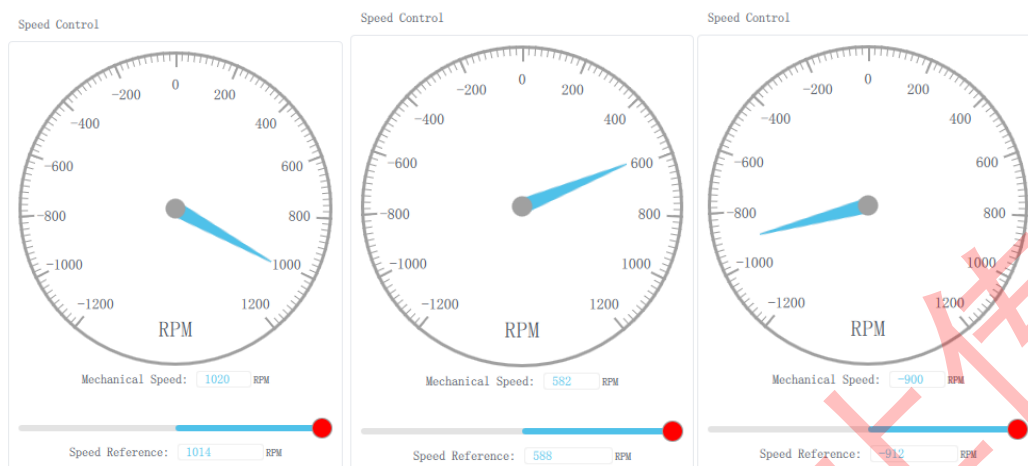
按以下图片接线，并将 USB 转 UART 模块连接至电脑：



① 编译并下载代码，打开 ST 上位机，使用 UART 连接开发板，并启动电机；



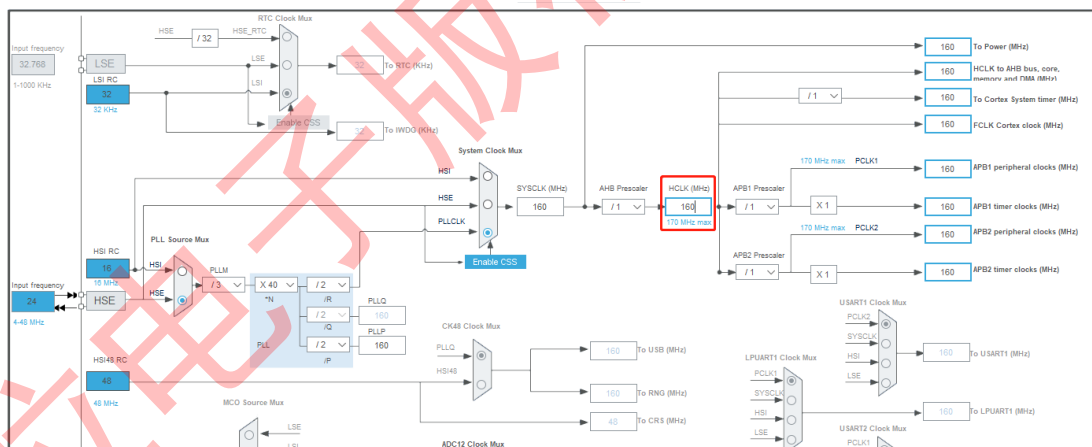
② 使用螺丝刀，旋转电位器，可以看到速度会随着 while 循环中的代码逻辑发生改变；



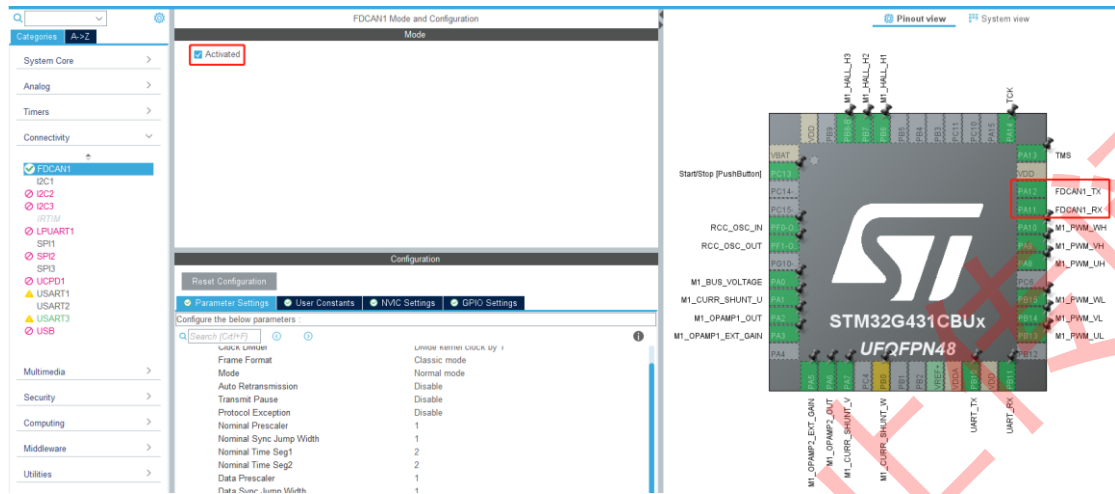
#### 4. 使用 CAN 上位机调节 BLDC 转速

基于第一小节的 BLDC 霍尔有感代码的基础上增加 CAN 通讯代码;

- ① 打开工程中的 IOC 文件，首先将系统时钟改为 160M，修改为 160M 主要是为了方便计算波特率；

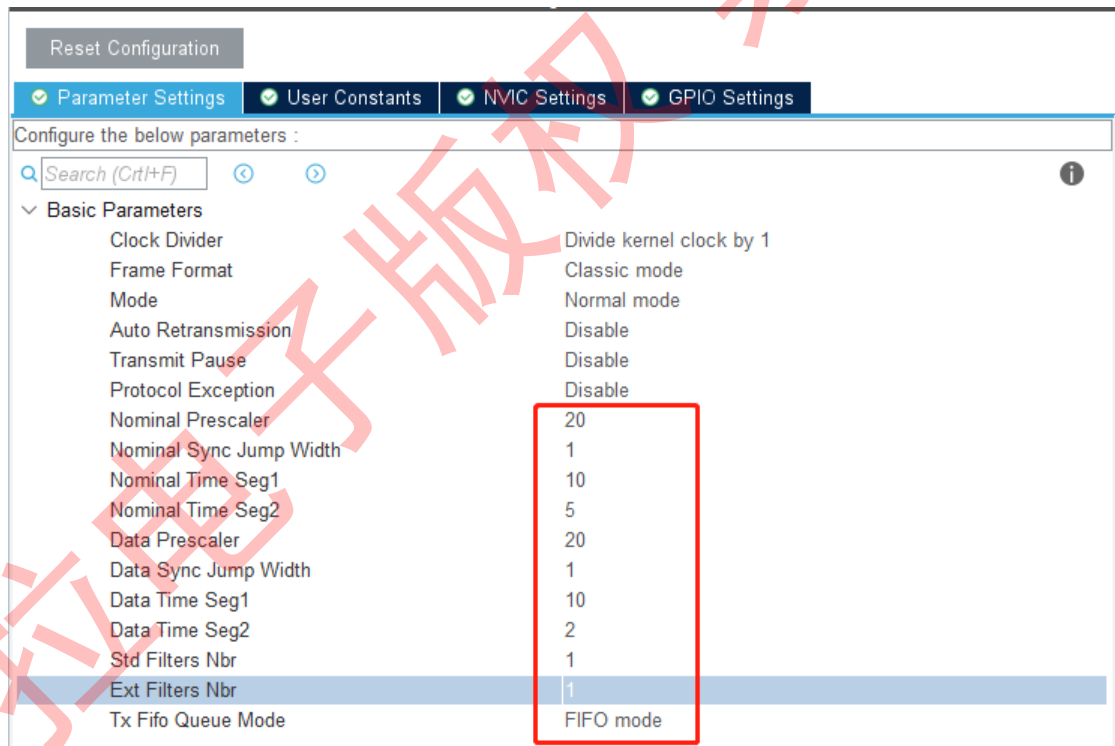


- ② 使能 CANFD 通讯;



③ 配置仲裁段和数据段的分频系数跳转位宽，使能 FIFO 模式；

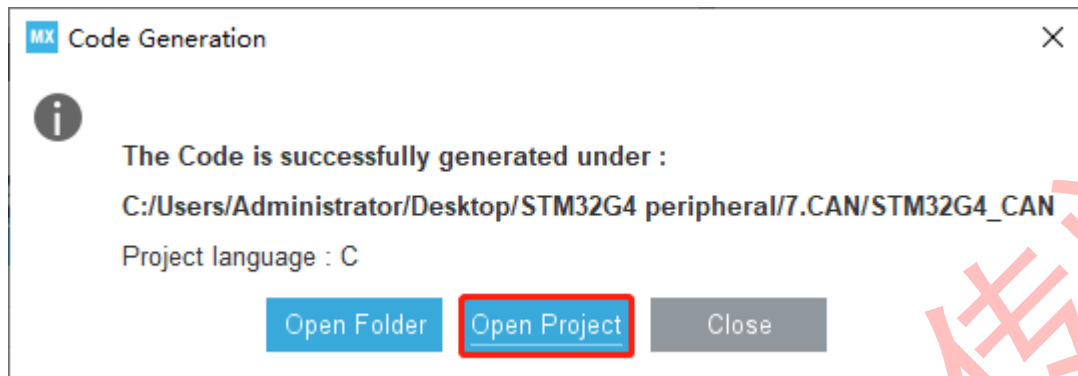
波特率：500k = 160M/20/(1+10+5)



④ 使能中断；

| Parameter Settings   | User Constants                      | NVIC Settings       | GPIO Settings |
|----------------------|-------------------------------------|---------------------|---------------|
| NVIC Interrupt Table |                                     |                     |               |
|                      | Enabled                             | Preemption Priority | Sub Priority  |
| FDCAN1 interrupt 0   | <input checked="" type="checkbox"/> | 0                   | 0             |
| FDCAN1 interrupt 1   | <input type="checkbox"/>            | 0                   | 0             |

⑤ 点击生成代码 **GENERATE CODE**，打开 Keil 工程；



⑥ 定义 RX 和 TX 数据及中间变量；

```
60  /* USER CODE BEGIN PV */
61  FDCAN_RxHeaderTypeDef RxHeader;
62  FDCAN_TxHeaderTypeDef TxHeader;
63  uint8_t RxData[8]={NULL};
64  uint8_t TxData[8] = {NULL};
65  /* USER CODE END PV */
```

⑦ 配置 RX 和 TX 数据帧类型，RX 接收的 ID 范围；TX ID 和数据长度等参数；

```
75  /* USER CODE BEGIN PFP */
76  void FDCAN2_Config(void);
77  /* USER CODE END PFP */

140  FDCAN2_Config();
141  /* USER CODE END 2 */

327 void FDCAN2_Config(void)
328 {
329     FDCAN_FilterTypeDef sFilterConfig;
330
331     HAL_FDCAN_ActivateNotification(&hfdcan1, FDCAN_IT_RX_FIFO0_NEW_MESSAGE, 0);
332
333     sFilterConfig.IdType = FDCAN_EXTENDED_ID;
334     sFilterConfig.FilterIndex = 0;
335     sFilterConfig.FilterType = FDCAN_FILTER_RANGE;
336     sFilterConfig.FilterConfig = FDCAN_FILTER_TO_RXFIFO0;
337     sFilterConfig.FilterID1 = 0x00000000;
338     sFilterConfig.FilterID2 = 0x01ffffff;
339     HAL_FDCAN_ConfigFilter(&hfdcan1, &sFilterConfig);
340
341     TxHeader.Identifier = 0x1B;
342     TxHeader.IdType = FDCAN_EXTENDED_ID;
343     TxHeader.TxFrametype = FDCAN_DATA_FRAME;
344     TxHeader.DataLength = FDCAN_DLC_BYTES_8;
345     TxHeader.ErrorStateIndicator = FDCAN_ESI_ACTIVE;
346     TxHeader.BitRateSwitch = FDCAN_BRS_OFF;
347     TxHeader.FDFormat = FDCAN_CLASSIC_CAN;
348     TxHeader.TxEventFifoControl = FDCAN_NO_TX_EVENTS;
349     TxHeader.MessageMarker = 0x52;
350
351     HAL_FDCAN_Start(&hfdcan1);
352 }
```

⑧ 在 it.c 文件中，在 FDCAN 中断里面，接收上位机发送的数据；

```

65  /* USER CODE BEGIN EV */
66  extern FDCAN_HandleTypeDef hfdcan1;
67  extern FDCAN_RxHeaderTypeDef RxHeader;
68  extern uint8_t RxData[8];
69  /* USER CODE END EV */

241 void FDCAN1_IT0_IRQHandler(void)
242 {
243     /* USER CODE BEGIN FDCAN1_IT0_IRQHandler 0 */
244
245     /* USER CODE END FDCAN1_IT0_IRQHandler 0 */
246     HAL_FDCAN_IRQHandler(&hfdcan1);
247     /* USER CODE BEGIN FDCAN1_IT0_IRQHandler 1 */
248     HAL_FDCAN_GetRxMessage(&hfdcan1, FDCAN_RX_FIFO0, &RxHeader, RxData);
249     /* USER CODE END FDCAN1_IT0_IRQHandler 1 */
250 }
251

```

- ⑨ 增加 math 及 string 头文件；

```

22  #include <math.h>
23  #include <string.h>

```

- ⑩ 定义速度、模式及方向等需要上传的变量；

```

67  float AverageMecSpeed = 0;
68  STC_Modality_t ControlMode;
69  uint16_t Speed = 0;
70  uint8_t Direction = 0;
71  uint8_t mode = 0;

```

- ⑪ 在 while 循环中增加获取当前电机速度、方向、控制模式等函数；并将实时参数上传至上位机；同时接收上位机下发的电机启动、速度等指令；



```

145 while (1)
146 {
147     AverageMecSpeed = MC_GetAverageMecSpeedMotor1_F();
148     ControlMode = MC_GetControlModeMotor1();
149     if(AverageMecSpeed >= 0)
150     {
151         Direction = 1;
152         Speed = (uint16_t)AverageMecSpeed;
153     }
154     else
155     {
156         Direction = 0;
157         Speed = (uint16_t)fabs(AverageMecSpeed);
158     }
159     if(ControlMode == STC_TORQUE_MODE)
160     {
161         mode = 0;
162     }
163     else
164     {
165         mode = 1;
166     }
167     TxData[0] = mode;
168     TxData[1] = Direction;
169     TxData[2] = Speed>>8;
170     TxData[3] = Speed;
171     HAL_FDCAN_AddMessageToTxFifoQ(&hfdcan1, &TxHeader, TxData);
172     if(0x01 == RxData[0])
173     {
174         if(0x01 == RxData[4])
175         {
176             MC_StartMotor1();
177         }
178         else
179         {
180             MC_StopMotor1();
181         }
182         if(0x01 == RxData[1])
183         {
184             MC_ProgramSpeedRampMotor1((RxData[2]*256 + RxData[3])/6, 1000);
185         }
186         else
187         {
188             MC_ProgramSpeedRampMotor1((-RxData[2]*256 + RxData[3])/6, 1000);
189         }
190     }
191     memset((void *)RxData, 0, 8);
192     HAL_Delay(500);
193     /* USER CODE END WHILE */

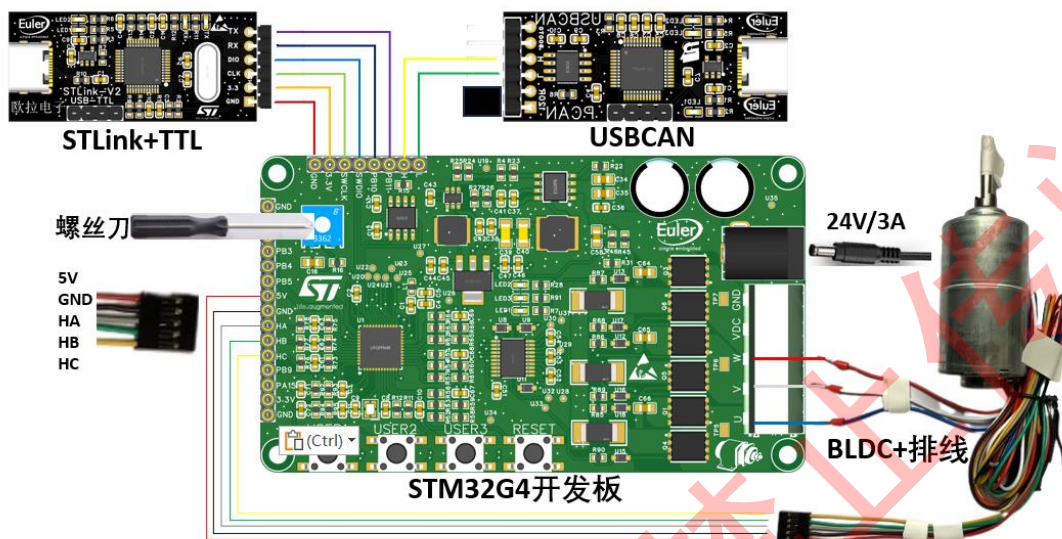
```

⑫ 编译并下带代码;

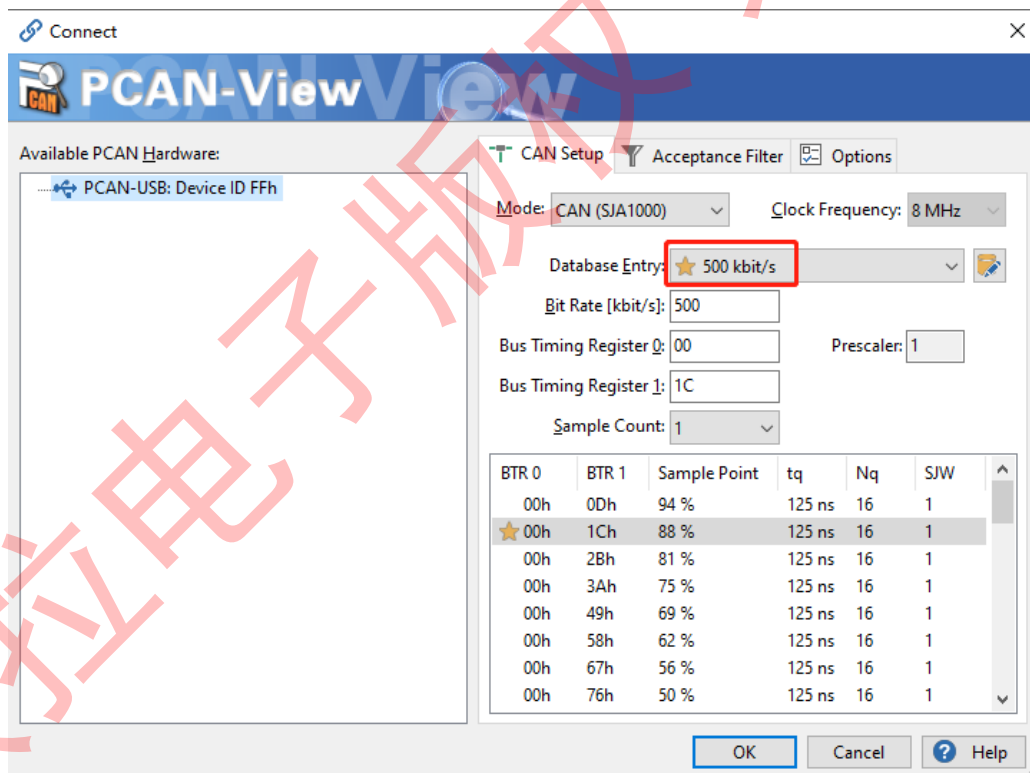
## 4.2 实验现象

按以下图片接线, 并将 USB 转 UART 及 USBCAN 模块连接至电脑:





- ① 打开 ST 上位机及 PCAN 上位机，并连接，CAN 通讯速率 500kbits;



Receive / Transmit

Trace

PCAN-USB

| CAN-ID    | Type | Length | Data                    | Cycle Time | Count |
|-----------|------|--------|-------------------------|------------|-------|
| 0000001Bh |      | 8      | 01 01 00 00 00 00 00 00 | 501.2      | 6     |

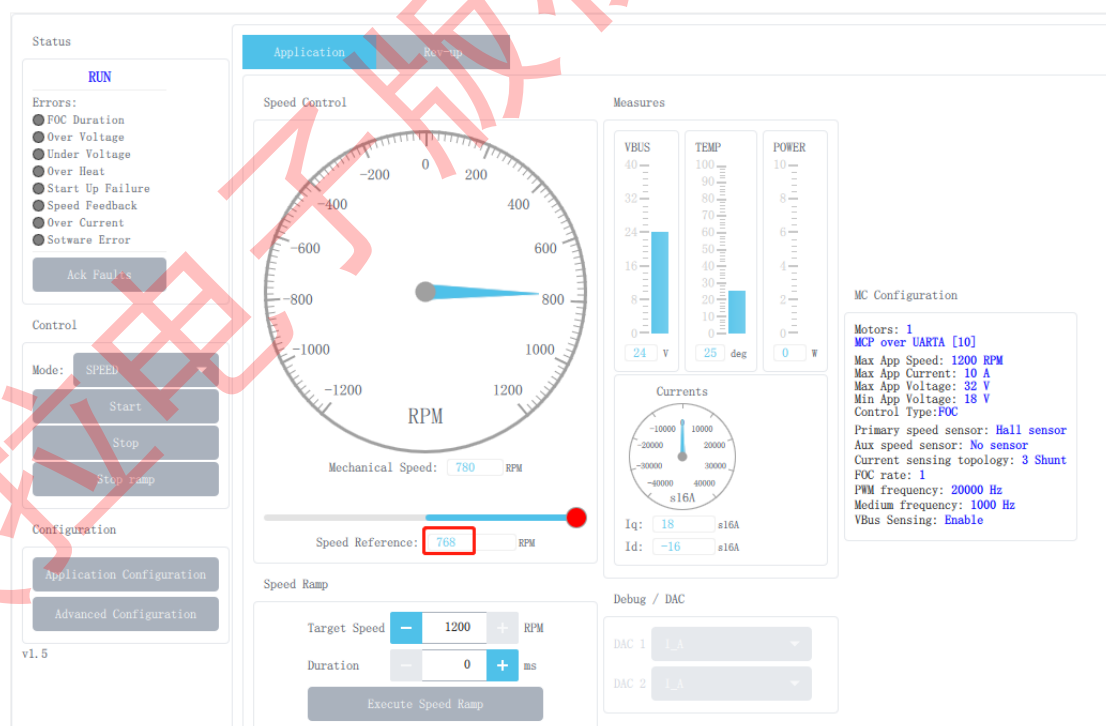
Receive

- ② 在发送数据栏中，发送该条数据即可启动电机，其中第 0 位为 1 并且第四位为 1 即可启动电机；第 1 位为 1 代表正向启动；第 2 及第三位代表速度， $RxData[2]*256 + RxData[3]$ ；

Transmit

| <input type="checkbox"/> CAN-ID | Type | Length | Data                    | Cycle Time                               | Count | Trigger | Comment |
|---------------------------------|------|--------|-------------------------|------------------------------------------|-------|---------|---------|
| 0000011Bh                       |      | 8      | 01 01 03 03 01 00 00 00 | <input checked="" type="checkbox"/> 1000 | 71    | Time    |         |

- ③ 观察 ST 上位机的速度，与我们期望参考速度一致；

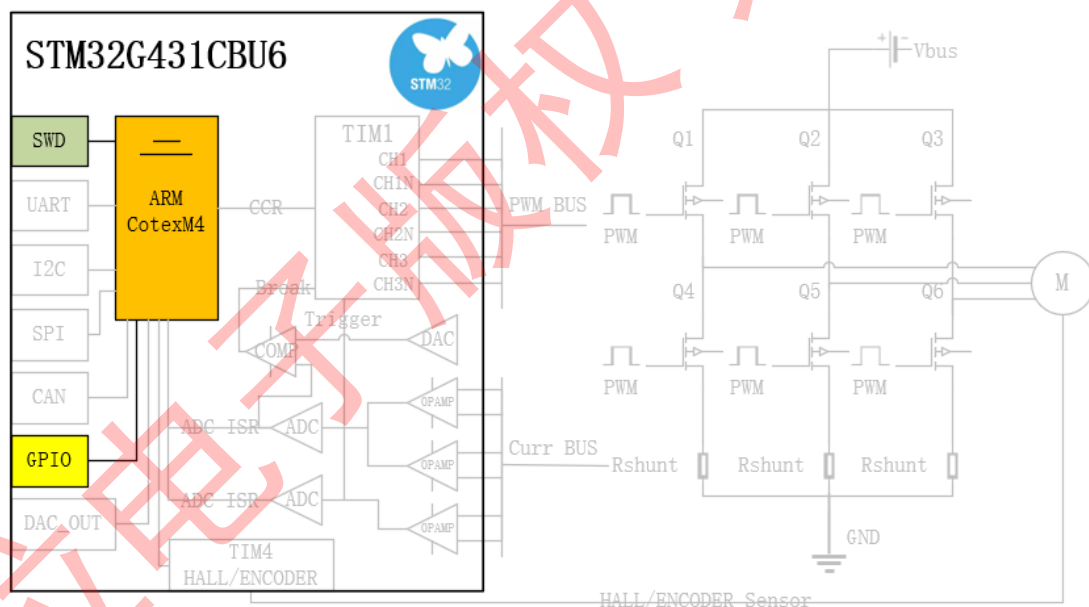


- ④ 修改第 2 及 3 位数据即可修改电机参考转速；将第 0 位为 1 并且第四位为 0 即可停止电机；

| CAN-ID    | Type | Length | Data                 | Cycle Time | Count | Trigger | Comment |
|-----------|------|--------|----------------------|------------|-------|---------|---------|
| 0000011Bh |      | 8      | 01 03 03 00 00 00 00 | 1000       | 224   | Time    |         |

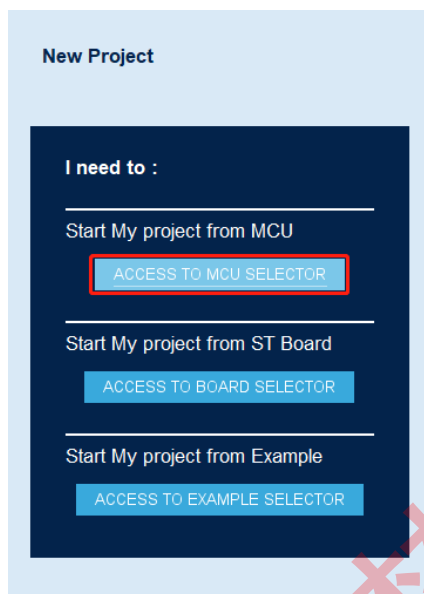
## STM32G4 电机外设篇

### 第五章：MCU 基础配置及 GPIO、EXTI 外设

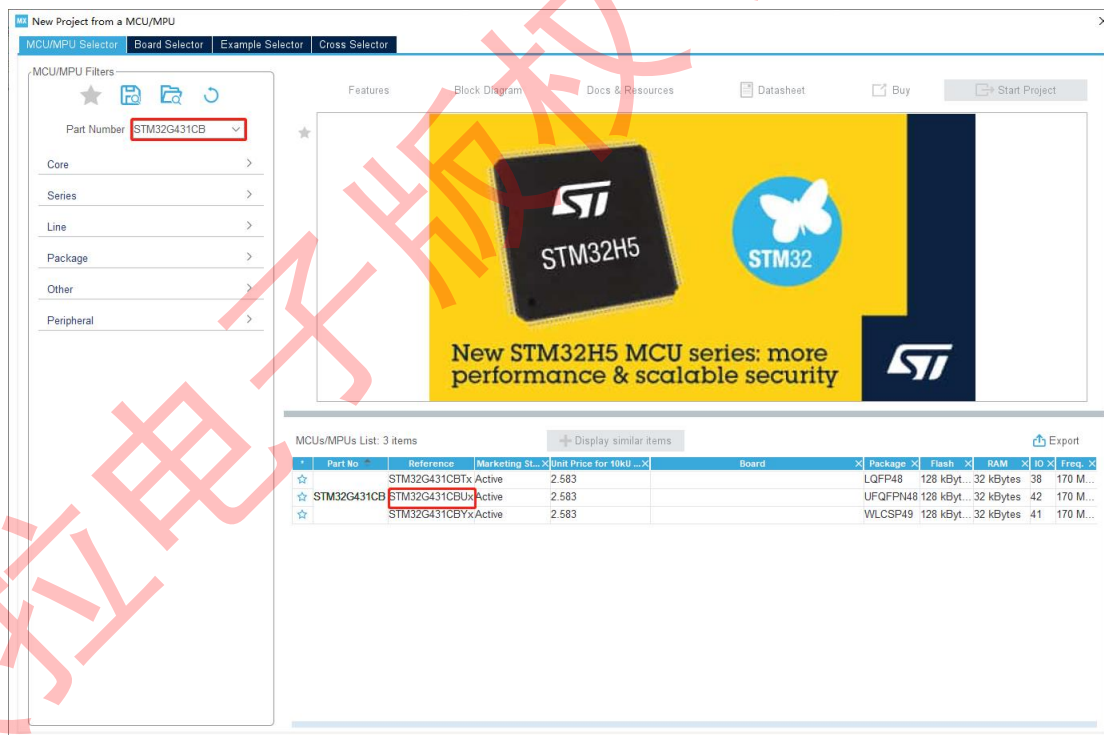


#### 1. STM32CUBEMX 配置及 Keil 代码

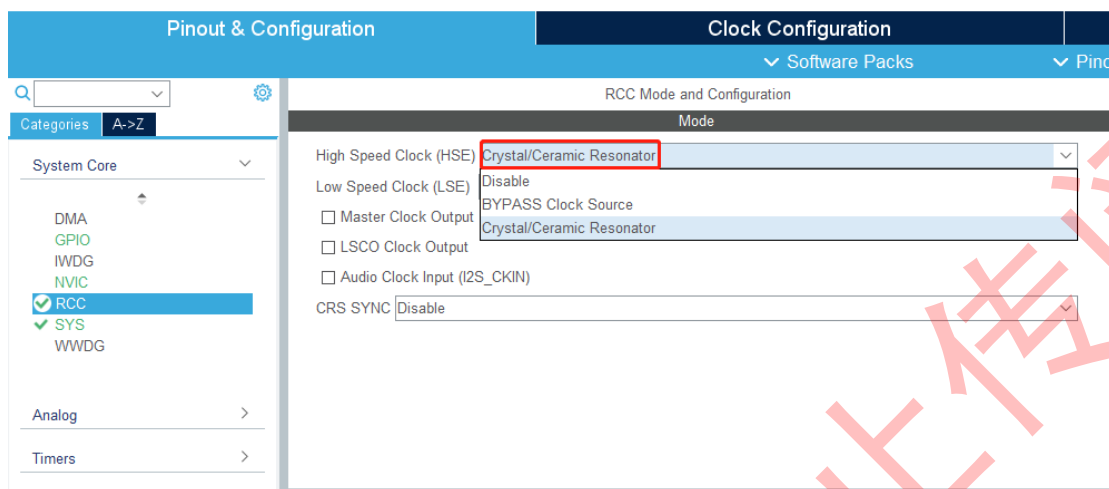
- ① 双击打开 STM32CUBEMX，选择 MCU 型号；



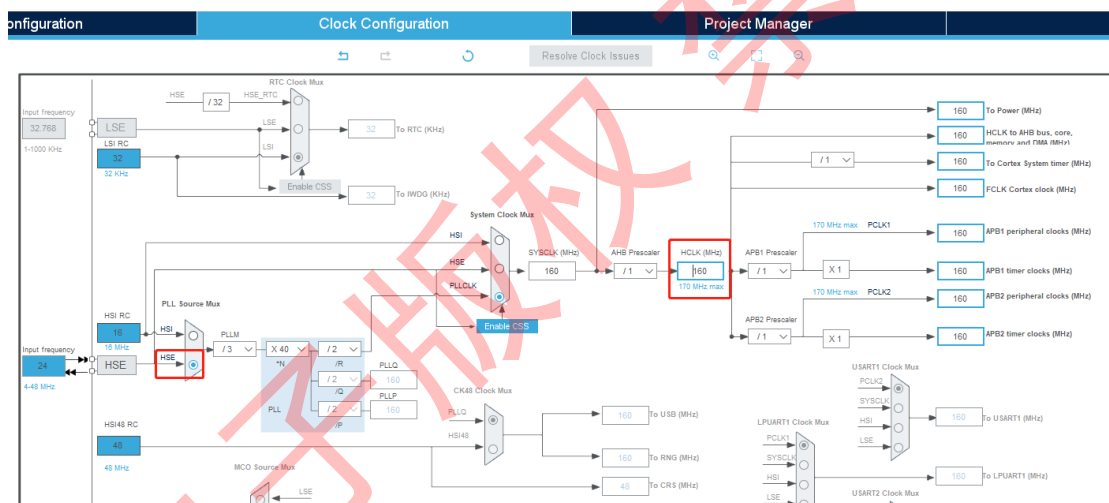
② 选择 STM32G431CBUX;



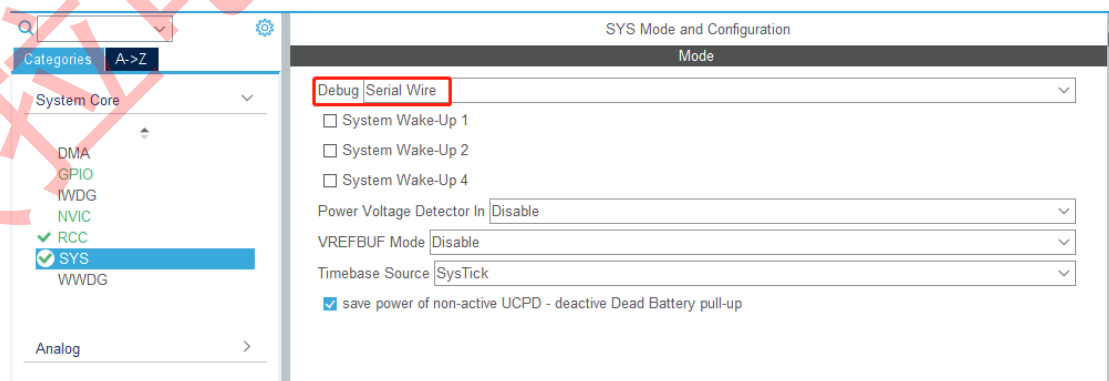
③ 选择 MCU HSE 为外部晶振;



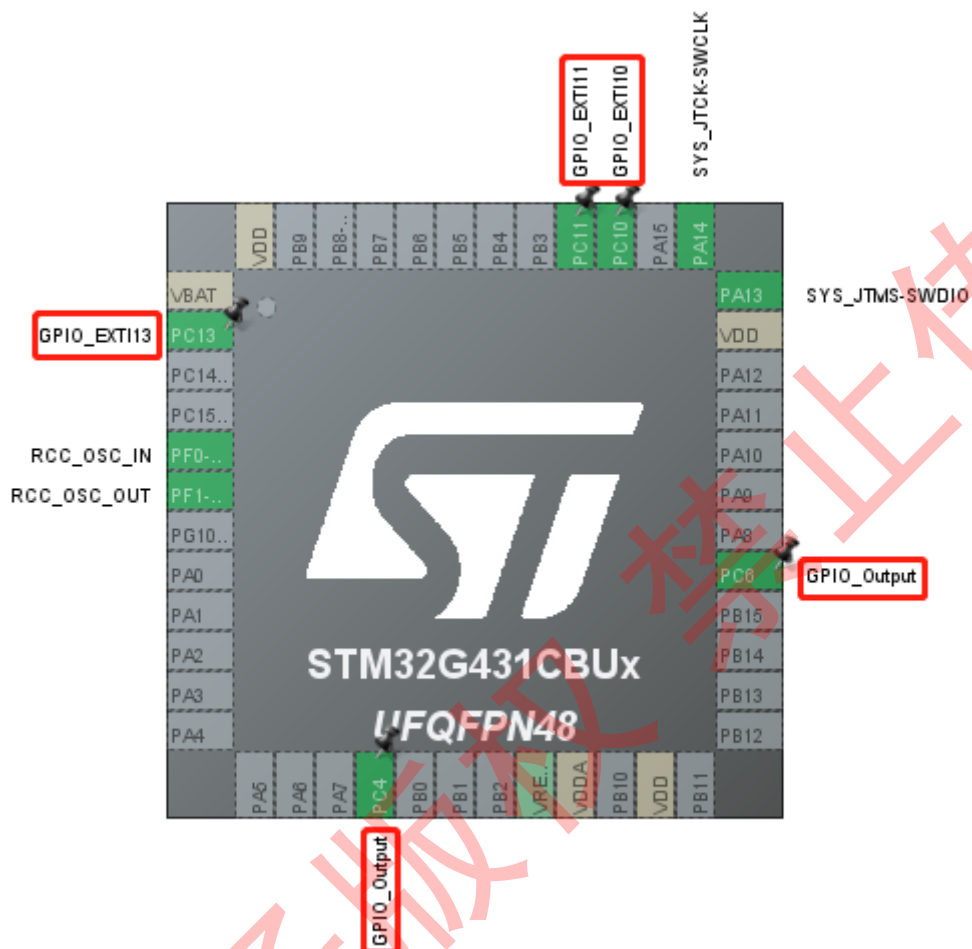
④ 配置时钟树，PLL 时钟源选择 HSE，外部晶振 24M，HCLK 配置为 160M：



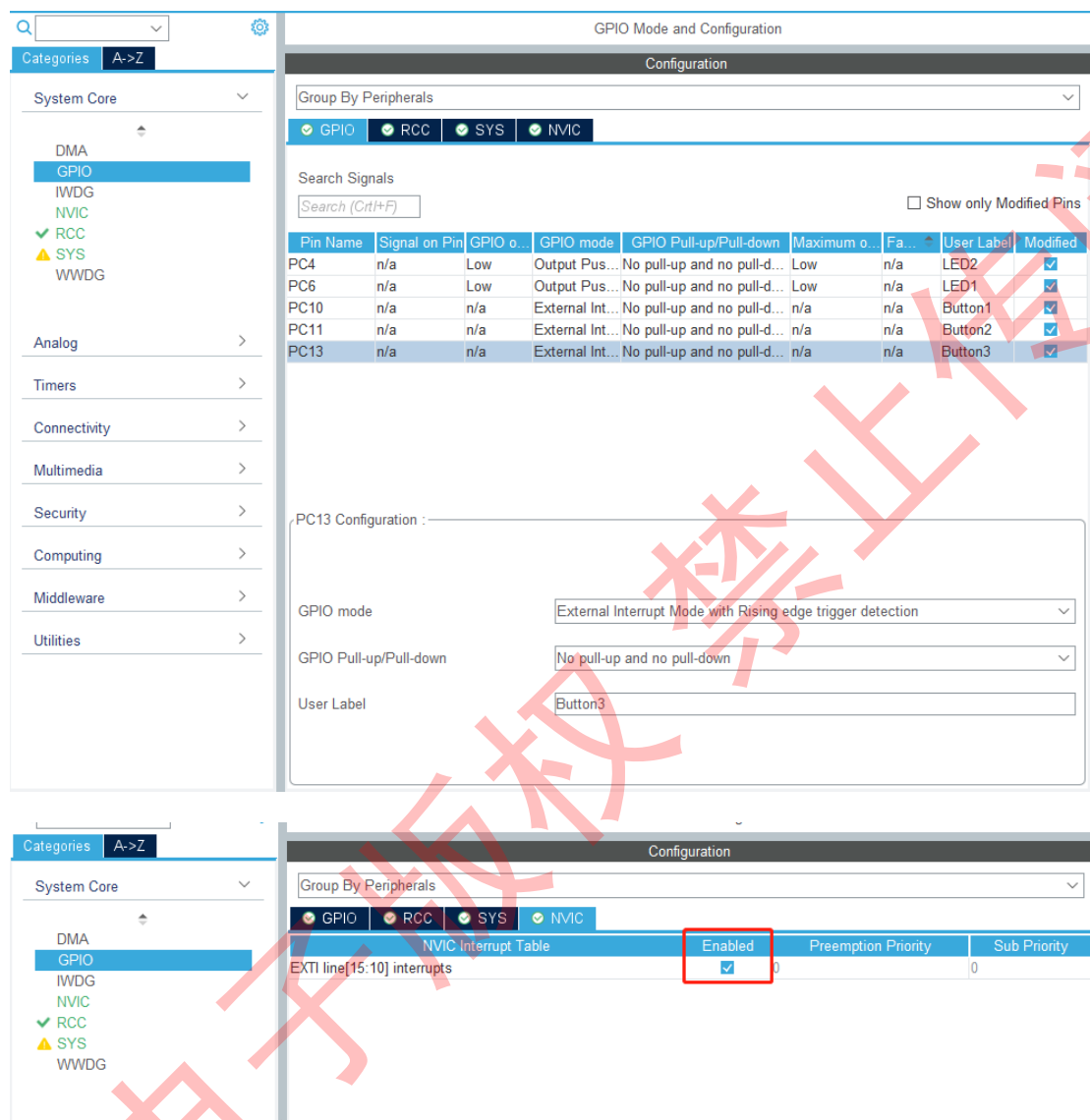
⑤ debug 接口配置为串行接口;



⑥ 配置 LED 等及按键的 GPIO 口，分别配置为 GPIO OUTPUT 和 GPIO EXTI;



- ⑦ 配置 GPIO 的模式，LED GPIO 配置为推挽输出，EXTI GPIO 配置为上升沿检测；同时修改各 GPIO 的名称；并使能 EXTI 中断；



⑧ 配置工程名称，路径，编译器；



**Pinout & Configuration**

**Project**

Project Settings

Project Name  
STM32G4\_GPIO

Project Location  
C:\Users\Administrator\Desktop\STM32G4 peripheral\1.GPIO

Application Structure  
Advanced

Toolchain Folder Location  
STM32G4\_GPIO\

Toolchain / IDE  
MDK-ARM

Min Version  
V5.32

**Code Generator**

Linker Settings

Minimum Heap Size  
0x200

Minimum Stack Size  
0x400

**Advanced Settings**

Thread-safe Settings

Cortex-M4NS

Enable multi-threaded support

Thread-safe Locking Strategy  
Default - Mapping suitable strategy depending on RTOS selection.

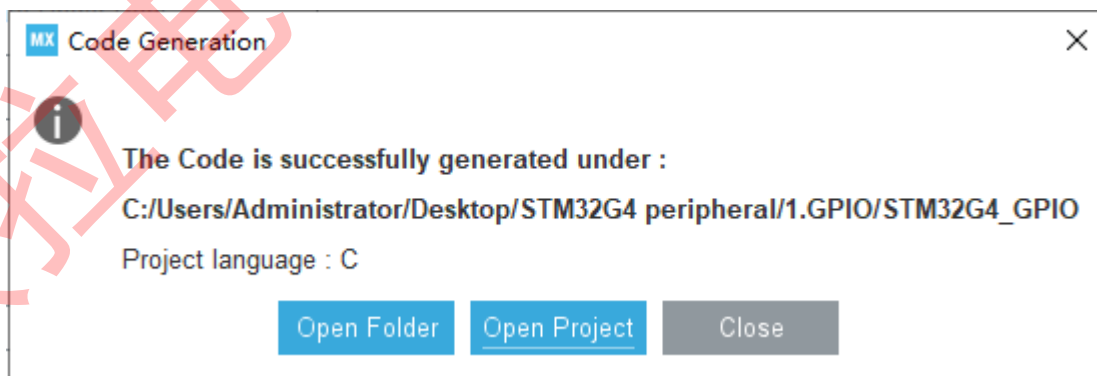
Mcu and Firmware Package

Mcu Reference  
STM32G431CBUx

Firmware Package Name and Version  
STM32Cube FW\_G4 V1.4.0

Use Default Firmware Location

⑨ 点击生成代码，并打开工程；



⑩ 在 main.c 中增加以下函数，函数功能为：按键 1 按下翻转 LED1，按键 2 按下翻转 LED2，按键 3 按下，LED1 和 LED2 全部打开；

```

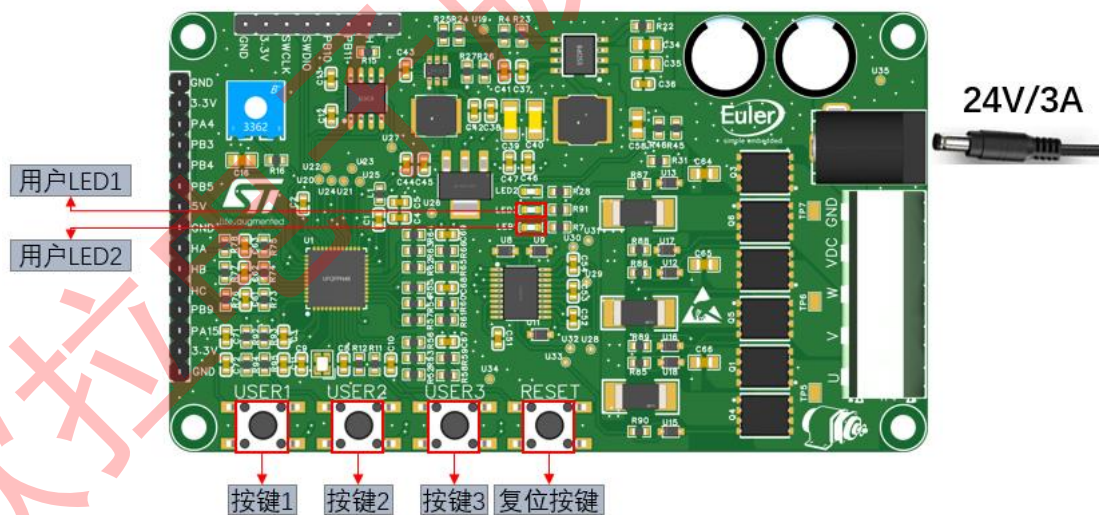
146  /* USER CODE BEGIN 4 */
147  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
148  {
149      /* Prevent unused argument(s) compilation warning */
150      UNUSED(GPIO_Pin);
151      if(Button1_Pin == GPIO_Pin)
152      {
153          HAL_GPIO_TogglePin(LED1_GPIO_Port,LED1_Pin);
154      }
155      if(Button2_Pin == GPIO_Pin)
156      {
157          HAL_GPIO_TogglePin(LED2_GPIO_Port,LED2_Pin);
158      }
159      if(Button3_Pin == GPIO_Pin)
160      {
161          HAL_GPIO_WritePin(LED1_GPIO_Port,LED1_Pin,GPIO_PIN_RESET);
162          HAL_GPIO_WritePin(LED2_GPIO_Port,LED2_Pin,GPIO_PIN_RESET);
163      }
164      /* NOTE: This function should not be modified, when the callback is needed,
165       the HAL_GPIO_EXTI_Callback could be implemented in the user file
166       */
167  }
168  /* USER CODE END 4 */

```

⑪ 编译并下载代码至开发板；

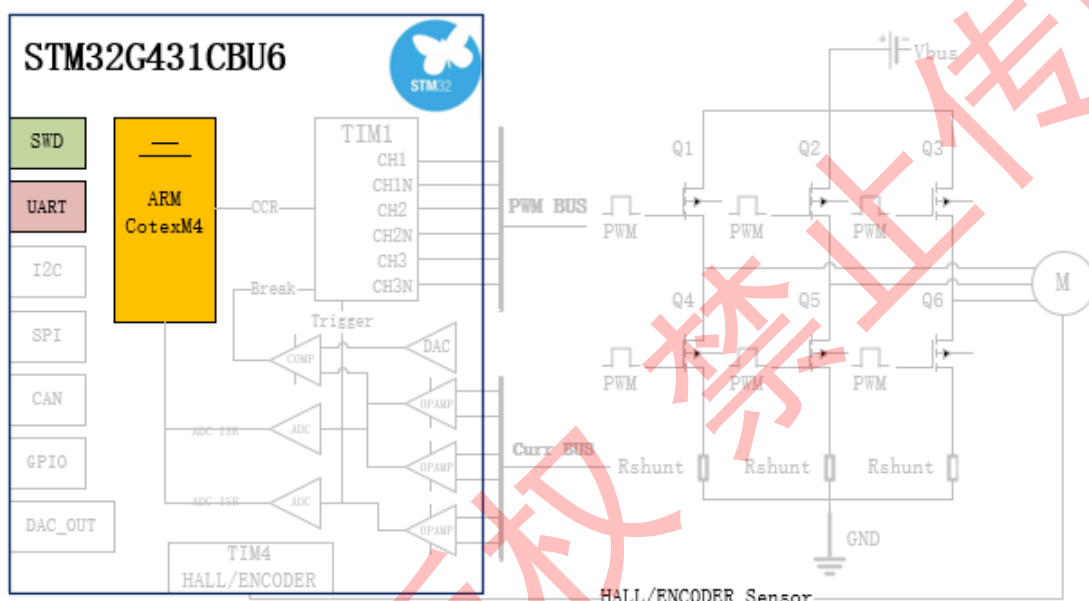
## 2. 实验现象

按以下图片接线：



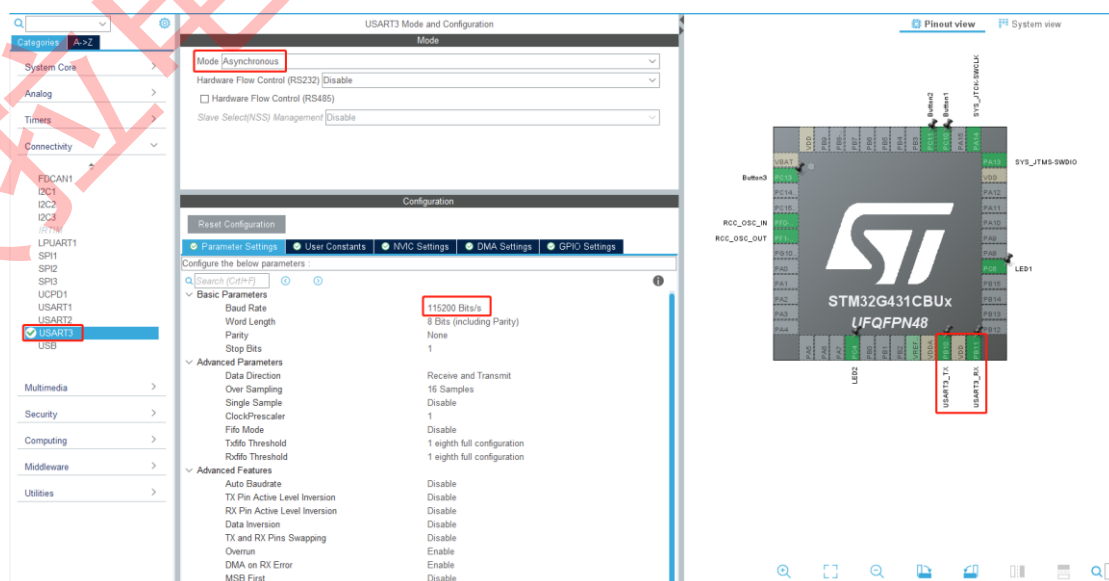
按下 RESET 按键，LED1 和 LED2 亮，按下按键 1，LED1 等翻转；按下按键 2，LED2 等翻转；按下按键 3，LED1 和 LED2 灯同时亮；

## 第六章: UART 外设及 VOFA+上位机 通讯

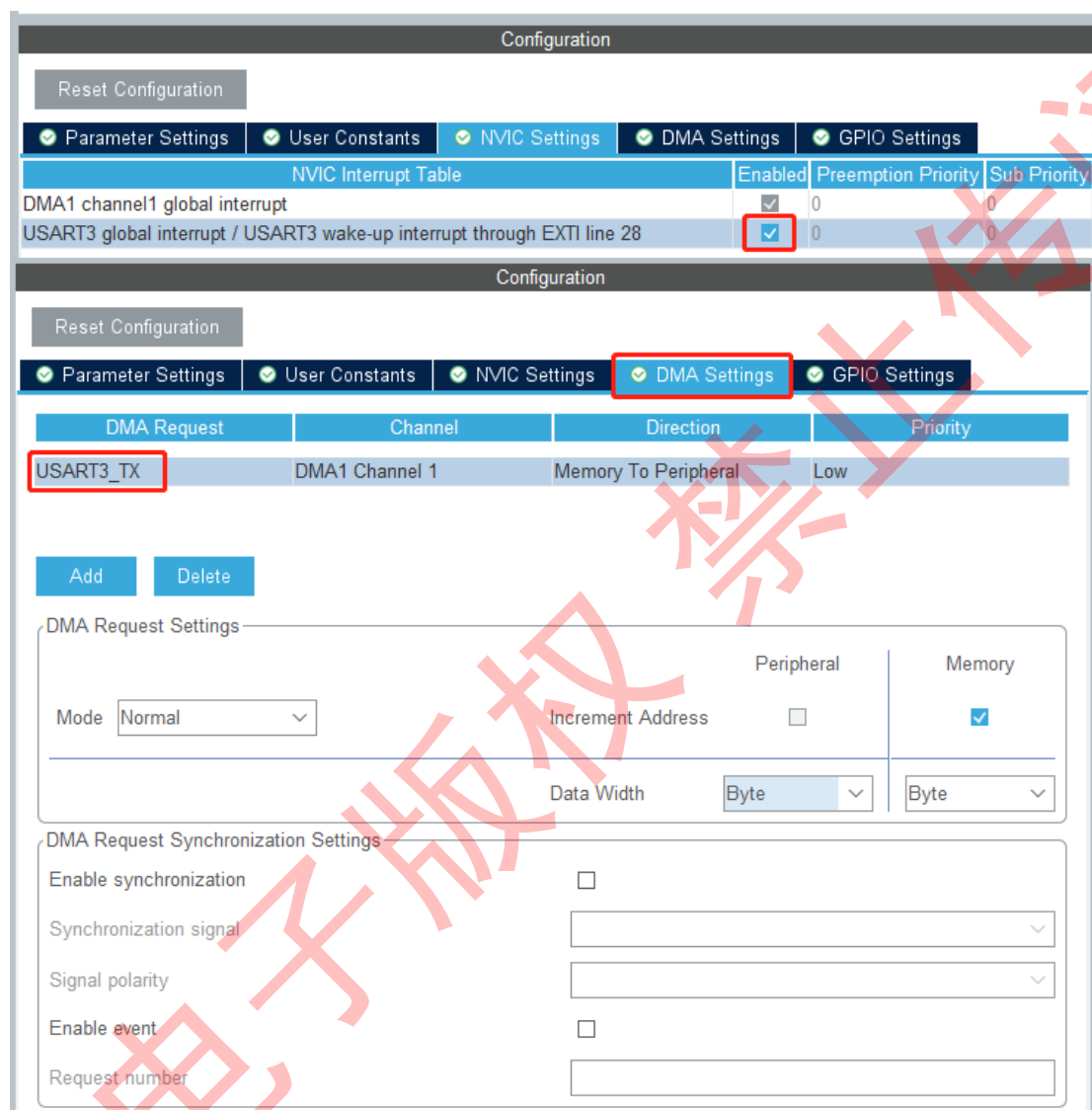


### 1. STM32CUBEMX 配置及 Keil 代码

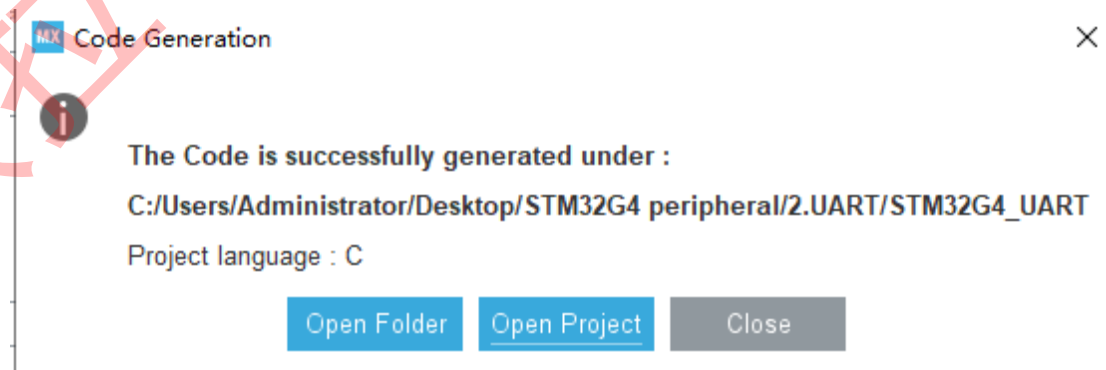
① 在上一章的基础上, 使能 UASRT3, 对应 PB10 和 PB11 IO 口, 将波特率调整为 115200Bits/s:



② 开启 UART 中断，使能 TX 的 DMA 功能：

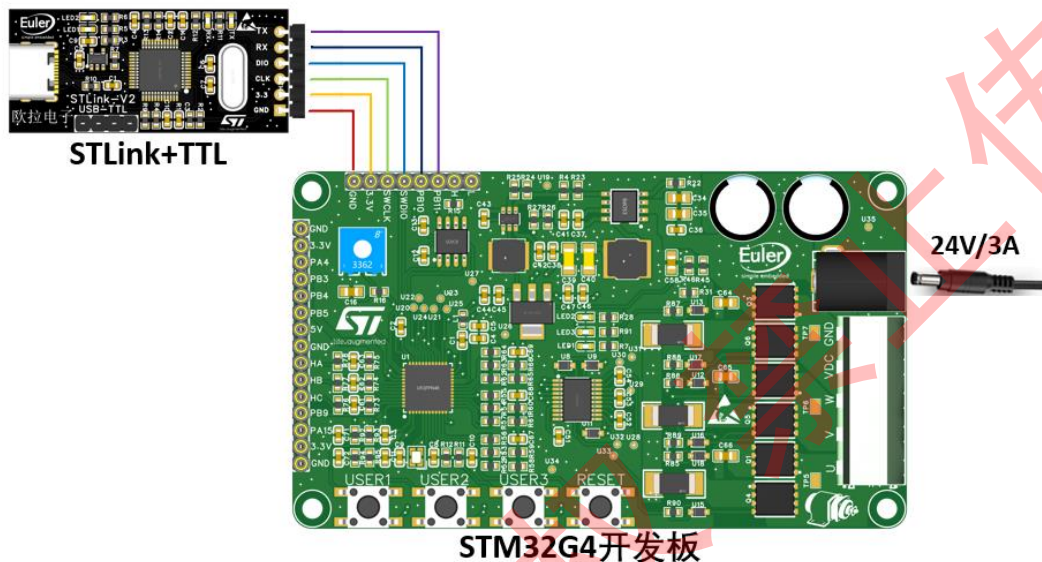


③ 生成代码，打开工程；



## 2. 实验现象

按以下图片接线，并将 USB 转 UART 模块连接至电脑：



### 2.1 UART 重定向 printf 打印数据

① 使用 UART 重定向，printf 发生数据至串口上位机；

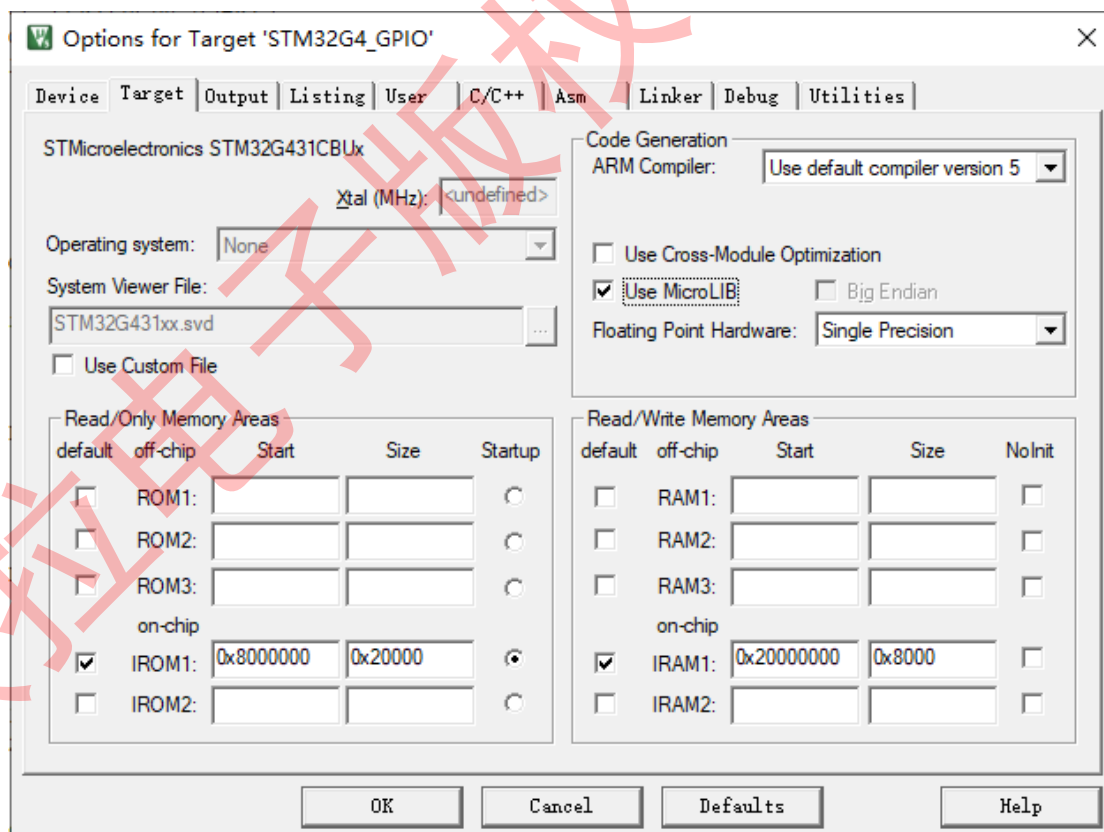
```
26  /* Private includes -----
27  /* USER CODE BEGIN Includes */
28  #include <stdio.h>
29  /* USER CODE END Includes */
...
182 int fputc(int ch, FILE *f)
183 {
184     while ((USART3->ISR & 0X40) == 0);
185     USART3->TDR = (uint8_t)ch;
186     return ch;
187 }
188 /* USER CODE END 4 */
189
```

```

159 /* USER CODE BEGIN 4 */
160 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
161 {
162     /* Prevent unused argument(s) compilation warning */
163     UNUSED(GPIO_Pin);
164     if(Button1_Pin == GPIO_Pin)
165     {
166         HAL_GPIO_TogglePin(LED1_GPIO_Port,LED1_Pin);
167         printf("LED1 Toggle/r/n");
168     }
169     if(Button2_Pin == GPIO_Pin)
170     {
171         HAL_GPIO_TogglePin(LED2_GPIO_Port,LED2_Pin);
172         printf("LED2 Toggle/r/n");
173     }
174     if(Button3_Pin == GPIO_Pin)
175     {
176         HAL_GPIO_WritePin(LED1_GPIO_Port,LED1_Pin,GPIO_PIN_RESET);
177         HAL_GPIO_WritePin(LED2_GPIO_Port,LED2_Pin,GPIO_PIN_RESET);
178         printf("LED1 and LED2 Open/r/n");
179     }
180     /* NOTE: This function should not be modified, when the callback is needed,
181        the HAL_GPIO_EXTI_Callback could be implemented in the user file
182     */
183 }

```

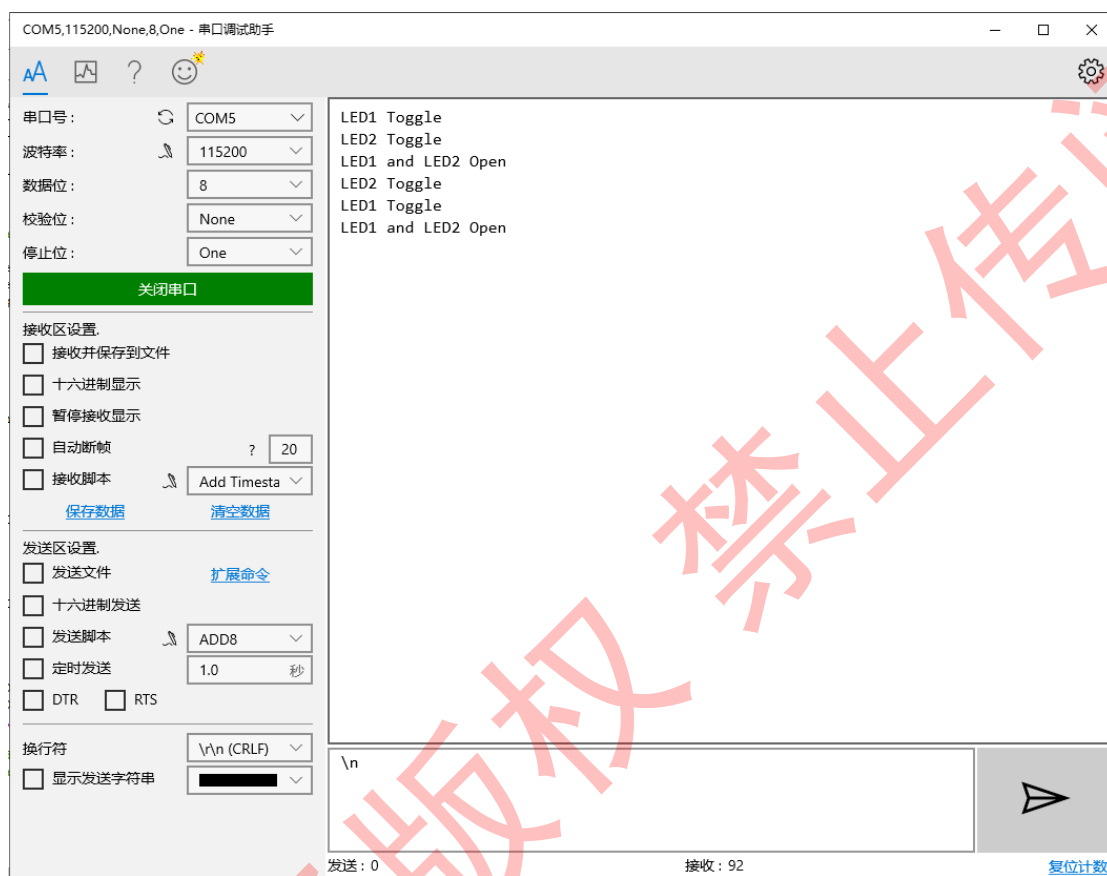
② 勾选 use MicroLIB 后编译，下载程序，打开上位机；



③ 打开上位机，连接串口；按下按键 1，LED1 翻转，上位机显示 LED1

Toggle；按下按键 2，LED2 翻转，上位机显示 LED2 Toggle；按下按键 3，

LED1 和 LED2 亮，上位机显示 LED1 and LED2 Open；



## 2.2 UART DMA 上传数据

① 使用 DMA 发送数据：

```
47 /* USER CODE BEGIN PV */
48 extern DMA_HandleTypeDef hdma_usart3_tx;
49 uint8_t DataB1[32] = "LED1 Toggle\r\n";
50 uint8_t DataB2[32] = "LED2 Toggle\r\n";
51 uint8_t DataB3[32] = "LED1 and LED2 Open\r\n";
52 /* USER CODE END PV */
```



```

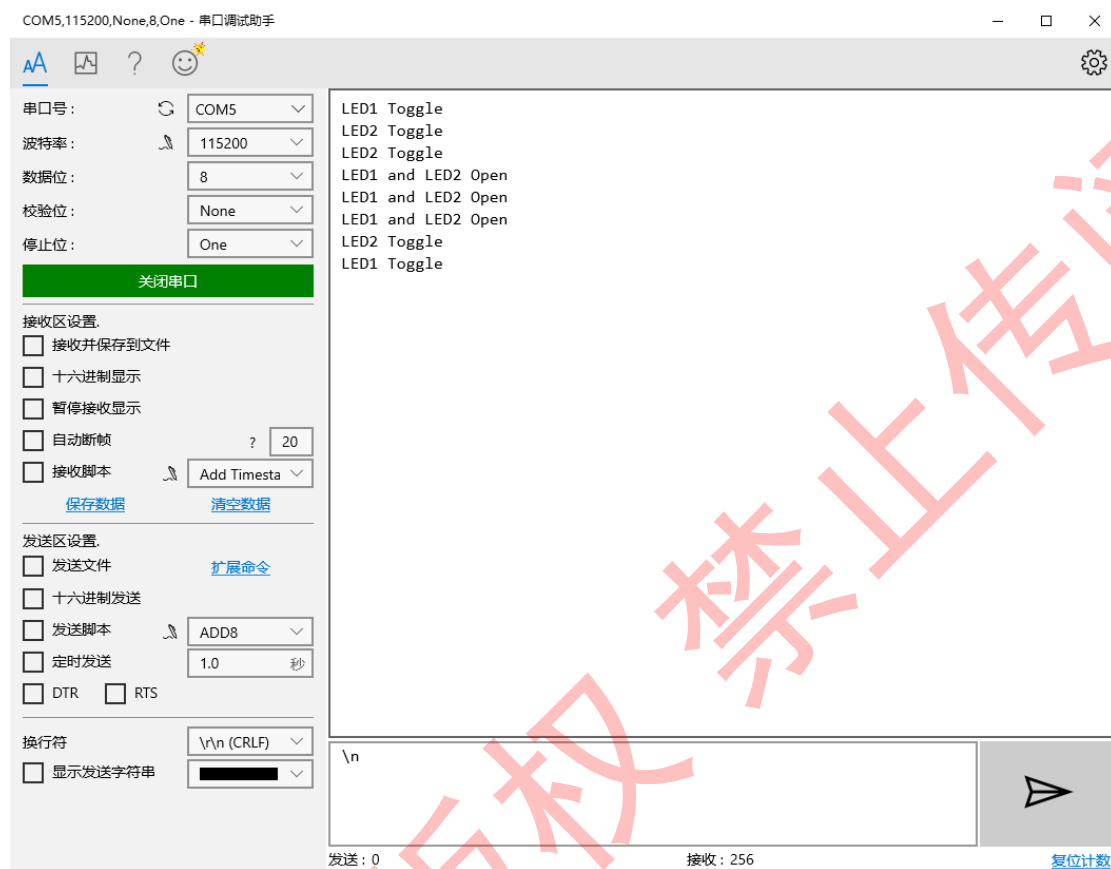
189 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
190 {
191     /* Prevent unused argument(s) compilation warning */
192     UNUSED(GPIO_Pin);
193     if(Button1_Pin == GPIO_Pin)
194     {
195         HAL_GPIO_TogglePin(LED1_GPIO_Port,LED1_Pin);
196         while(HAL_DMA_GetState(&hdma_usart3_tx) != HAL_DMA_STATE_READY);
197         HAL_UART_Transmit_DMA(&huart3,DataB1,sizeof(DataB1));
198     }
199     if(Button2_Pin == GPIO_Pin)
200     {
201         HAL_GPIO_TogglePin(LED2_GPIO_Port,LED2_Pin);
202         while(HAL_DMA_GetState(&hdma_usart3_tx) != HAL_DMA_STATE_READY);
203         HAL_UART_Transmit_DMA(&huart3,DataB2,sizeof(DataB1));
204     }
205     if(Button3_Pin == GPIO_Pin)
206     {
207         HAL_GPIO_WritePin(LED1_GPIO_Port,LED1_Pin,GPIO_PIN_RESET);
208         HAL_GPIO_WritePin(LED2_GPIO_Port,LED2_Pin,GPIO_PIN_RESET);
209         while(HAL_DMA_GetState(&hdma_usart3_tx) != HAL_DMA_STATE_READY);
210         HAL_UART_Transmit_DMA(&huart3,DataB3,sizeof(DataB1));
211     }
212     /* NOTE: This function should not be modified, when the callback is needed,
213        the HAL_GPIO_EXTI_Callback could be implemented in the user file
214     */
215 }

```

② 打开上位机，连接串口；按下按键 1，LED1 翻转，上位机显示 LED1

Toggle；按下按键 2，LED2 翻转，上位机显示 LED2 Toggle；按下按键 3，

LED1 和 LED2 亮，上位机显示 LED1 and LED2 Open；



## 2.3 UART 中断接收数据

① 使用 UART 中断接收上位机发送数据；

```

26  /* Private includes -----
27  /* USER CODE BEGIN Includes */
28  #include <stdio.h>
29  #include <string.h>
30  /* USER CODE END Includes */

54  #define RXBUFFERSIZE 256
55  char RxBuffer[RXBUFFERSIZE];
56  uint8_t aRxBuffer;
57  uint8_t Uart1_Rx_Cnt = 0;
58  /* USER CODE END PV */

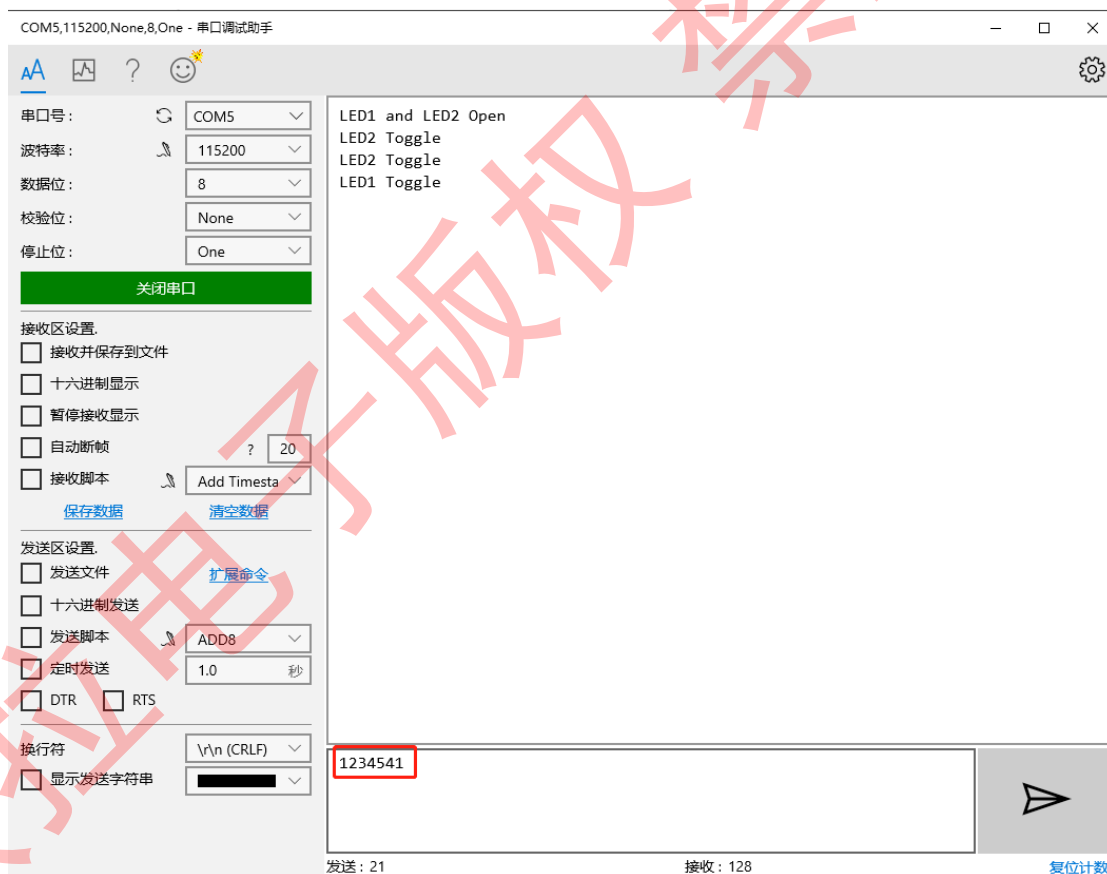
100  HAL_UART_Receive_IT(&huart3, (uint8_t *)&aRxBuffer, 1);
101  /* USER CODE BEGIN 2 */
102  HAL_UART_Receive_IT(&huart3, (uint8_t *)&aRxBuffer, 1);
103  /* USER CODE END 2 */

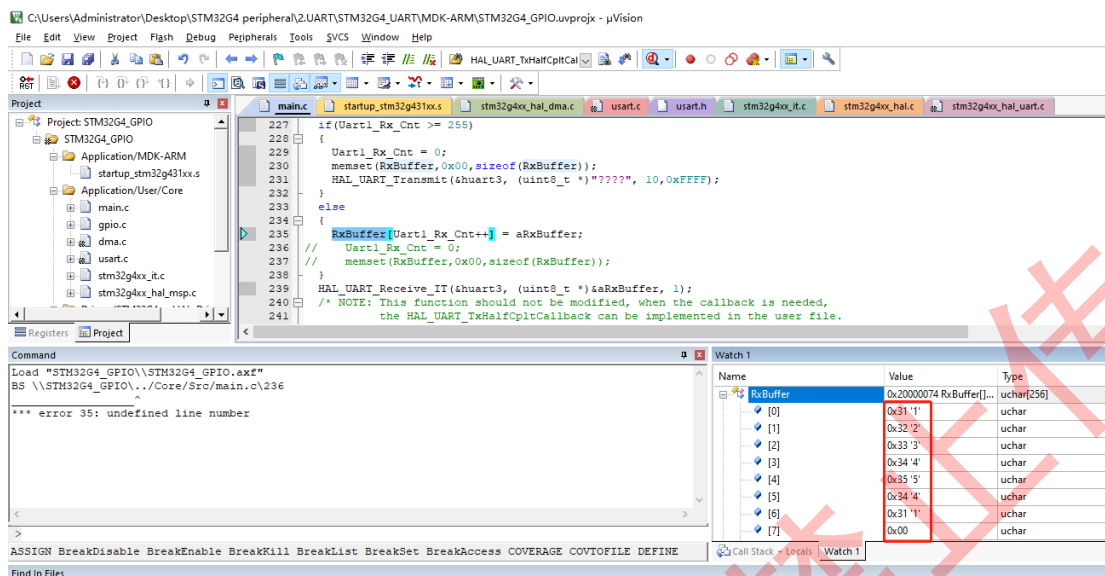
```

```

223 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
224 {
225     /* Prevent unused argument(s) compilation warning */
226     UNUSED(huart);
227     if(Uart1_Rx_Cnt >= 255)
228     {
229         Uart1_Rx_Cnt = 0;
230         memset(RxBuffer, 0x00, sizeof(RxBuffer));
231         HAL_UART_Transmit(&huart3, (uint8_t *) "????", 10, 0xFFFF);
232     }
233     else
234     {
235         RxBuffer[Uart1_Rx_Cnt++] = aRxBuffer;
236         // Uart1_Rx_Cnt = 0;
237         // memset(RxBuffer, 0x00, sizeof(RxBuffer));
238     }
239     HAL_UART_Receive_IT(&huart3, (uint8_t *) &aRxBuffer, 1);
240     /* NOTE: This function should not be modified, when the callback is needed,
241        the HAL_UART_TxHalfCpltCallback can be implemented in the user file.
242     */
243 }

```





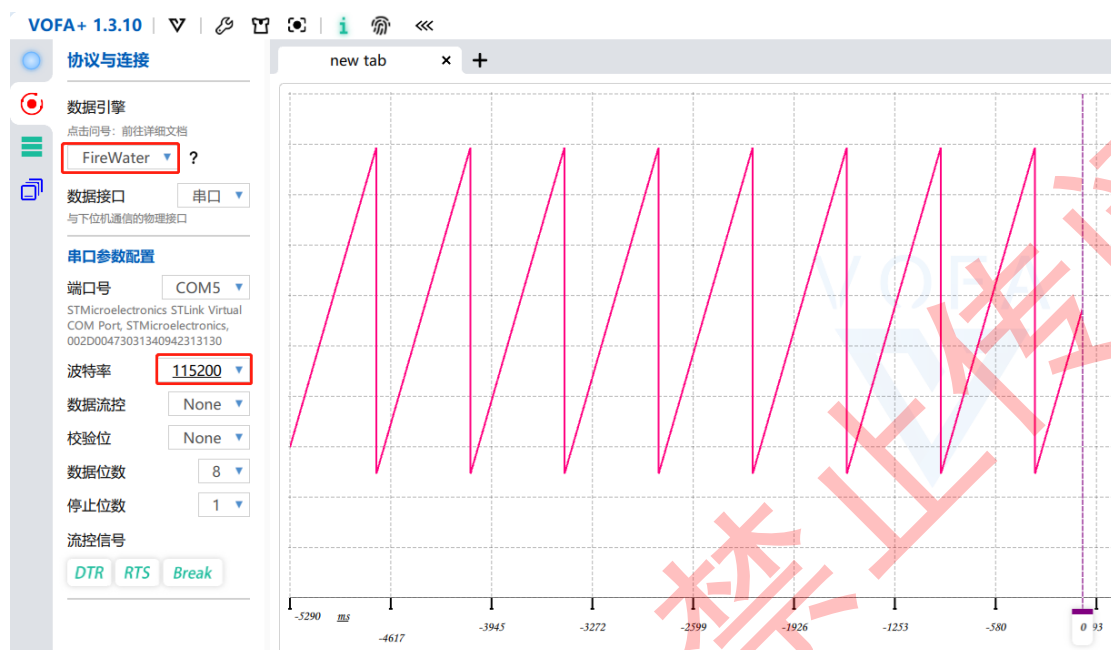
### 3. VOFA 上位机显示波形

可先参考 附录三: vofa+上位机使用教程

① 使用 VOFA 上位机显示波形, stm32 使用 printf, VOFA 使用 FireWater 协议, 115200 波特率, 点击连接按钮:

```
59 float temp[1];

110 while (1)
111 {
112     temp[0] += 0.01f;
113     if(temp[0] > 6.28f)
114     {
115         temp[0] = 0;
116     }
117     printf("%f\r\n", temp[0]);
118     // memcpy(tempData, (uint8_t *)&temp, sizeof(temp));
119     // HAL_UART_Transmit_DMA(&huart3, (uint8_t *)&tempData, 8);
120     HAL_Delay(1);
```



② 使用 VOFA 上位机显示波形，stm32 使用 UART DMA，VOFA 使用

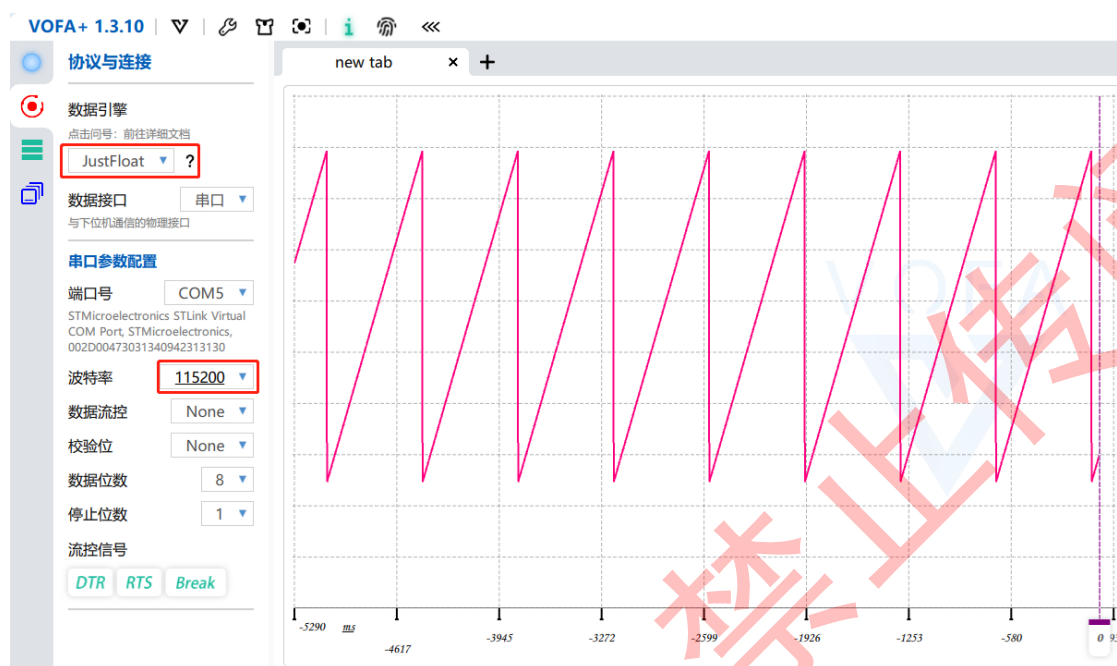
JustFloat 协议，115200 波特率，点击连接按钮：

```

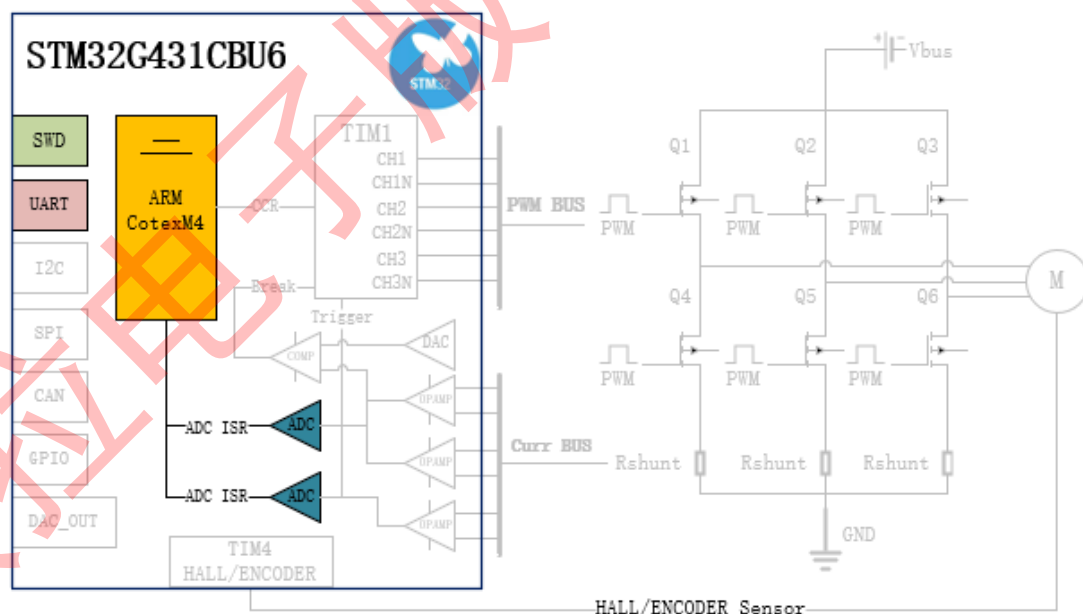
59 float temp[1];
60 static uint8_t tempData[8] = {0,0,0,0,0,0,0x80,0x7F};

110 while (1)
111 {
112     temp[0] += 0.01f;
113     if(temp[0] > 6.28f)
114     {
115         temp[0] = 0;
116     }
117     // printf("%f\r\n",temp[0]);
118     memcpy(tempData, (uint8_t *)&temp, sizeof(temp));
119     HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 8);
120     HAL_Delay(1);

```



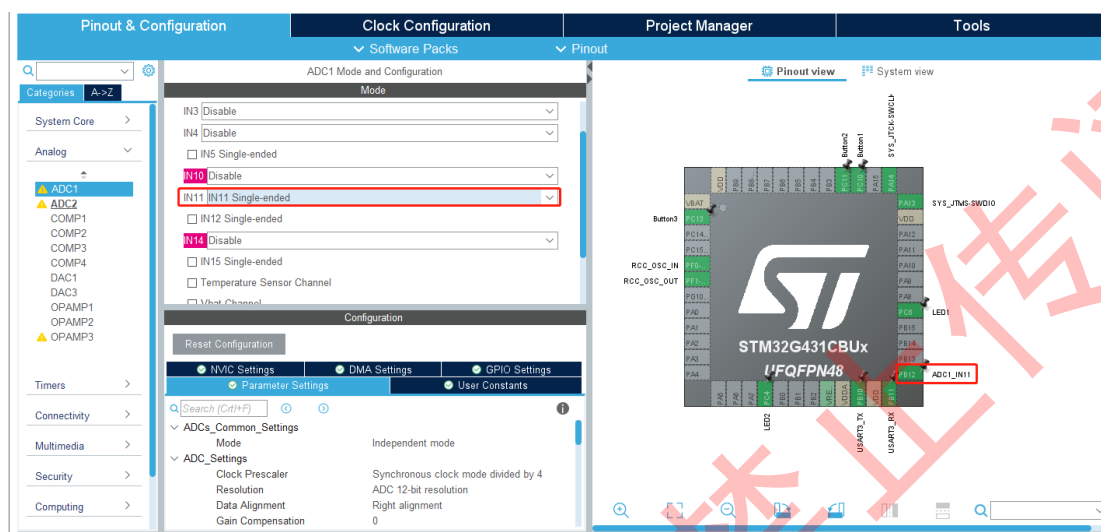
## 第七章：ADC 采样电位器电压



### 1. STM32CUBEMX 配置及 Keil 代码

- ① 在上一章的基础上增加 ADC 功能，使能 ADC IN11 通道，对应 PB12，对应

板载电位器接口；



② 配置以下参数，使能 ADC 中断；



③ 生成代码，并打开工程；



## ④ ADC1 自校验:

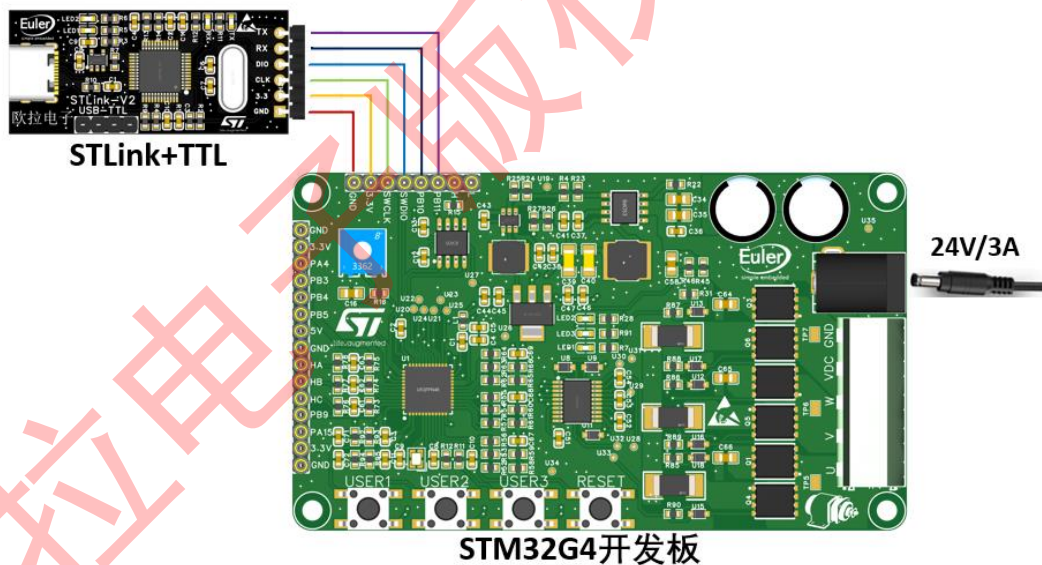
```
108 | HAL_ADCEx_Calibration_Start( &hadc1, ADC_SINGLE_ENDED);
```

## ⑤ 开启 ADC 采样，并将采样值通过 UART 上传到 VOFA 上位机:

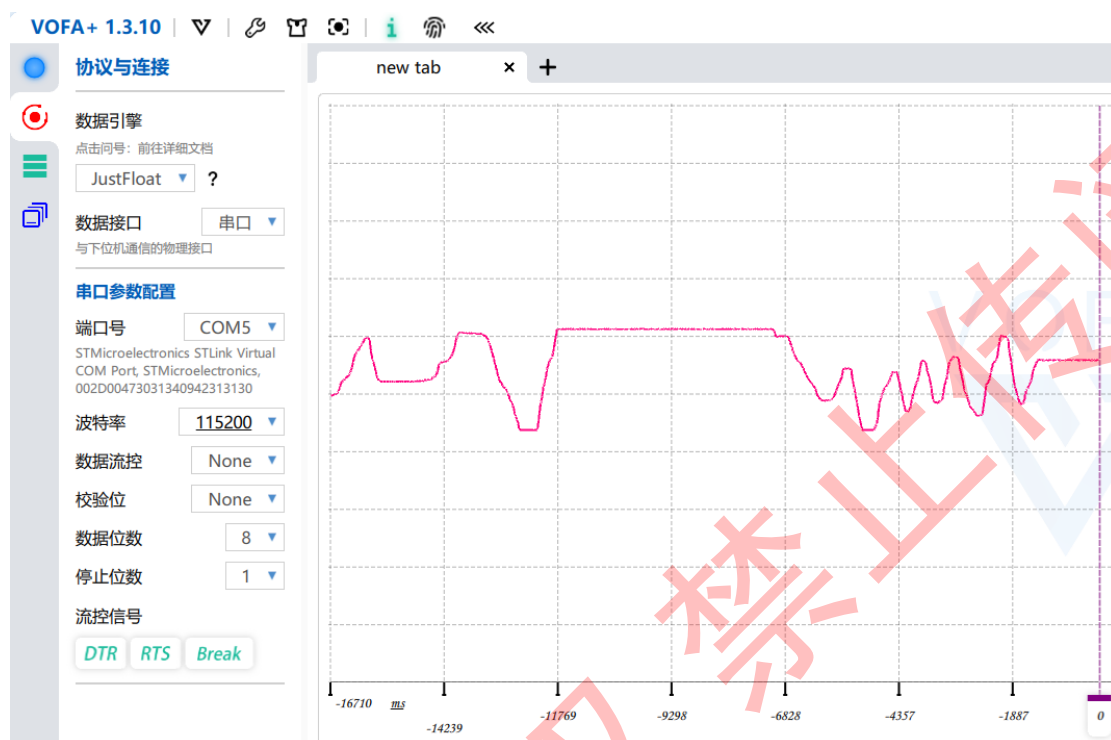
```
113 | while (1)
114 | {
115 |     HAL_ADC_Start(&hadc1);
116 |     temp[0] = HAL_ADC_GetValue(&hadc1);
117 |     memcpy(tempData, (uint8_t *)&temp, sizeof(temp));
118 |     HAL_UART_Transmit_DMA(&huart3, (uint8_t *)&tempData, 8);
119 |     HAL_Delay(1);
120 |     /* USER CODE END WHILE */
```

## 2. 实验现象

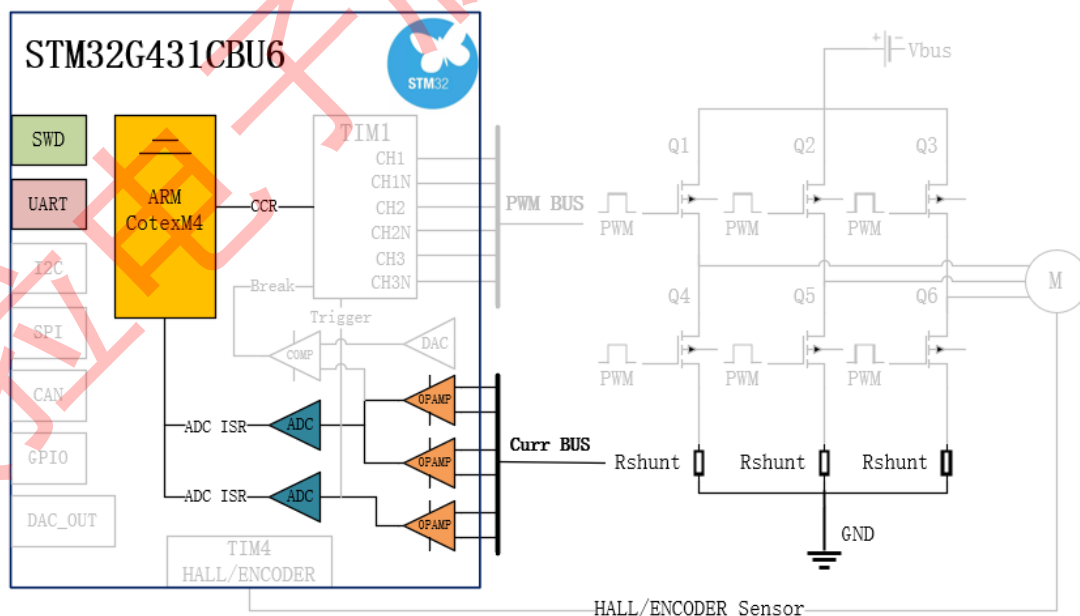
按以下图片接线，并将 USB 转 UART 模块连接至电脑:



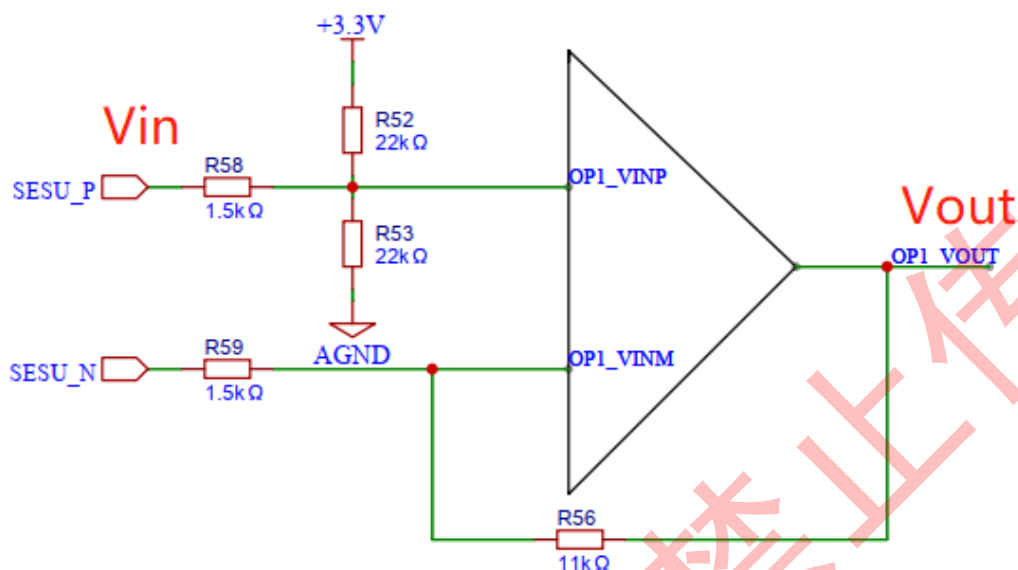
## ① 旋转开发板电位器，VOFA 上位机的值随机改变;



## 第八章：MCU 内部 OPAMP-电流采样



增益具体计算方法请参考附录一：带偏置的差分运放参数设计



R58,R59 作用是为了输入的正向端和负向端输入阻抗匹配；

R52, R53 为分压电阻，作用是将输入正向端电压正向偏置 1.65V；

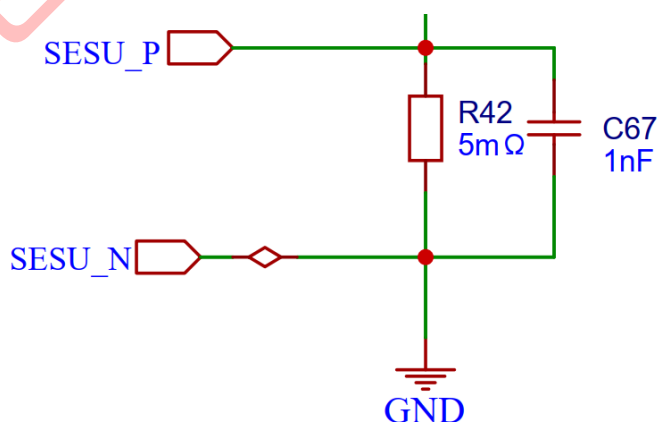
R56 与 R59 形成负反馈增益，运放整体增益计算如下：

虚断：OP1\_VINP 输入电压为  $V_{in}$ ；

虚短：

$$\frac{V_{in}}{R59} = \frac{V_{out}}{R59 + R56}$$

$$Gain = \frac{V_{out}}{V_{in}} = 1 + \frac{R56}{R59}$$



电流计算公式：

$$\frac{V_{adc}}{V_{in}} = \frac{R56}{R59}$$

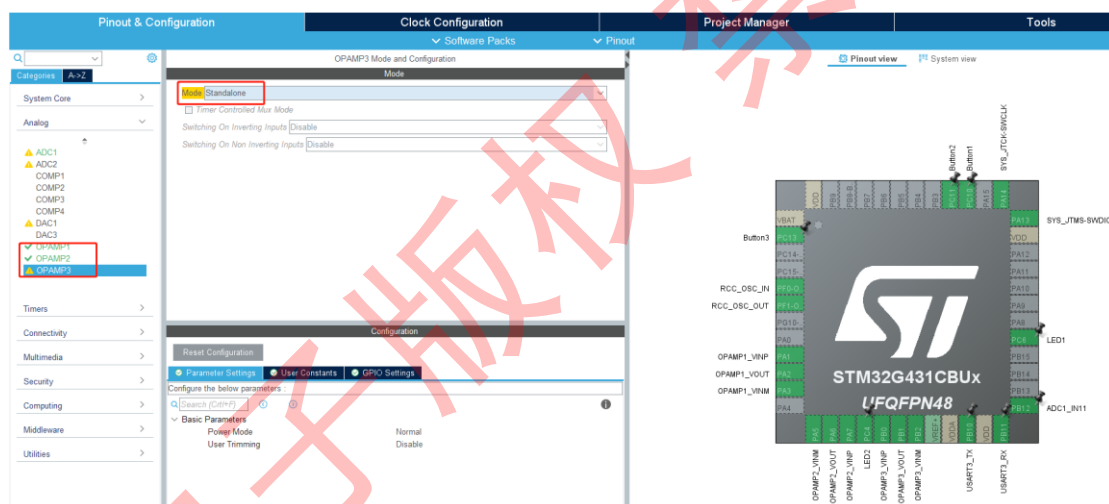
$$V_{adc} = \frac{ADC\_Value}{4096} * 3.3$$

$$V_{in} = I_{in} * R_{42}$$

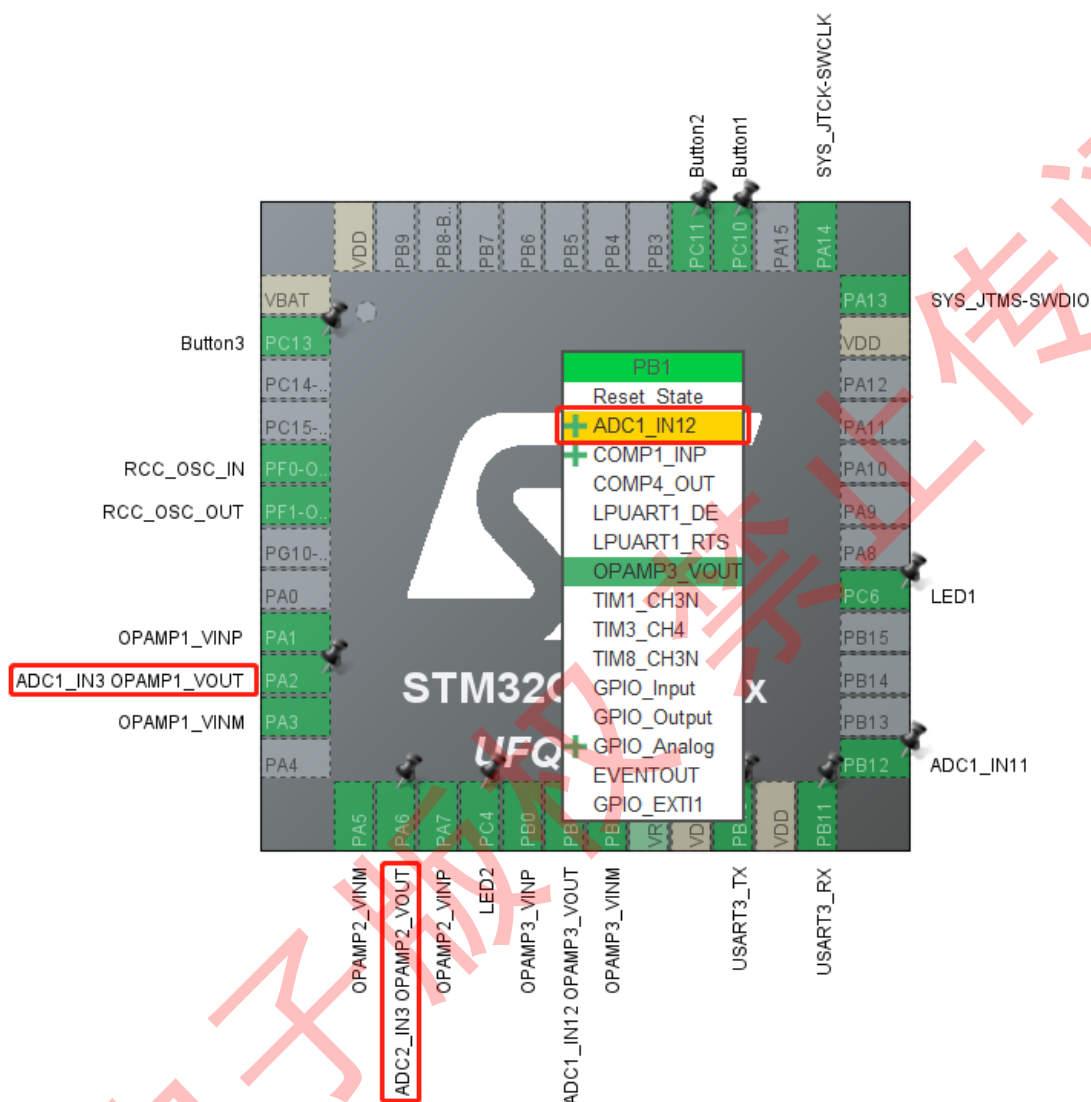
$$I_{in} = \frac{ADC\_Value * 3.3}{4096 * R_{42} * \left(\frac{R_{56}}{R_{59}}\right)}$$

## 1. STM32CUBEMX 配置及 Keil 代码

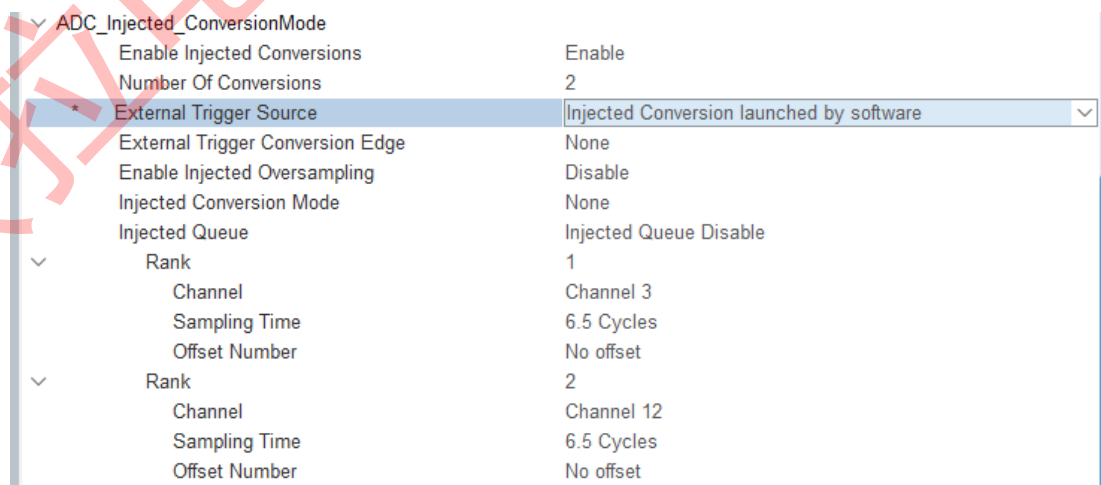
- ① 在上一章的工程基础上我们增加运放的外设，使能 OPAMP1、2、3 为 standalone 模式；



- ② 增加 OPAMP\_OUT Pin 的 ADC 功能；



③ 使能三个 ADC 通道，并配置为注入通道；



|                                  |                                          |
|----------------------------------|------------------------------------------|
| ADC_Injected_ConversionMode      |                                          |
| Enable Injected Conversions      | Enable                                   |
| Number Of Conversions            | 1                                        |
| * External Trigger Source        | Injected Conversion launched by software |
| External Trigger Conversion Edge | None                                     |
| Enable Injected Oversampling     | Disable                                  |
| Injected Conversion Mode         | None                                     |
| Injected Queue                   | Injected Queue Disable                   |
| Rank                             | 1                                        |
| Channel                          | Channel 3                                |
| Sampling Time                    | 6.5 Cycles                               |
| Offset Number                    | No offset                                |

- ④ 生成代码，使能 OPAMP，开启 ADC 自校验，开启规则组和注入组转换，并在 ADC 注入转换完成中断中读取 ADC 值并转换为三相电流值；

```

112 HAL_OPAMP_Start(&hopamp1);
113 HAL_OPAMP_Start(&hopamp2);
114 HAL_OPAMP_Start(&hopamp3);
115 HAL_UART_Receive_IT(&huart3, (uint8_t *)&aRxBuffer, 1);
116 HAL_ADCEX_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);
117 HAL_ADCEX_Calibration_Start(&hadc2, ADC_SINGLE_ENDED);

122 while (1)
123 {
124   HAL_ADC_Start(&hadc1);
125   HAL_ADCEX_InjectedStart_IT(&hadc1);
126   HAL_ADCEX_InjectedStart_IT(&hadc2);
127   temp[0] = HAL_ADC_GetValue(&hadc1);
128   temp[0] = temp[0]*3.3f/4096;
129   memcpy(tempData, (uint8_t *)&temp, sizeof(temp));
130   HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 5*4);
131   HAL_Delay(1);
132   /* USER CODE END WHILE */

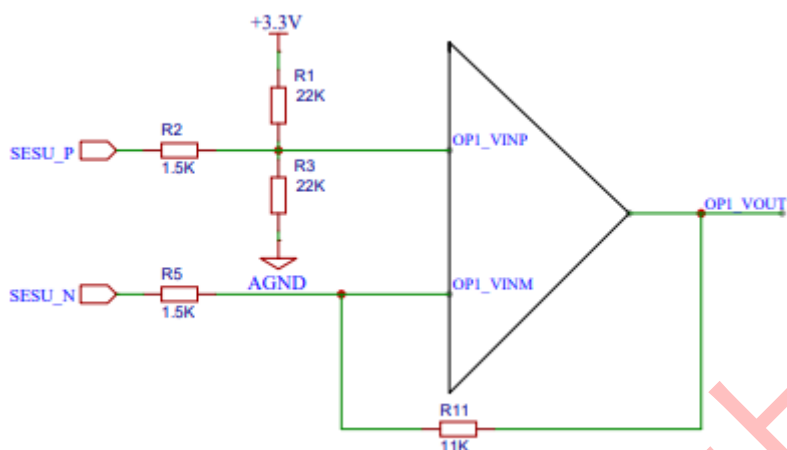
```

- ⑤ 结合原理图，OPAMP 的运放增益计算值：

负反馈增益系数： $7.33333 = 11k/1.5k$

总体增益系数： $0.02197265625 = 3.3/4096/7.33333/0.005$

增益具体计算方法请参考附录一：带偏置的差分运放参数设计

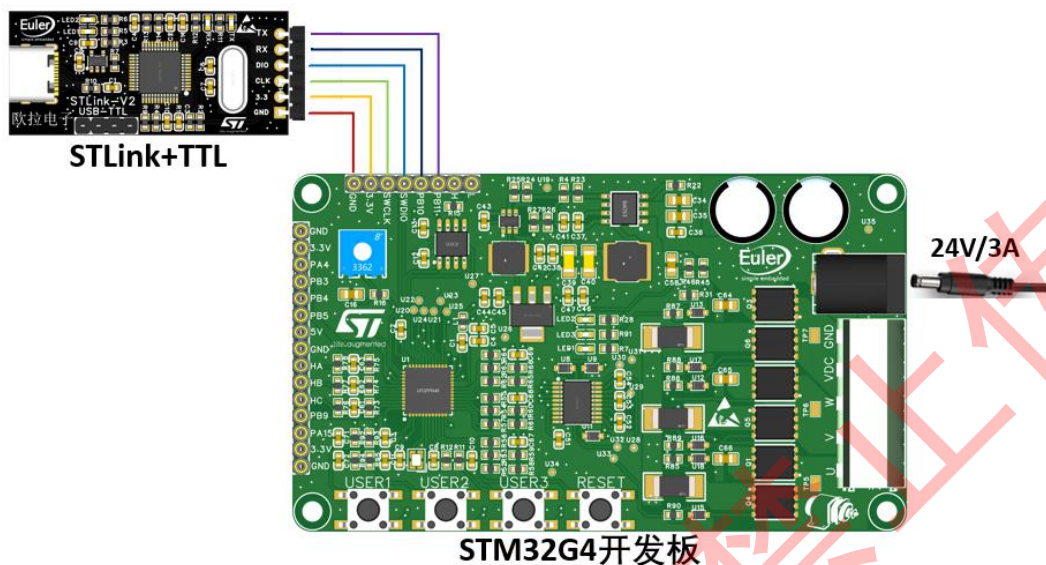


```
276 void HAL_ADCEx_InjectedConvCpltCallback(ADC_HandleTypeDef *hadc)
277 {
278     /* Prevent unused argument(s) compilation warning */
279     UNUSED(hadc);
280     if(hadc == &hadc1)
281     {
282         temp[1] = hadc1.Instance->JDR1;
283         temp[1] = (temp[1] - 0x7ef)*0.02197f;
284         temp[2] = hadc1.Instance->JDR2;
285         temp[2] = (temp[2] - 0x7f5)*0.02197f;
286     }
287     if(hadc == &hadc2)
288     {
289         temp[3] = hadc2.Instance->JDR1;
290         temp[3] = (temp[3] - 0x7e8)*0.02197f;
291     }
292     /* NOTE : This function should not be modified. When the callback is needed,
293             function HAL_ADCEx_InjectedConvCpltCallback must be implemented in the user file.
294     */
295 }
```

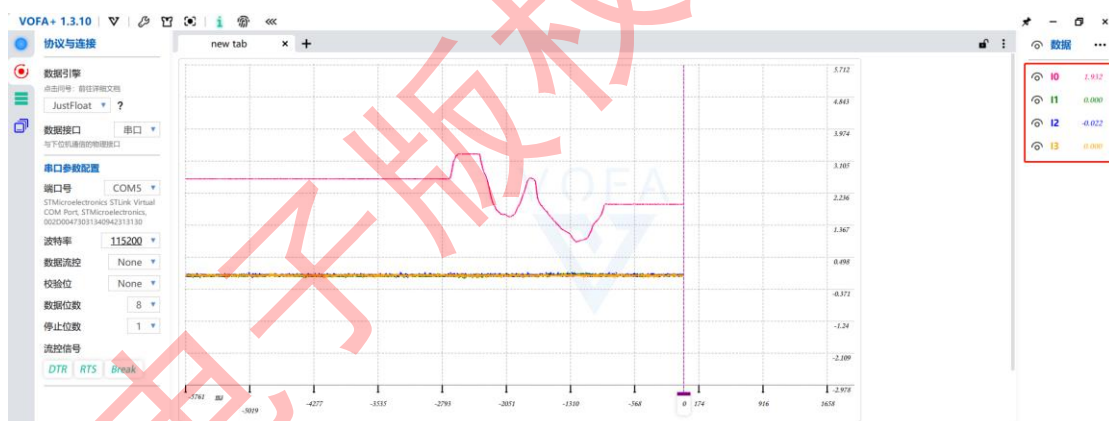
## 2. 实验现象

按以下图片接线，并讲 USB 转 UART 模块连接至电脑：



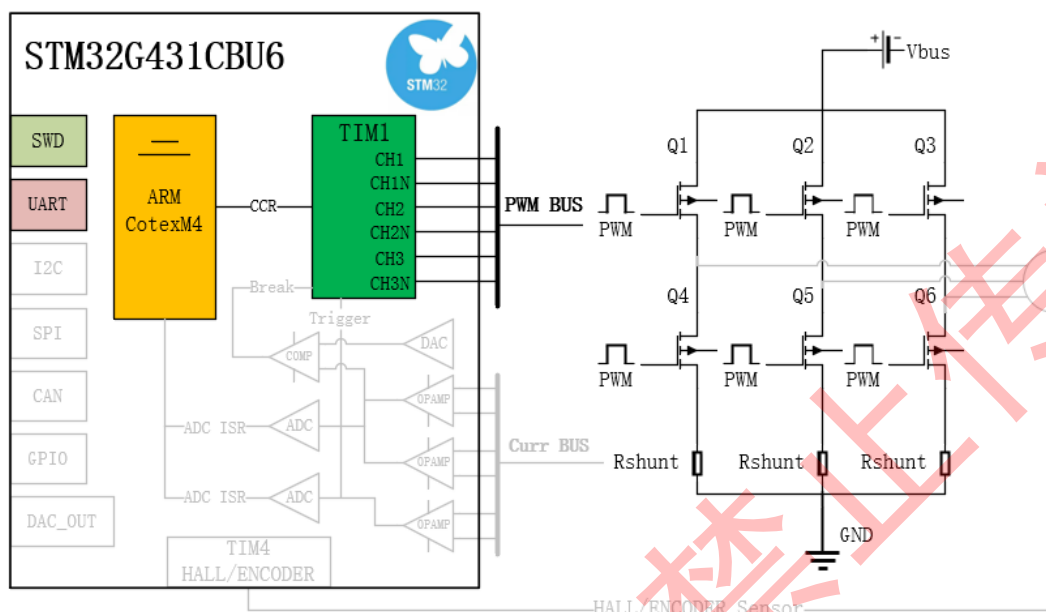


① 使用 VOFA 上位机显示电位器及三相电流值：



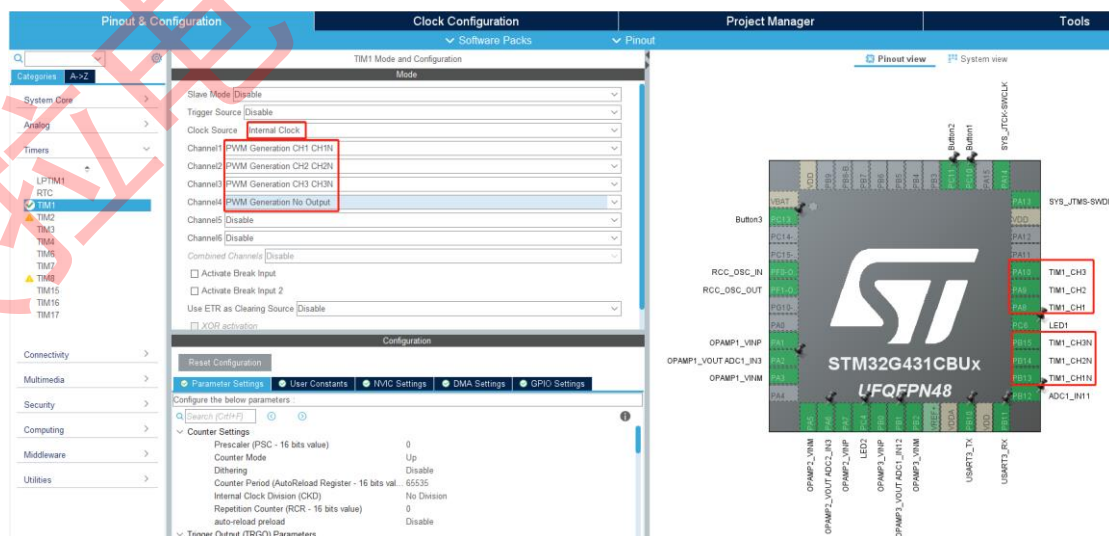
## 第九章：TIM1 生成互补 PWM

本章内容为大家介绍 STM32G4 的高级定时器模块，使用 TIM1 发出 3 对互补的 PWM 波。

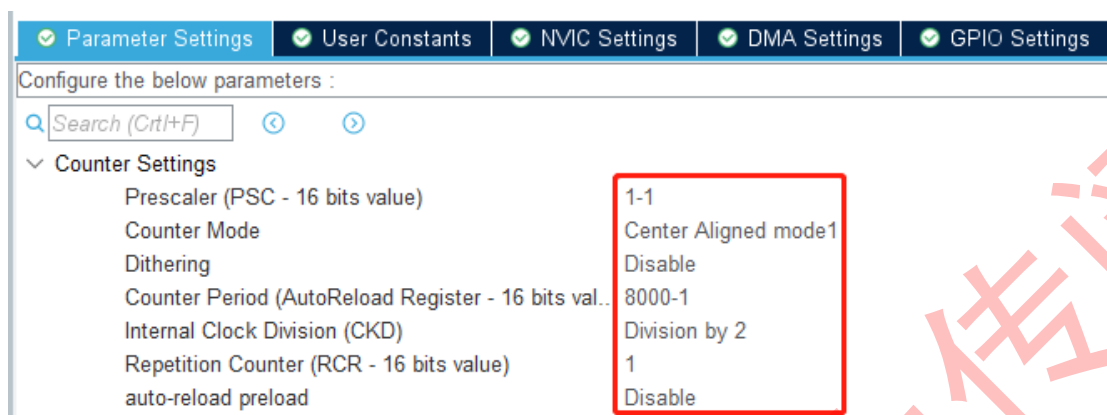


## 1. STM32CUBEMX 配置及 Keil 代码

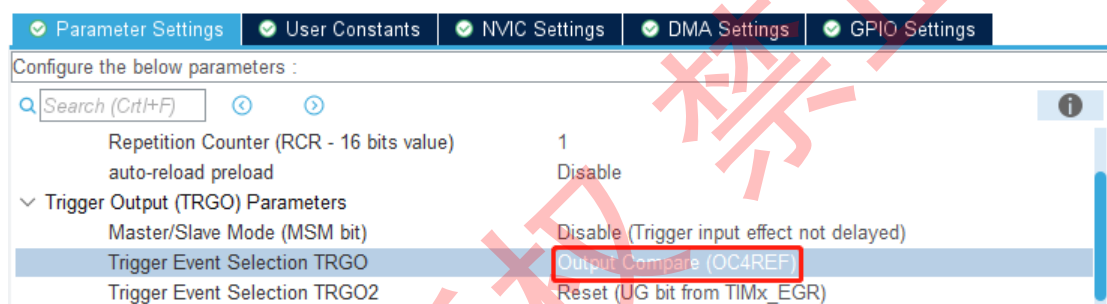
- ① 在上一章的工程基础上我们增加 TIM1 的外设，生成 3 对互补的 PWM 信号；选择内部时钟源，根据原理图配置输出的 PWM IO 口，选择 channel 1, 2, 3 为 PWM 互补输出，channel 配置为 PWM 模式但不输出，用于后续 ADC 触发；



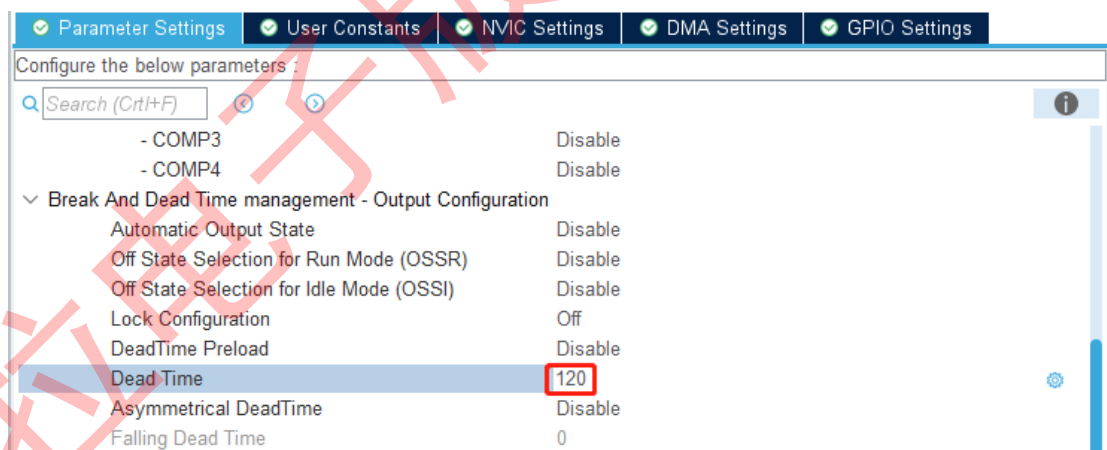
- ② 配置时基单元;



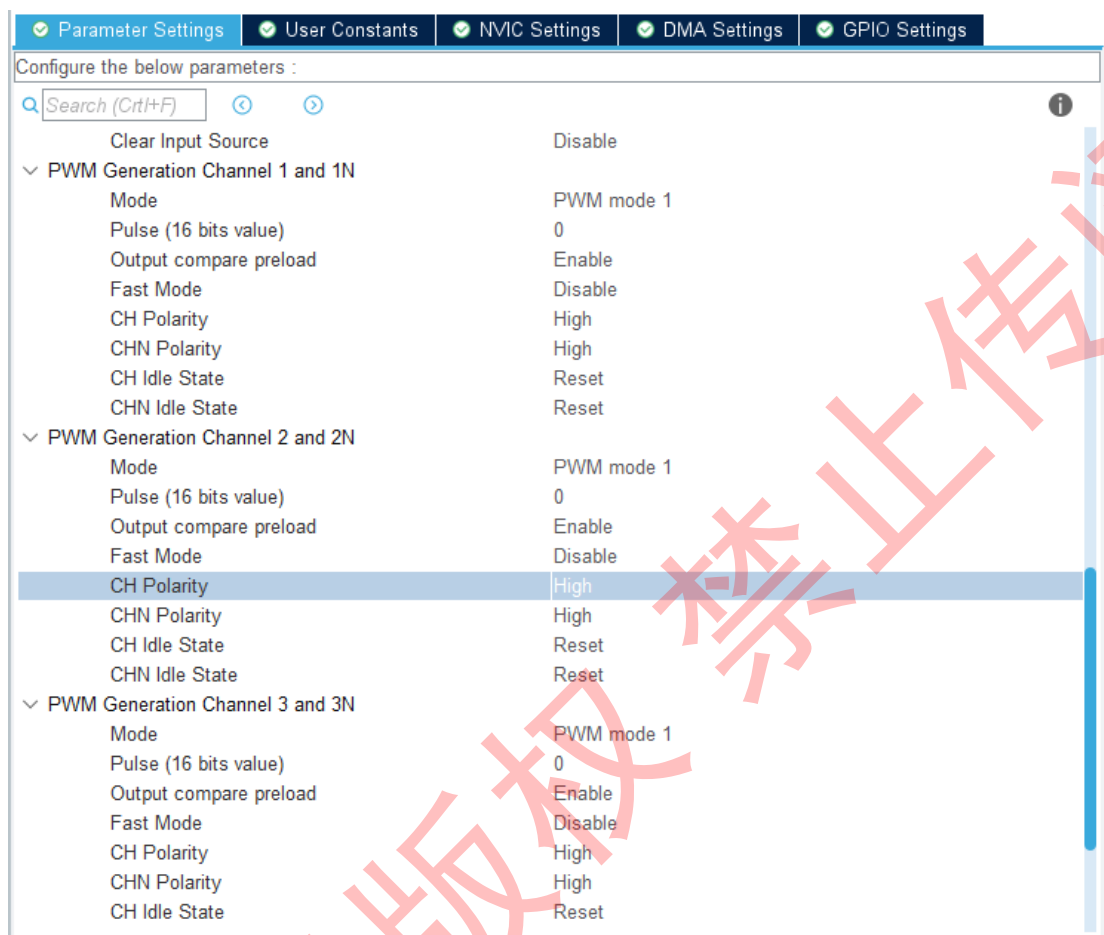
③ 配置 TRGO 触发源，后续用于 ADC 电流采样触发；



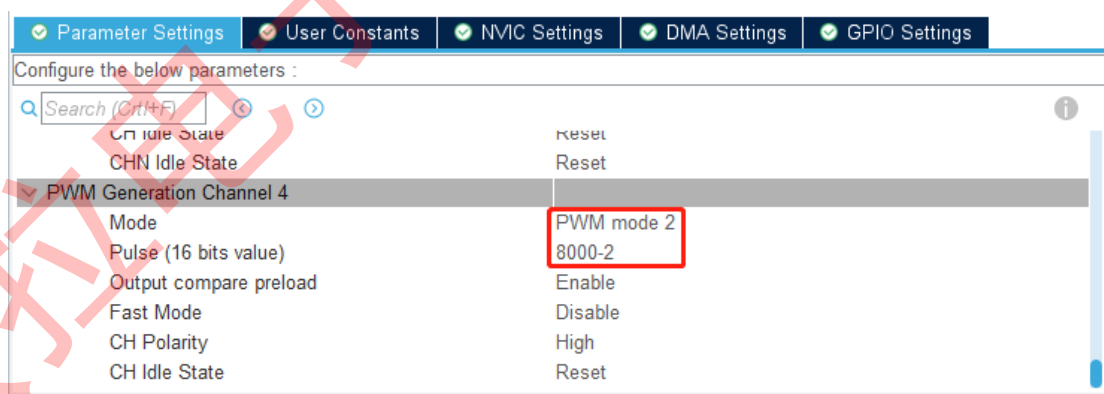
④ 配置死区时间 120；



⑤ 配置 channel1, 2, 3 为 PWM 模式 1，其他默认；



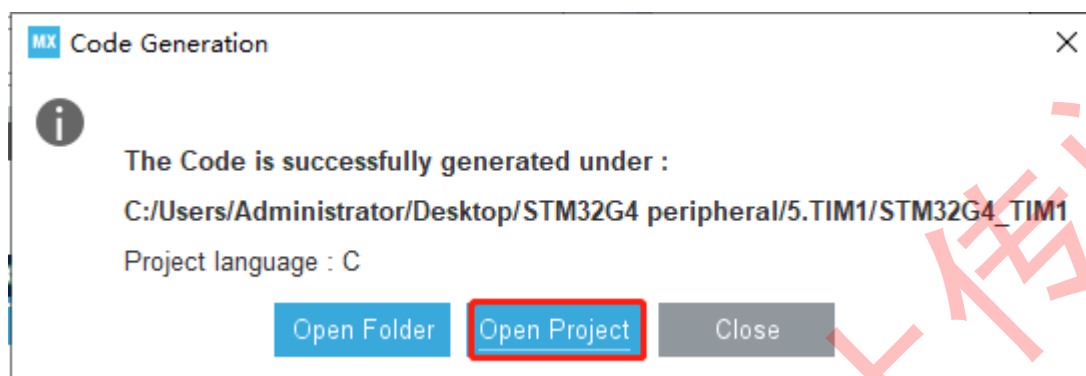
⑥ 配置 channel 4 为 PWM 模式 2，周期值为 8000-2；



⑦ 配置 TIM1 中断；

| Parameter Settings                                                 | User Constants                      | NVIC Settings       | DMA Settings | GPIO Settings |
|--------------------------------------------------------------------|-------------------------------------|---------------------|--------------|---------------|
| NVIC Interrupt Table                                               |                                     |                     |              |               |
|                                                                    | Enabled                             | Preemption Priority | Sub Priority |               |
| TIM1 break interrupt and TIM15 global interrupt                    | <input checked="" type="checkbox"/> | 0                   | 0            |               |
| TIM1 update interrupt and TIM16 global interrupt                   | <input checked="" type="checkbox"/> | 0                   | 0            |               |
| TIM1 trigger and commutation interrupts and TIM17 global interrupt | <input type="checkbox"/>            | 0                   | 0            |               |
| TIM1 capture compare interrupt                                     | <input type="checkbox"/>            | 0                   | 0            |               |

- ⑧ 生成代码，打开 Keil 工程；



- ⑨ 在 TIM1 初始化完成后，从新设置分频系数，ARR 及 CCR1,2,3 的值，启动 TIM1 计数及 PWM 输出；

```

120 TIM1->PSC = 30000;
121 TIM1->ARR = 10000;
122 TIM1->CCR1 = 2000;
123 TIM1->CCR2 = 5000;
124 TIM1->CCR3 = 8000;
125 HAL_TIM_Base_Start( &htim1);
126 HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_1);
127 HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_2);
128 HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_3);
129 HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_1);
130 HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_2);
131 HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_3);
132 /* USER CODE END 2 */

```

- ⑩ 在 while 循环中，注释掉之前 ADC 代码；读取 PWM channel1(PA8), channel2(PA9), channel3(PA10) IO 状态，并通过 VOFA 上位机打印波形；

```

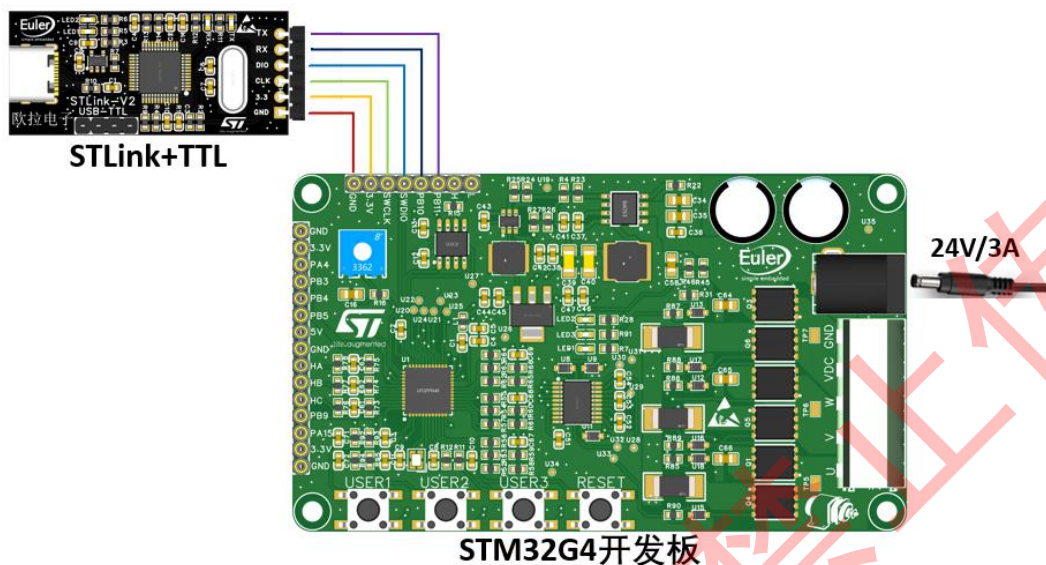
136 while (1)
137 {
138     // HAL_ADC_Start(&hadc1);
139     // HAL_ADCEX_InjectedStart_IT(&hadc1);
140     // HAL_ADCEX_InjectedStart_IT(&hadc2);
141     // temp[0] = HAL_ADC_GetValue(&hadc1);
142     // temp[0] = temp[0]*3.3f/4096;
143     if((GPIOA->IDR & GPIO_PIN_8) != 0)
144     {
145         temp[0] = 1.0f;
146     }
147     else
148     {
149         temp[0] = 0.0f;
150     }
151     if((GPIOA->IDR & GPIO_PIN_9) != 0)
152     {
153         temp[1] = 3.0f;
154     }
155     else
156     {
157         temp[1] = 2.0f;
158     }
159     if((GPIOA->IDR & GPIO_PIN_10) != 0)
160     {
161         temp[2] = 5.0f;
162     }
163     else
164     {
165         temp[2] = 4.0f;
166     }
167     memcpy(tempData, (uint8_t *)&temp, sizeof(temp));
168     HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 5*4);
169     // HAL_Delay(1);

```

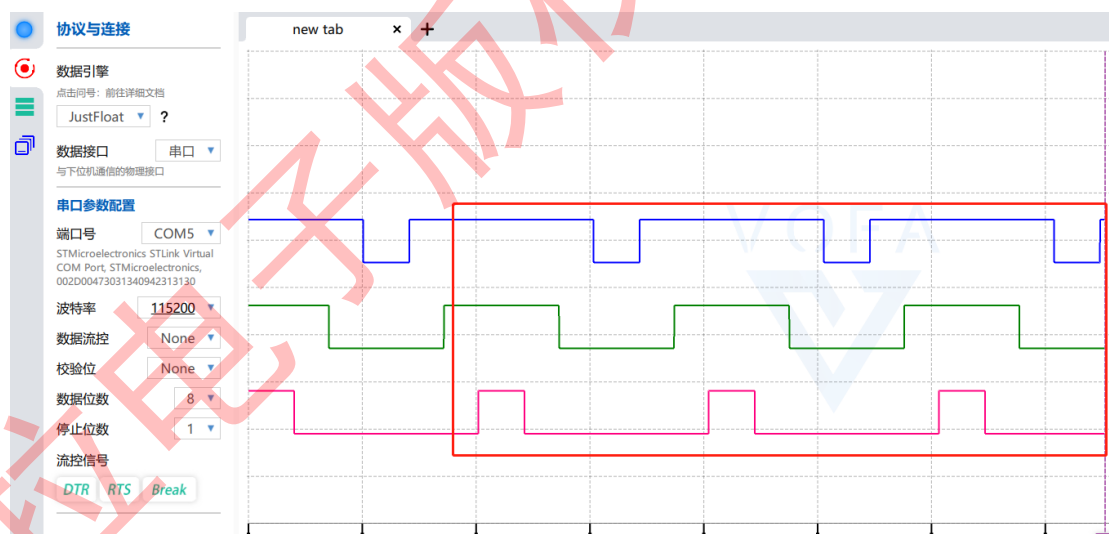
## 2. 实验现象

按以下图片接线，并讲 USB 转 UART 模块连接至电脑：



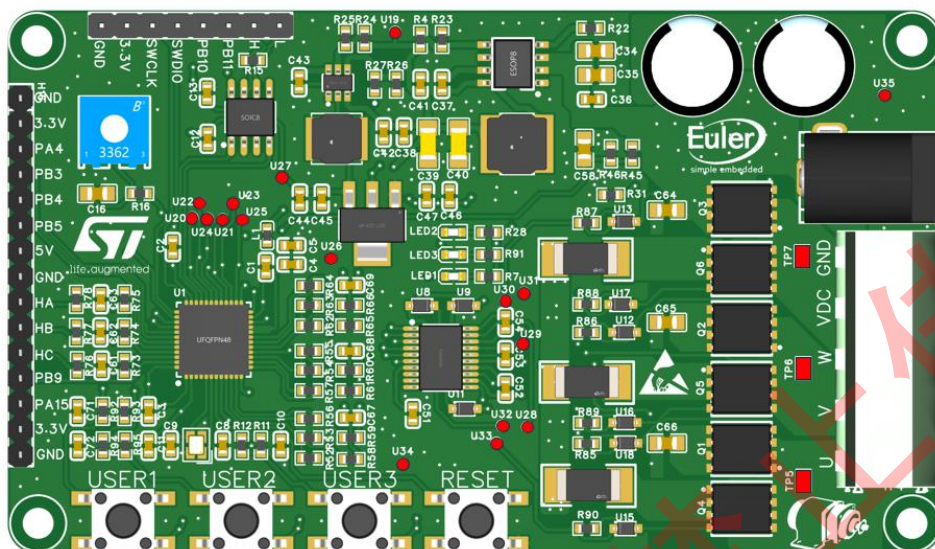


VOFA 上位机显示三个通道的 PWM 波形，波形与配置需要达到的效果完全移植；



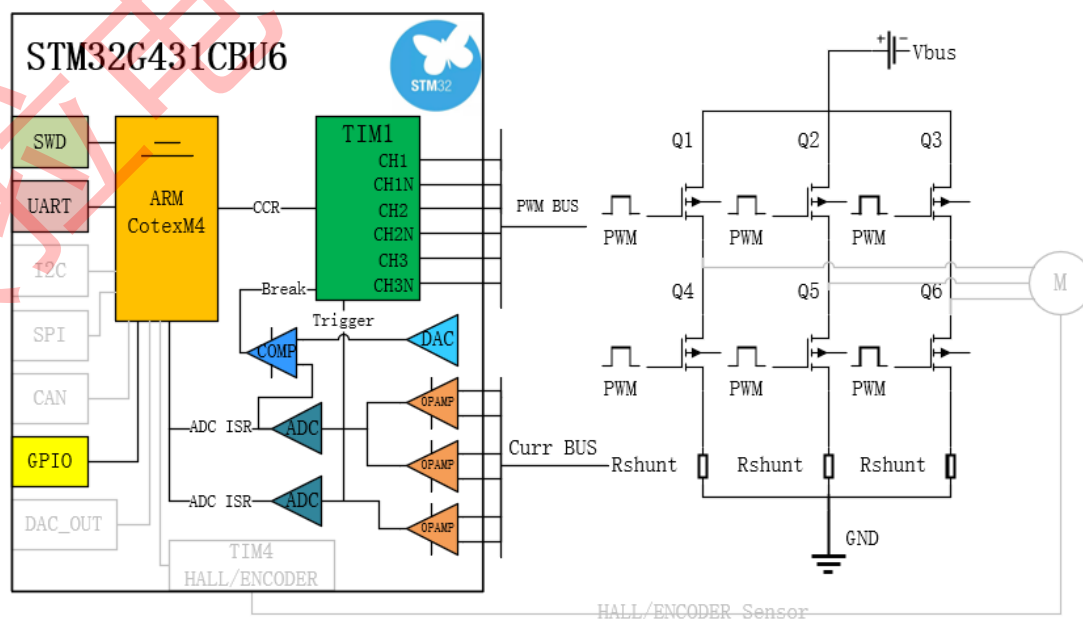
通过板子测试点，使用示波器或逻辑分析仪也可查看相应波形；



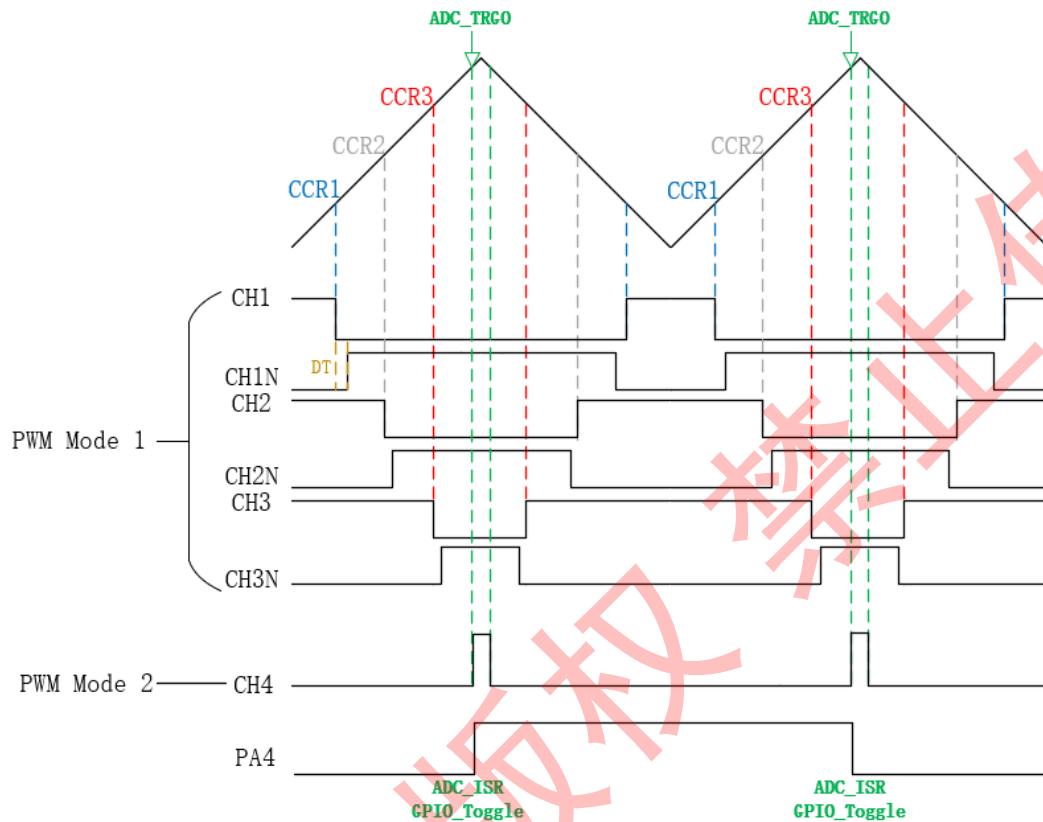


## 第十章：TIM1 ADC COMP DAC 级联使用

电机控制环路主要涉及的外设功能包含高级定时器 TIM1 的发波；OPAMP 及 ADC 准确的采样三相电流；并在三相电流过流时及时封波；本章将会使用 STM32G4 内部 TIM1 ADC COMP DAC 级联使用，完成以上内容。



## TIM1 发波时序及电流采样触发：



根据 MCU 内部比较器的资源分配，我们 INP 输入为 PB1，对应比较器三的  
输出脚，同时也为 ADC1\_IN12 通道；INM 输入选择 DAC3\_CH1，通过设置  
DAC 的模拟值与 ADC1 通道 12 的采样值进行比较，合理设置 DAC\_CH1 模拟值  
就能实现过流保护。

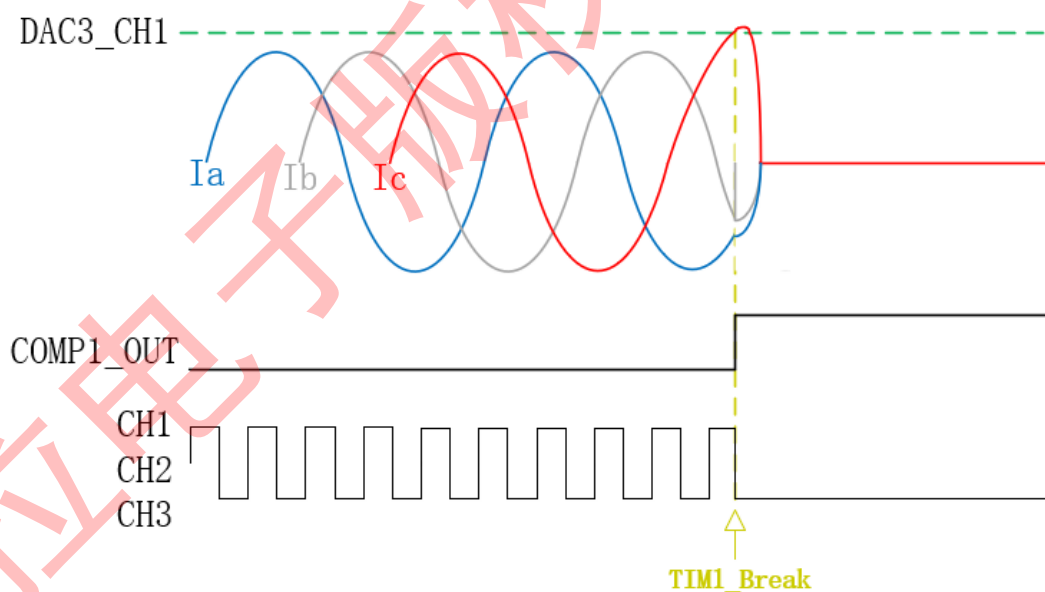
Table 196. COMPx non-inverting input assignment

| INPSEL | COMP1_<br>INP | COMP2_<br>INP | COMP3_<br>INP | COMP4_<br>INP | COMP5_<br>INP | COMP6_<br>INP | COMP7_<br>INP |
|--------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 0      | PA1           | PA7           | PA0           | PB0           | PB13          | PB11          | PB14          |
| 1      | PB1           | PA3           | PC1           | PE7           | PD12          | PD11          | PD14          |

Table 197. COMPx inverting input assignment

| INMSEL<br>[2:0] | COMP1_<br>INM           | COMP2_<br>INM | COMP3_<br>INM | COMP4_<br>INM | COMP5_<br>INM | COMP6_<br>INM | COMP7_<br>INM |
|-----------------|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 000             | 1/4 V <sub>REFINT</sub> |               |               |               |               |               |               |
| 001             | 1/2 V <sub>REFINT</sub> |               |               |               |               |               |               |
| 010             | 3/4 V <sub>REFINT</sub> |               |               |               |               |               |               |
| 011             | V <sub>REFINT</sub>     |               |               |               |               |               |               |
| 100             | DAC3_CH1                | DAC3_CH2      | DAC3_CH1      | DAC3_CH2      | DAC4_CH1      | DAC4_CH2      | DAC4_CH1      |
| 101             | DAC1_CH1                | DAC1_CH2      | DAC1_CH1      | DAC1_CH1      | DAC1_CH2      | DAC2_CH1      | DAC2_CH1      |
| 110             | PA4                     | PA5           | PF1           | PE8           | PB10          | PD10          | PD15          |
| 111             | PA0                     | PA2           | PC0           | PB2           | PD13          | PB15          | PB12          |

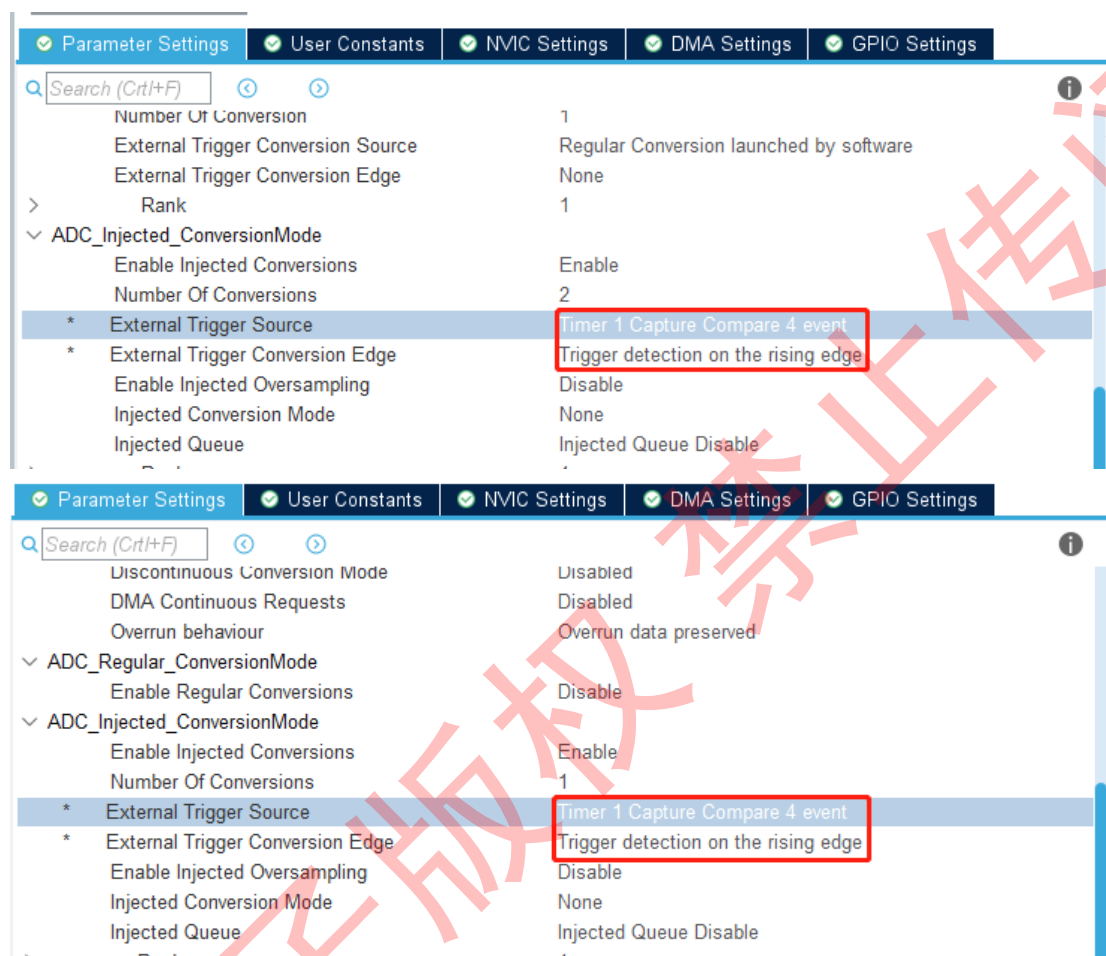
三相电流过流封锁时序：



## 1. STM32CUBEMX 配置及 Keil 代码

- ① 在上一章的工程基础上我们增加 TIM1 channel 4 的触发源，作为 ADC 电流采样注入组的触发源；分别修改 ADC1 和 ADC2 的注入组转换触发源为

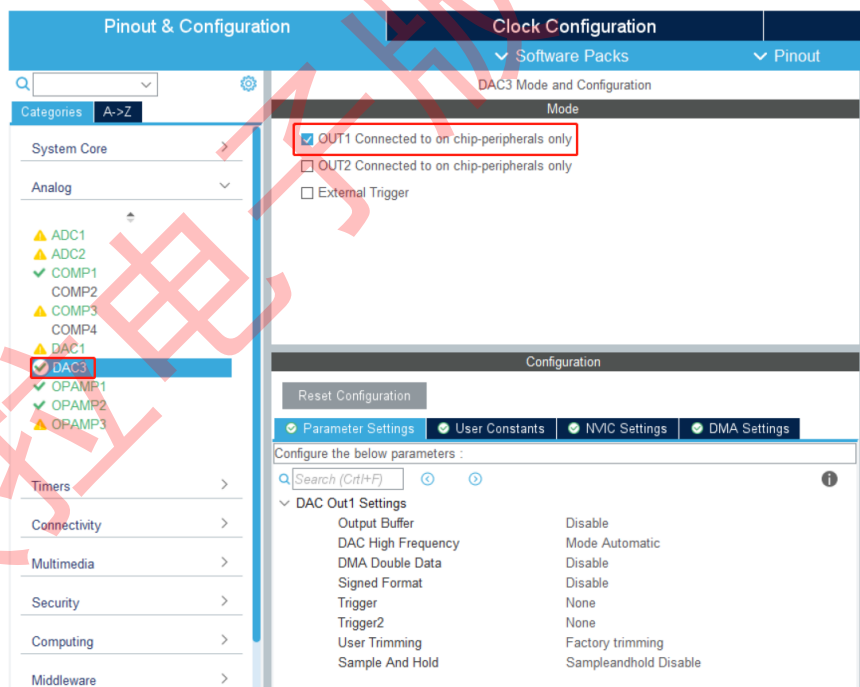
TIM1 比较 4 事件：



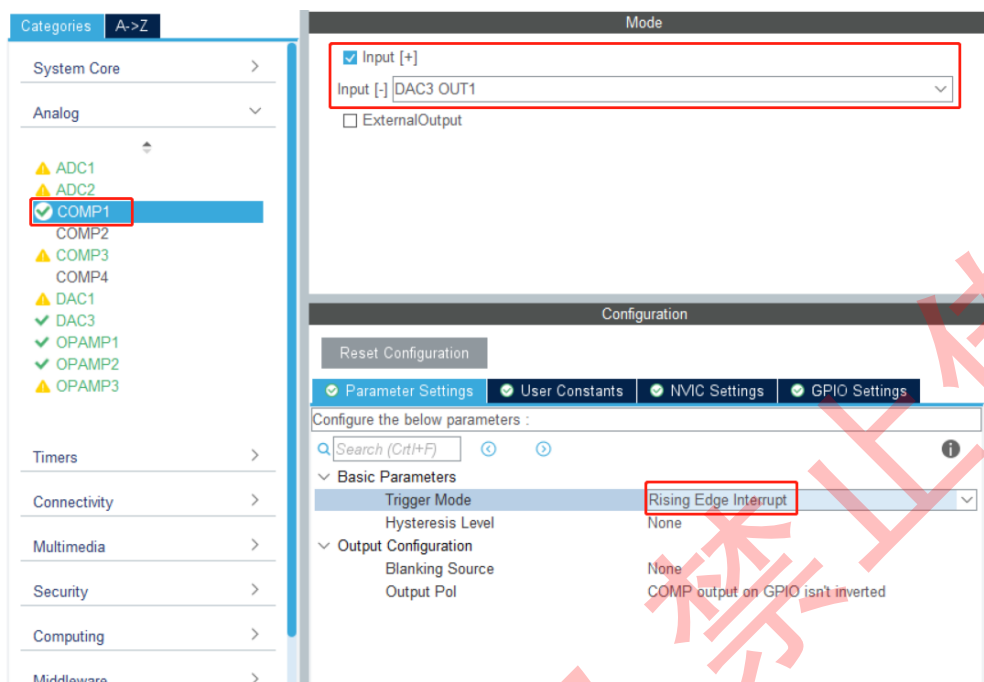
② 首先将 PB1 配置增加 COMP1\_INP 功能；



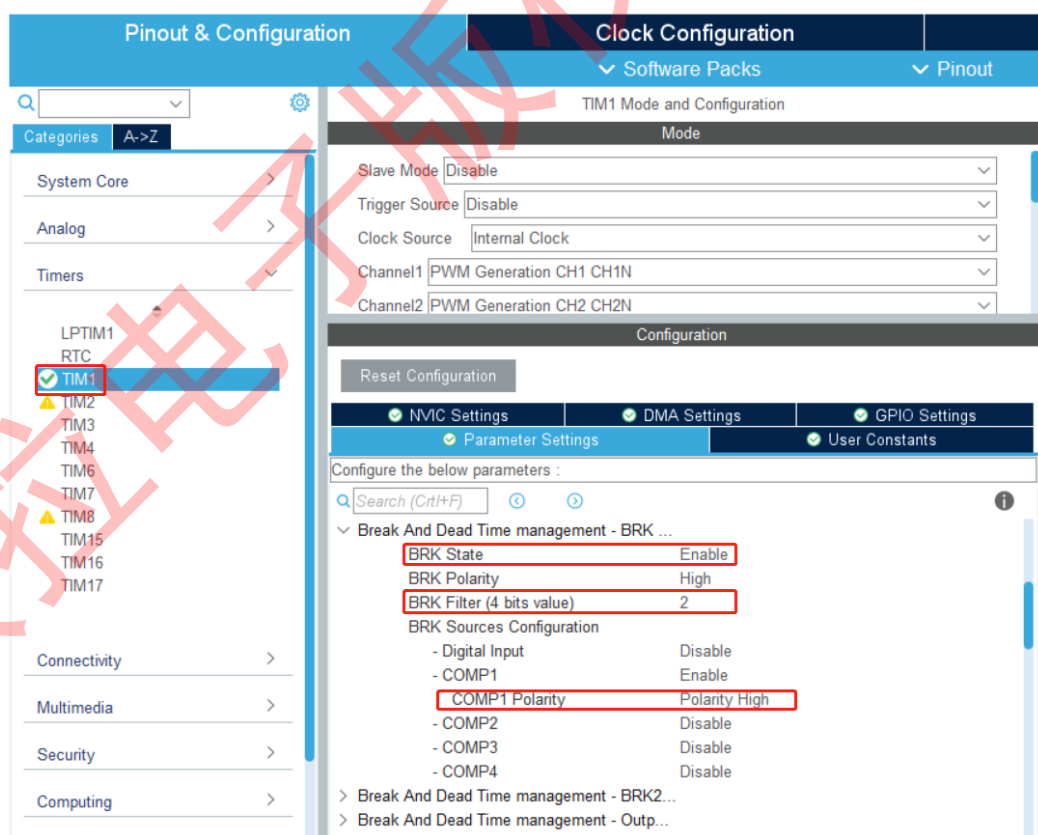
③ 设置 DAC\_CHI 连接 MCU 内部外设，其他参数默认，不需要修改；



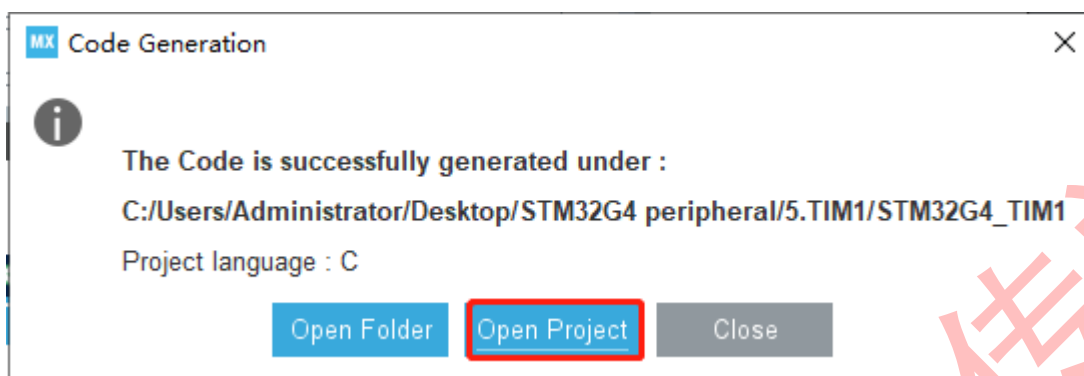
④ 设置 COMP1 的 INP 及 INM，配置上升沿触发中断；



⑤ 使能 TIM1 的 break 功能，滤波器配置为 2，高电平触发 break；



⑥ 点击生成代码 **GENERATE CODE**，打开 Keil 工程；



- ⑦ 修改上传 VOFA 数据个数, ia, ib, ic, Vbus, V 电位器;

```
62 float temp[5];
63 static uint8_t tempData[24] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0x80,0x7F};
```

- ⑧ 设置 TIM1 ARR 值 8000-1; CCR4 值 8000-2; 开启 channel 4 pwm; 开启

ADC 注入组采样;

PWM 开关频率 =  $160\text{M}/(8000*2) = 10\text{k}$ ;

```
126 TIM1->ARR = 8000 - 1;
127 TIM1->CCR4 = 8000 - 2;
128 HAL_TIM_Base_Start(&htim1);
129 HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
130 HAL_ADCEX_InjectedStart_IT(&hadc1);
131 HAL_ADCEX_InjectedStart(&hadc2);
```

- ⑨ 启动 DAC, 并设置 DAC 阈值; 启动比较器;

```
131 HAL_ADCEX_InjectedStart(&hadc2);
132 HAL_DAC_Start(&hdac3, DAC_CHANNEL_1);
133 HAL_DAC_SetValue(&hdac3, DAC_CHANNEL_1, DAC_ALIGN_12B_R, 3000);
134 HAL_COMP_Start(&hcomp1);
```

- ⑩ 在 while 循环中对 VDC 和电位器电压采样;

VDC 电压采样增益:  $26 = 78\text{k}/3\text{k}$ ; (基于原理图电阻分压)

```
136 while (1)
137 {
138     HAL_ADC_Start(&hadc1);
139     HAL_ADC_Start(&hadc2);
140     temp[3] = HAL_ADC_GetValue(&hadc1);
141     temp[3] = temp[3]*3.3f/4096;
142     temp[4] = HAL_ADC_GetValue(&hadc2);
143     temp[4] = temp[4]*3.3f/4096*26;
144     HAL_Delay(10);
```

- ⑪ 按键 2 开始 PWM 输出, 按键 3 停止 PWM 输出;



```

210 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
211 {
212     /* Prevent unused argument(s) compilation warning */
213     UNUSED(GPIO_Pin);
214     if(Button2_Pin == GPIO_Pin)
215     {
216         HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
217         HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
218         HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
219         HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);
220         HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
221         HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_3);
222     }
223     if(Button3_Pin == GPIO_Pin)
224     {
225         HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1);
226         HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_2);
227         HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_3);
228         HAL_TIMEx_PWMN_Stop(&htim1, TIM_CHANNEL_1);
229         HAL_TIMEx_PWMN_Stop(&htim1, TIM_CHANNEL_2);
230         HAL_TIMEx_PWMN_Stop(&htim1, TIM_CHANNEL_3);
231     }
232     /* NOTE: This function should not be modified, when the callback is needed,
233             the HAL_GPIO_EXTI_Callback could be implemented in the user file
234     */
235 }
236

```

⑫ 在 ADC 注入组转换完成回调函数中计算三相电流偏置值及 ia, ib, ic 值，

同时使用 UART DMA 上传数据到 VOFA;

```

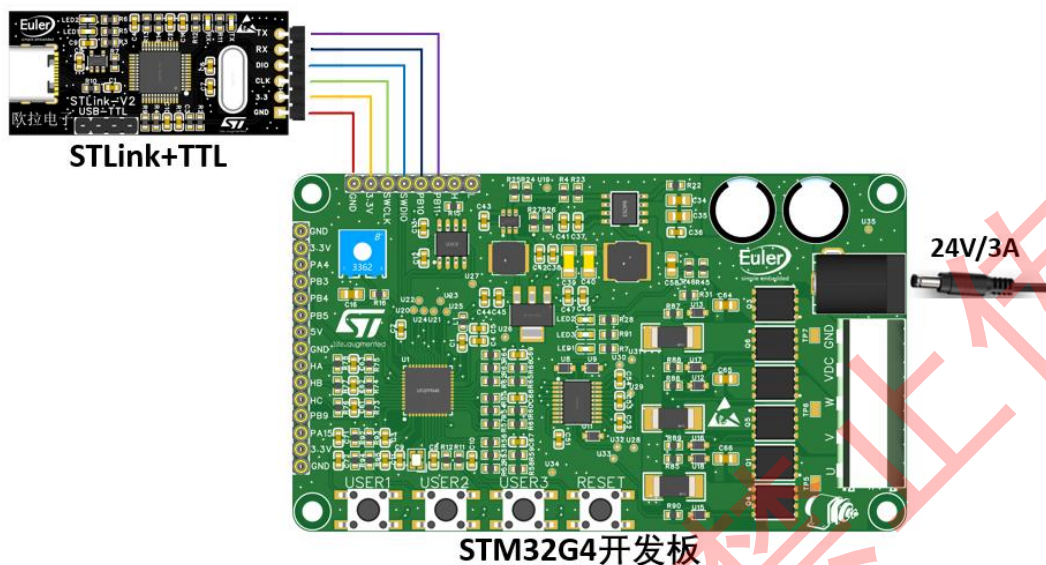
270 void HAL_ADCEx_InjectedConvCpltCallback(ADC_HandleTypeDef *hadc)
271 {
272     static uint8_t cnt;
273     /* Prevent unused argument(s) compilation warning */
274     UNUSED(hadc);
275     if(hadc == &hadc1)
276     {
277         cnt++;
278         adc1_in1 = hadc1.Instance->JDR1;
279         adc1_in2 = hadc2.Instance->JDR1;
280         adc1_in3 = hadc1.Instance->JDR2;
281         IA_Offset += adc1_in1;
282         IB_Offset += adc1_in2;
283         IC_Offset += adc1_in3;
284         if(cnt >= 10)
285         {
286             ADC_offset = 1;
287             IA_Offset = IA_Offset/10;
288             IB_Offset = IB_Offset/10;
289             IC_Offset = IC_Offset/10;
290         }
291     }
292     else
293     {
294         adc1_in1 = hadc1.Instance->JDR1;
295         Ia = (adc1_in1 - IA_Offset)*0.0193359375f;
296         adc1_in2 = hadc2.Instance->JDR1;
297         Ib = (adc1_in2 - IB_Offset)*0.0193359375f;
298         adc1_in3 = hadc1.Instance->JDR2;
299         Ic = (adc1_in3 - IC_Offset)*0.0193359375f;
300         TIM1->CCR1 = 2000;
301         TIM1->CCR2 = 4000;
302         TIM1->CCR3 = 6000;
303         load_data[0] = Ia;
304         load_data[1] = Ib;
305         load_data[2] = Ic;
306         load_data[3] = 0;
307         load_data[4] = 0;
308         memcpy(tempData, (uint8_t *)&load_data, sizeof(load_data));
309         HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 6*4);
310     }
311 }

```

⑬ 编译并下载代码；

## 2. 实验现象

按以下图片接线，并将 USB 转 UART 模块连接至电脑：

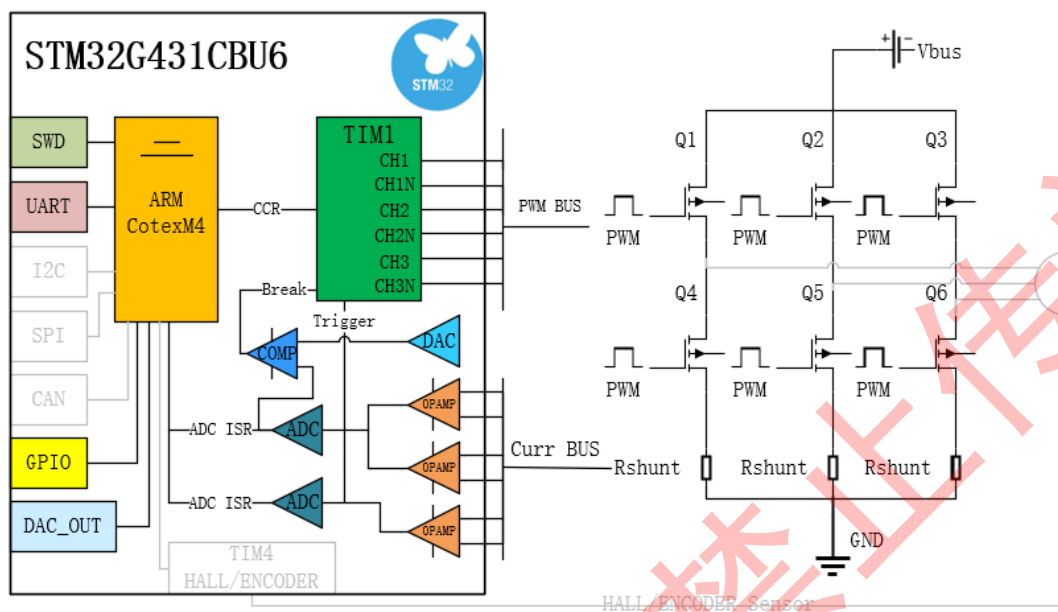


打开 VOFA 上位机显示三相电流  $i_a$ ,  $i_b$ ,  $i_c$  及 VDC、电位器电压



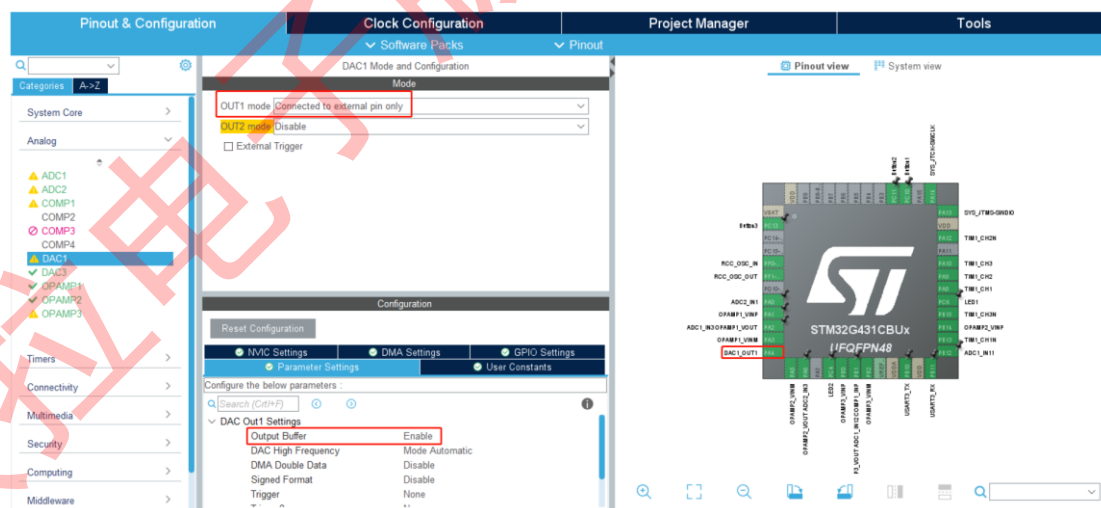
## 第十一章： DAC 输出电流波形

本章为大家介绍 STM32G4 DAC 在 buffer 模式下通过 GPIO 输出波形至示波器或者逻辑分析仪。



## 1. STM32CUBEMX 配置及 Keil 代码

- ① 配置 DAC1\_CH1 为输出模式，只连接外部 Pin，配置外部 Pin 为 PA4；使能 buffer 模式；



- ② 点击生成代码；
- ③ 启动 DAC1 通道 1；

```
137 HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);
```

- ④ 在 ADC 中断函数中增加修改 DAC 输出值的函数；DAC\_wave 为一个正弦波  
版本：V13

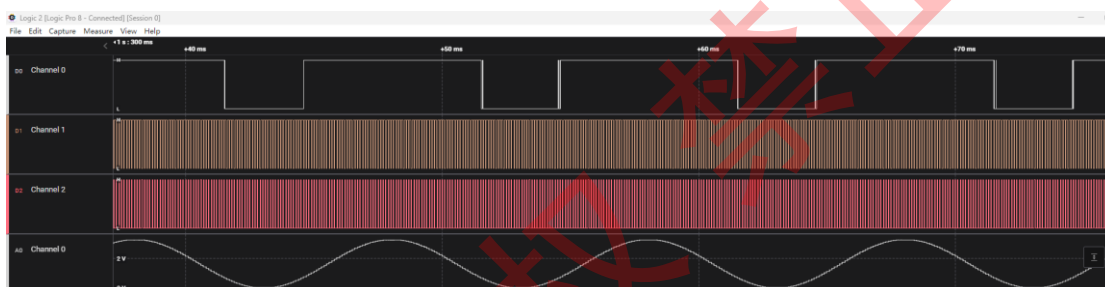
数组；

```
323     i++;  
324     if(i >= 100)  
325     {  
326         HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, DAC_wave[i]);  
327     }  
328 }
```

⑤ 编译并下载程序；

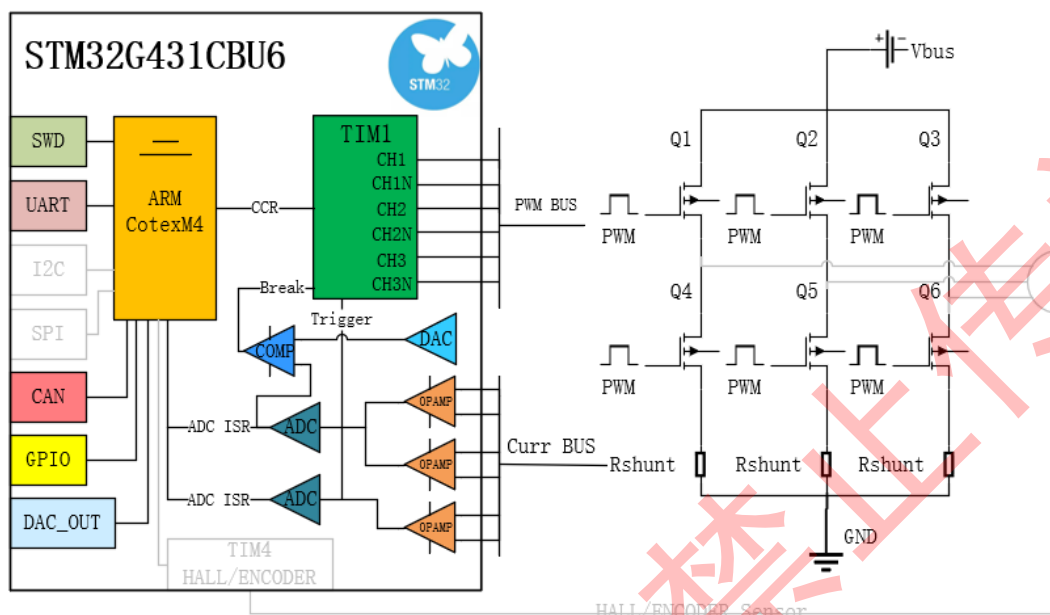
## 2. 实验现象

使用逻辑分析仪或者示波器连接逻辑分析仪，查看上传波形；



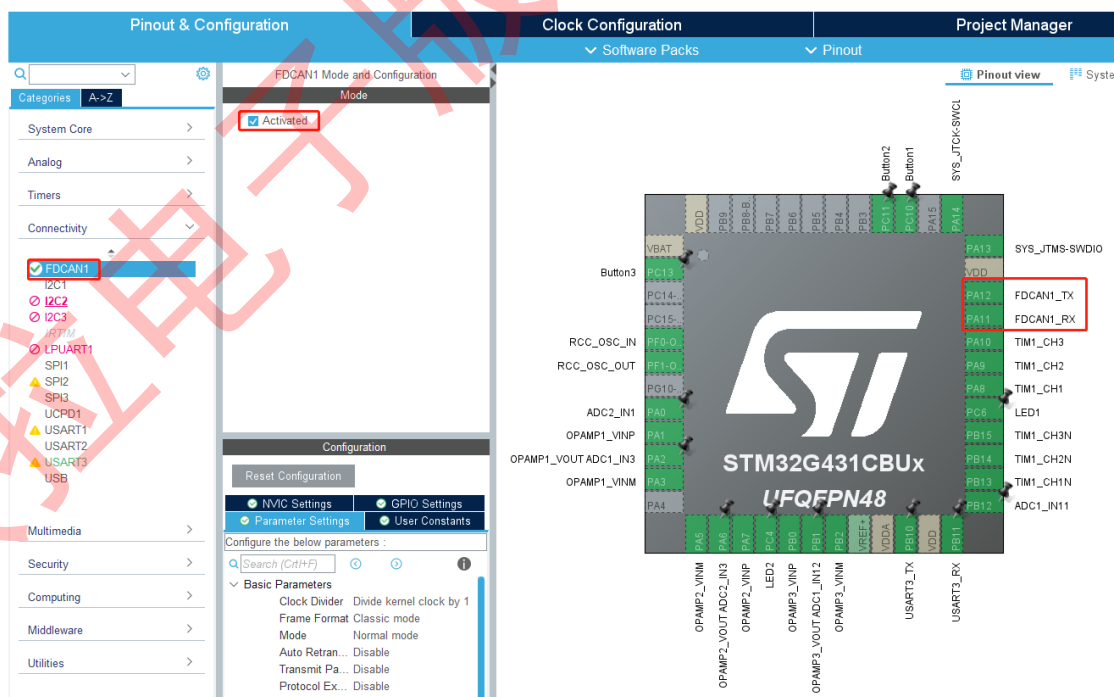
## 第十二章：CAN/CANFD 通讯

本章为大家介绍使用 STM32CUBEMX 建立 CAN 通讯，由于 CAN 通讯自身的特点，在电机控制的场景中使用非常常见，了解 CAN 通讯非常有必要。



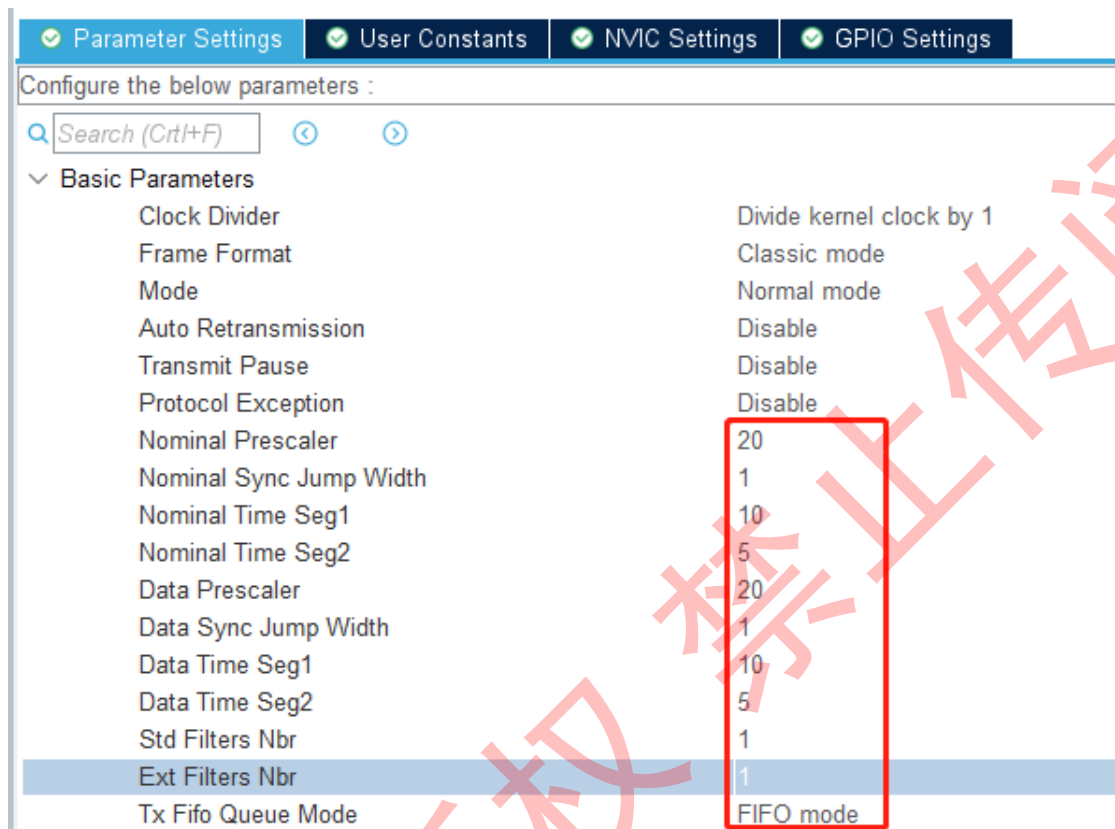
## 1. STM32CUBEMX 配置及 Keil 代码

① 在上一章的工程基础上我们增加 CAN 通讯，使能 CAN 外设；

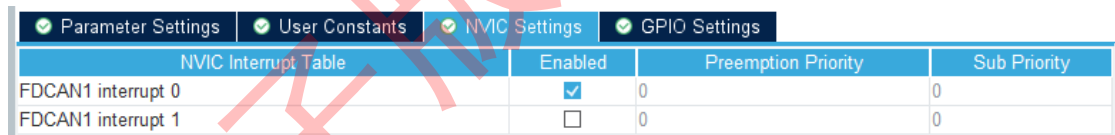


② 配置仲裁段和数据段的分频系数跳转位宽，使能 FIFO 模式；

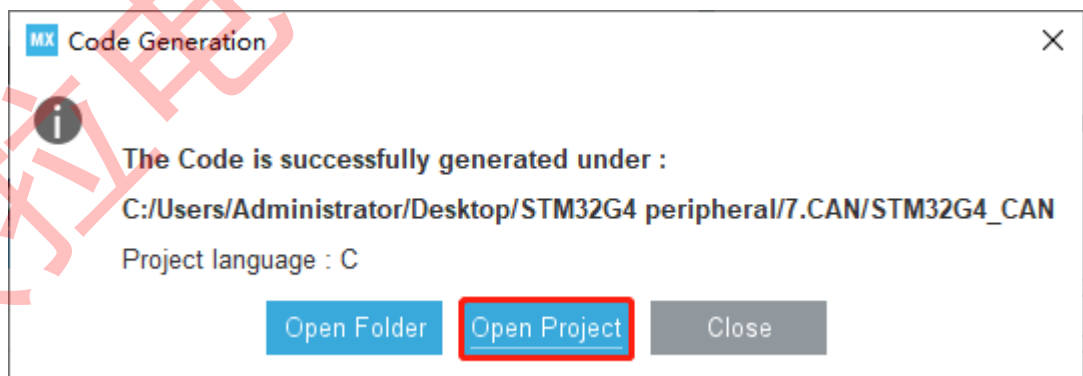
波特率：500k =  $160M/20/(1+10+5)$



③ 使能中断；



④ 点击生成代码 **GENERATE CODE**，打开 Keil 工程；



⑤ 定义 RX 和 TX 数据及中间变量；



```

66 FDCAN_RxHeaderTypeDef RxHeader;
67 FDCAN_TxHeaderTypeDef TxHeader;
68 uint8_t RxData[8]={NULL};
69 uint8_t TxData[8] = {NULL};
70 float CANtemp[1];
71 /* USER CODE END PV */

```

- ⑥ 配置 RX 和 TX 数据帧类型，RX 接收的 ID 范围；TX ID 和数据长度等参数；

```

75 /* USER CODE BEGIN PFP */
76 void FDCAN2_Config(void);
77 /* USER CODE END PFP */

140 FDCAN2_Config();
141 /* USER CODE END 2 */

327 void FDCAN2_Config(void)
328 {
329     FDCAN_FilterTypeDef sFilterConfig;
330
331     HAL_FDCAN_ActivateNotification(&hfdcan1, FDCAN_IT_RX_FIFO0_NEW_MESSAGE, 0);
332
333     sFilterConfig.IdType = FDCAN_EXTENDED_ID;
334     sFilterConfig.FilterIndex = 0;
335     sFilterConfig.FilterType = FDCAN_FILTER_RANGE;
336     sFilterConfig.FilterConfig = FDCAN_FILTER_TO_RXFIFO0;
337     sFilterConfig.FilterID1 = 0x00000000;
338     sFilterConfig.FilterID2 = 0x01ffffff;
339     HAL_FDCAN_ConfigFilter(&hfdcan1, &sFilterConfig);
340
341     TxHeader.Identifier = 0x1B;
342     TxHeader.IdType = FDCAN_EXTENDED_ID;
343     TxHeader.TxFrameType = FDCAN_DATA_FRAME;
344     TxHeader.DataLength = FDCAN_DLC_BYTES_8;
345     TxHeader.ErrorStateIndicator = FDCAN_ESI_ACTIVE;
346     TxHeader.BitRateSwitch = FDCAN_BRS_OFF;
347     TxHeader.FDFormat = FDCAN_CLASSIC_CAN;
348     TxHeader.TxEventFifoControl = FDCAN_NO_TX_EVENTS;
349     TxHeader.MessageMarker = 0x52;
350
351     HAL_FDCAN_Start(&hfdcan1);
352 }

```

- ⑦ 在 while 循环中讲 VDC 电压上传至上位机；

```

144 /* USER CODE BEGIN WHILE */
145 while (1)
146 {
147     HAL_ADC_Start(&hadc1);
148     HAL_ADC_Start(&hadc2);
149     temp[3] = HAL_ADC_GetValue(&hadc1);
150     temp[3] = temp[3]*3.3f/4096;
151     temp[4] = HAL_ADC_GetValue(&hadc2);
152     temp[4] = temp[4]*3.3f/4096*26;
153     CANtemp[0] = temp[4];
154     memcpy(TxData, (uint8_t *) &CANtemp, sizeof(CANtemp));
155     HAL_FDCAN_AddMessageToTxFifoQ(&hfdcan1, &TxHeader, TxData);
156     HAL_Delay(10);
157 /* USER CODE END WHILE */

```

- ⑧ 在 it.c 文件中, 在 FDCAN 中断里面, 接收上位机发送的数据;

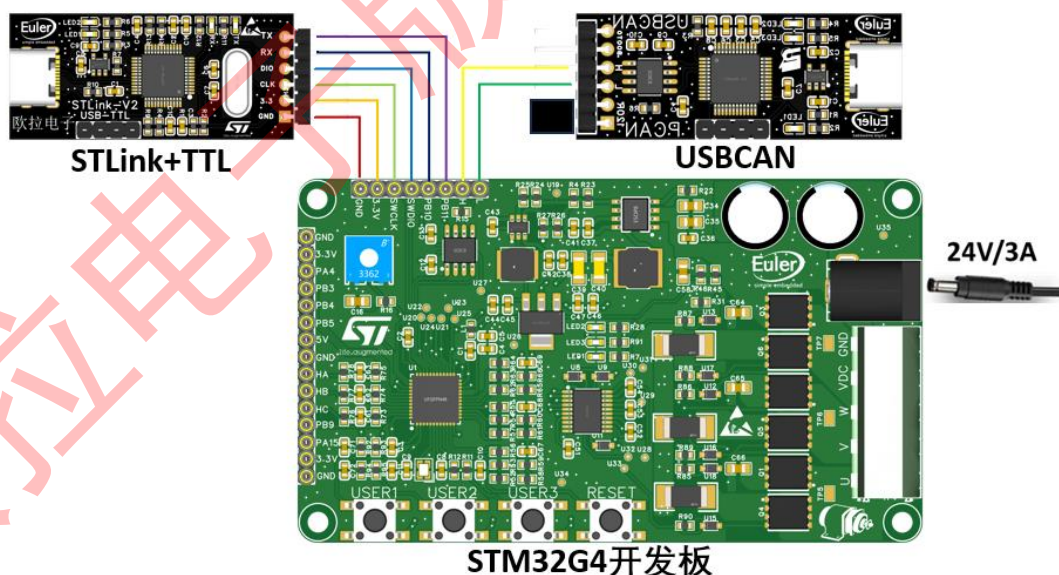
```
65  /* USER CODE BEGIN EV */
66  extern FDCAN_HandleTypeDef hfdcan1;
67  extern FDCAN_RxHeaderTypeDef RxHeader;
68  extern uint8_t RxData[8];
69  /* USER CODE END EV */

241 void FDCAN1_IT0_IRQHandler(void)
242 {
243     /* USER CODE BEGIN FDCAN1_IT0_IRQn 0 */
244
245     /* USER CODE END FDCAN1_IT0_IRQn 0 */
246     HAL_FDCAN_IRQHandler(&hfdcan1);
247     /* USER CODE BEGIN FDCAN1_IT0_IRQn 1 */
248     HAL_FDCAN_GetRxMessage(&hfdcan1, FDCAN_RX_FIFO0, &RxHeader, RxData);
249     /* USER CODE END FDCAN1_IT0_IRQn 1 */
250 }
251
```

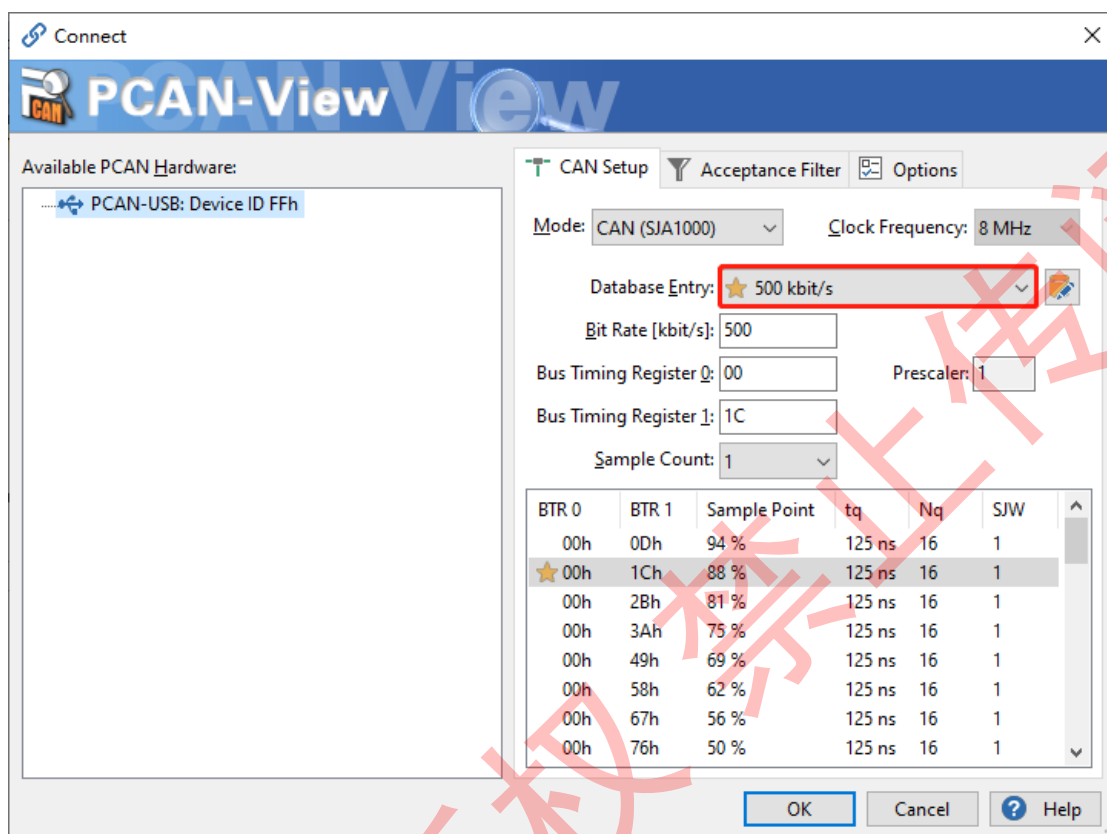
- ⑨ 编译并下载代码;

## 2. 实验现象

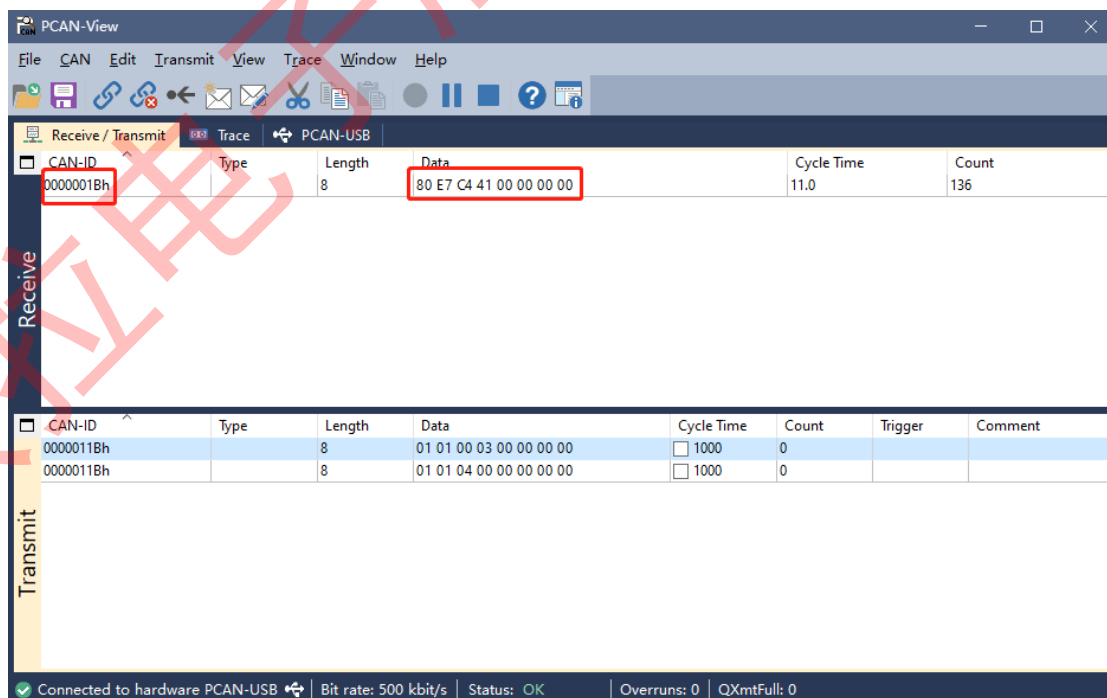
按以下图片接线, 并讲 USB 转 UART 模块及 USBCAN 模块连接至电脑:



- ① 打开 PCAN  上位机, 选择 500k 波特率, 点击 OK;



- ② 可以看到接收的数据帧 ID 与 MCU 中设置一致，上传的 VDC 电压值实时更新；



- ③ 点击下载数据，在 KEIL debug 窗口中查看接收数据；

| CAN-ID    | Type | Length | Data                    | Cycle Time                               | Count | Trigger | Comment |
|-----------|------|--------|-------------------------|------------------------------------------|-------|---------|---------|
| 0000011Bh |      | 8      | 01 01 00 03 00 00 00 00 | <input checked="" type="checkbox"/> 1000 | 2     | Time    |         |
| 0000011Bh |      | 8      | 01 01 04 00 00 00 00 00 | <input type="checkbox"/> 1000            | 0     |         |         |

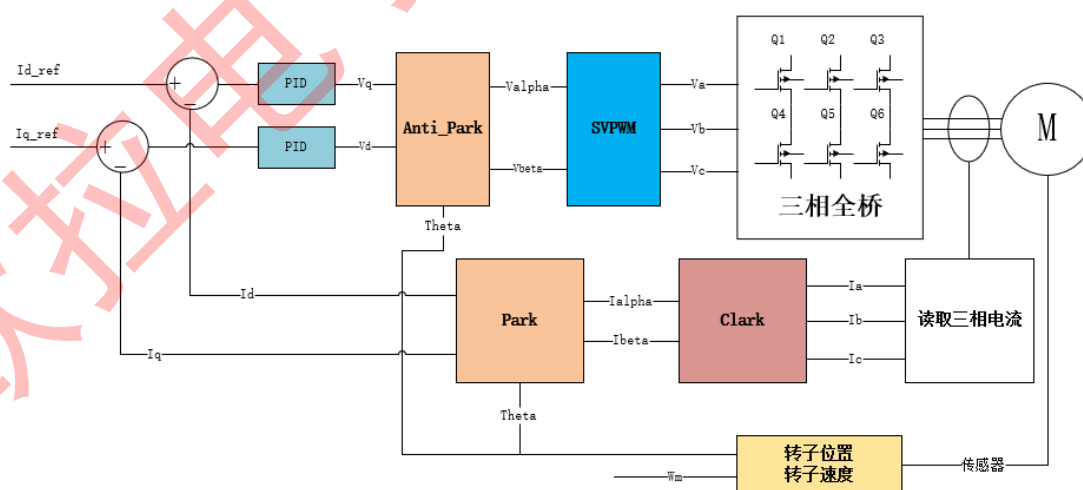
Transmit

Connected to hardware PCAN-USB | Bit rate: 500 kbit/s | Status: OK | Overruns: 0 | QXmtFull: 0

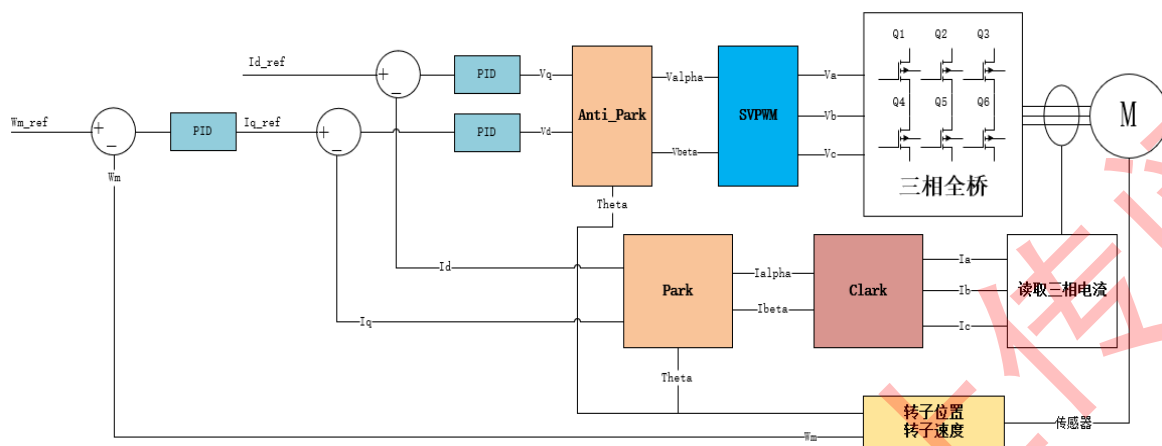
| Name   | Value                     | Type     |
|--------|---------------------------|----------|
| RxData | 0x20000088 RxData[] "..." | uchar[8] |
| [0]    | 0x01                      | uchar    |
| [1]    | 0x01                      | uchar    |
| [2]    | 0x00                      | uchar    |
| [3]    | 0x03                      | uchar    |
| [4]    | 0x00                      | uchar    |
| [5]    | 0x00                      | uchar    |
| [6]    | 0x00                      | uchar    |
| [7]    | 0x00                      | uchar    |

## MATLAB 控制算法篇

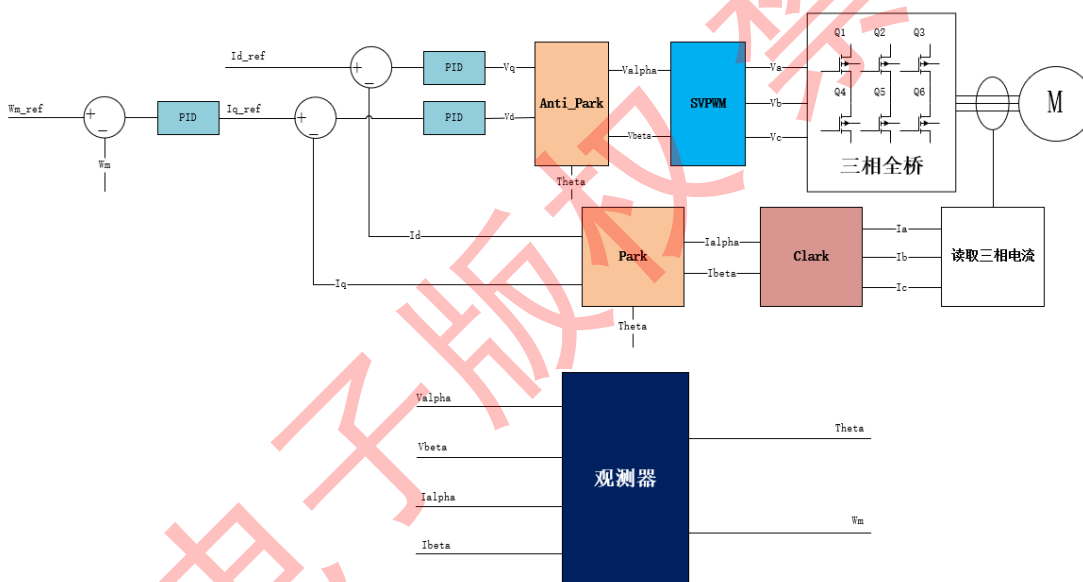
电流环：



速度环：



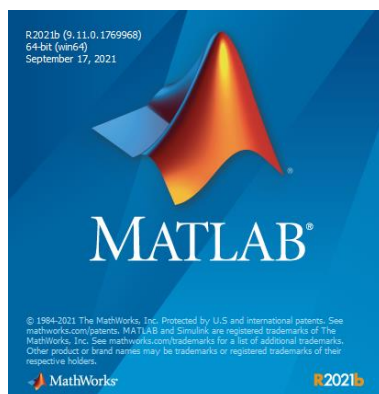
无感观测器:



## 第十三章：Clark 变换及代码生成

### 1. Clark 变换建模

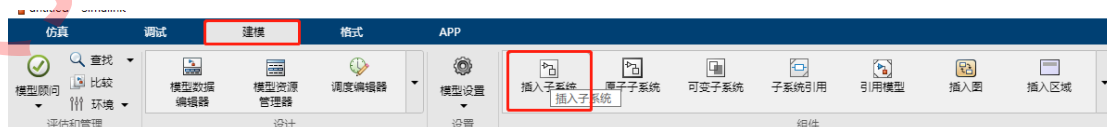
① 双击打开  MATLAB 软件;

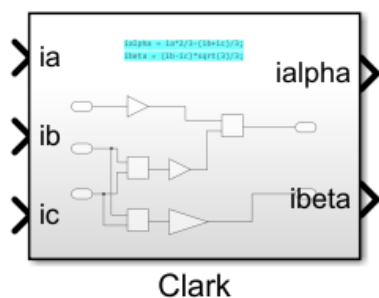


② 点击 Simulink，新建空白模型；

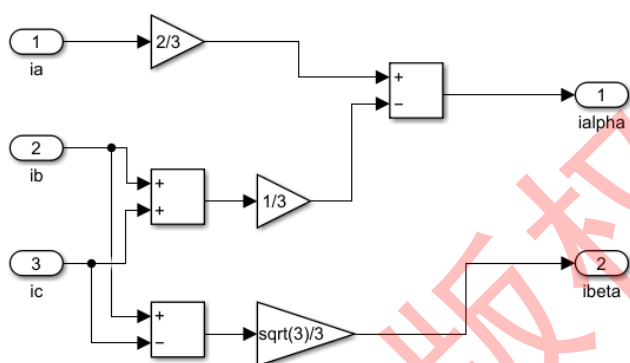


③ 新建子系统，命名为 Clark 变换；并在子系统中基于公式建立 Clark 变换模型；



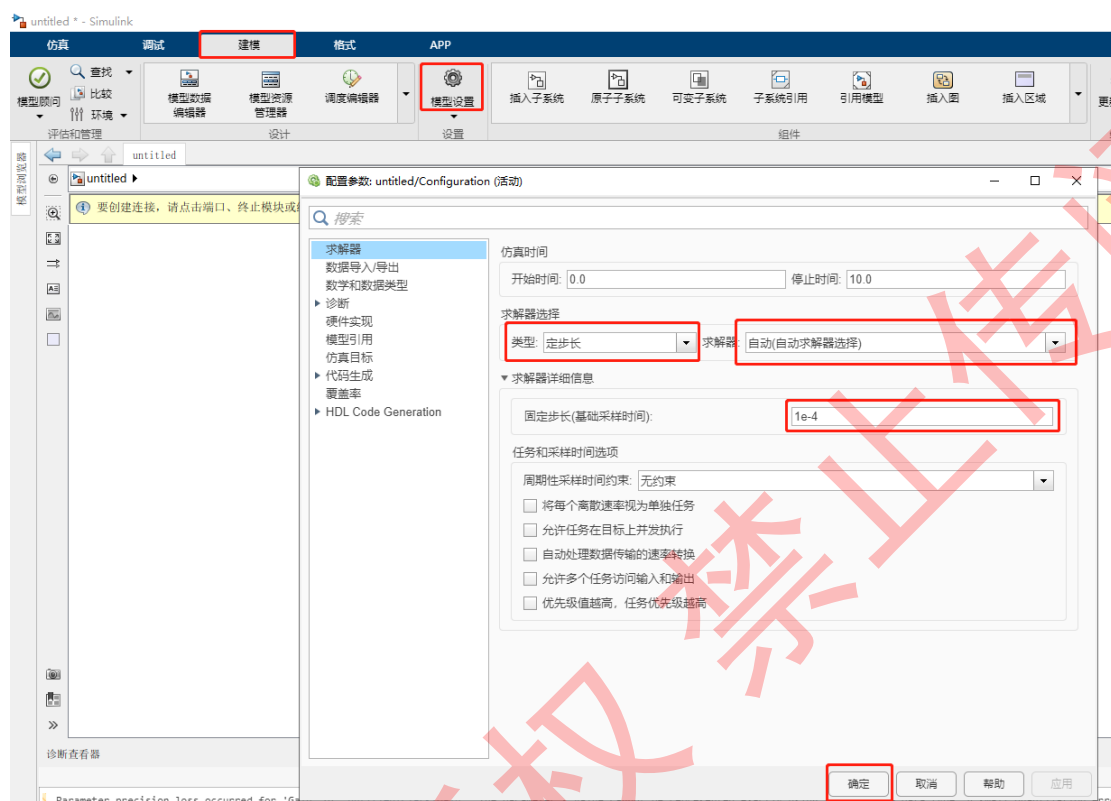


$$\begin{aligned} i_{\alpha} &= i_a \cdot \frac{2}{3} - \frac{i_b + i_c}{3} \\ i_{\beta} &= \frac{i_b - i_c}{\sqrt{3}} \end{aligned}$$

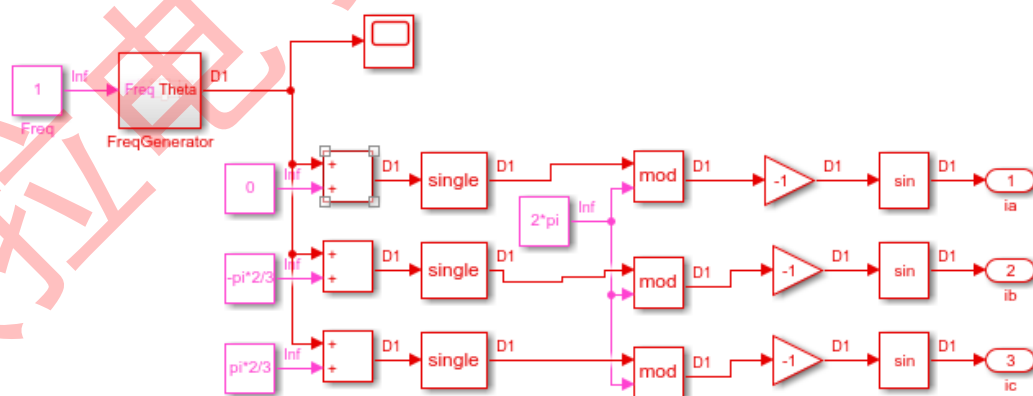


④ 修改模型仿真步长及采样时间：建模->模型设置->类型->固定步长；

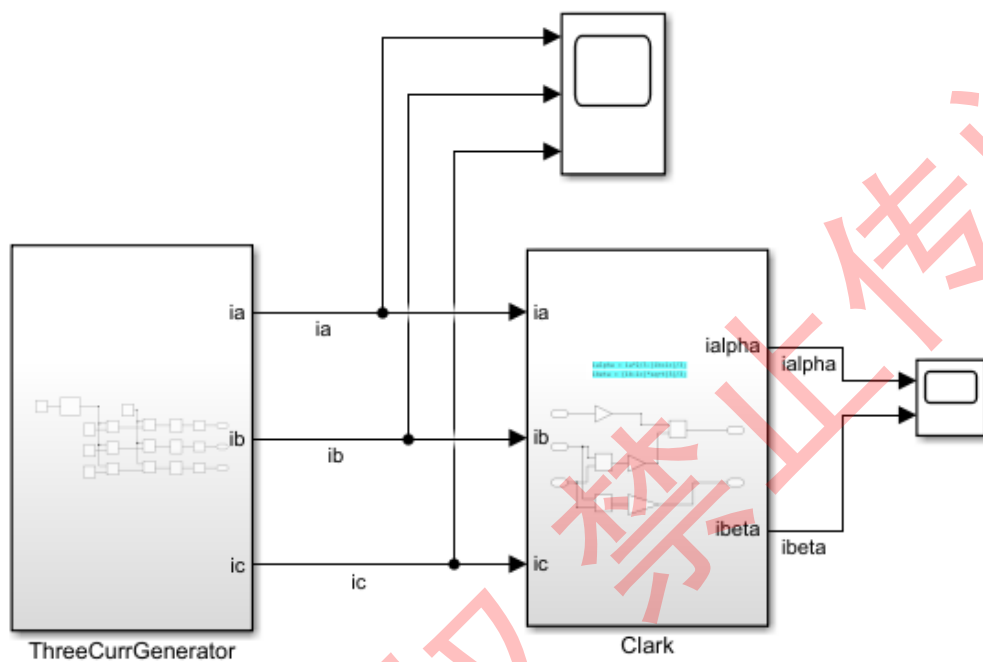


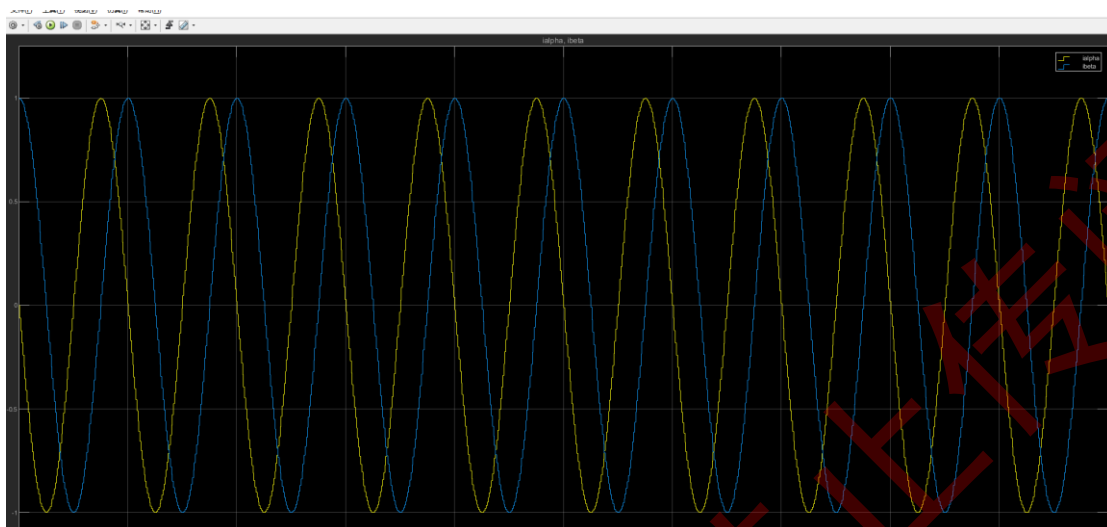


- ⑤ 增加三相电流生成模块；其中包含  $\theta$  角度生成模块；生成的  $\theta$  乘-1 为了与 Simulink 电机模块生成三相电流相位一致；该模块感兴趣的小伙伴可以自行研究一下；



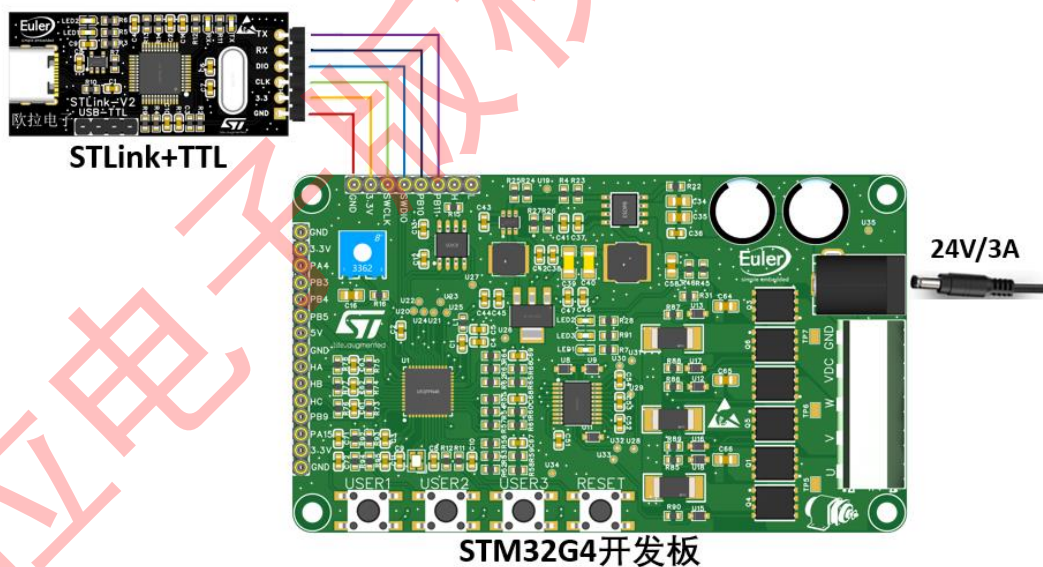
- ⑥ 增加示波器，点击运行按钮 ，观察三相电流及变换后的  $\alpha$  和  $\beta$  电流；



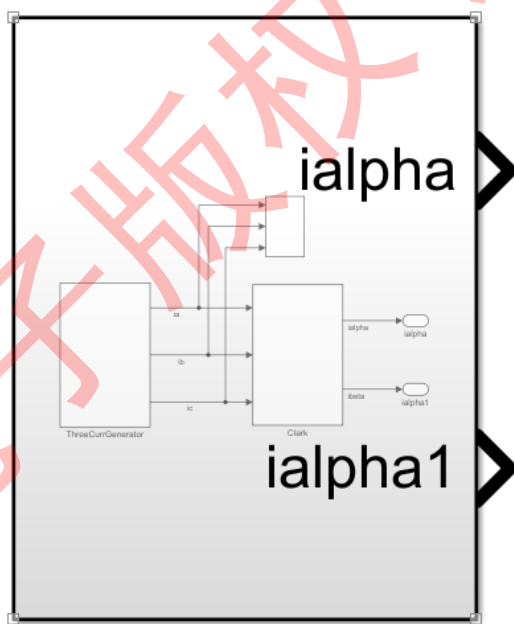
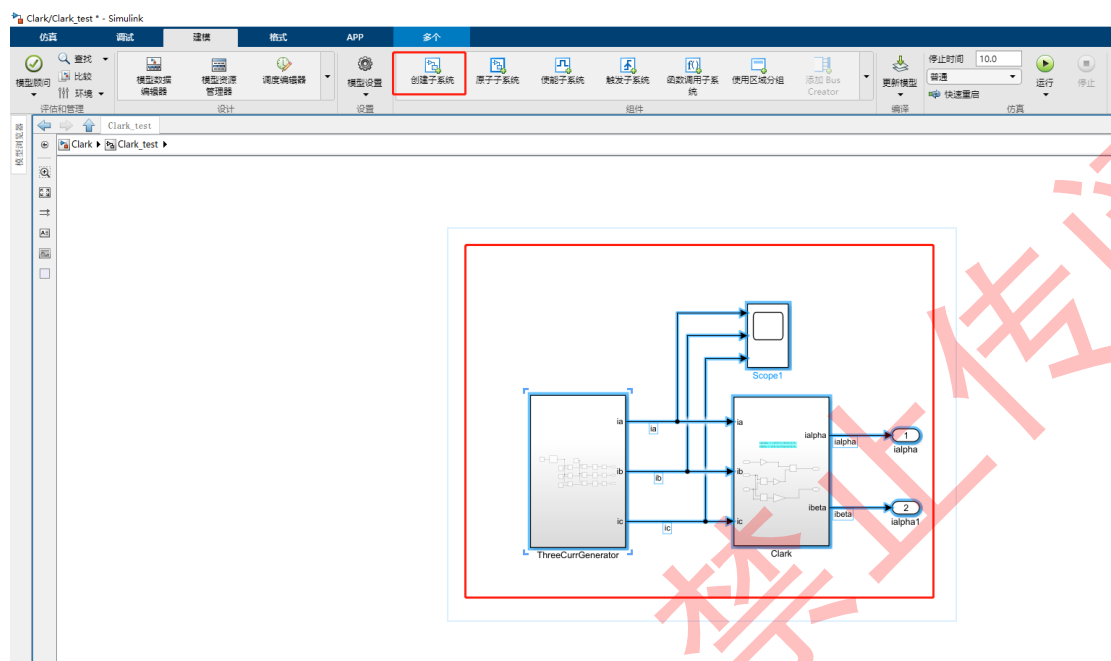


## 2. 实验现象

按以下图片接线，并将 USB 转 UART 模块连接至电脑：

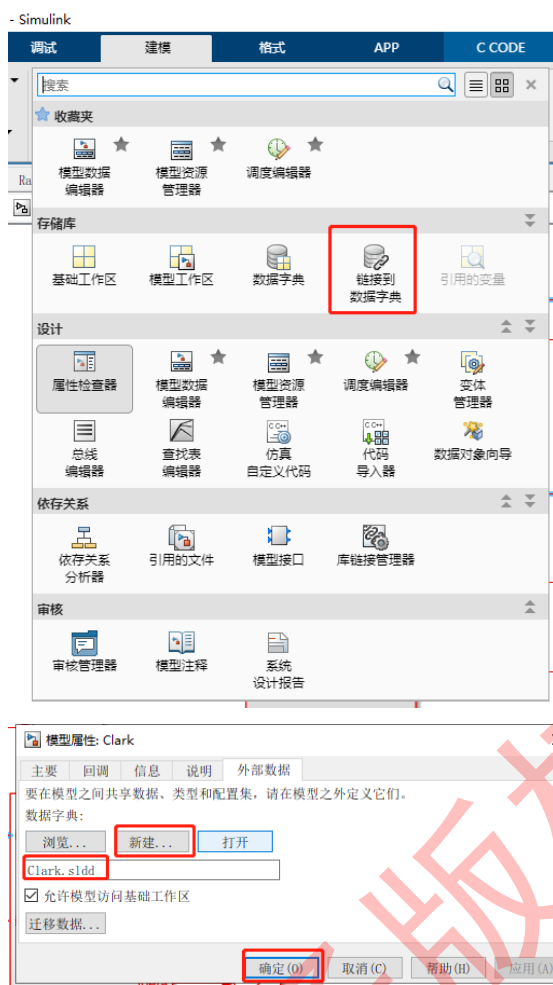


- ① 选中所有模型，点击创建子系统，将其封装成同一个模块；

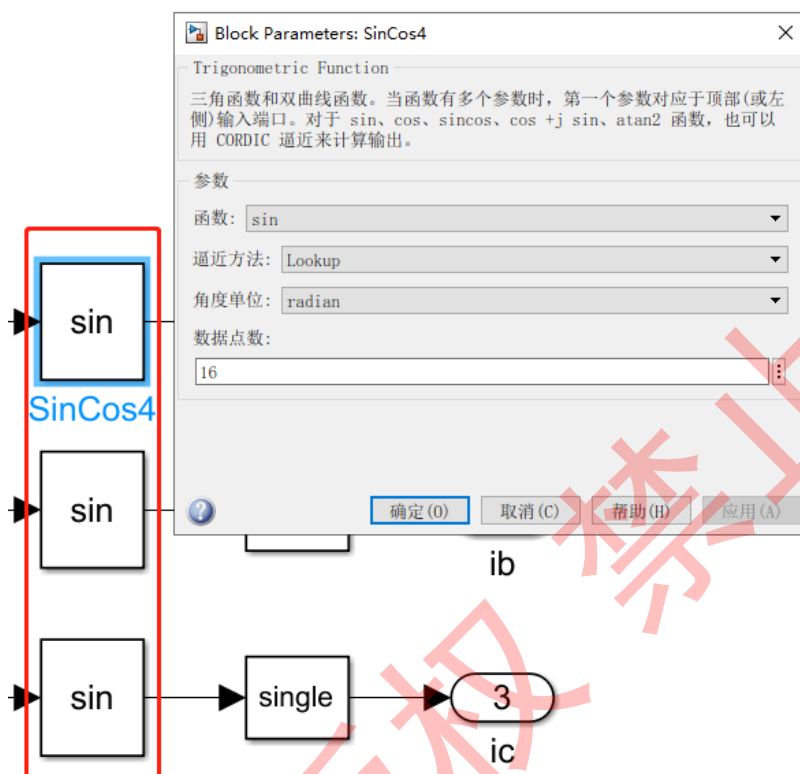


Clark\_test

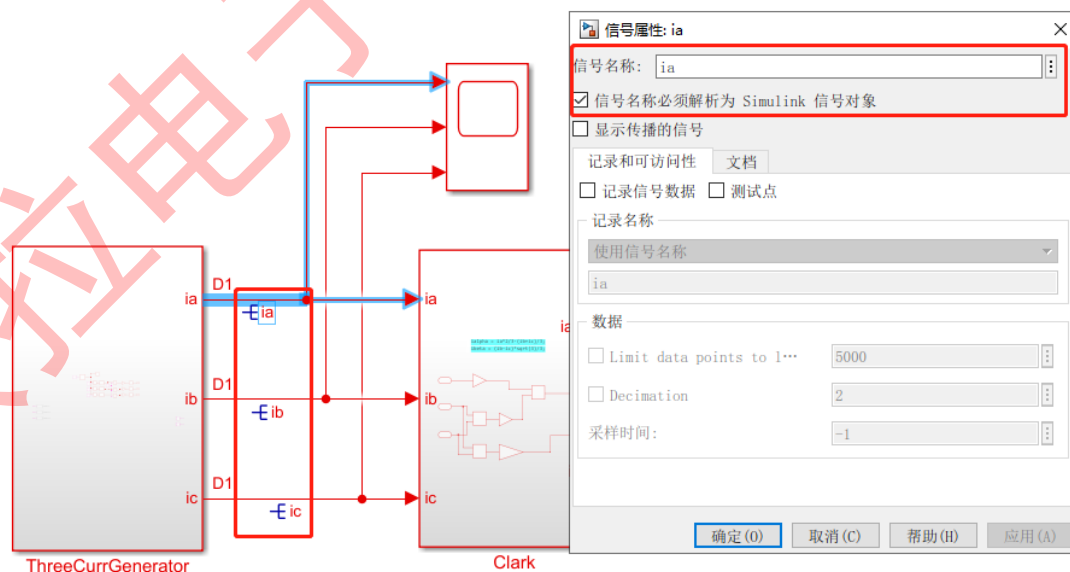
② 创建数据字典；

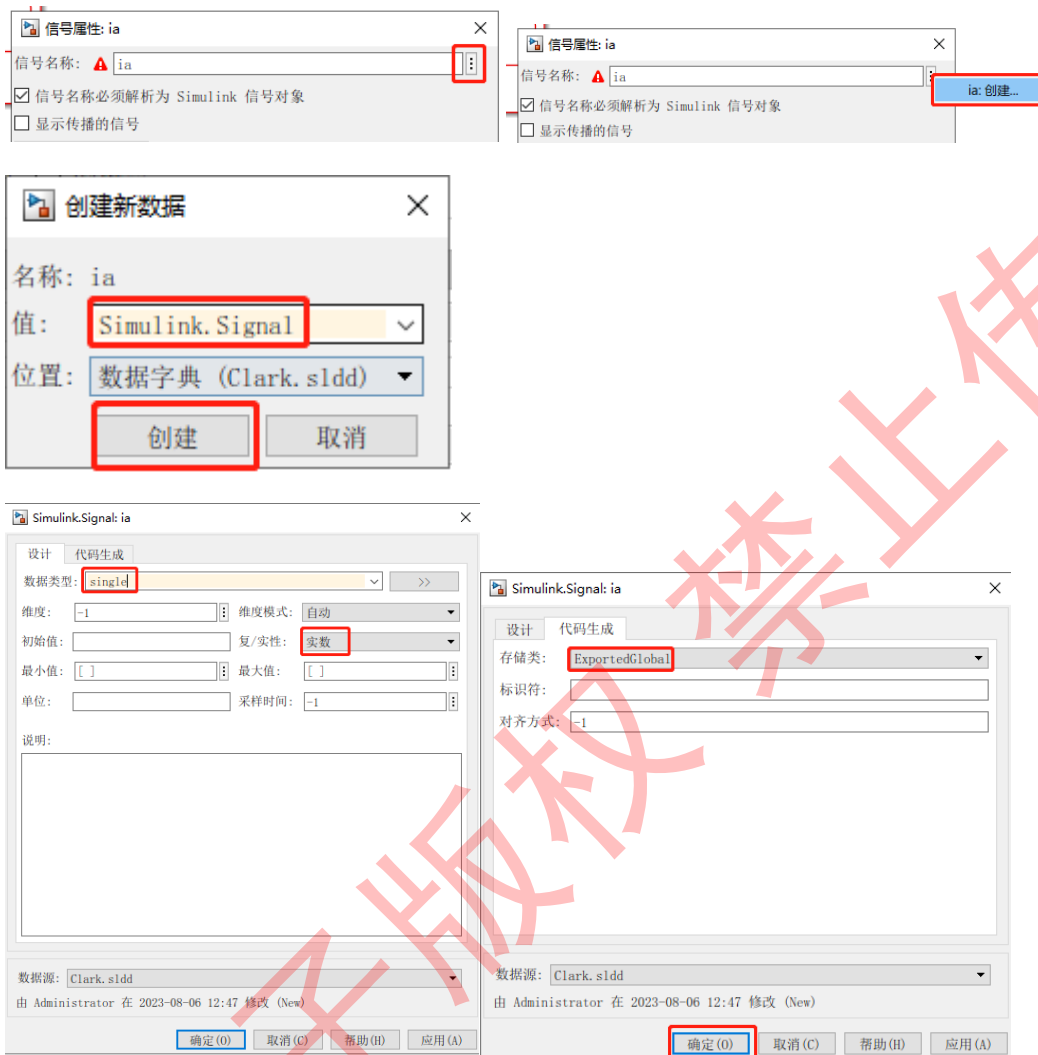


③ 将三相电流生成模块中的三个 sin 计算模块修改成以下配置：

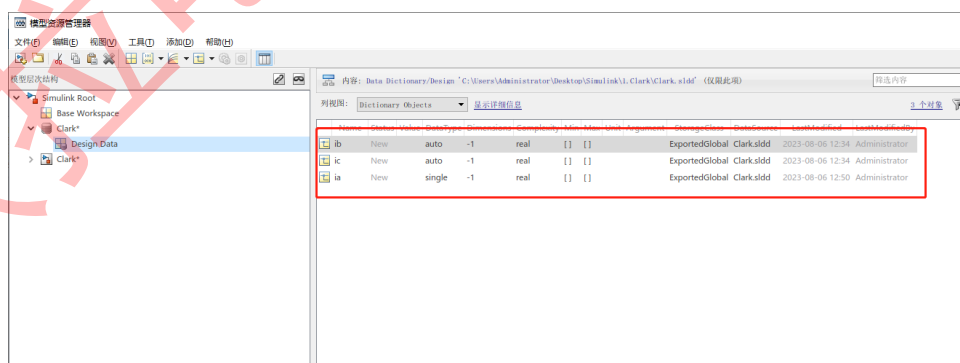


- ④ 将 ia, ib, ic 定义为全局的 Simulink signal，右键点击信号线，打开属性窗口，方便后续的在代码中观测；

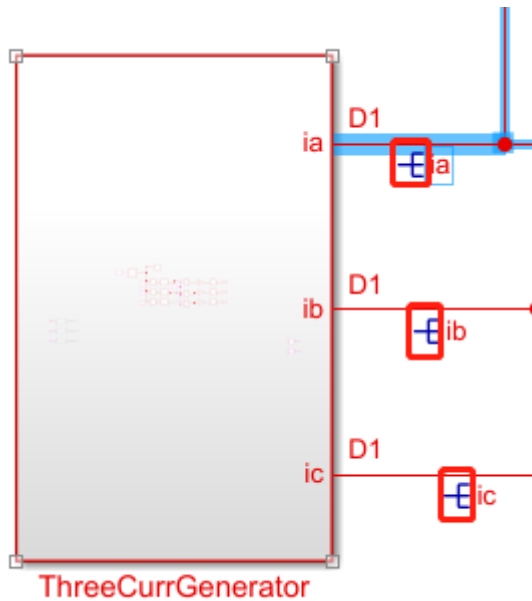




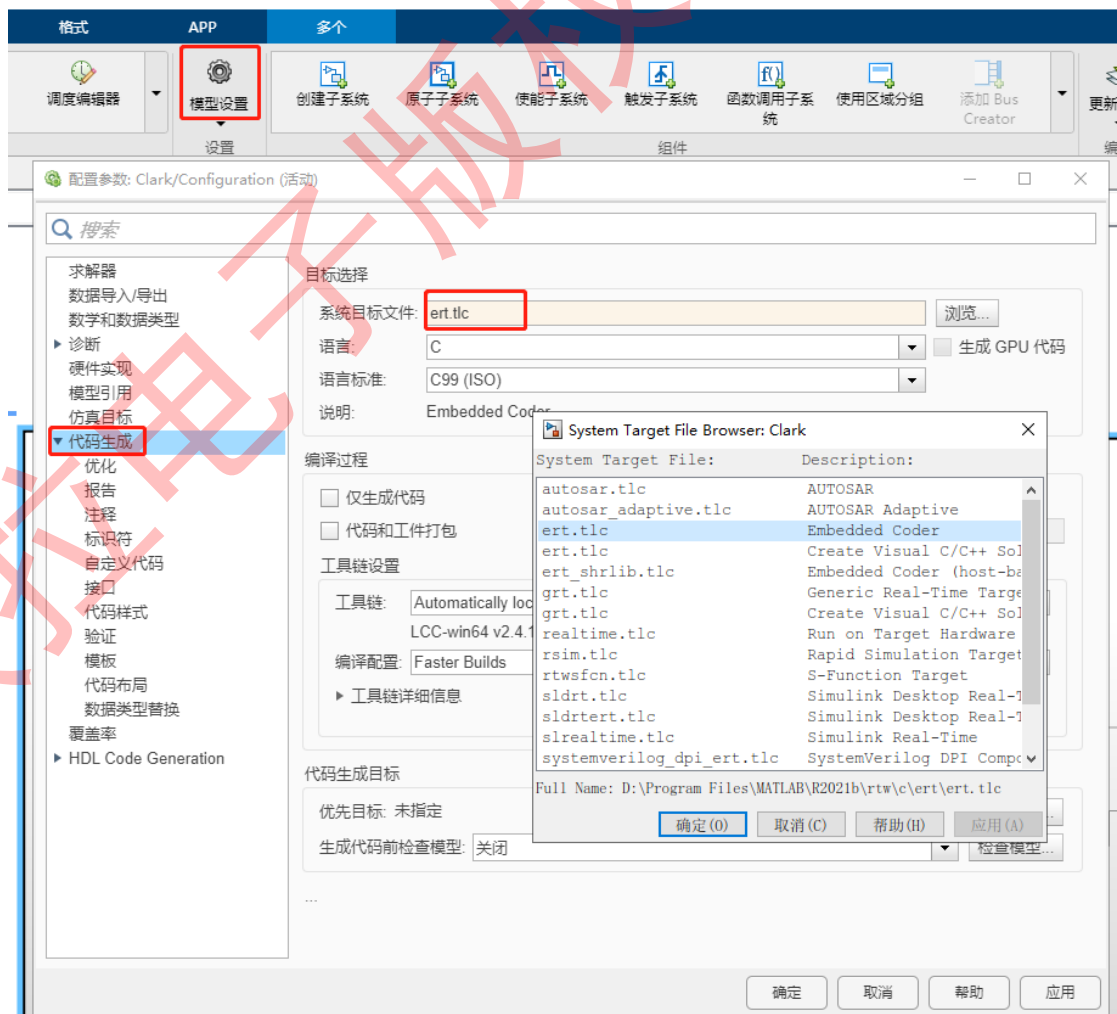
- ⑤ 打开数据字典，发现已经建立好 ia, ib, ic；模型中 ia, ib, ic 前方也多了一个叉子形状；



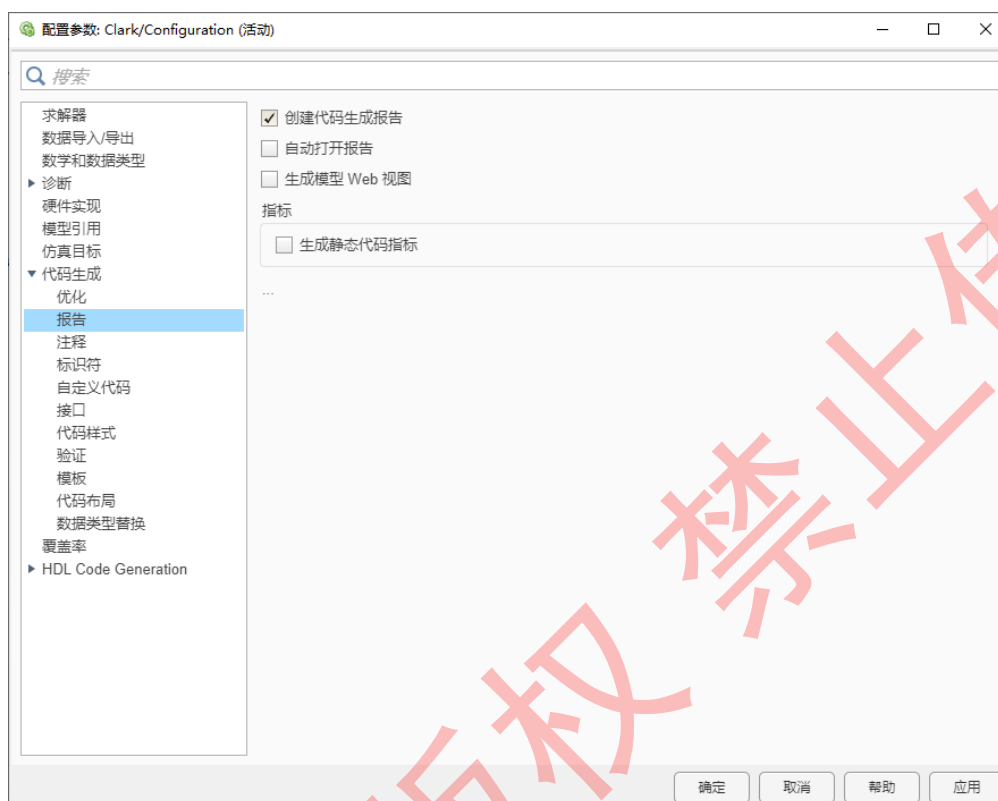




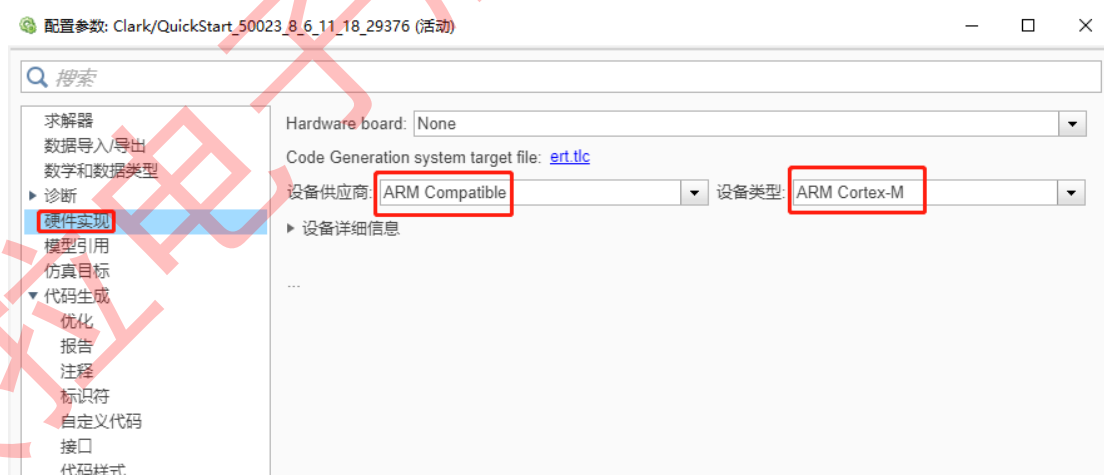
⑥ 点击模型设置，代码生成，将系统目标文件改为 ert.tlc,嵌入式代码生成；



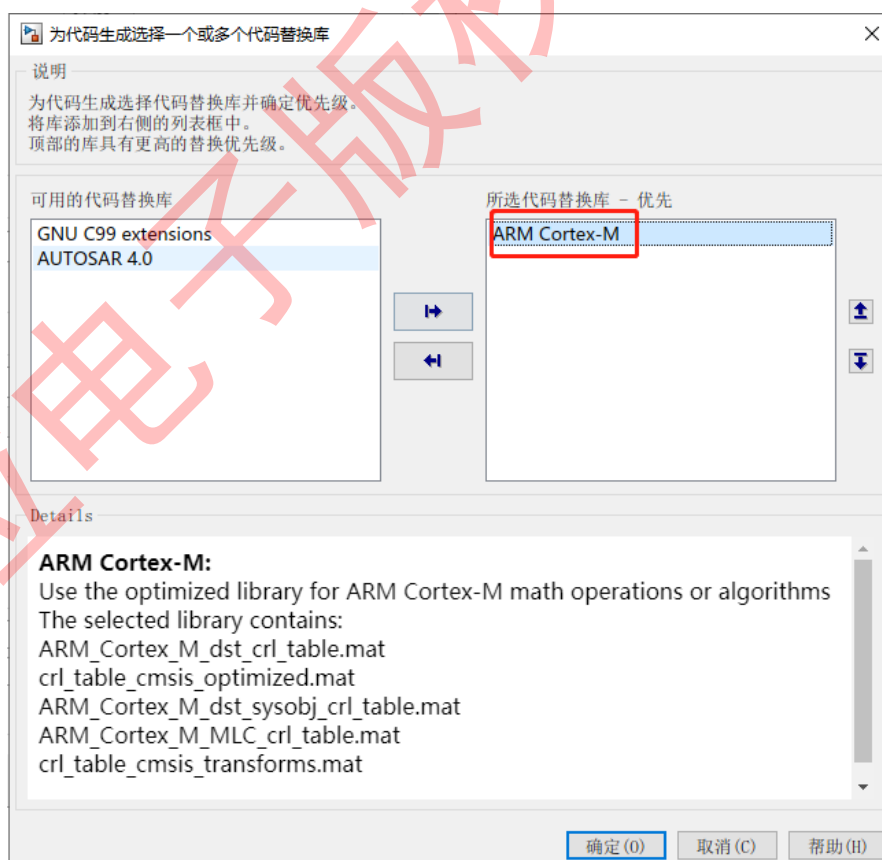
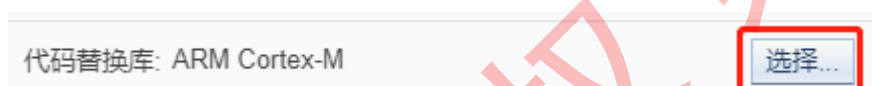
⑦ 在报告中打开创建代码生成报告；



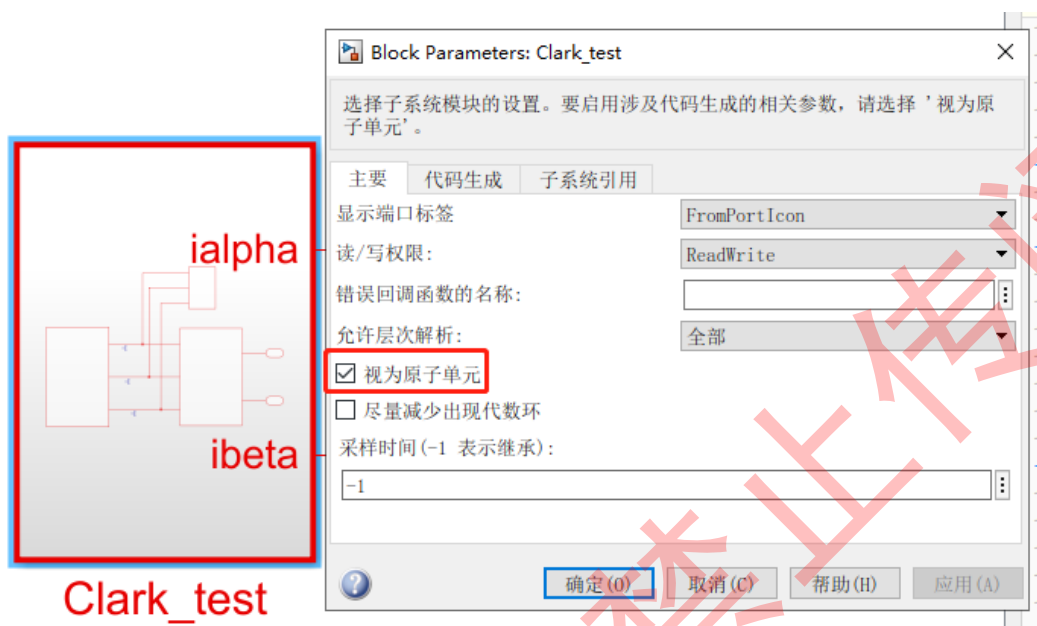
⑧ 在硬件实现栏选择 ARM CORTEX-M；



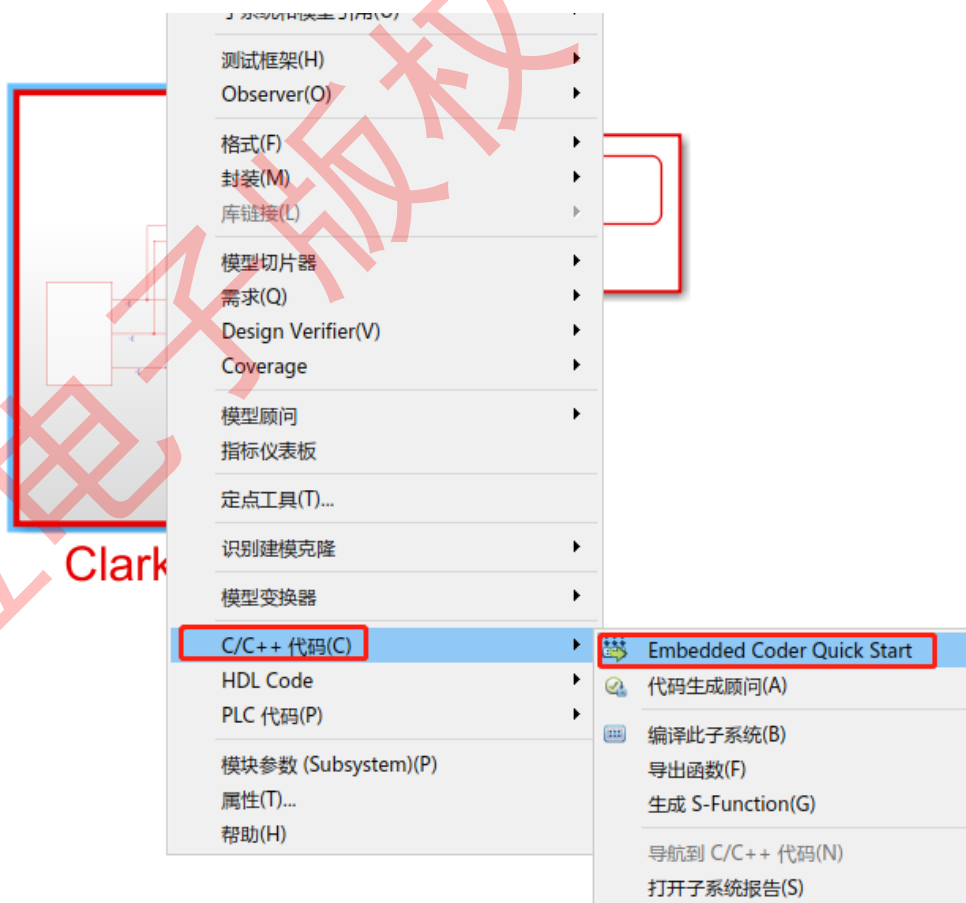
⑨ 在接口栏将代码替换库改成 ARM CORTEX-M，选择浮点数；

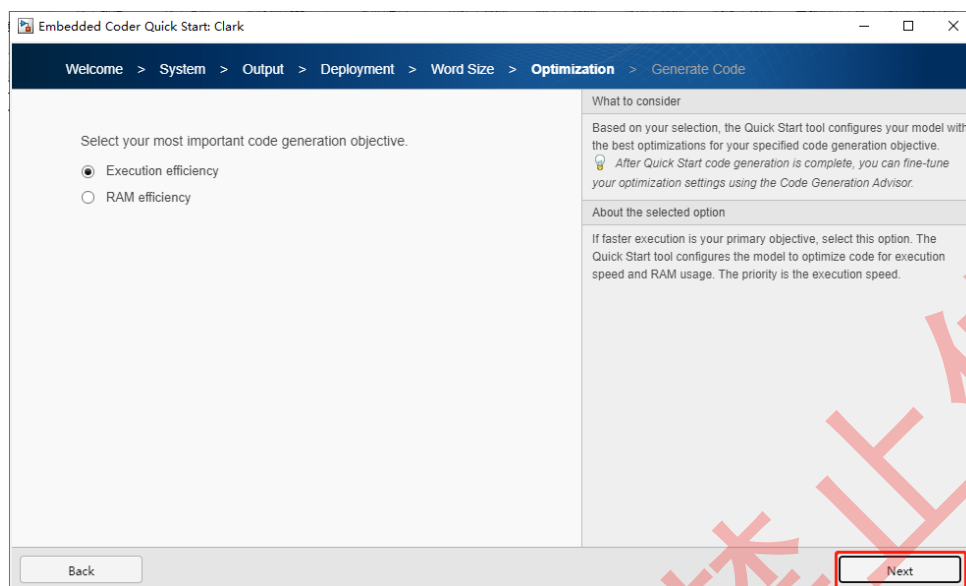


⑩ 将 Clark\_test 模型视为原子子单元；



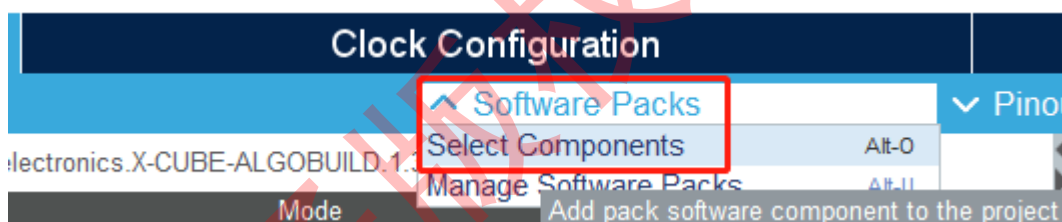
⑪ 右键该模块，选择嵌入式代码生成；生成过程一直点 NEXT 即可；



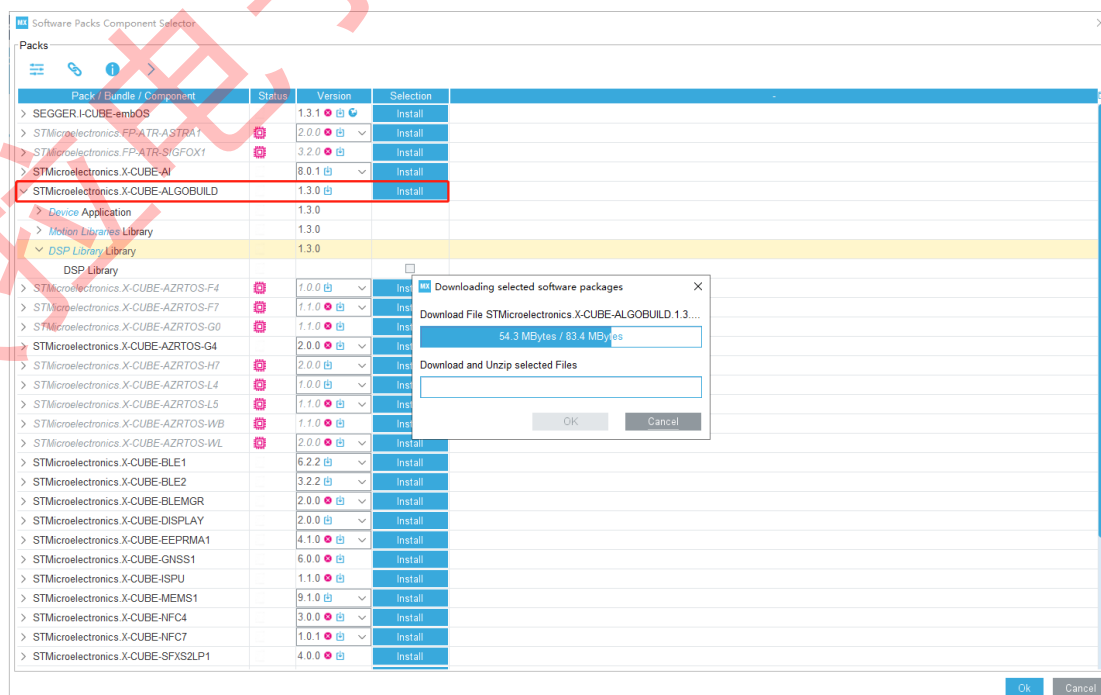


⑫ 复制第九章 TIM1 触发 ADC 电流采样工程，并打开 IOC 文件；添加 DSP 库；

安装流程详细可参考附录二：附录二：STM32CubeMX 添加 DSP 库方法



⑬ 点击安装 DSP 库；



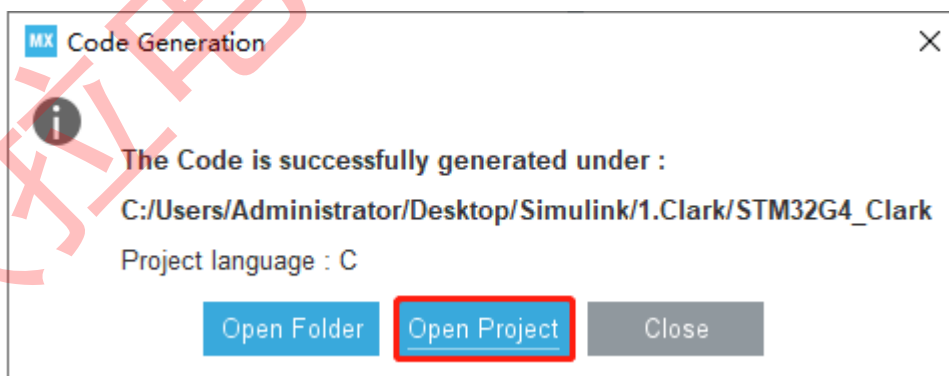
⑭ 安装完成后选中 DSP 库：

|                                       |   |       |                                     |
|---------------------------------------|---|-------|-------------------------------------|
| ▼ STMicroelectronics.X-CUBE-ALGOBUILD | ✓ | 1.3.0 |                                     |
| ▼ Device Application                  |   | 1.3.0 |                                     |
| Application                           |   |       | Not selected ▼                      |
| ▼ Motion Libraries Library            |   | 1.3.0 |                                     |
| Motion Libraries                      |   |       | <input type="checkbox"/>            |
| ▼ DSP Library Library                 | ✓ | 1.3.0 |                                     |
| DSP Library                           | ✓ |       | <input checked="" type="checkbox"/> |

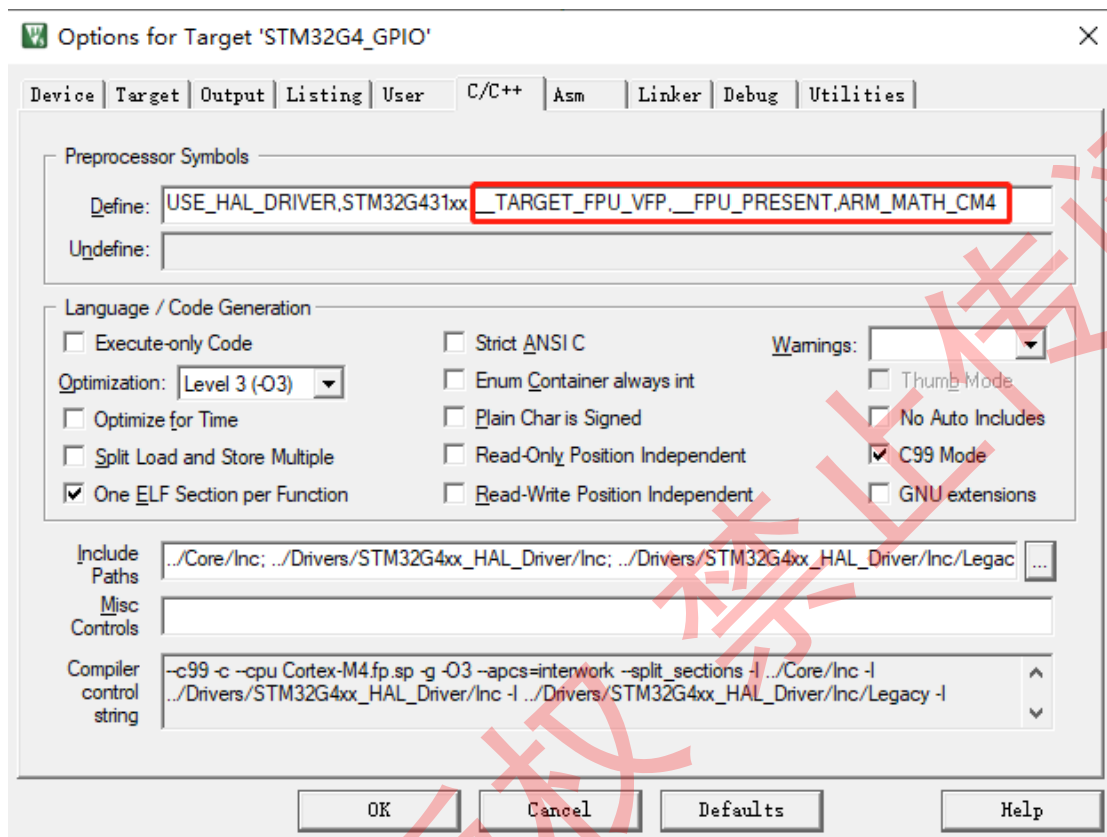
⑮ 修改 uart3 的波特率：



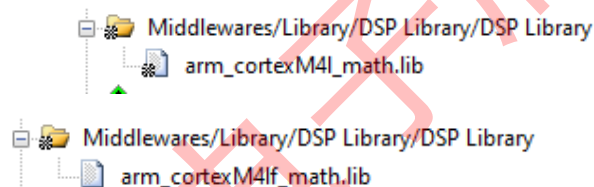
⑯ 生成代码 **GENERATE CODE**，并打开 keil 工程：



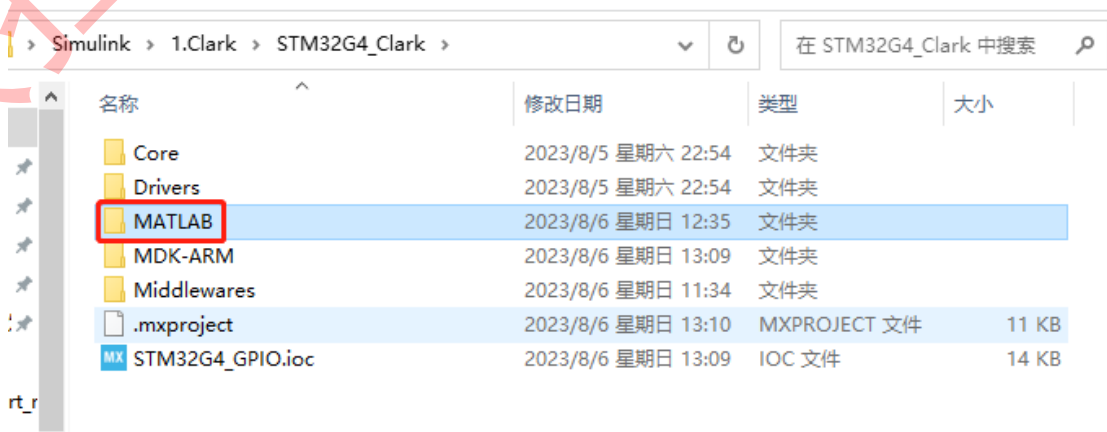
⑰ 增加宏定义：, \_TARGET\_FPU\_VFP, \_FPU\_PRESENT, ARM\_MATH\_CM4



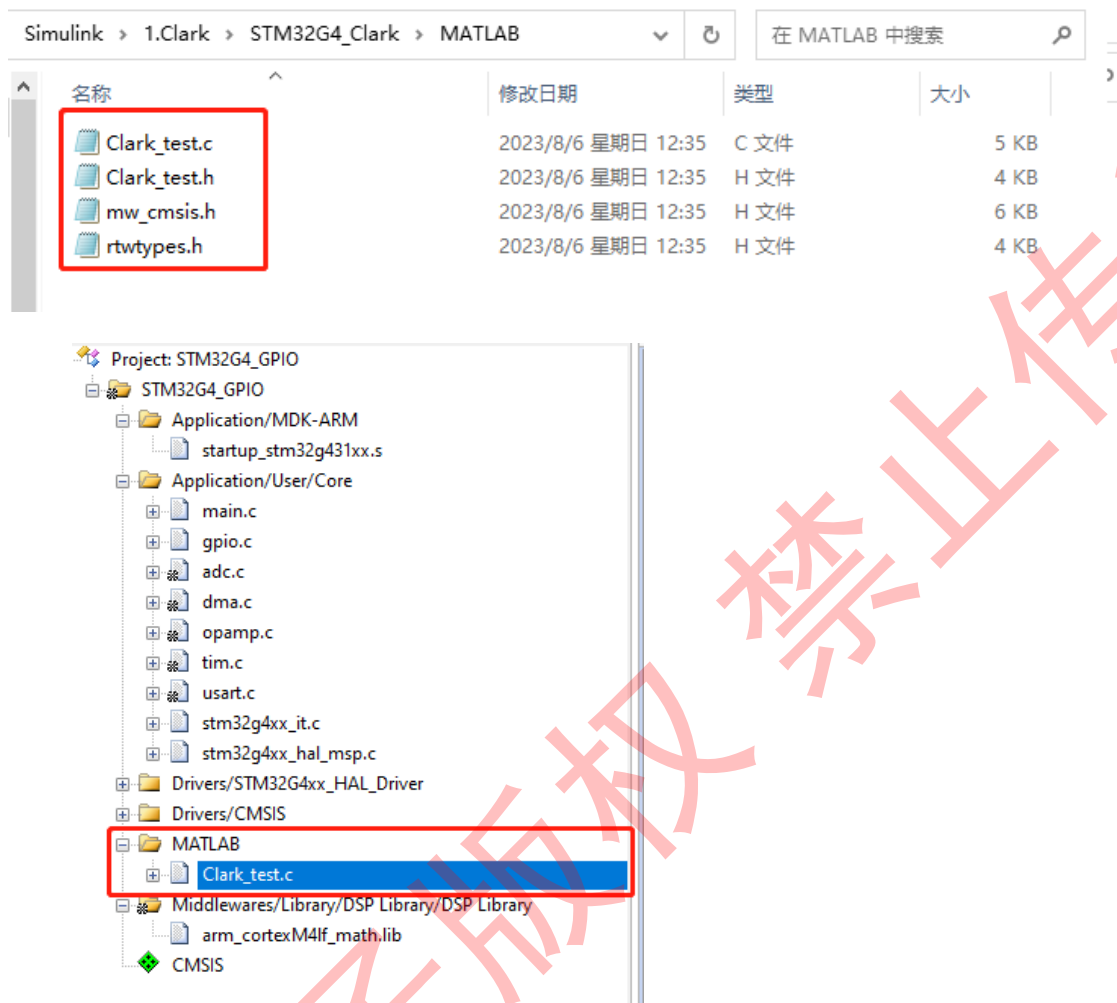
⑱ 将 DSP 库由 M4I 改为 M4LF;



⑲ 在 keil 工程中新建 MATLAB 文件夹; 并将 MATLAB 生成的代码放进来; 并添加到 Keil 工程中;







② 在 main.c 中包换头文件；注释掉 PWM1, 2, 3 通道；注释 while 循环代码；

```

29  /* Private includes -----
30  /* USER CODE BEGIN Includes */
31  #include <stdio.h>
32  #include <string.h>
33  #include "Clark_test.h"
34  /* USER CODE END Includes */

65  extern ExtY rtY;
66  extern real32_T ib;          /* '<S3>/Sin1' */
67  extern real32_T ic;          /* '<S3>/Sin2' */
68  extern real32_T ia;          /* '<S3>/Sin' */

128 // HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_1);
129 // HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_2);
130 // HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_3);
131 HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_4);
132 // HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_1);
133 // HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_2);
134 // HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_3);

```

```

141 while (1)
142 {
143     // HAL_ADC_Start(&hadc1);
144     // HAL_ADC_Start(&hadc2);
145     // temp[3] = HAL_ADC_GetValue(&hadc1);
146     // temp[3] = temp[3]*3.3f/4096;
147     // temp[4] = HAL_ADC_GetValue(&hadc2);
148     // temp[4] = temp[4]*3.3f/4096*26;
149     // HAL_Delay(10);
150     /* USER CODE END WHILE */
151
152     /* USER CODE BEGIN 3 */
153 }
154 /* USER CODE END 3 */

```

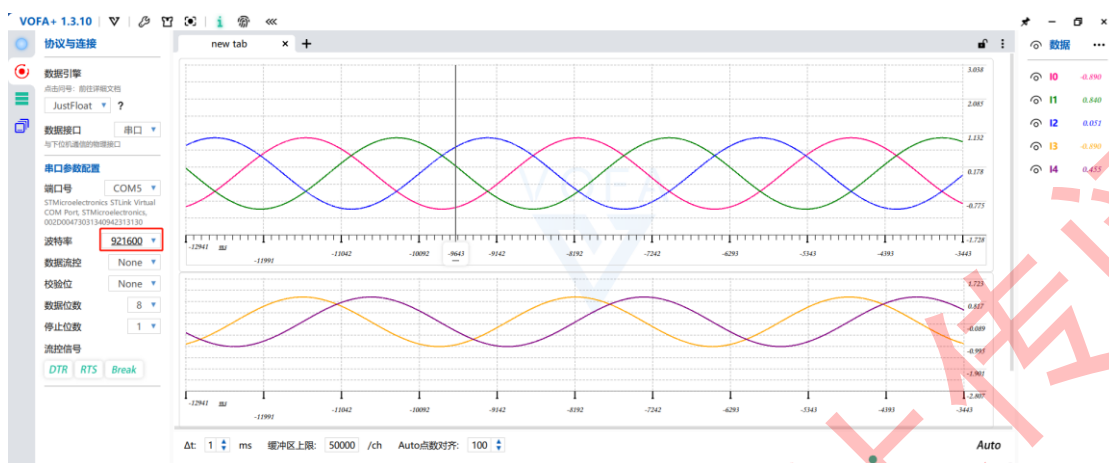
21 ADC 中断函数添加以下代码;

```

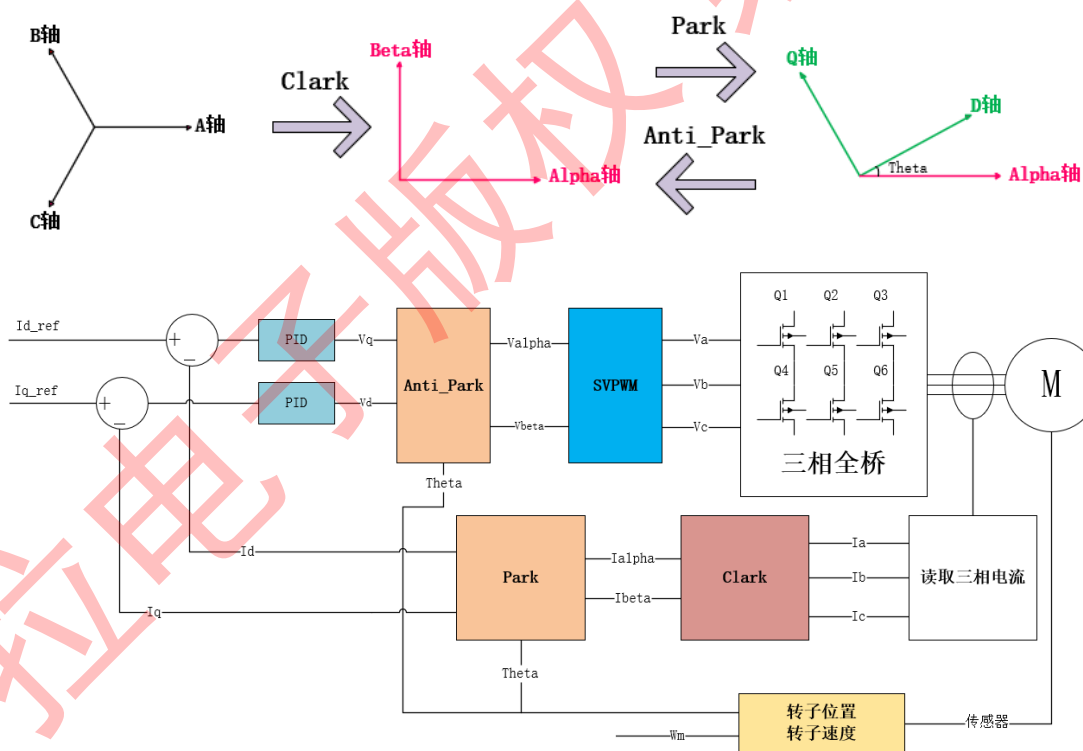
294 void HAL_ADCEx_InjectedConvCpltCallback(ADC_HandleTypeDef *hadc)
295 {
296     /* Prevent unused argument(s) compilation warning */
297     UNUSED(hadc);
298     if(hadc == &hadc1)
299     {
300         // temp[0] = hadc1.Instance->JDR1;
301         // temp[0] = (temp[0] - 0x7ef)*0.02197f;
302         // temp[1] = hadc1.Instance->JDR2;
303         // temp[1] = (temp[1] - 0x7f5)*0.02197f;
304         // temp[2] = hadc2.Instance->JDR1;
305         // temp[2] = (temp[2] - 0x7e8)*0.02197f;
306         // TIM1->CCR1 = 2000;
307         // TIM1->CCR2 = 4000;
308         // TIM1->CCR3 = 6000;
309         Clark_test_step();
310         temp[3] = rtY.ialpha;
311         temp[4] = rtY.ibeta;
312         temp[0] = ia;
313         temp[1] = ib;
314         temp[2] = ic;
315         memcpy(tempData, (uint8_t *)&temp, sizeof(temp));
316         HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 6*4);
317     }

```

22 编译并下载代码; 打开 VOFA 上位机;

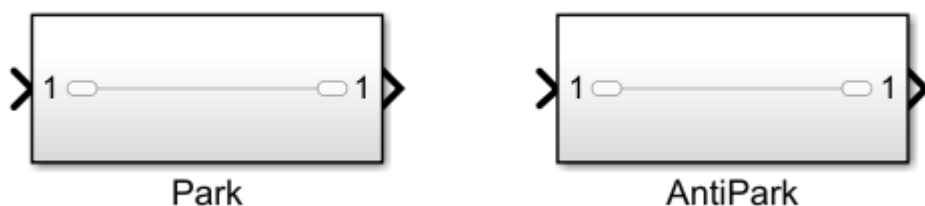


## 第十四章：Park 及反 Park 变换



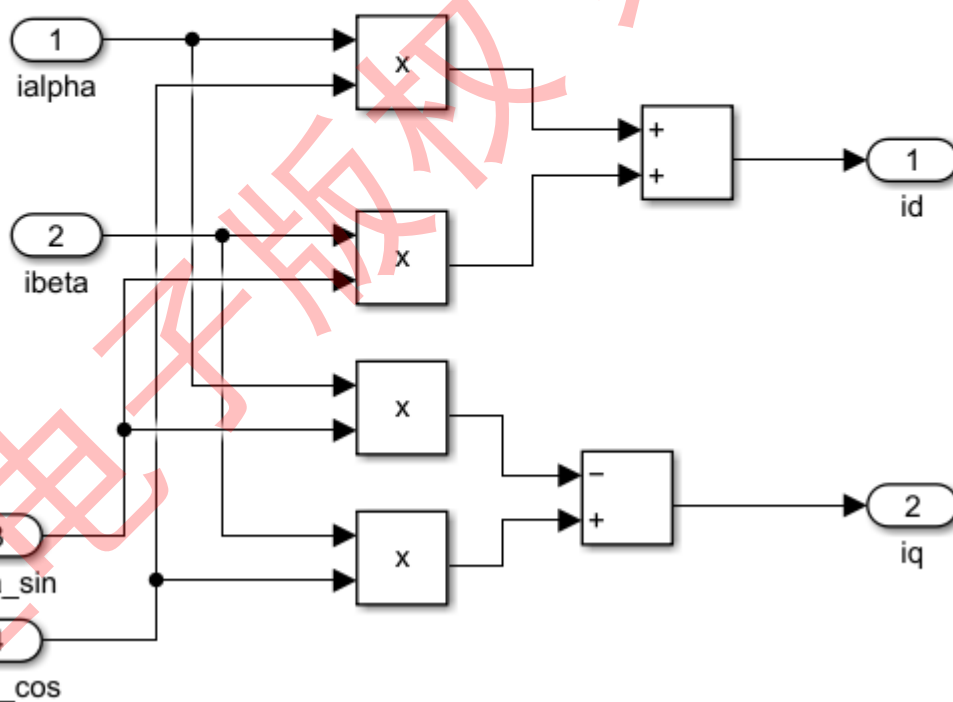
### 1. Park 及反 Park 变换建模

- ① 在 Clark 变换的基础上我们增加，Park 及反 Park 变换模块；



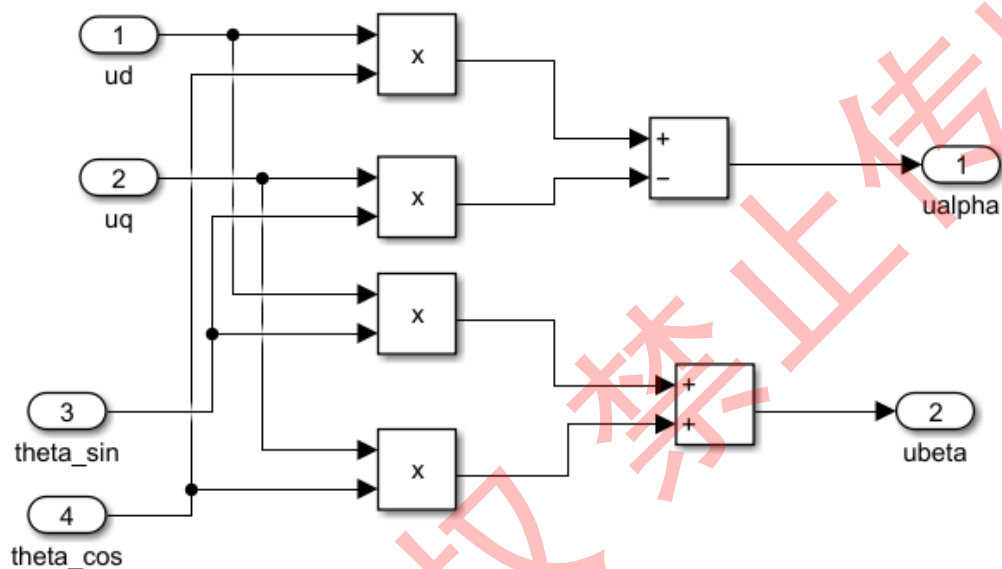
② 根据 Park 变换公式，搭建 Park 变换模型；

```
id = ialpha*cos(theta)+ibeta*sin(theta);
iq = -ialpha*sin(theta)+ibeta*cos(theta);
```

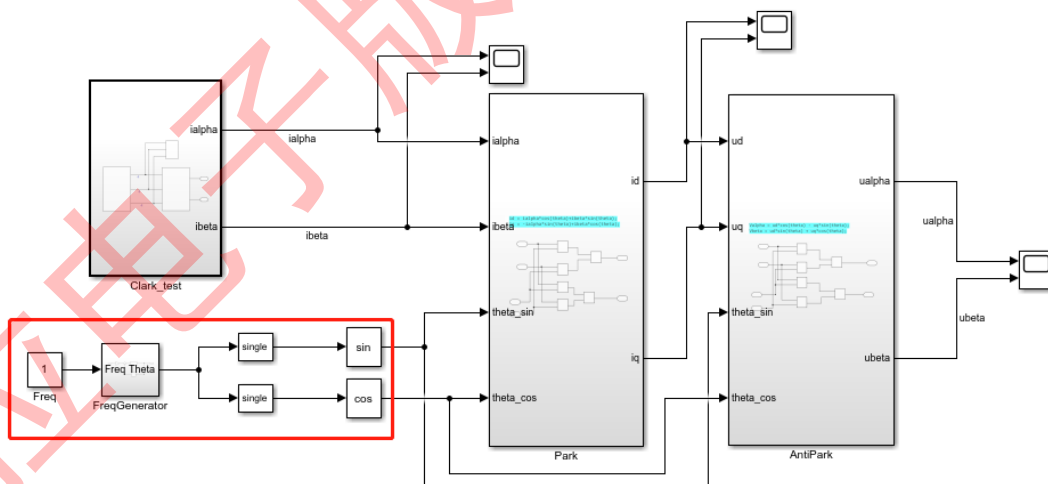


③ 根据 AntiPark 变换公式，搭建 AntiPark 变换模型；

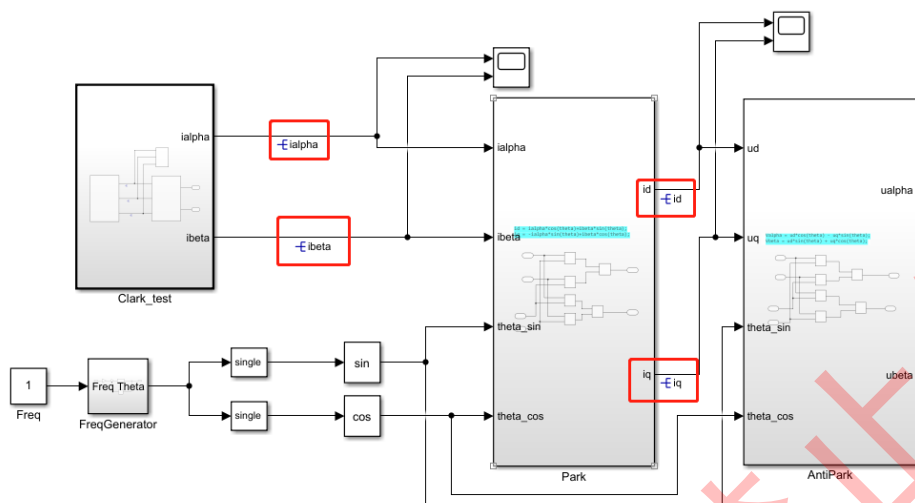
```
Valpha = ud*cos(theta) - uq*sin(theta);
Vbeta = ud*sin(theta) + uq*cos(theta);
```



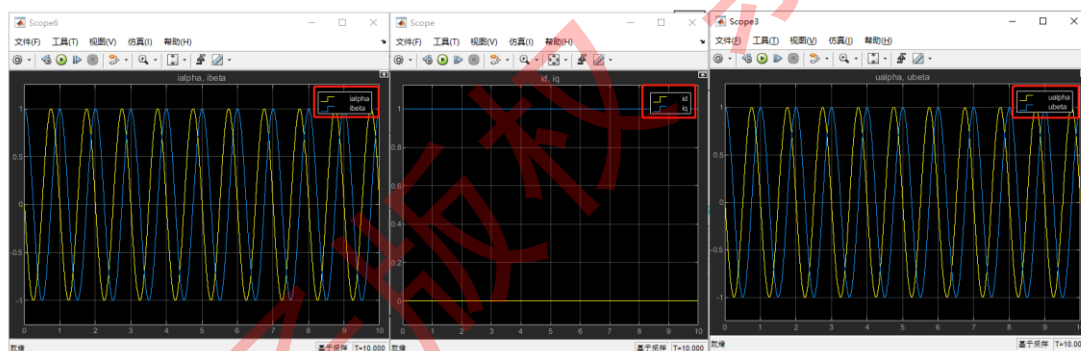
④ 增加 theta 角度生成模块；与 Clark 角度保持同步；



⑤ 将 ialpha, ibeta, id 和 iq 配置成全局 Simulink signal，便于程序代码观测；

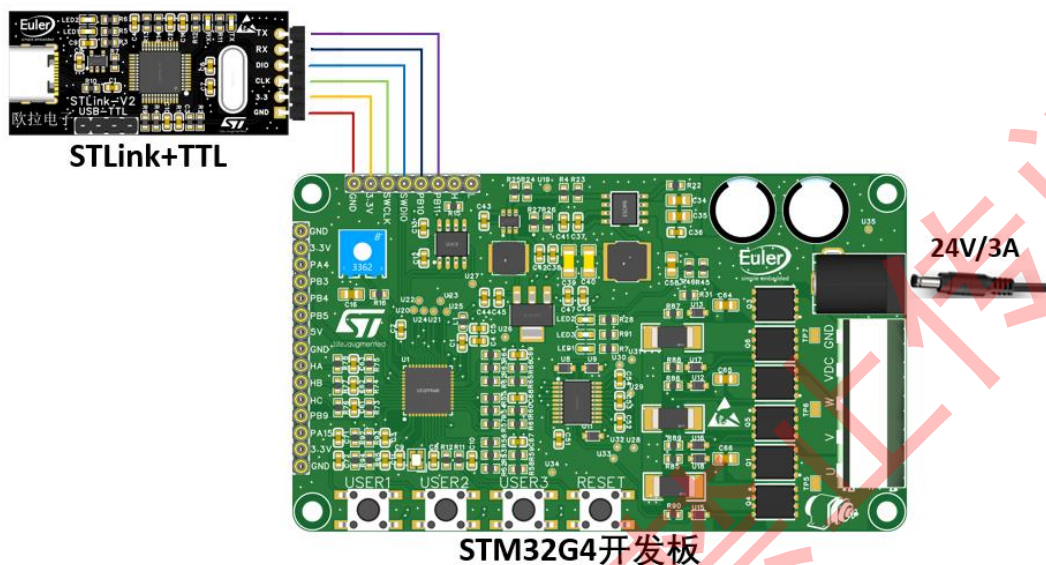


⑥ 点击仿真，观察波形；

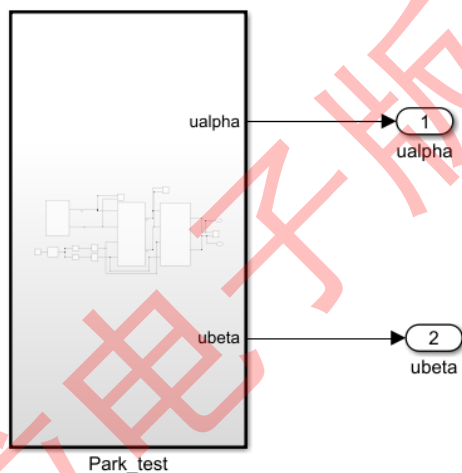


## 2. 实验现象

按以下图片接线，并讲 USB 转 UART 模块连接至电脑：



- ① 选中所有模型，点击创建子系统，将其封装成同一个模块；并定义为原子子单元；



- ② 右键生成嵌入式代码；
- ③ 复制 Clark 变换的工程代码，将 MATLAB 中代码替换成刚生成的代码；

| Code > STM32G4_Park > MATLAB |                    |      |      |  | 在 MATLAB 中搜索 |  |
|------------------------------|--------------------|------|------|--|--------------|--|
| 名称                           | 修改日期               | 类型   | 大小   |  |              |  |
| mw_cmsis.h                   | 2023/8/6 星期日 13:51 | H 文件 | 6 KB |  |              |  |
| Park_test.c                  | 2023/8/6 星期日 13:51 | C 文件 | 8 KB |  |              |  |
| Park_test.h                  | 2023/8/6 星期日 13:51 | H 文件 | 6 KB |  |              |  |
| rtwtypes.h                   | 2023/8/6 星期日 13:51 | H 文件 | 4 KB |  |              |  |



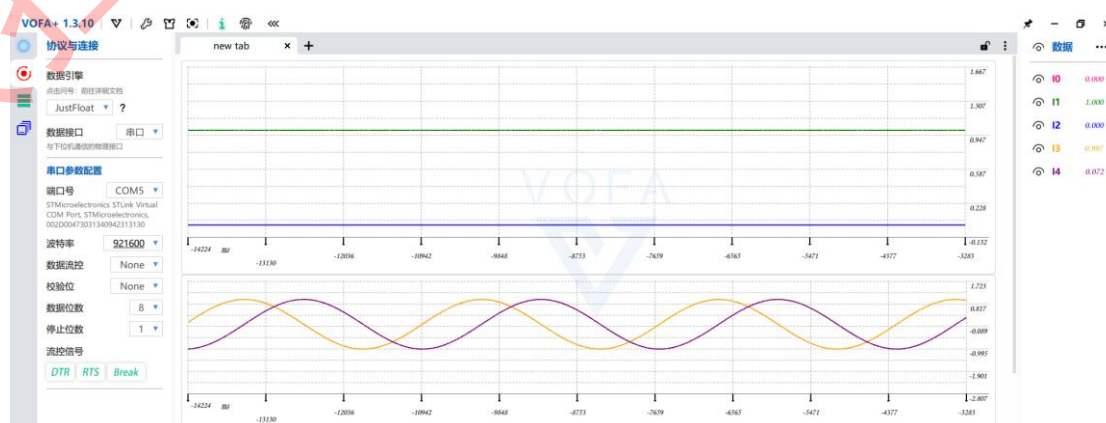
④ 修改 keil 工程；

```

29  /* Private includes -----
30  /* USER CODE BEGIN Includes */
31  #include <stdio.h>
32  #include <string.h>
33  #include "Park_test.h"
34  /* USER CODE END Includes */
35
62
63  float temp[5];
64  static uint8_t tempData[24] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0x80,0x7f};
65  extern ExtY rtY;
66  extern real32_T id;          /* '<S5>/Add' */
67  extern real32_T iq;          /* '<S5>/Add1' */
68  extern real32_T ialpha;      /* '<S6>/Add1' */
69  extern real32_T ibeta;       /* '<S6>/Gain2' */
70  /* USER CODE END PV */
71
294
295  void HAL_ADCEx_InjectedConvCpltCallback(ADC_HandleTypeDef *hadc)
296  {
297      /* Prevent unused argument(s) compilation warning */
298      UNUSED(hadc);
299      if(hadc == &hadc1)
300      {
301          // temp[0] = hadc1.Instance->JDR1;
302          // temp[0] = (temp[0] - 0x7ef)*0.02197f;
303          // temp[1] = hadc1.Instance->JDR2;
304          // temp[1] = (temp[1] - 0x7f5)*0.02197f;
305          // temp[2] = hadc2.Instance->JDR1;
306          // temp[2] = (temp[2] - 0x7e8)*0.02197f;
307          // TIM1->CCR1 = 2000;
308          // TIM1->CCR2 = 4000;
309          // TIM1->CCR3 = 6000;
310          Park_test_step();
311          temp[3] = rtY.ualpha;
312          temp[4] = rtY.ubeta;
313          temp[0] = id;
314          temp[1] = iq;
315          memcpy(tempData, (uint8_t *)&temp, sizeof(temp));
316          HAL_UART_Transmit_DMA(&uart3, (uint8_t *)tempData, 6*4);
317      }
318  }

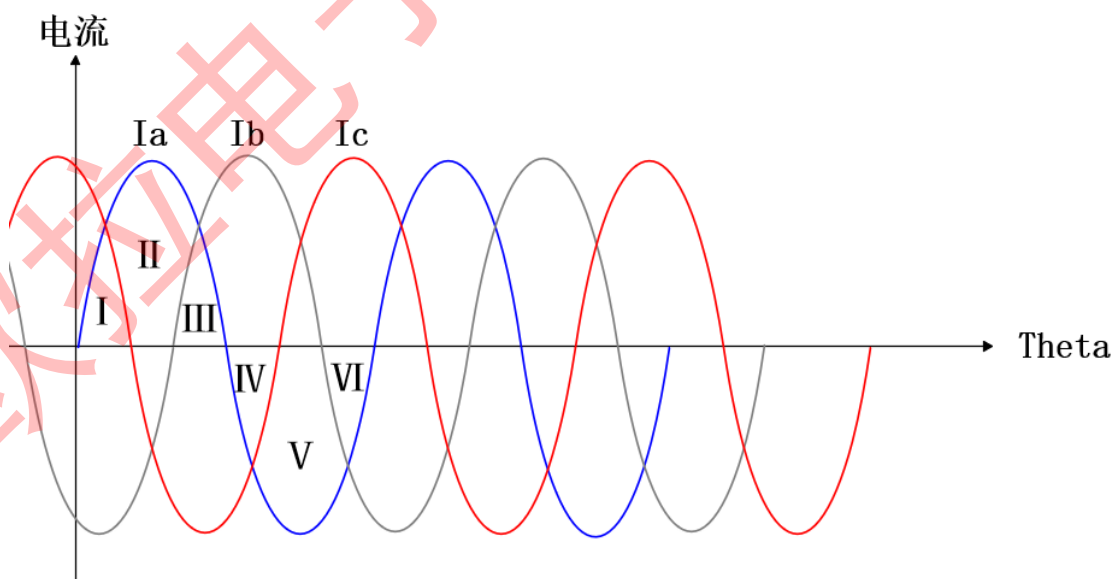
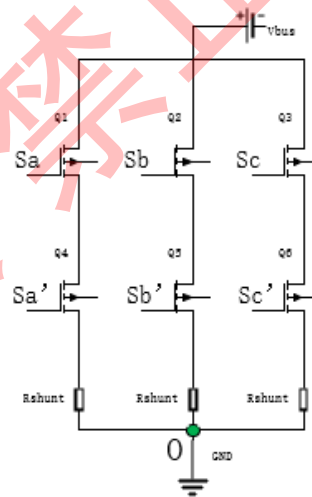
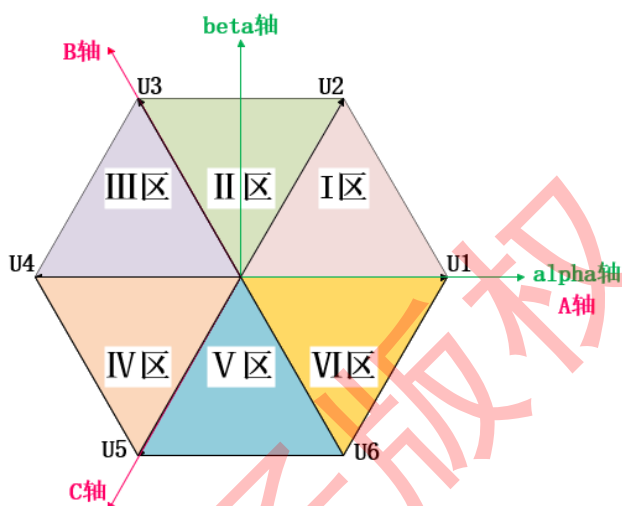
```

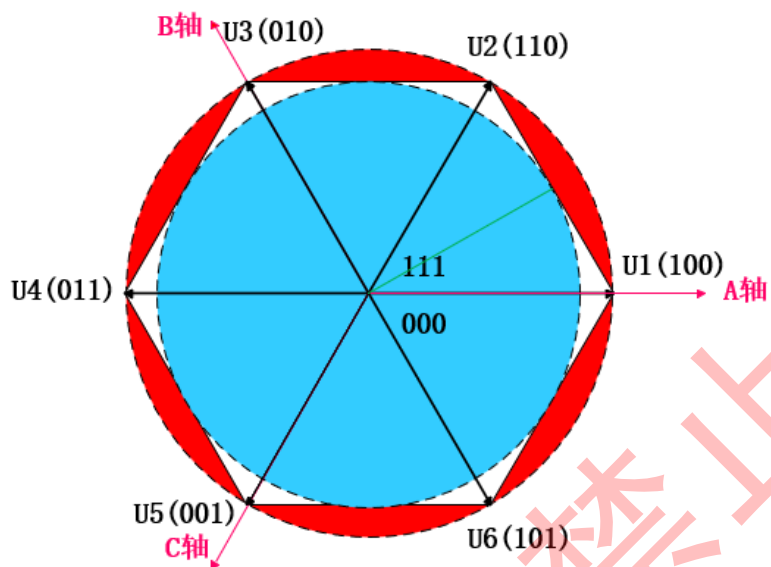
⑤ 编译并下载代码；打开 VOFA 上位机；



## 第十五章：SVPWM,VF 开环强托电机

SVPWM 控制策略是依据变流器空间电压（电流）矢量切换来控制变流器的种新颖思路和控制策略，其主要思想在于抛弃原有的 SPWM 算法，采用逆变器空间电压矢量的切换以获得准圆形旋转磁场，从而在不高的开关频率条件下，使得交流电机获得较 SPWM 算法更好的控制性能。

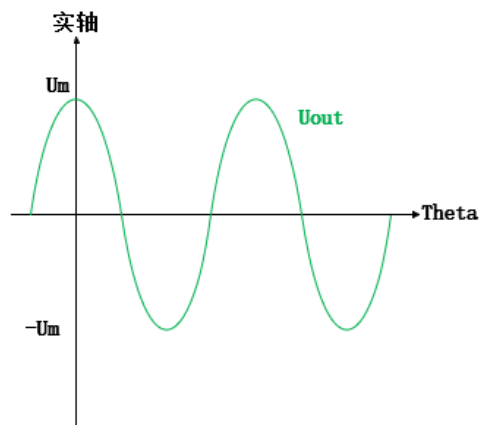
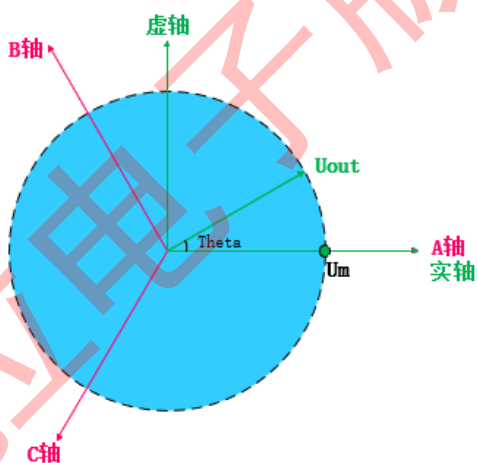




$$U_{out} = U_1 * \frac{\sqrt{3}}{2}$$

$$U_1 = \frac{2}{3} U_{dc}$$

$$U_{out} = \frac{\sqrt{3}}{3} U_{dc}$$

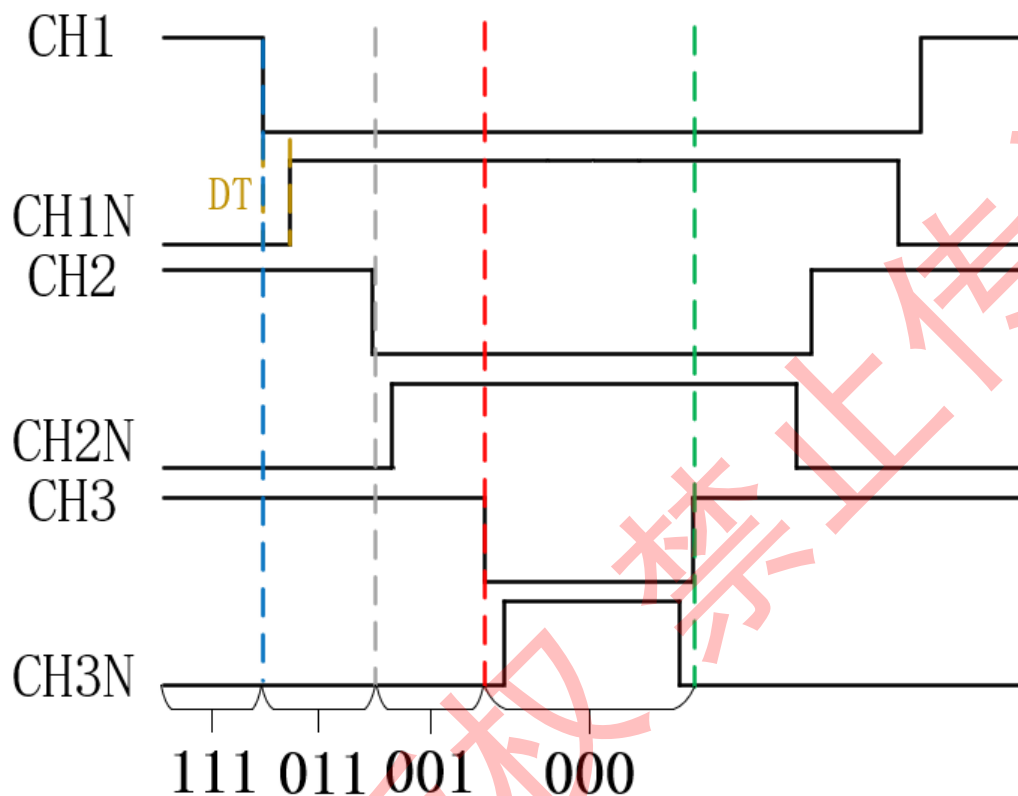


欧拉公式：

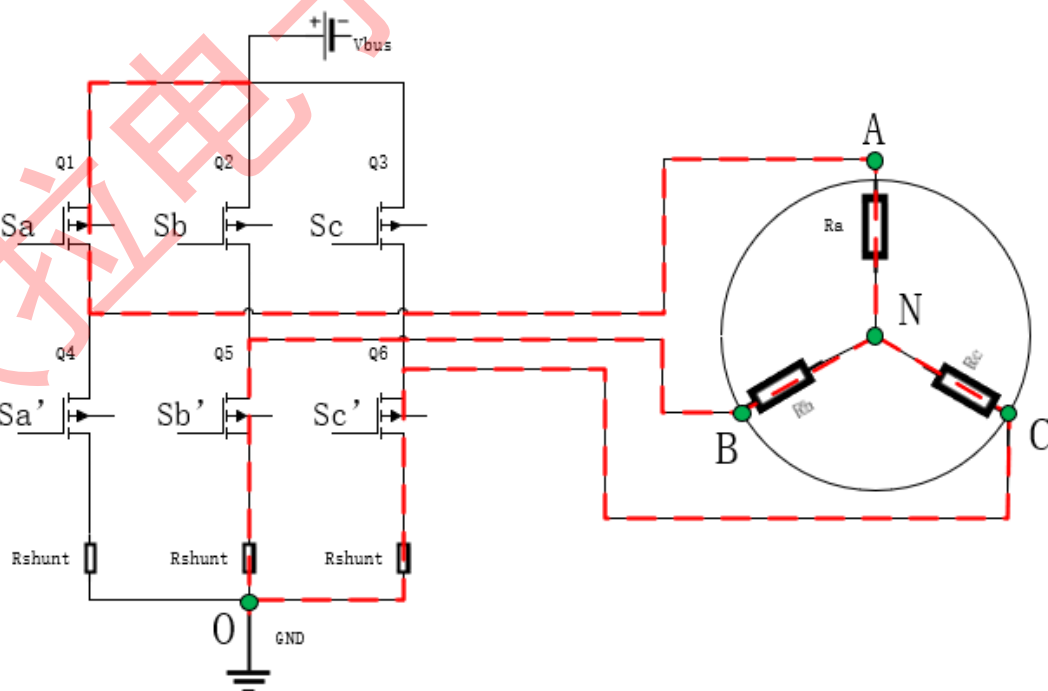
$$e^{i*theta} = \cos(theta) + i * \sin(theta)$$

其中，实部：

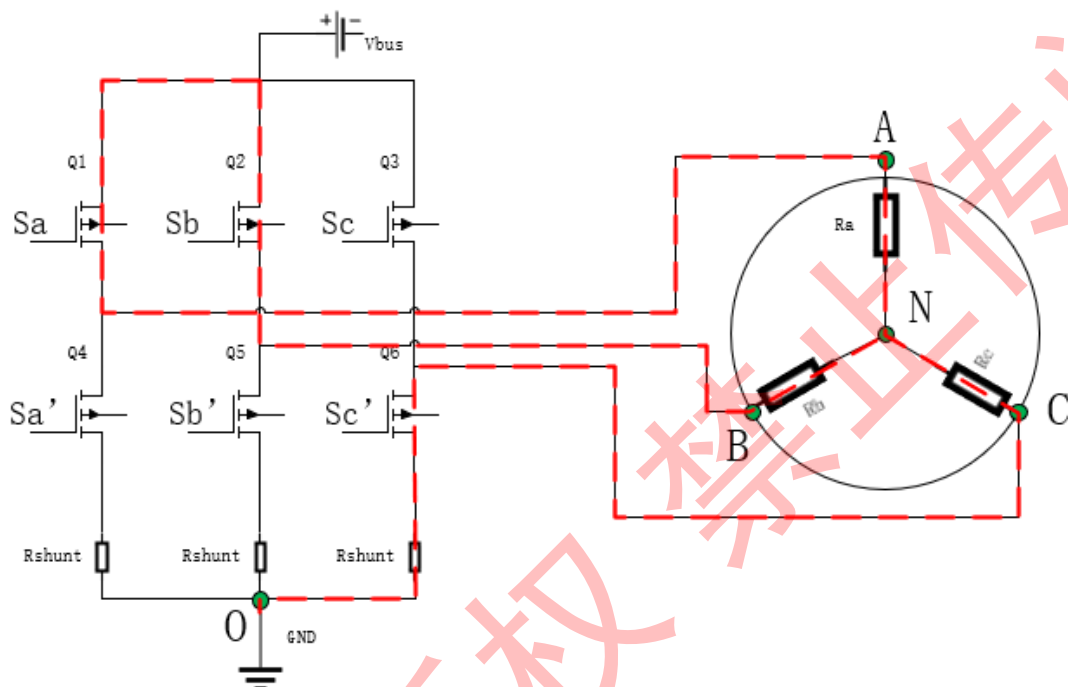
$$U_{out} = U_m * \cos(theta)$$



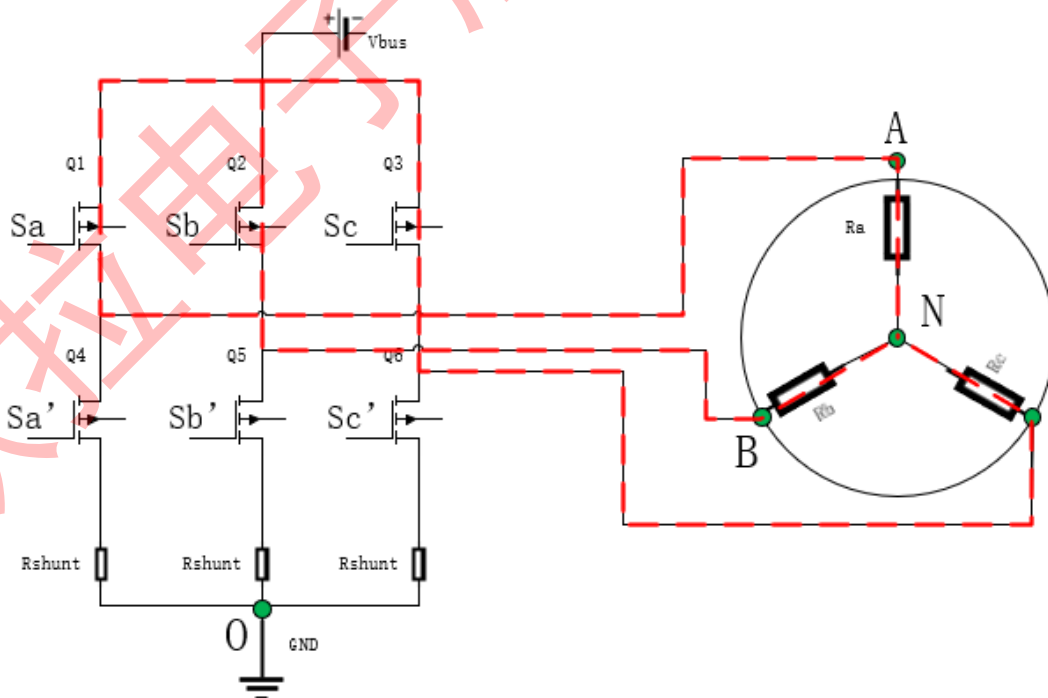
U1 (100)



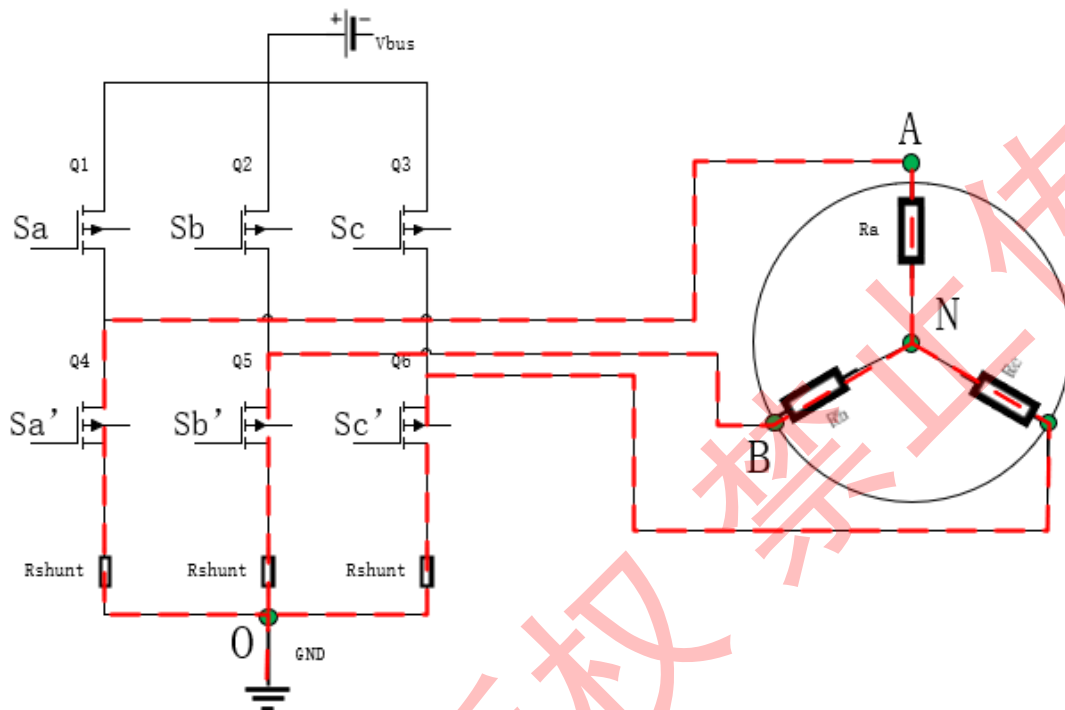
## U2(110)



## 111(电感续流)



## 000 (电感续流)



| Sa | Sb | Sc | Van              | Vbn              | Vcn              | Vout                                   |
|----|----|----|------------------|------------------|------------------|----------------------------------------|
| 0  | 0  | 0  | 0                | 0                | 0                | 0                                      |
| 1  | 0  | 0  | $2U_d$<br>$c/3$  | -<br>$U_{dc}/3$  | -<br>$U_{dc}/3$  | $\frac{2}{3}U_{dc}$                    |
| 0  | 1  | 0  | -<br>$U_{dc}/3$  | $2U_d$<br>$c/3$  | -<br>$U_{dc}/3$  | $\frac{2}{3}U_{dc}e^{j\frac{2\pi}{3}}$ |
| 1  | 1  | 0  | $U_{dc}$<br>$/3$ | $U_{dc}$<br>$/3$ | -<br>$2U_{dc}/3$ | $\frac{2}{3}U_{dc}e^{j\frac{\pi}{3}}$  |
| 0  | 0  | 1  | -<br>$U_{dc}/3$  | -<br>$U_{dc}/3$  | $2U_d$<br>$c/3$  | $\frac{2}{3}U_{dc}e^{j\frac{4\pi}{3}}$ |

|   |   |   |                      |                      |                    |                                        |
|---|---|---|----------------------|----------------------|--------------------|----------------------------------------|
| 1 | 0 | 1 | $\frac{U_{dc}}{3}$   | $-\frac{2U_{dc}}{3}$ | $\frac{U_{dc}}{3}$ | $\frac{2}{3}U_{dc}e^{j\frac{5\pi}{3}}$ |
| 0 | 1 | 1 | $-\frac{2U_{dc}}{3}$ | $\frac{U_{dc}}{3}$   | $\frac{U_{dc}}{3}$ | $\frac{2}{3}U_{dc}e^{j\pi}$            |
| 1 | 1 | 1 | 0                    | 0                    | 0                  | 0                                      |

## 1. SVPWM 建模

为了提高直流电压的利用率，考虑在调制波信号中注入零序分量，其调制波表达式为：

$$\begin{cases} V_{am} = V_{m1} \sin \omega t + V_0 \\ V_{bm} = V_{m1} \sin \left( \omega t - \frac{2}{3} \pi \right) + V_0 \\ V_{cm} = V_{m1} \sin \left( \omega t + \frac{2}{3} \pi \right) + V_0 \end{cases}$$

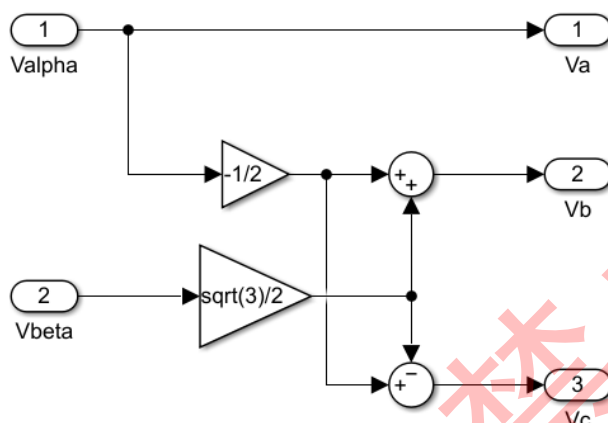
$$V_0 = -\frac{1}{2}(V_{\max} + V_{\min})$$

基于以上公式，在 simulink 中建立模型；打开上一章模型；

- ① 将 AntiPark 变换得到的 Valpha 及 Vbeta 电压，通过 AntiClark 变换得到 Va, Vb, Vc;

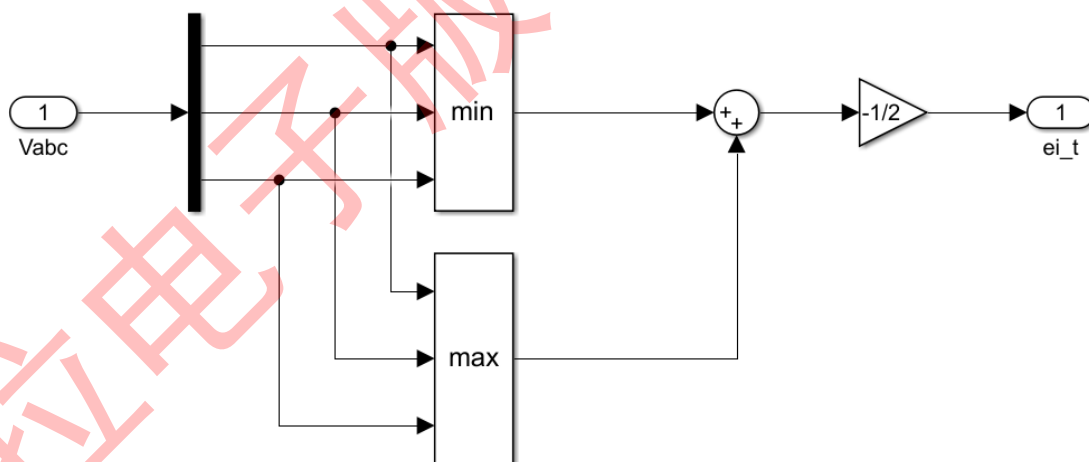


$$\begin{aligned} V_a &= V_{\alpha}; \\ V_b &= V_{\alpha} * (-1/2) + V_{\beta} * \sqrt{3}/3 \\ V_c &= V_{\alpha} * (-1/2) - V_{\beta} * \sqrt{3}/3 \end{aligned}$$

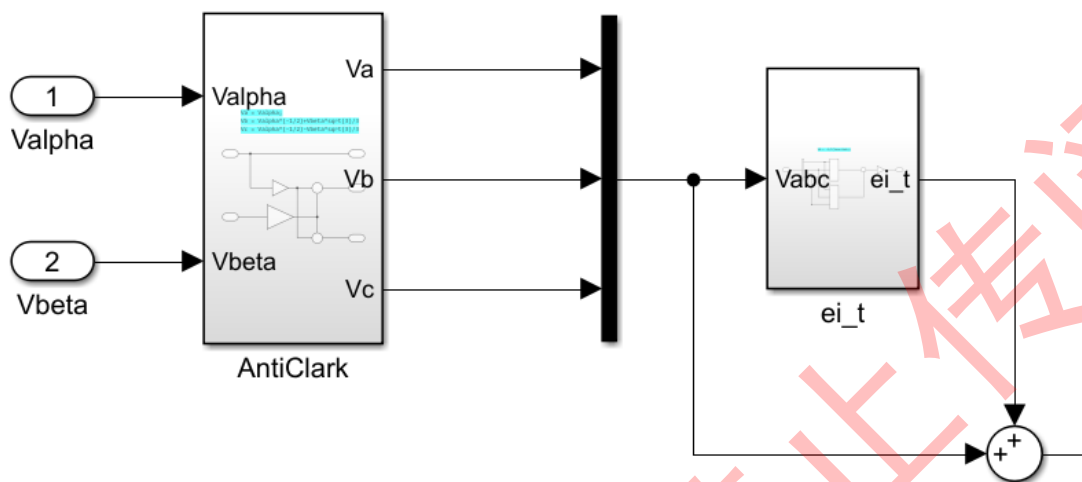


② 找到  $V_a$ ,  $V_b$ ,  $V_c$  中的最大值及最小值，计算  $V_0$ ;

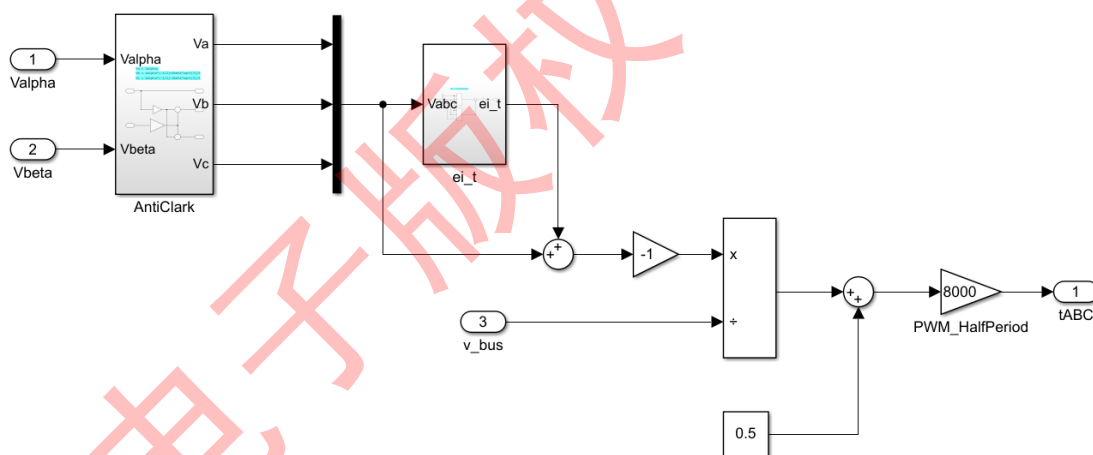
$$V_0 = -1/2(V_{\max} + V_{\min})$$



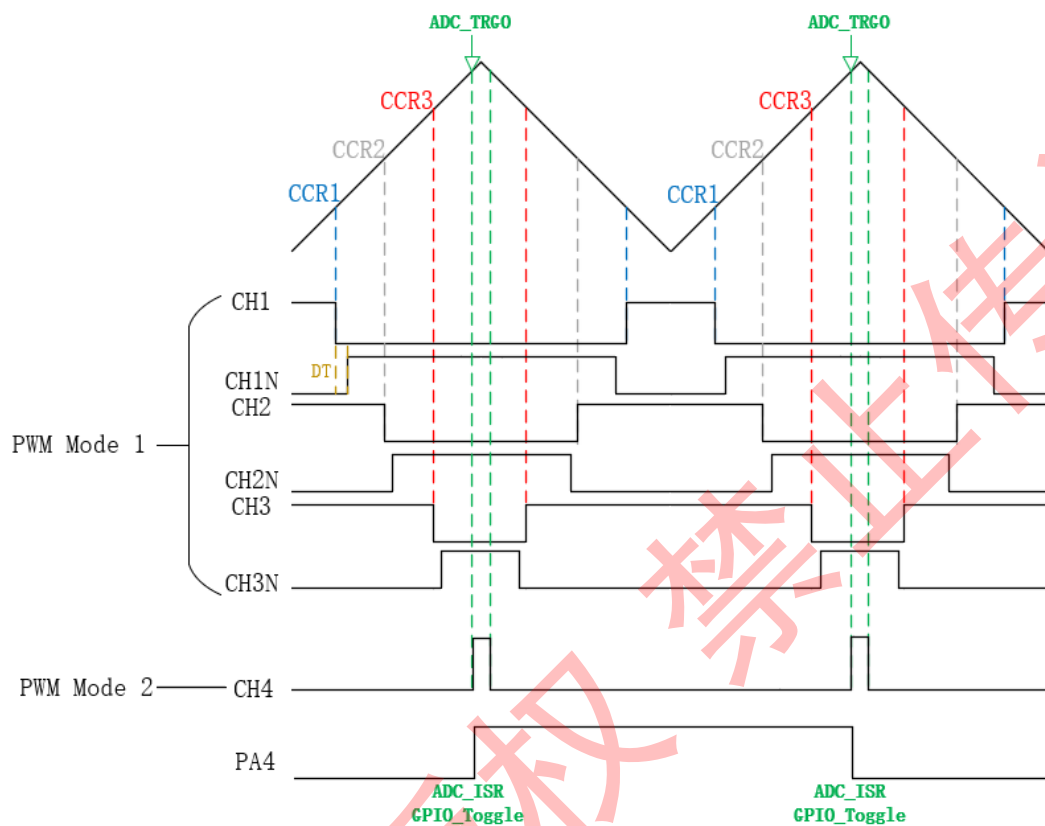
③  $V_a$ ,  $V_b$ ,  $V_c$  加  $V_0$ ，得到调制波;



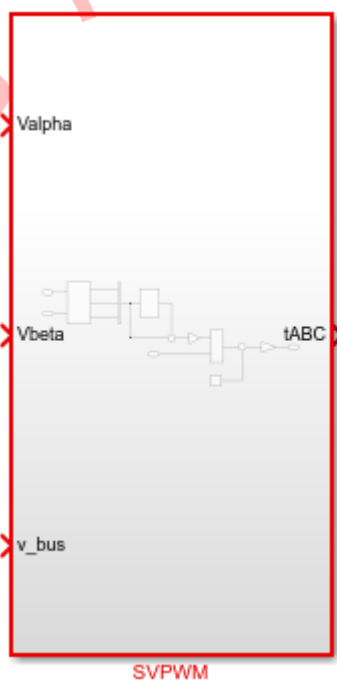
- ④ 计算 a,b,c 三相电压占空比；使用三相调制波除以母线电压  $V_{dc}$ ，然后乘以 MCU 定时器的计数值，即得到  $T_a$ ,  $T_b$ ,  $T_c$  的三相电压切换时间；



模型中计算出来的三相调制波电压乘-1；乘-1的原因与的 STM32G4 的 TIM1 的发波模式有关，如下图，CH1, 2, 3 为 PWM mode 1，即 TIM 计数值大于 CCR 值，CH1 为低电平，而我们此处计算出来的三相调制波电压是对应 PWM mode 2，与 mode 1 刚好相反，故此处需要乘-1.



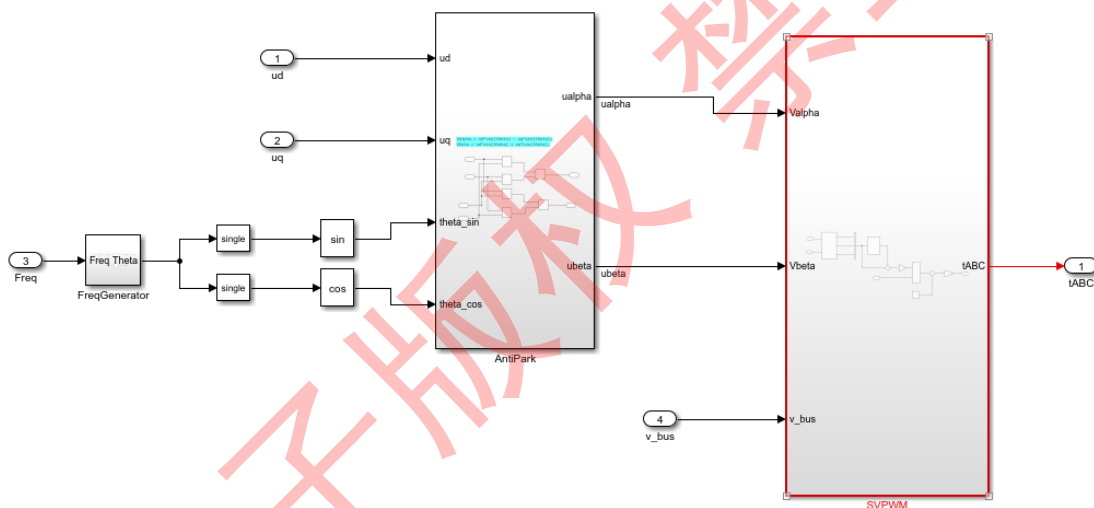
⑤ SVPWM 算法模块建立完成;



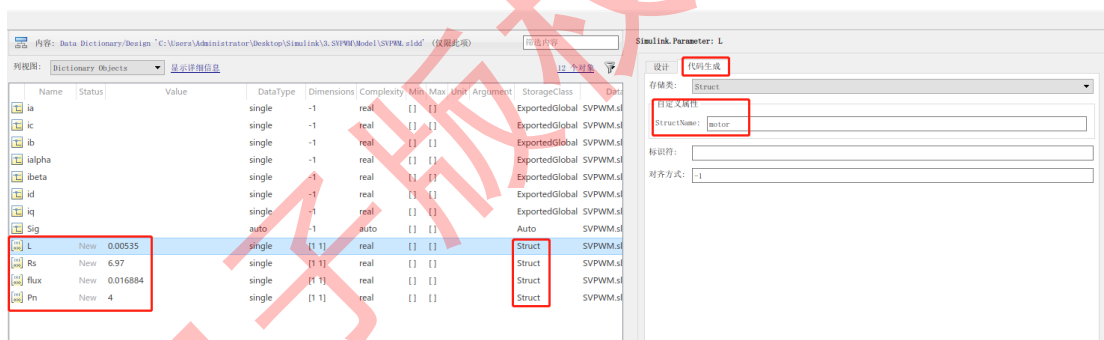
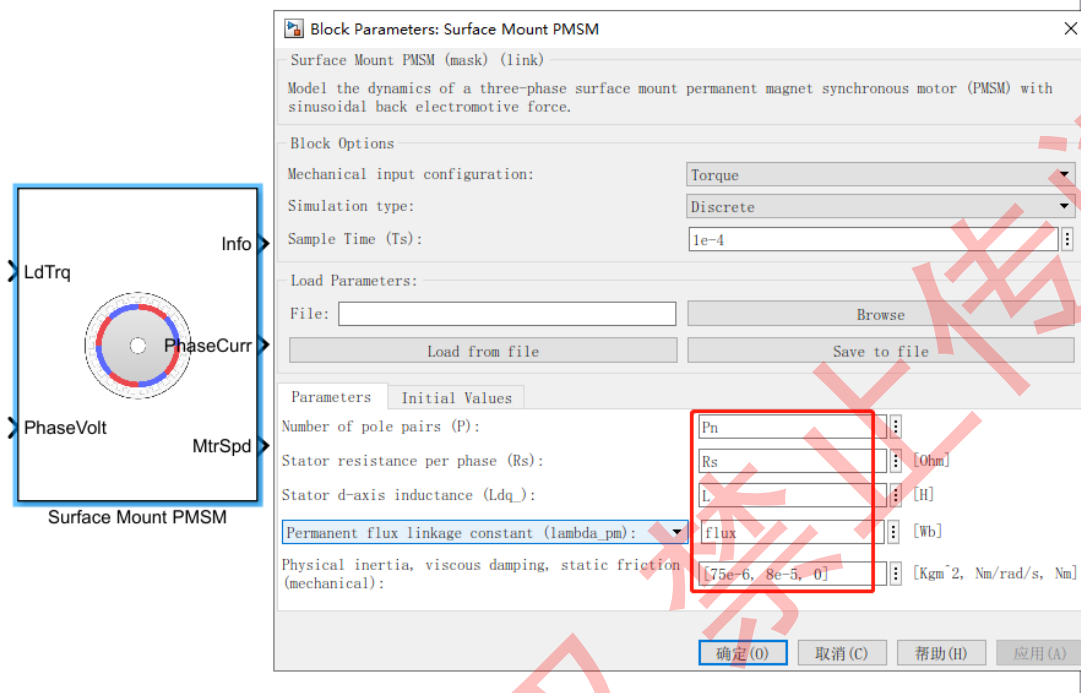
## 2. VF 模式建模

结合上一章的 AntiPark 模块及 SVPWM 模块即可实现 VF 模式来驱动电机开环转动；

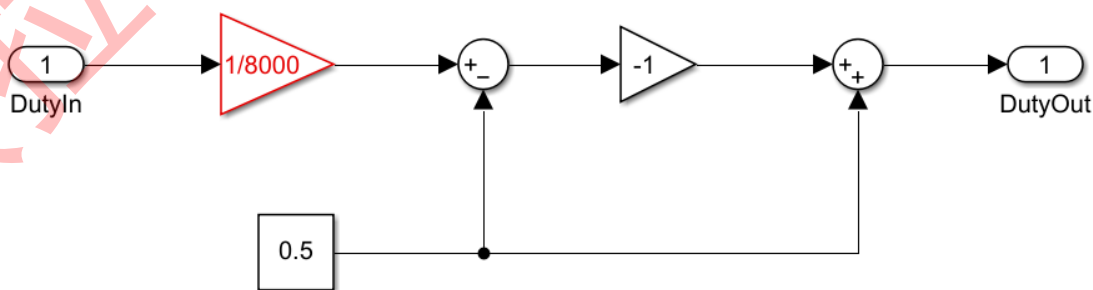
- ① 将 AntiPark 模块及 SVPWM 模型按如下连接；其中输入的  $u_d$ ， $u_q$ ，Freq 及  $V_{bus}$  均为标量，可在软件中自行定义其值；输出的  $T_a$ ， $T_b$ ， $T_c$  直接写入 MCU TIMER 的比较值寄存器；



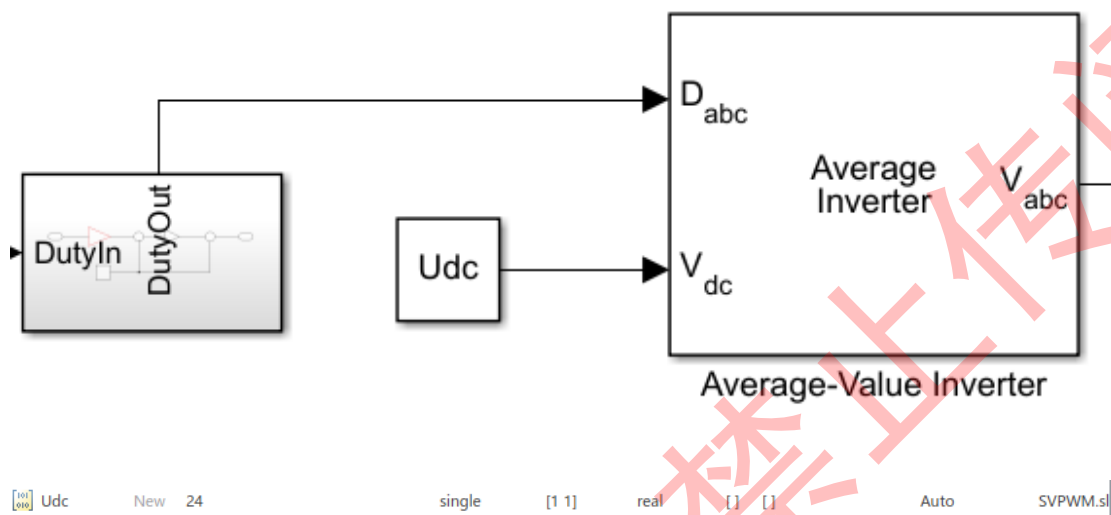
- ② 增加表贴式 PMSM 电机模型，写入电机参数及采样时间；并在数据字典中增加电机相关参数，将增加的电机参数的类定义为 struct，并将 struct 命名为 motor；该方法生成的代码为结构体数据，便于后期更改电机参数；



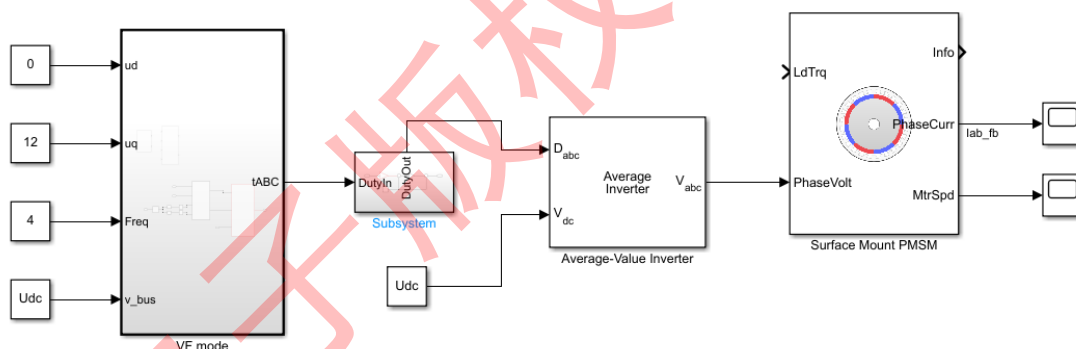
③ 增加一个将 SVPWM 输出的 Ta, Tb, Tc 时间转化为一个占空比时间的模块;



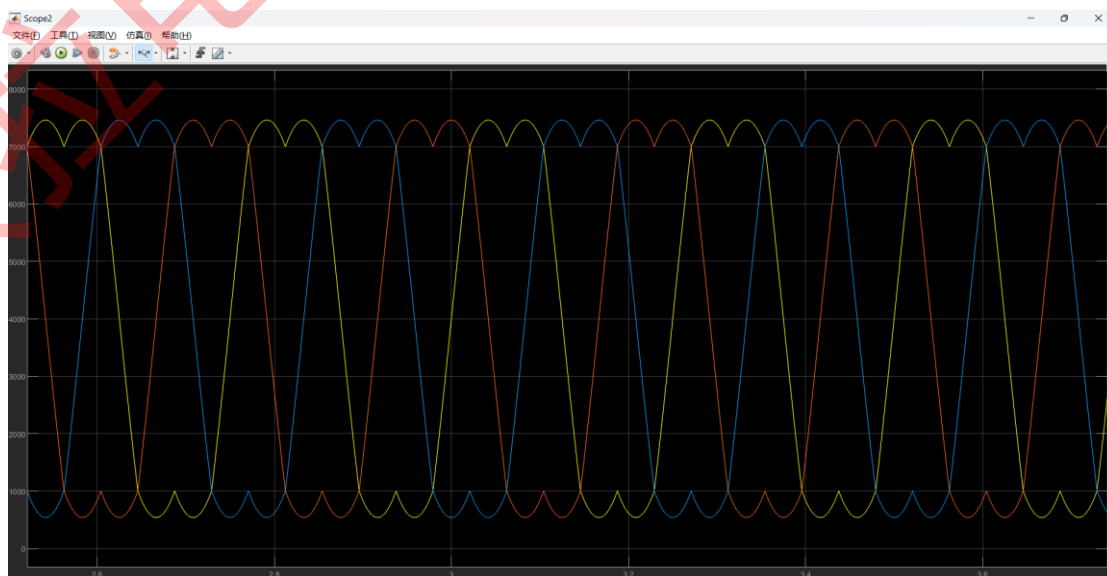
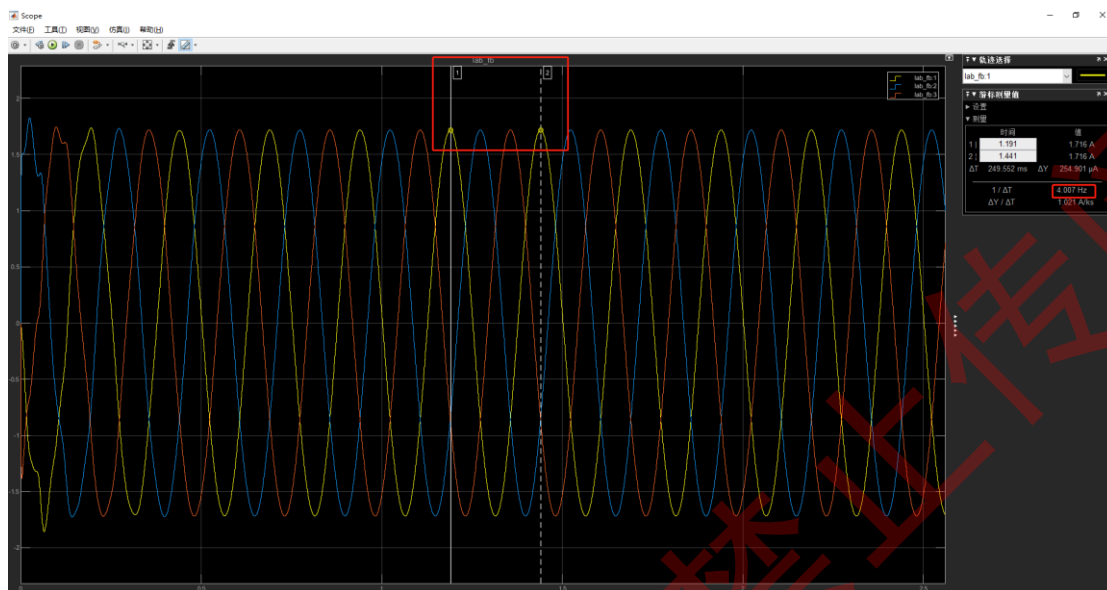
④ 使用 Simulink 自带的逆变器转换模块, 将三相电压转空比转化为实际的三相电压; 并在数据字典中增加母线电压 Udc 参数, 24V;



⑤ 将各个模块连接



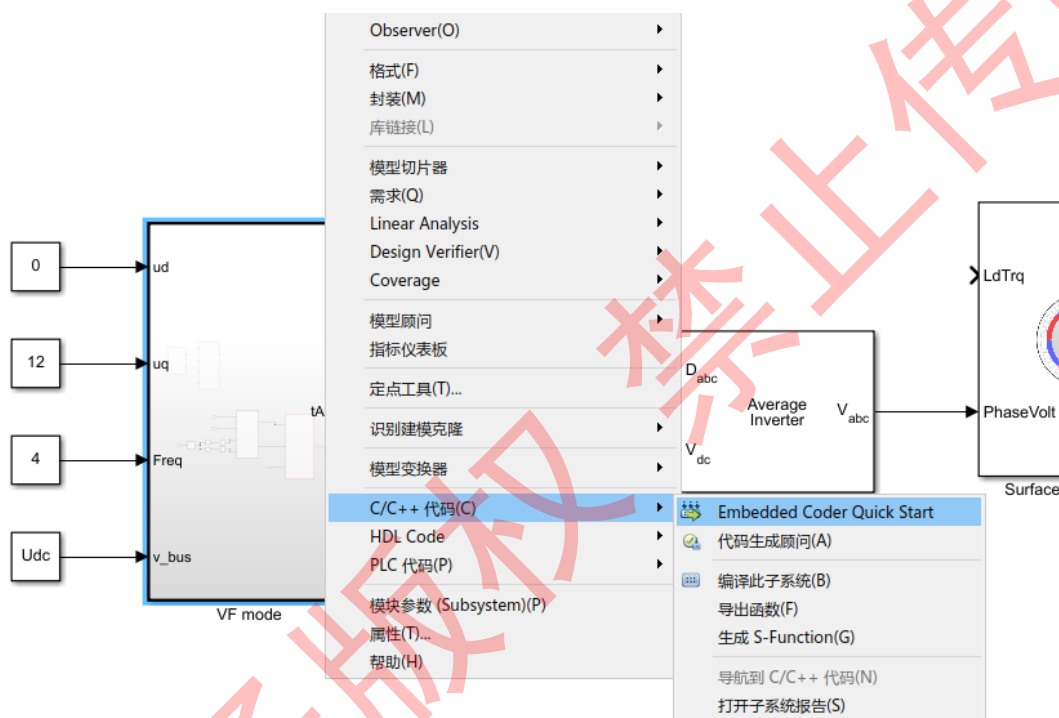
⑥ 电机运行，查看三相电流及速度波形；电流的频率与我们设计输入的频率一致；电机转速稳定在 8 左右；tABC 输出标准的马鞍波；



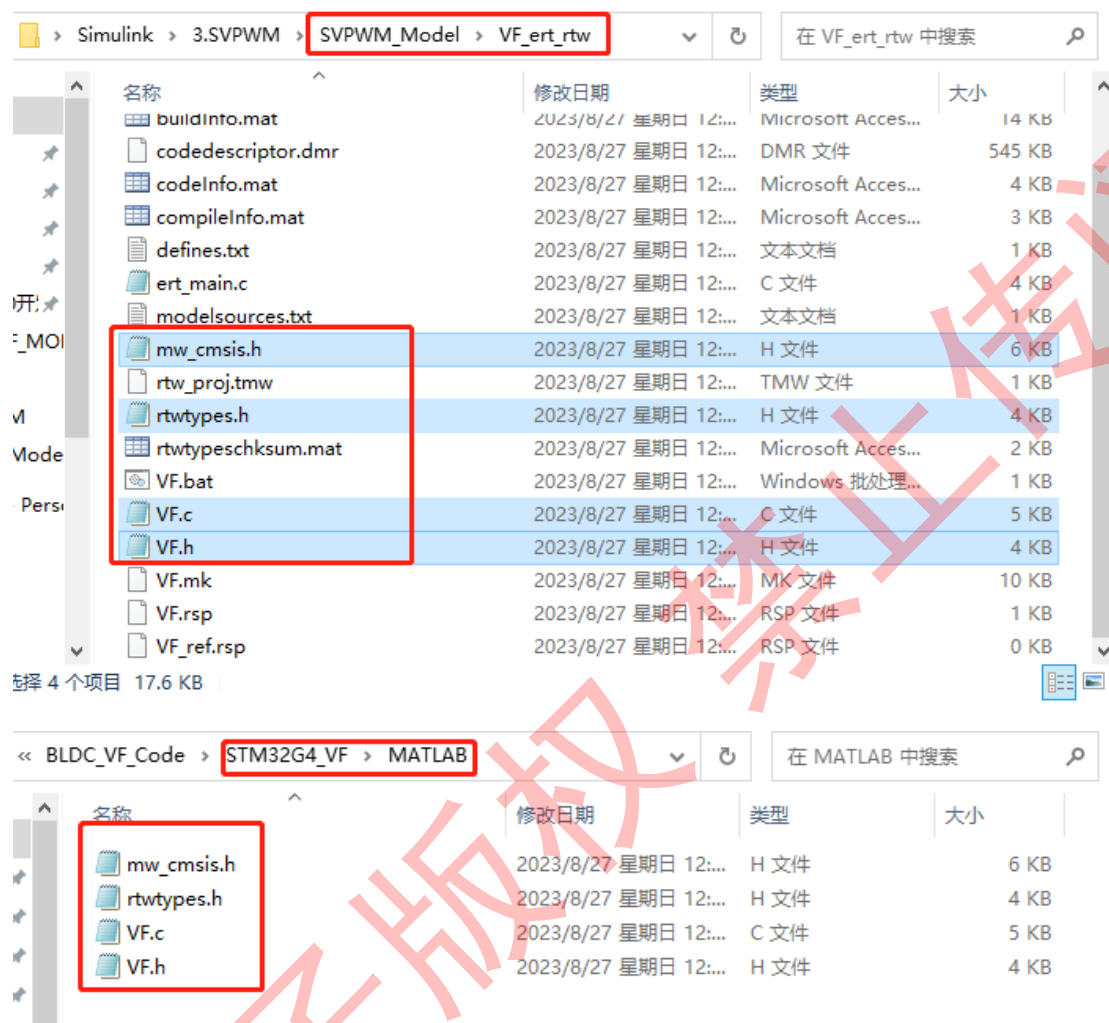


### 3. VF 启动电机实验现象

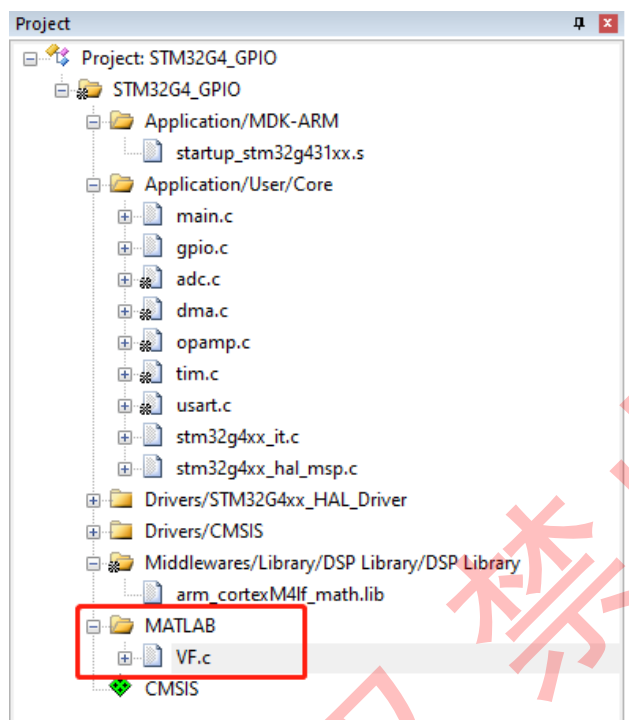
- ① 将 simulink 中 VF mode 模块，右键点击生成嵌入式代码；生成代码流程可参考第十三章；



- ② 将生成的代码，拷贝至 keil 工程中的 MATLAB 文件夹；



③ 打开 Keil 工程，在 Project 窗口中 MATLAB 文件夹中增加 VF.c 文件；



- ④ 在外设初始化函数后增加对 VF 算法的输入赋值，在 while 循环中实时采样

Vbus 电压；

```

139 HAL_TIM_Base_Start(&htim1);
140 HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
141 rtU.ud = 0;
142 rtU.uq = 12;
143 rtU.Freq = 4;
144 /* USER CODE END 2 */
145
146 /* Infinite loop */
147 /* USER CODE BEGIN WHILE */
148 while (1)
149 {
150     HAL_ADC_Start(&hadc2);
151     adc_vbus = HAL_ADC_GetValue(&hadc2);
152     rtU.v_bus = adc_vbus*3.3f/4096*26;
153     HAL_Delay(10);

```

- ⑤ 在按键三的回调函数中，启动和暂停 PWM 输出；

```

214  /* USER CODE BEGIN 4 */
215  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
216  {
217      /* Prevent unused argument(s) compilation warning */
218      UNUSED(GPIO_Pin);
219      if(Button3_Pin == GPIO_Pin)
220      {
221          Motor_state = ~Motor_state;
222          if(0 == Motor_state)
223          {
224              HAL_TIM_PWM_Stop( &htim1, TIM_CHANNEL_1);
225              HAL_TIM_PWM_Stop( &htim1, TIM_CHANNEL_2);
226              HAL_TIM_PWM_Stop( &htim1, TIM_CHANNEL_3);
227              HAL_TIMEx_PWMN_Stop( &htim1, TIM_CHANNEL_1);
228              HAL_TIMEx_PWMN_Stop( &htim1, TIM_CHANNEL_2);
229              HAL_TIMEx_PWMN_Stop( &htim1, TIM_CHANNEL_3);
230          }
231          else
232          {
233              HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_1);
234              HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_2);
235              HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_3);
236              HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_1);
237              HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_2);
238              HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_3);
239          }
240      }

```

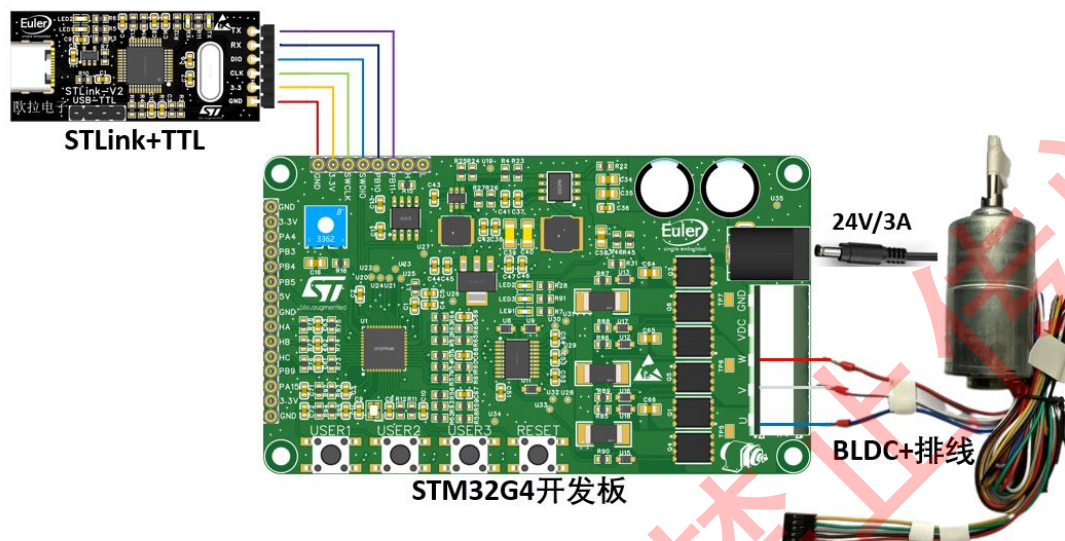
- ⑥ 在 ADC 中断回调函数中，读取三相电流值；将 VF 算法的输出的三相计数  
值写入 CCR 寄存器；并将电流及计数值上传至 VOFA 上位机实时显示；

```

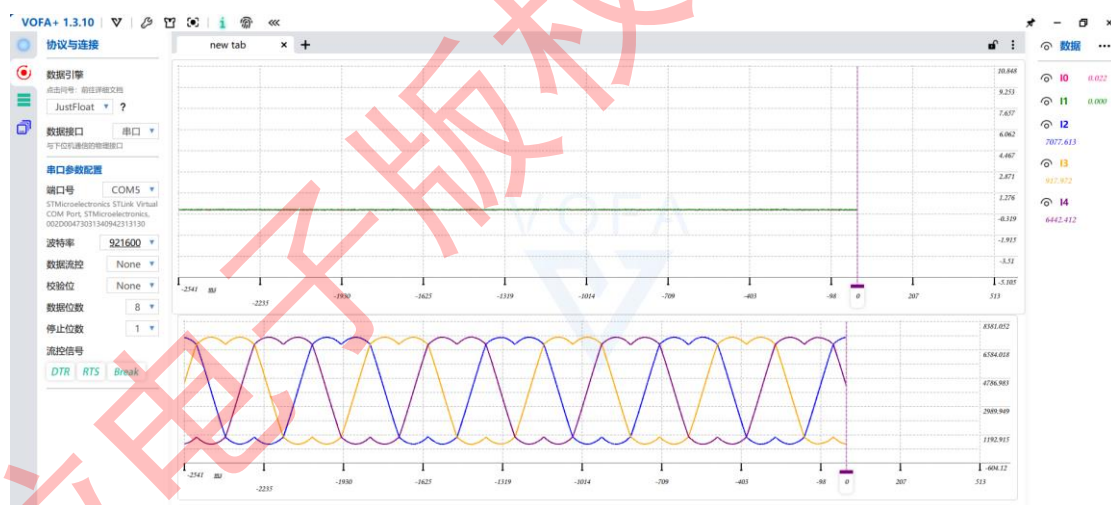
300 void HAL_ADCEx_InjectedConvCpltCallback(ADC_HandleTypeDef *hadc)
301 {
302     static uint8_t cnt;
303     static uint16_t observer_cnt;
304     /* Prevent unused argument(s) compilation warning */
305     UNUSED(hadc);
306     if(hadc == &hadc1)
307     {
308         if(ADC_offset == 0)
309         {
310             cnt++;
311             adc1_in1 = hadc1.Instance->JDR1;
312             adc1_in2 = hadc2.Instance->JDR1;
313             adc1_in3 = hadc1.Instance->JDR2;
314             IA_Offset += adc1_in1;
315             IB_Offset += adc1_in2;
316             IC_Offset += adc1_in3;
317             if(cnt >= 10)
318             {
319                 ADC_offset = 1;
320                 IA_Offset = IA_Offset/10;
321                 IB_Offset = IB_Offset/10;
322                 IC_Offset = IC_Offset/10;
323             }
324         }
325         else
326         {
327             adc1_in1 = hadc1.Instance->JDR1;
328             adc1_in3 = hadc1.Instance->JDR2;
329             adc1_in2 = hadc2.Instance->JDR1;
330             Ia = (adc1_in1 - IA_Offset)*0.02197f;
331             Ib = (adc1_in2 - IB_Offset)*0.02197f;
332             Ic = (adc1_in3 - IC_Offset)*0.02197f;
333             VF_step();
334             TIM1->CCR1 = rtY.tABC[0];
335             TIM1->CCR2 = rtY.tABC[1];
336             TIM1->CCR3 = rtY.tABC[2];
337             load_data[0] = Ia;
338             load_data[1] = Ib;
339             load_data[2] = rtY.tABC[0];
340             load_data[3] = rtY.tABC[1];
341             load_data[4] = rtY.tABC[2];
342             memcpy(tempData, (uint8_t *)&load_data, sizeof(load_data));
343             HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 6*4);
344         }
345     }
346 }

```

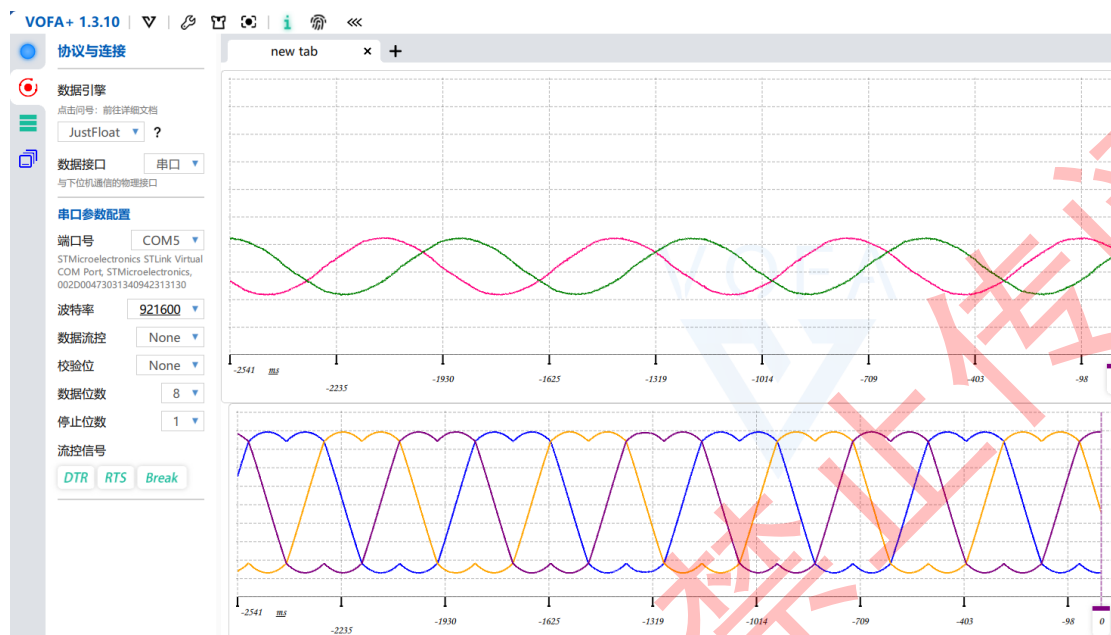
⑦ 按下图连接好 STLINK、UART 及电机线；



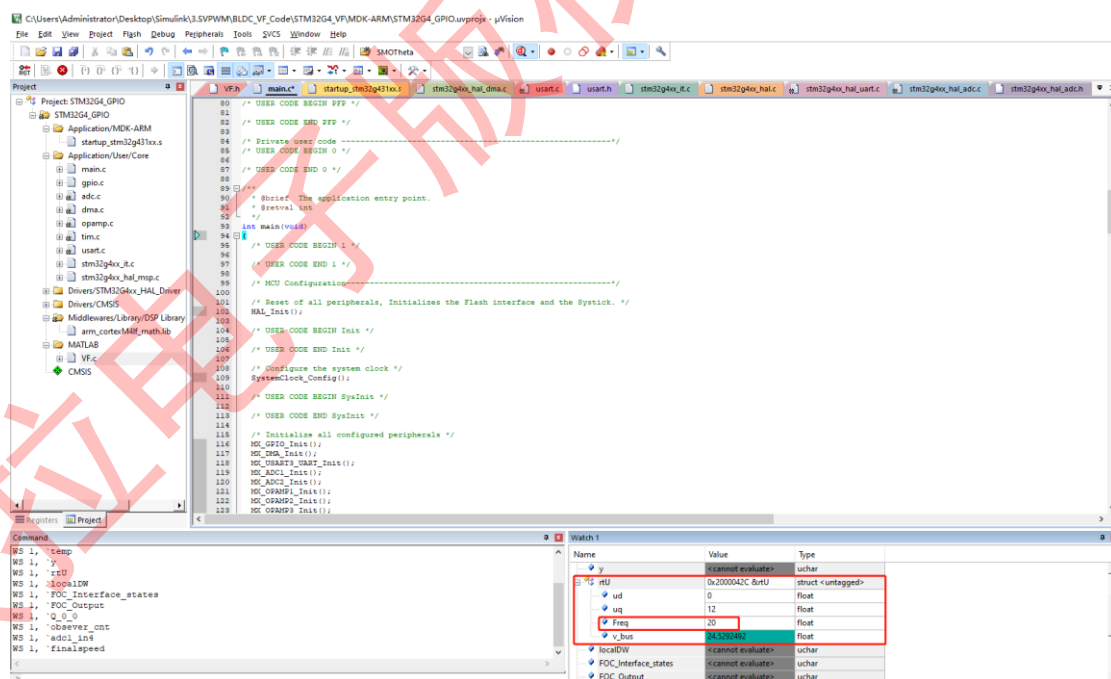
- ⑧ 编译并下载程序，打开 Keil debug 模式，打开 VOFA 上位机；可以看到输出的三相计数值为马鞍波；



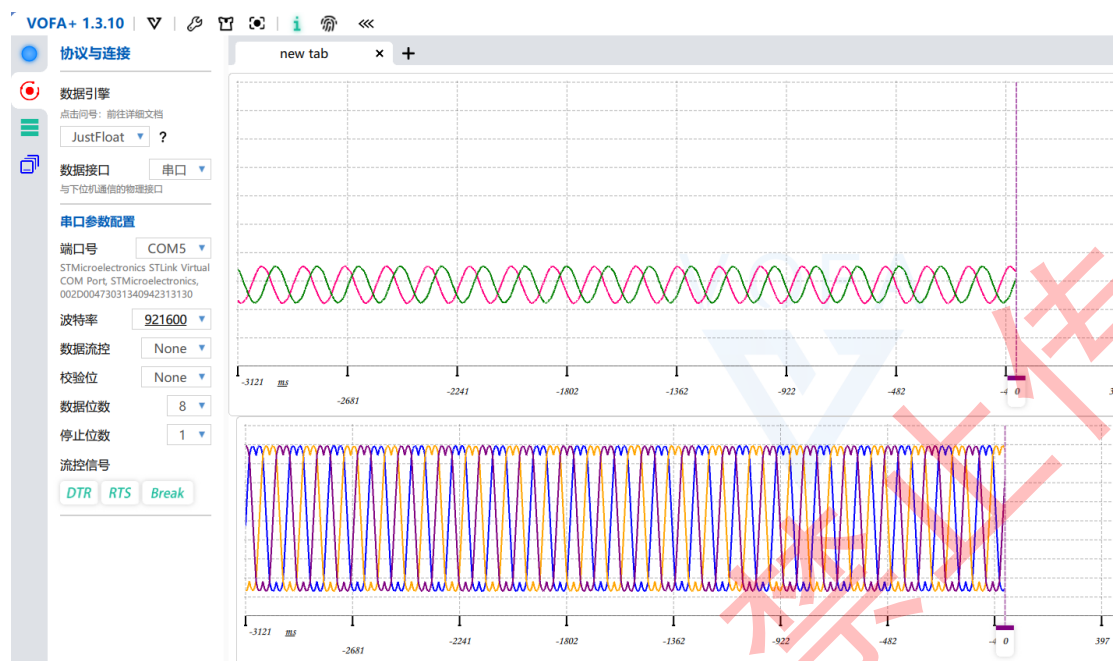
- ⑨ 按下按键三，电机旋转起来，a, b 相的电流值正常显示；



- ⑩ 在 keil 的 debug watch 串口中，输入变量 rtU，修改 Freq 值，即可修改电机转速；



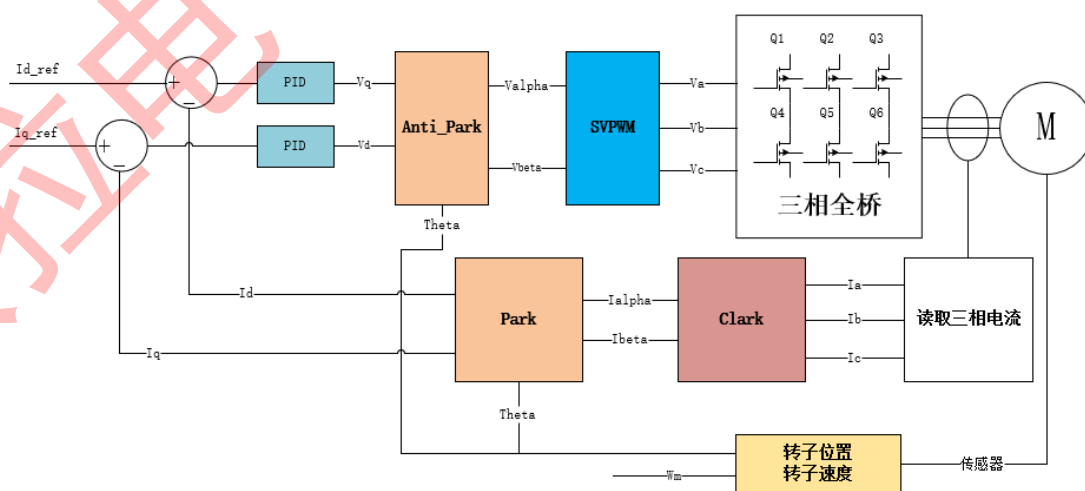




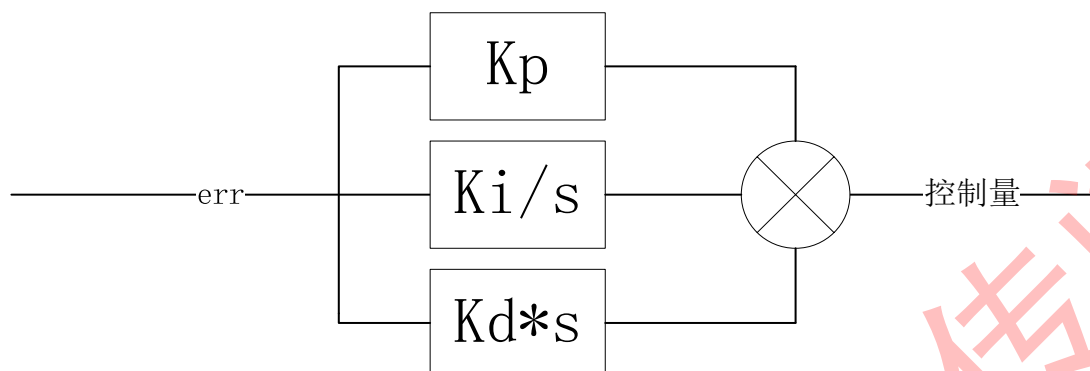
⑪ 在 VF 模式下不能长时间运行电机，电机发烫很严重；

## 第十六章：电流环闭环，IF 模式及启动状态机(Stateflow)

电流环框图：







$$C_{(t)} = \left( k_P e + k_I \int e dt + k_D \frac{de}{dt} \right) \cdot R_{(t)}$$

$$C_{(s)} = L[C_t] = \left( k_P + k_I \frac{1}{s} + k_D s \right) \cdot R_{(s)}$$

**PID:**

P 指比例部分

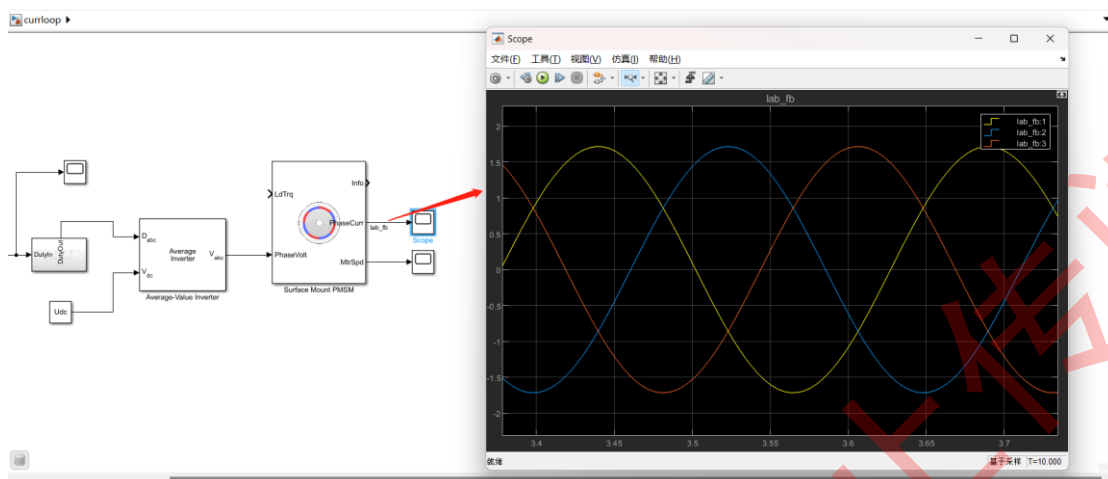
I 指积分部分

D 指微分部分；

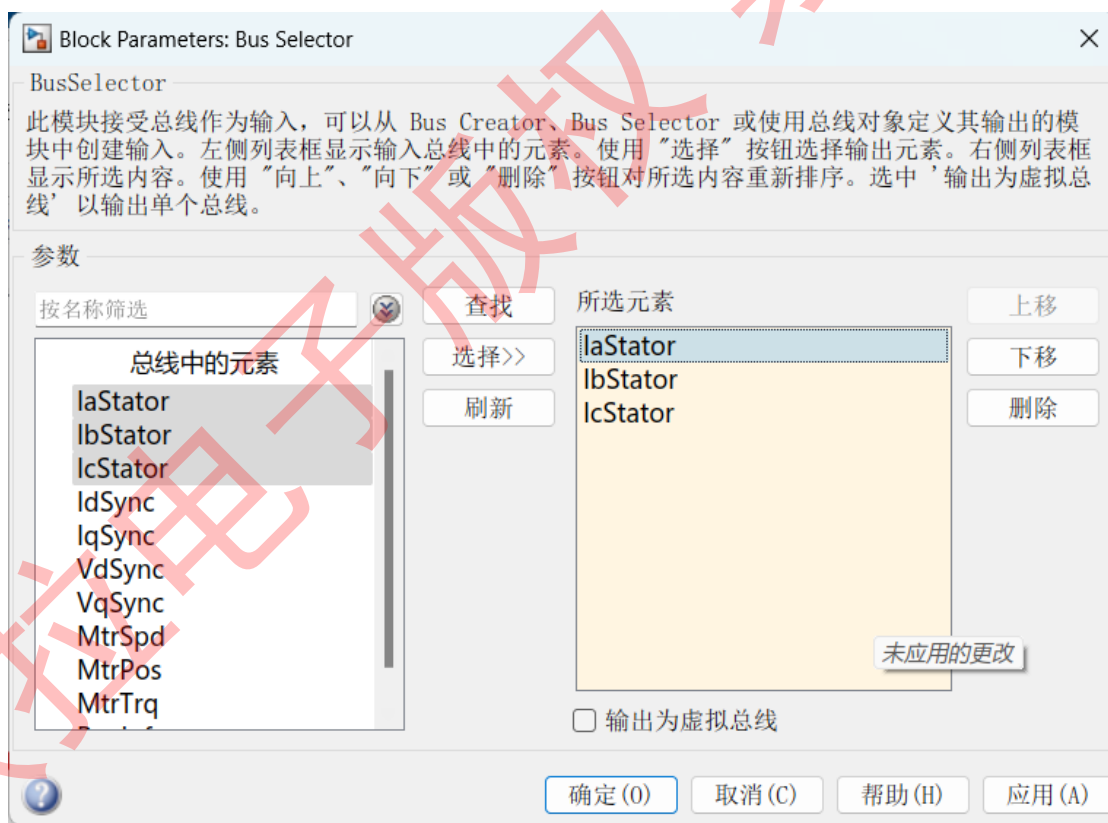
可以直观的描述出当前误差，过去误差，误差的变化趋势

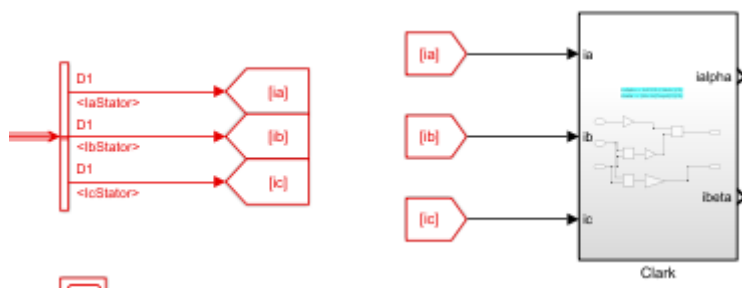
## 1. 电流环闭环建模

- ① 基于上一章 SVPWM 模型，我们可以得到电机的输出三相电流，得到电机的三相电流后我们可以通过 Calrk 变换将其转换为 alpha 及 beta 轴电流；

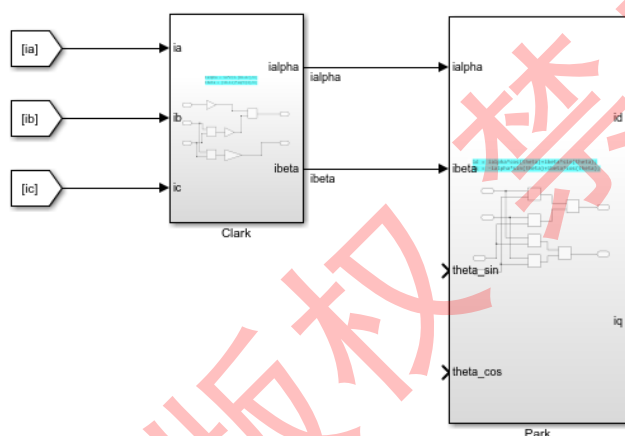


- ② 使用 Bus Selector 模块将电流总线输出为三相电流，并使用 Goto 模块及 From 模型，将电机输出的三相电流参数输入到 Clark 变换模块；



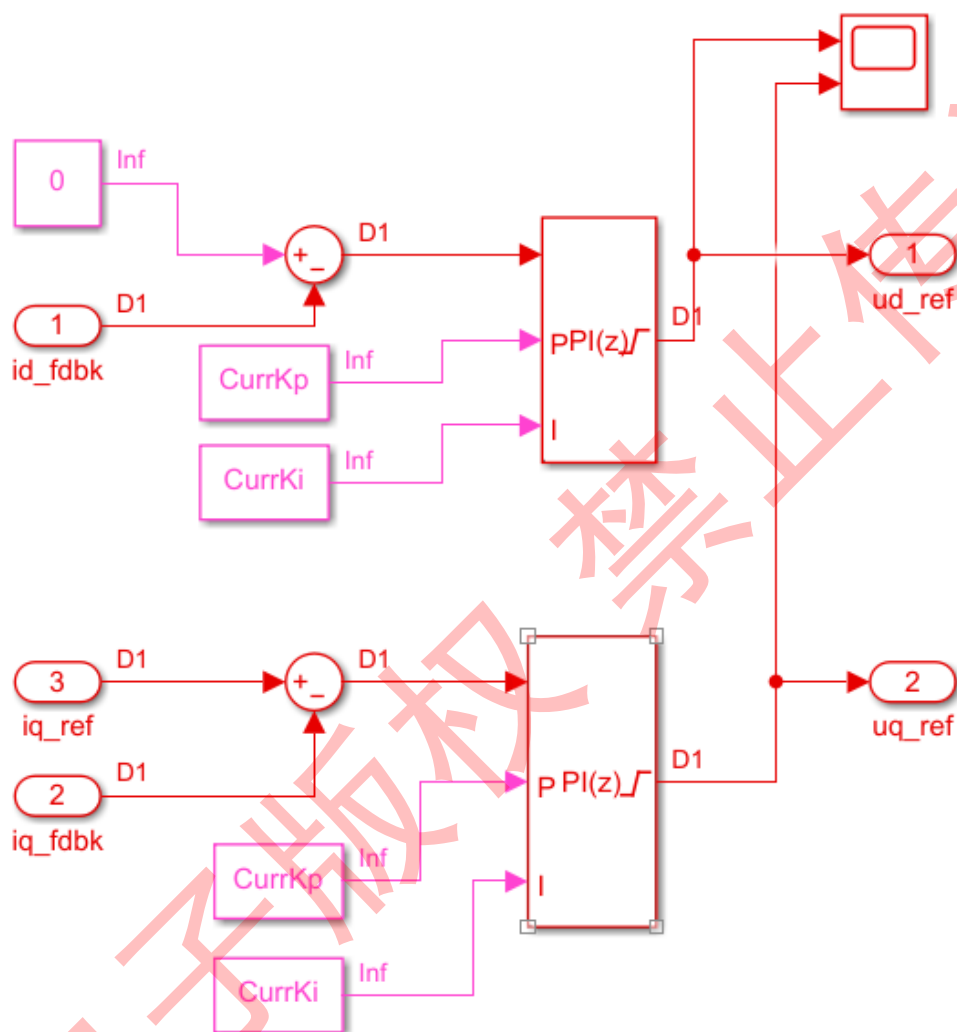


③ 得到 alpha 和 beta 轴电流后，再经过 Park 变换转换为实际 dq 轴电流；

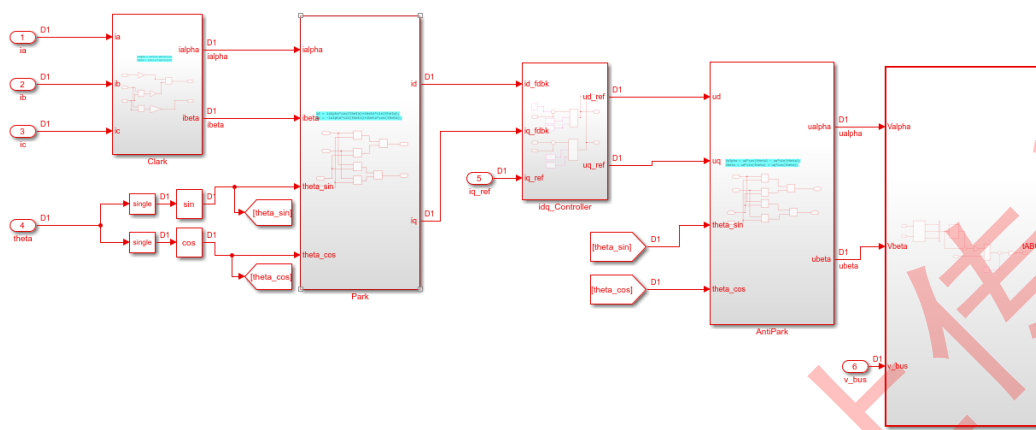


④ 电流环的目的就是将实际的 dq 轴电流变成我们期望的 dq 轴电流，故我们在期望的 dq 轴电流和实际的 dq 轴电流之间增加 PI 控制器；使用 constant 模块设置我们期望的 dq 轴电流，PID 模块用作 PI 控制器，Kp 和 Ki 参数配置外部输入；通常期望的 d 轴电流值都为 0；





- ⑤ PI 控制器的输出就是 dq 轴的电压经过 anti\_Park 变换就得到了 SVPWM 模块的输入 alpha 和 beta 轴电压；



- ⑥ 电流环的模型框架基本搭建完成，接下来确定我们的控制参数；电流环的 PI 控制参数可以按以下公式计算，其中  $\alpha$  为电流环的带宽可以选择 100 左右；

$$\begin{cases} K_{pd} = \alpha L_d \\ K_{id} = \alpha R \\ K_{pq} = \alpha L_q \\ K_{iq} = \alpha R \end{cases}$$

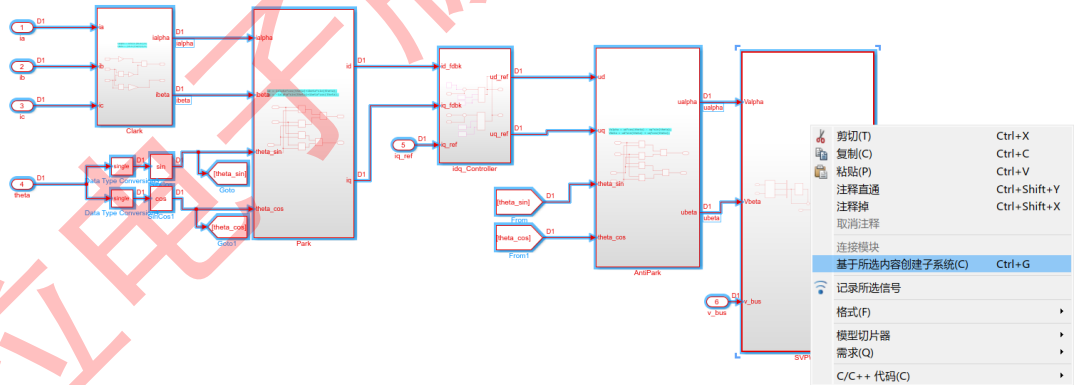
- ⑦ 在数据字典中建立 CurrKp 及 CurrKi 参数，并输入 PI 控制器；

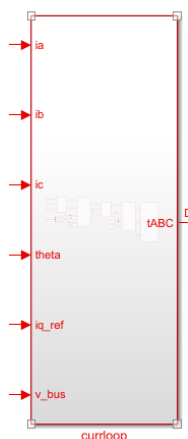
|  |        |     |      |        |       |      |     |     |      |
|--|--------|-----|------|--------|-------|------|-----|-----|------|
|  | CurrKp | New | 0.17 | single | [1 1] | real | [ ] | [ ] | Auto |
|  | CurrKi | New | 35   | single | [1 1] | real | [ ] | [ ] | Auto |

- ⑧ 设置 PI 控制器的输入 dq 轴电压阈值；

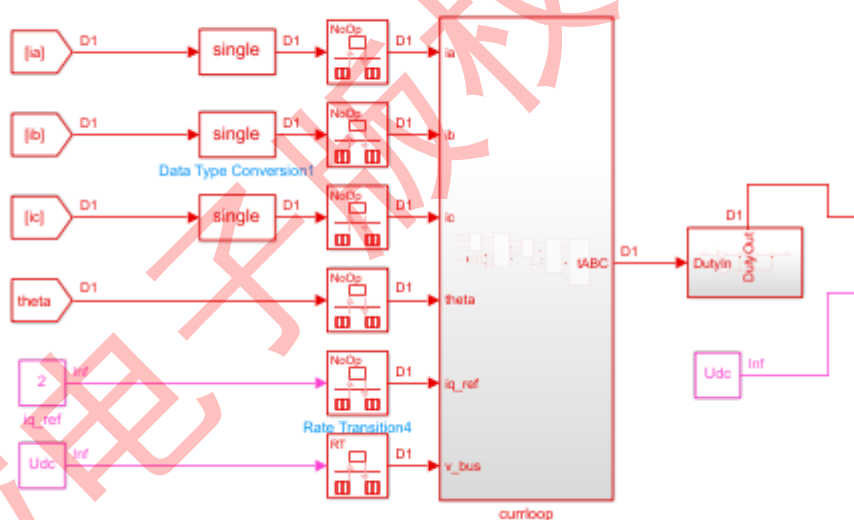


⑨ 将刚才所建立的算法部分选中并生成子系统，并命名为 currloop；

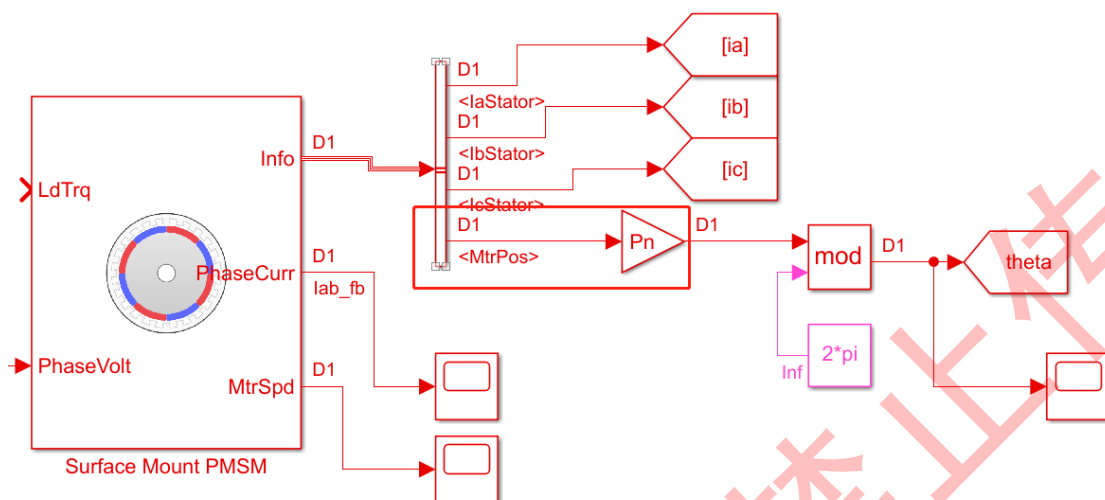




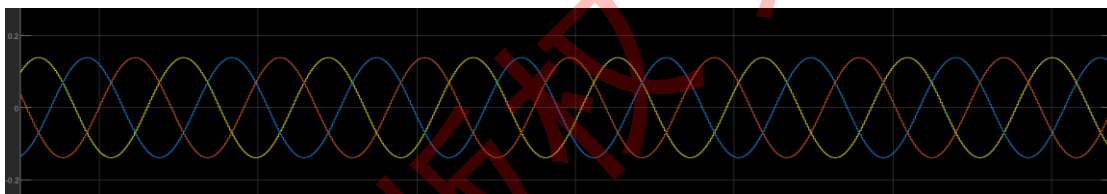
- ⑩ 连接外部模块，并增加 Rate Transition 模块，确保输入数据相同速率；其中 theta 角度来自电机的输出机械角度需要将其转化为电角度；



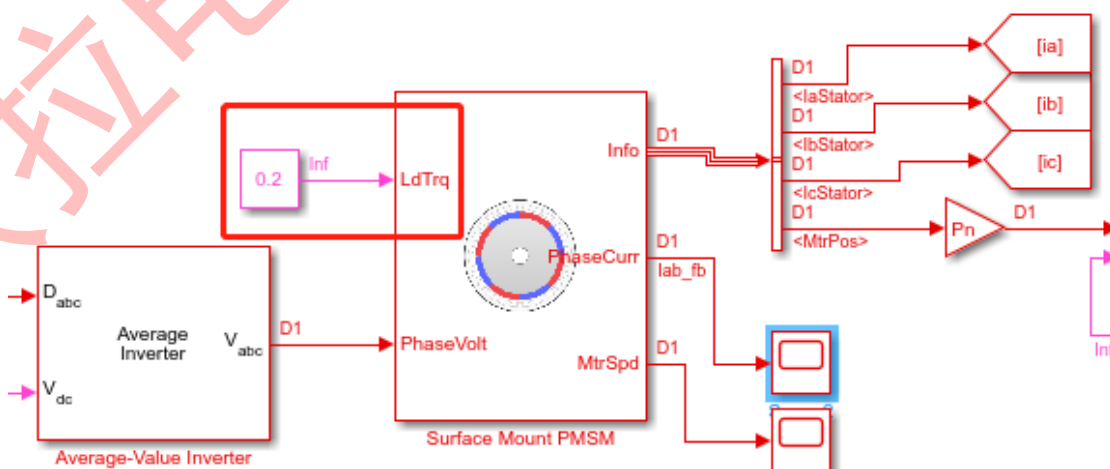


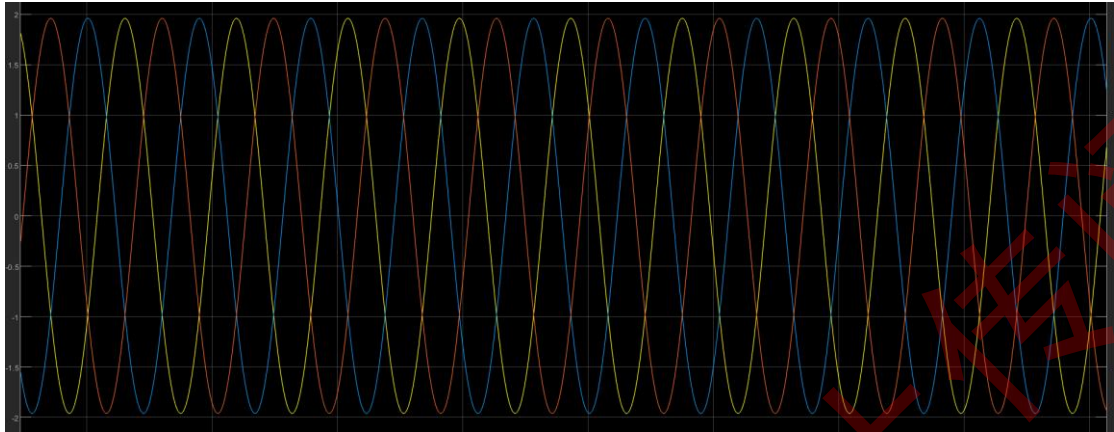


⑪ 点击仿真按钮，观察三相电流输出；



⑫ 如上图，可以看到三相电流为完美的正弦波，代表电机正常启动，但是电流幅值却不是我们期望的 q 轴电流大小；这是因为电机的输出转矩与负载不匹配，若想要我们输出的电流幅值达到我们设计的参考值，可以为增加负载；

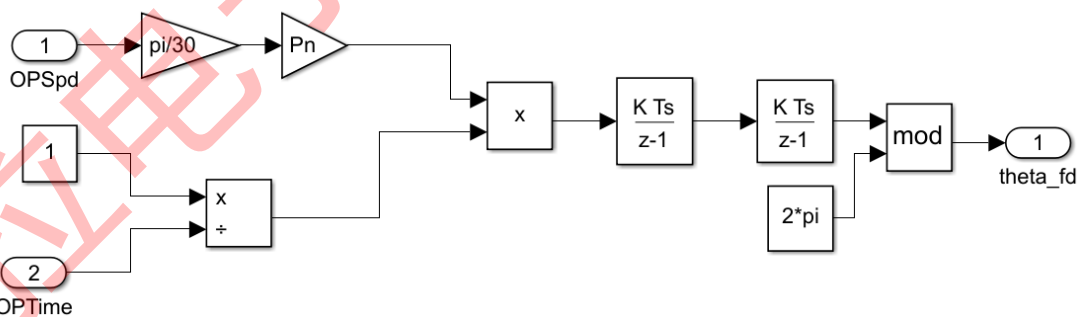




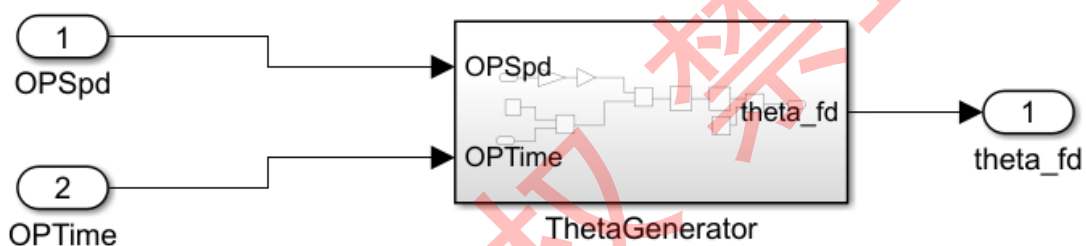
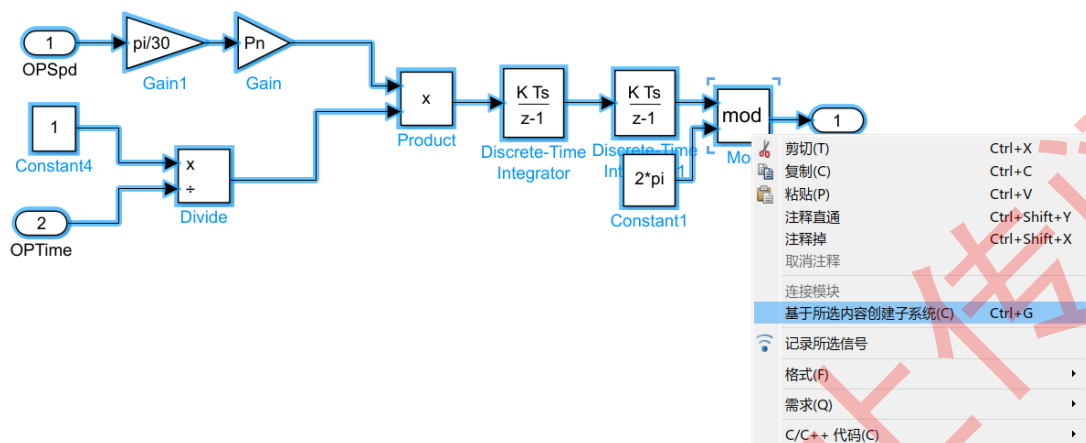
## 2. IF 模式建模

IF 模式是速度环开环，电流环闭环的模式来启动电机，在上一章电流环仿真的模型中我们已经建立起了电流环的闭环模型，而模型中的  $\theta$  角度来自电机输出的角度，然而在实际的电机系统中没有这样的角度可以输出，故在 IF 模式下我们需要自己来生成这样的角度。

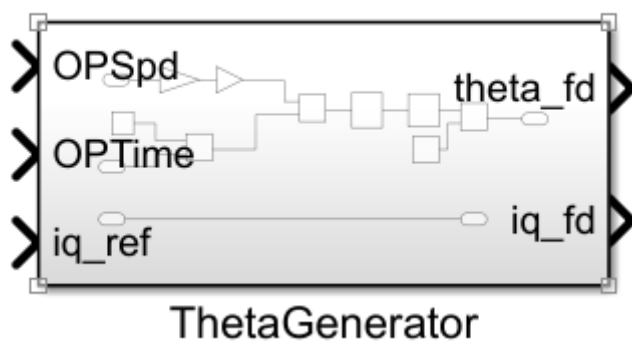
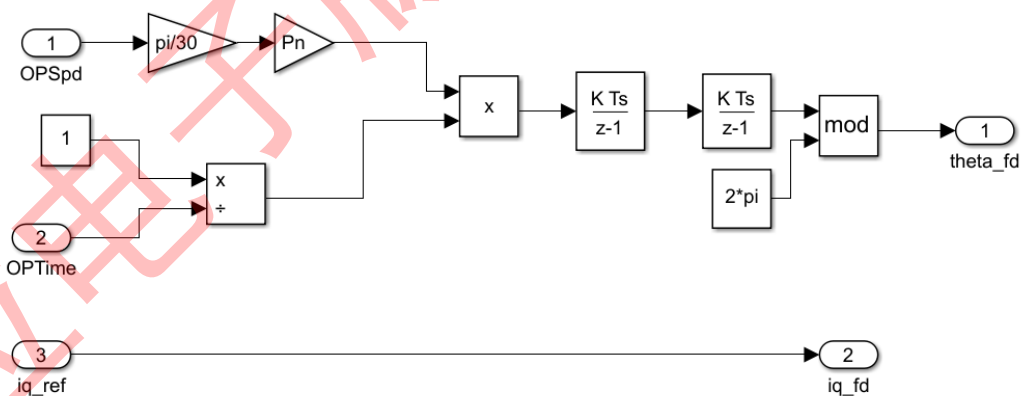
- ① 增加电机角度生成模块：电机的角度等于速度的积分，电机的速度来自于加速度的积分；加速度等于开环速度除以开环时间；



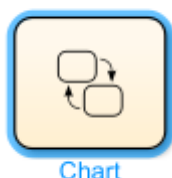
- ② 将该部分算法封装成子系统，并命名为 ThetaGenerator；



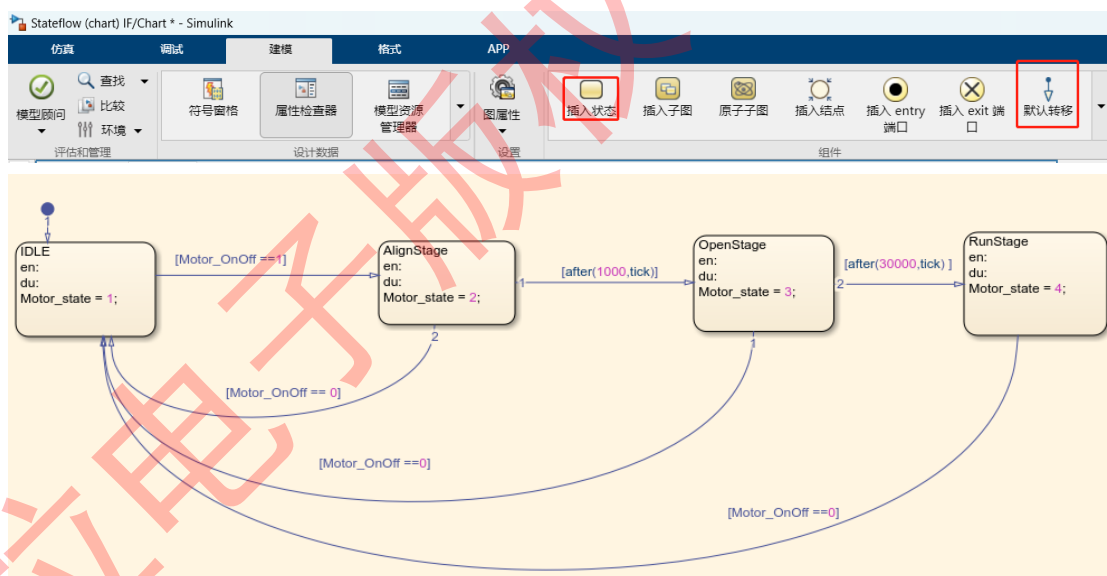
③ 在该模块中增加 q 轴的输入和输出参考电流；



- ④ 由于 IF 启动需要从 0 速缓慢拉起，其存在几个启动状态的切换，故需要使用 simulink 中的 Chart(stateflow)模块；



- ⑤ IF 启动分为转子初定位、开环拉起、切速度闭环几个状态，由于目前没有学习到速度环，在切速度闭环状态中，我们给一个开环固定速度让电机旋转；在 Chart 模块中使用插入状态和默认转移模块，搭建如下状态机；



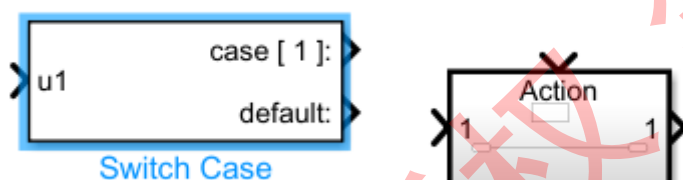
其中各状态之间的切换条件在默认转移线上有定义；Motor\_OnOff 为状态机输入的启动电机命令，Motor\_state 为状态机输出的状态，共计四个状态；

[Motor\_OnOff == 1]表示当 Motor\_OnOff 为 1 时，切换到下一个状态；

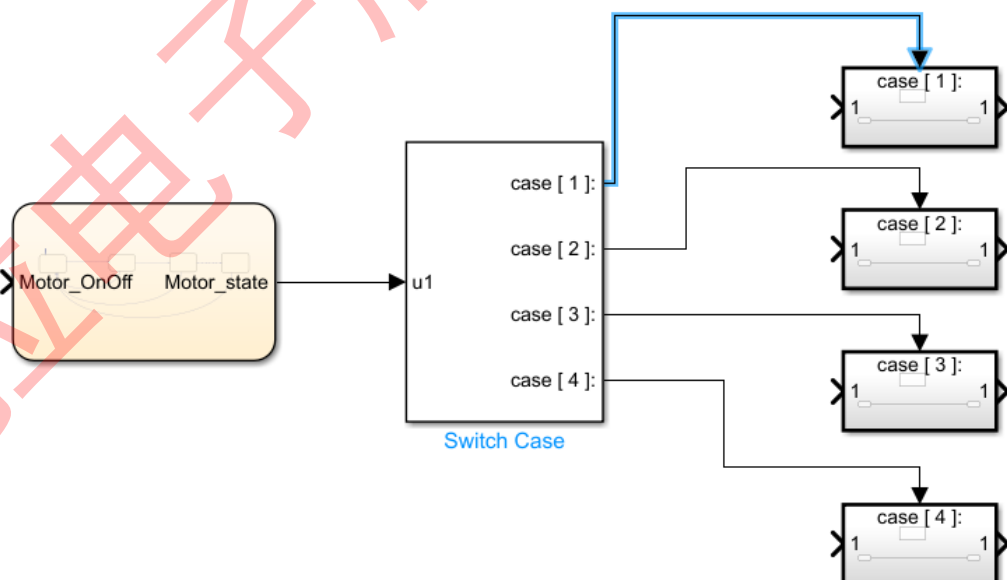
[after(1000,tick)]表示当该状态在 1000 个计数周期后切换至下一个状态；

| Symbols                                                                           |             |       |      |
|-----------------------------------------------------------------------------------|-------------|-------|------|
| TYPE                                                                              | NAME        | VALUE | PORT |
|  | Motor_OnOff |       | 1    |
|  | Motor_state |       | 1    |

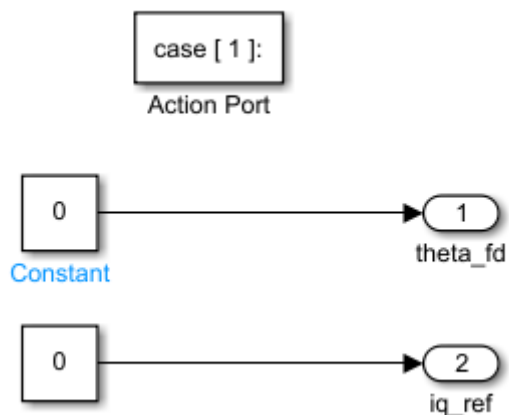
- ⑥ 使用 Switch Case 及 If Action Subsystem 模块来判断在不同状态下执行对应的内容；



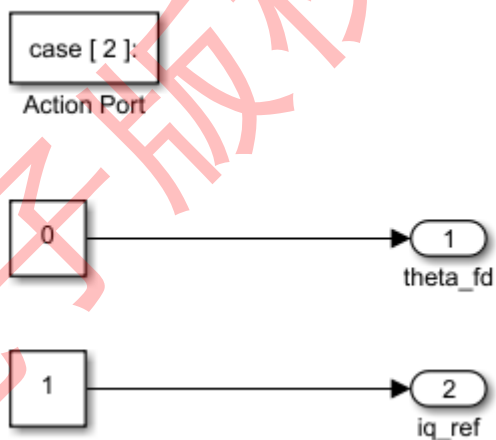
- ⑦ 状态机输出 4 个状态，则需要建立对应的 4 中状态下的执行系统；



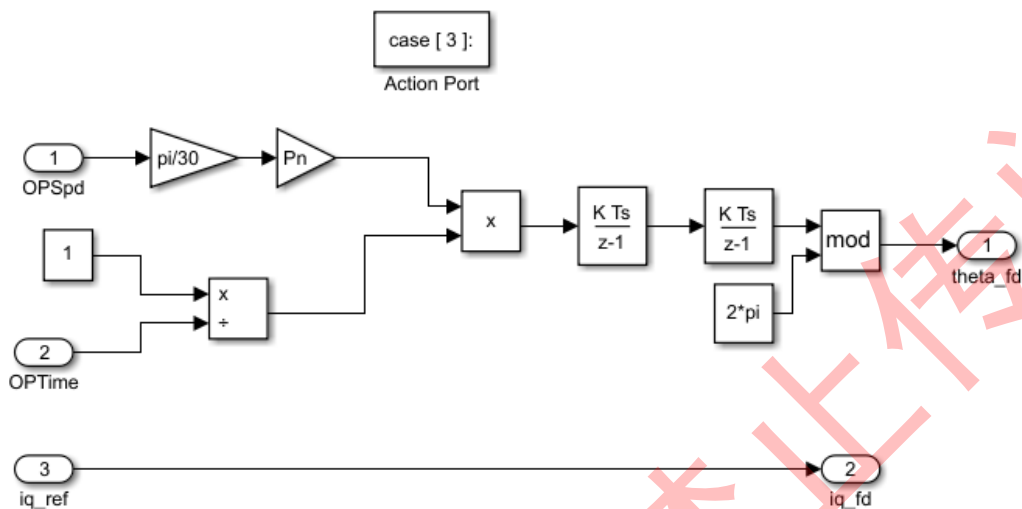
- ⑧ 状态一为空闲状态，此时输出的电机角度及 q 轴参考电流应为 0，在 case 1 中建立如下部分；



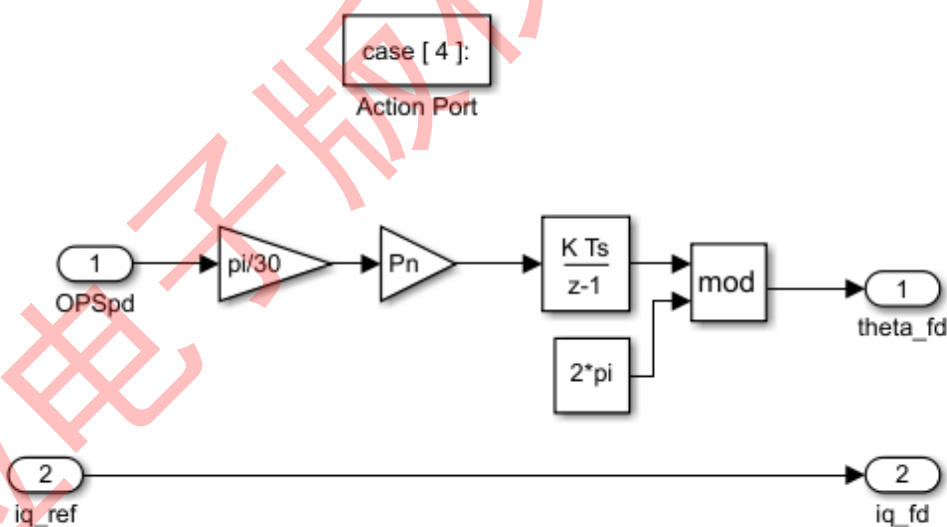
- ⑨ 状态二为转子角度对齐阶段，我们需要将转子角度对齐至 0 度，此时输出的 theta 角度需要为 0，q 轴参考电流有给一个能够驱动电机转动的电流，此处给 1；



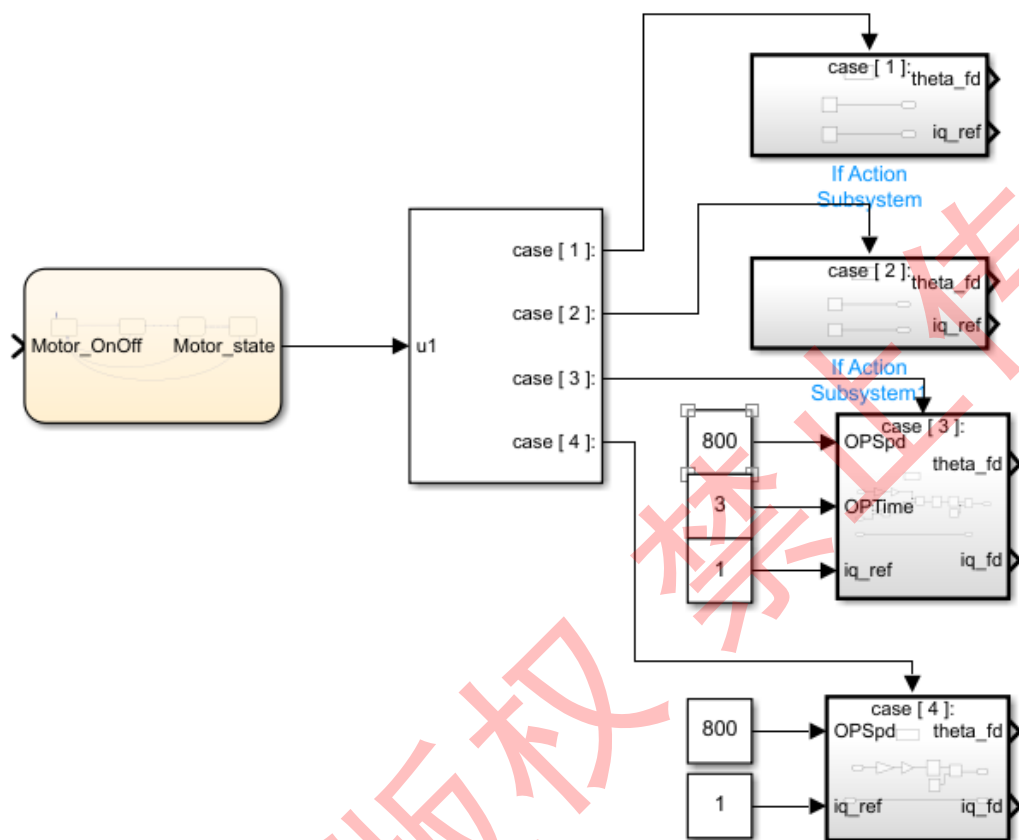
- ⑩ 状态三为电机开环拉起状态，我们将之前建立的 ThetaGenerator 模块内部的内容复制到 case 3 中；



- ⑪ 在状态四我们只需要将电机的转速开环控制在开环的转速上，将电机的开环速度进行积分即为电机的转子角度；在 case 4 中建立以下部分；

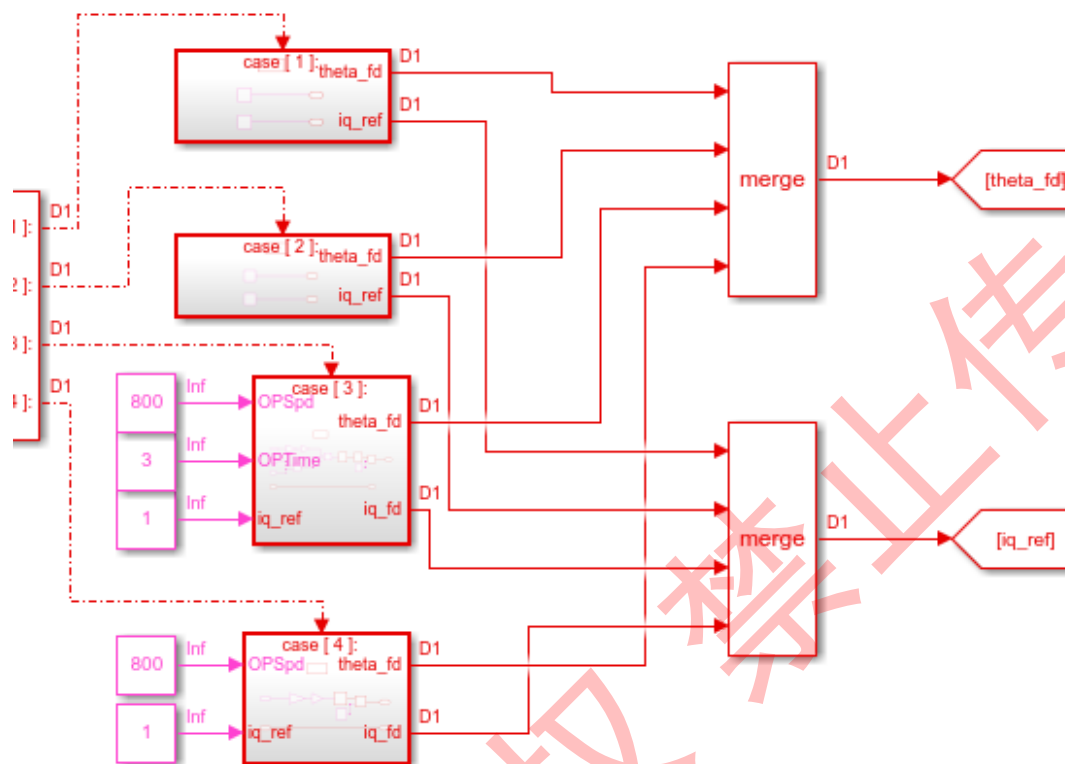


- ⑫ 整个状态机模型已经建立完成；我们将开环速度设置为 800rpm/min；开环时间为 3s，与状态机的[after(30000,tick)]相对应（ $30000 \times 1e-4$ ）；开环拉起阶段及运行阶段的参考电流设置为 1；

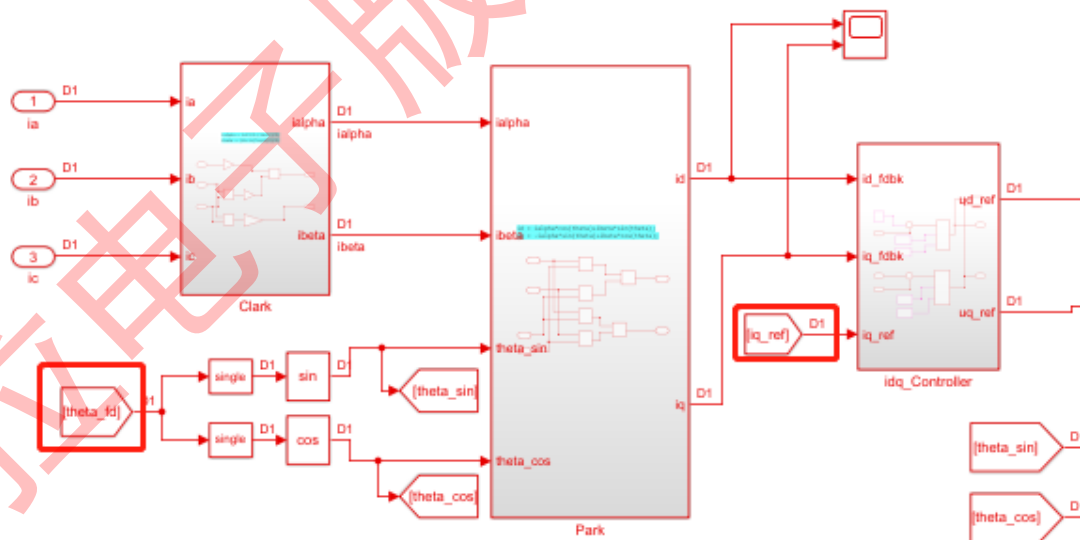


② 使用 Merge 模块，将四个状态的  $\theta$  和  $q$  轴电流组合输出；

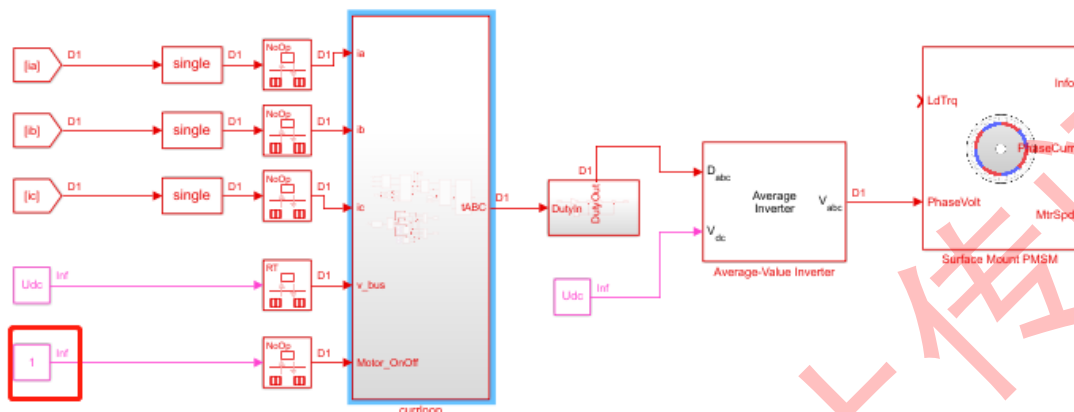




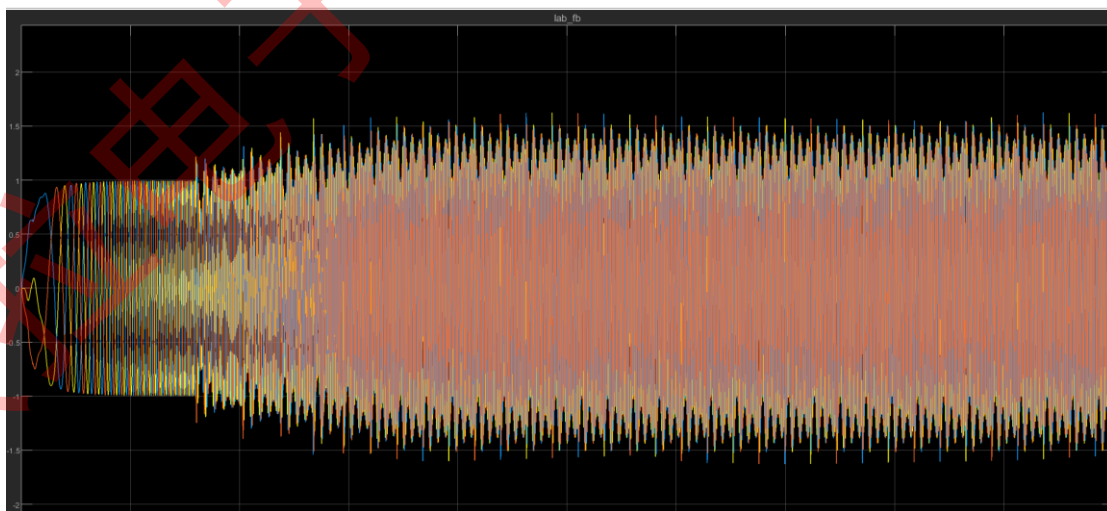
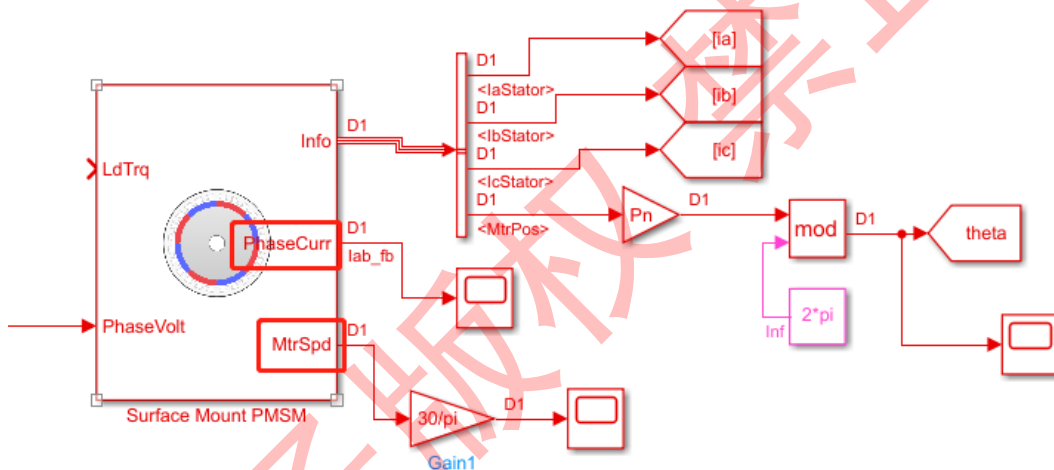
③ 将状态机输出的  $\theta$  及 q 轴电流参考值作为电流环的输入；

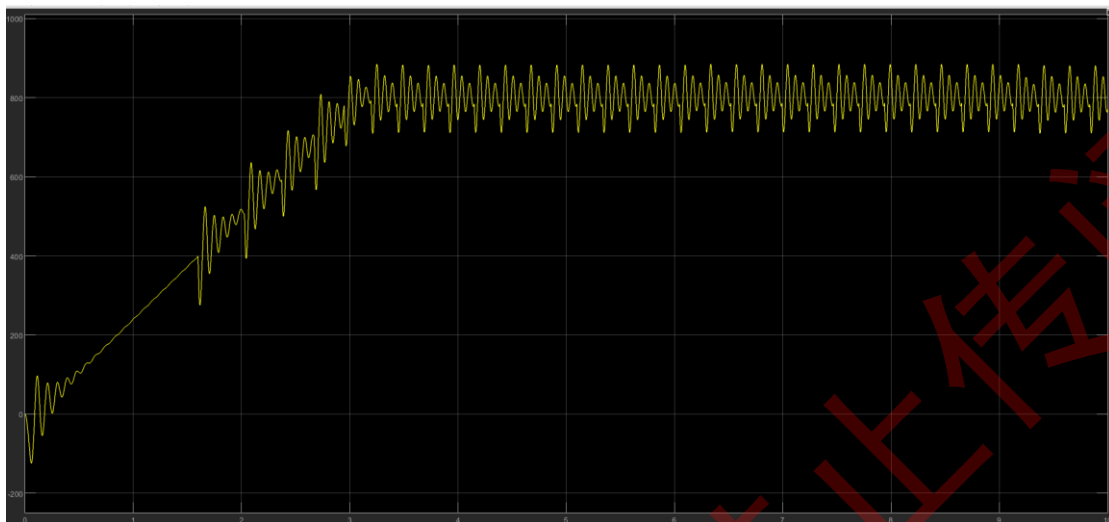


④ 外部增加 MotorOnOff 的输入为 1；



⑤ 点击仿真按钮，查看电机的速度值及三相电流值；

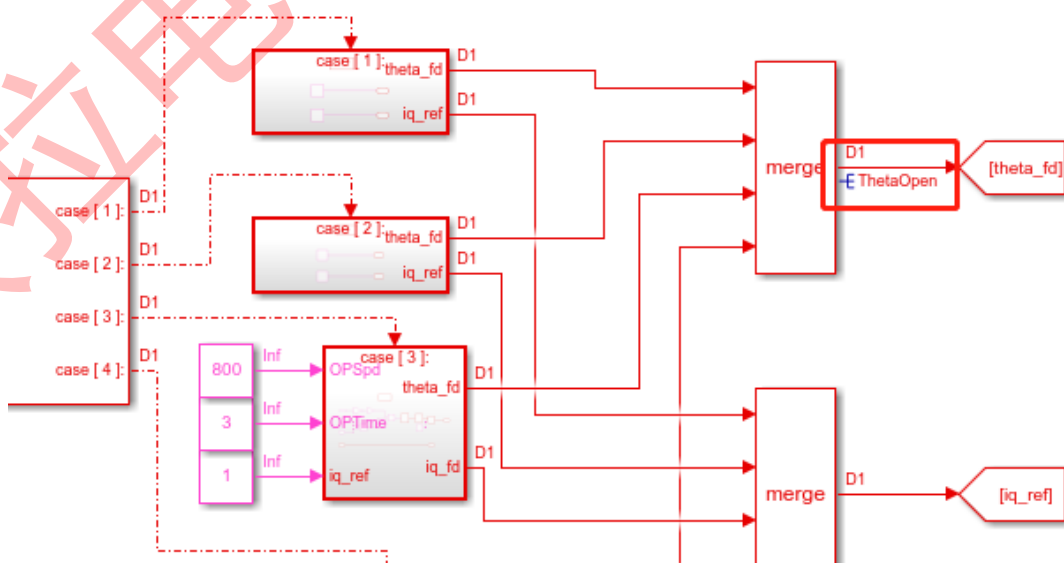




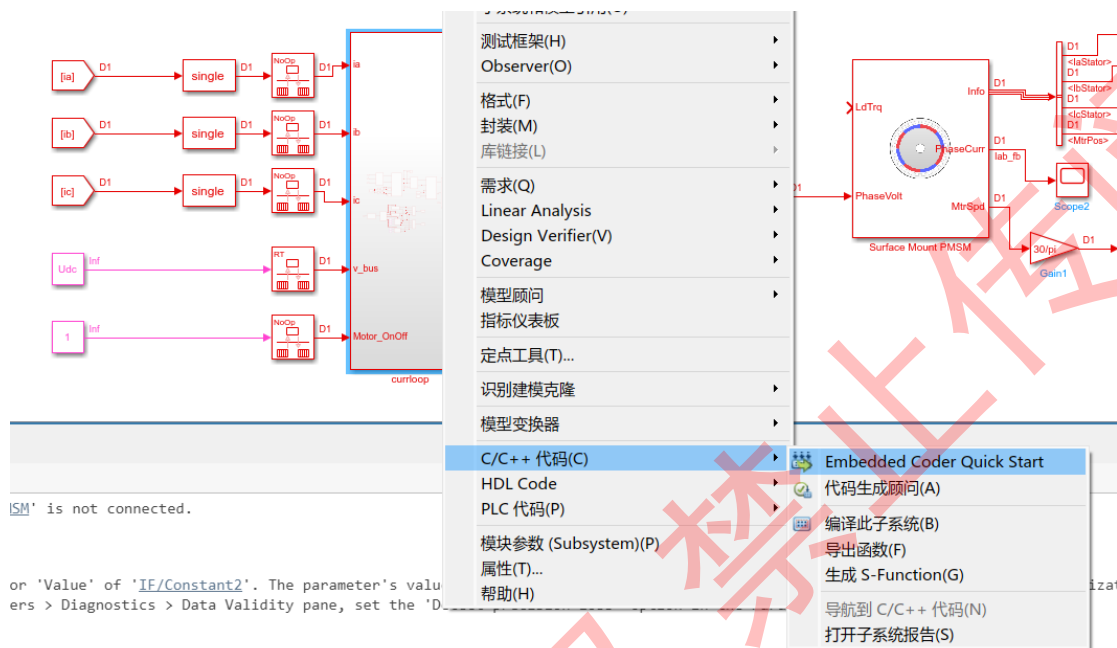
- ⑥ 通过上图可以看出转速从 0 上升到 800 后，在 800 上下波动，三相电流也存在波动，这是由于我们给的 q 轴电流参考值与实际负载不匹配导致；总体上看，电机从开环被拉到我们期望的开环转速，实现我们的目的。

### 3. IF 启动电机实验现象

- ① 我们需要用上位机显示开环的  $\theta$  角度，故将状态机的输出  $\theta$  角度作为全局变量输出；如何建立全局变量请参考之前章节内容；

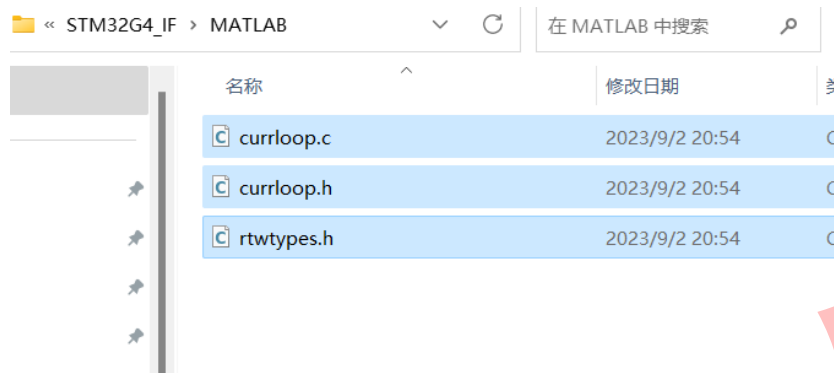


② 右键电流环模块，生成代码；生成代码流程请参考之前章节；

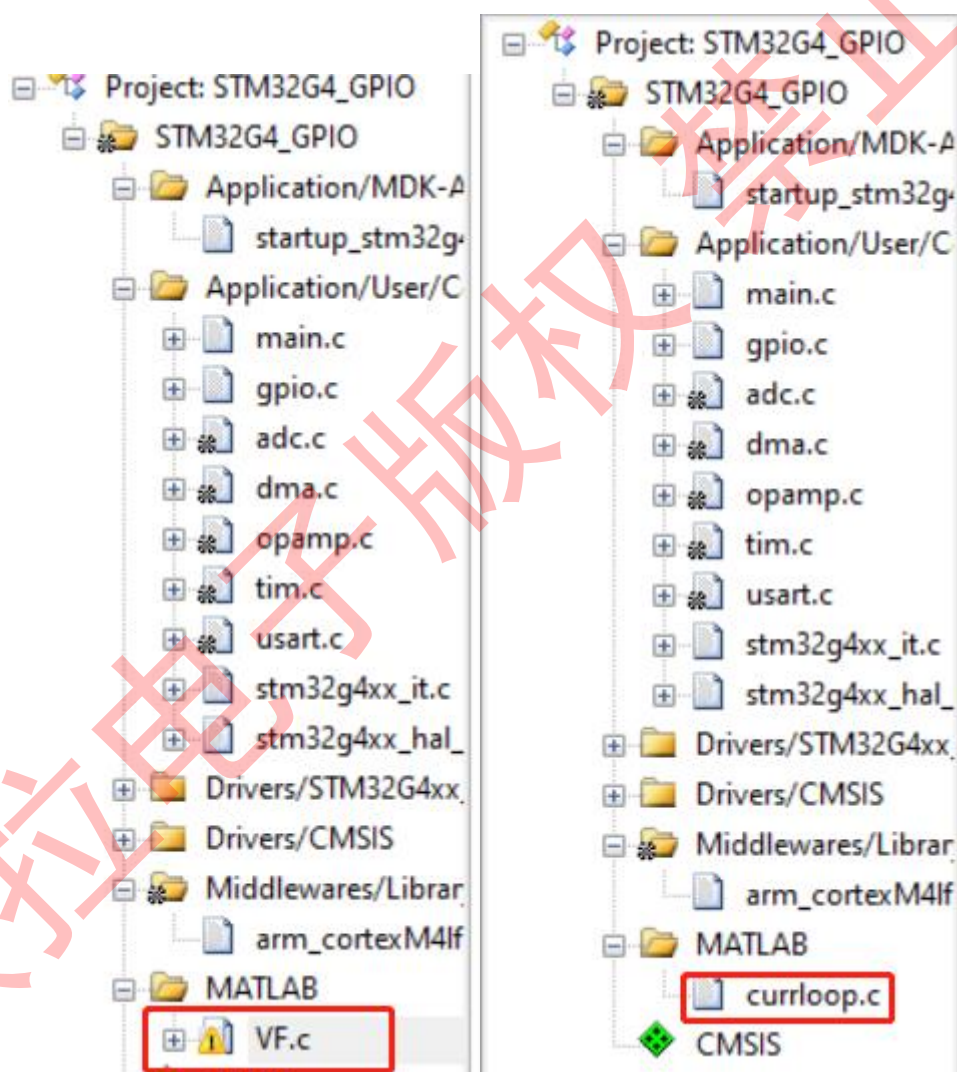


③ 复制 VF 工程代码，将工程中 MATLAB 中的文件替换为 currloop 模块生成的代码；

| 名称                 | 修改日期           | 类型             | 大小     |
|--------------------|----------------|----------------|--------|
| tmwinternal        | 2023/9/2 20:54 | 文件夹            |        |
| buildInfo.mat      | 2023/9/2 20:54 | MAT 文件         | 20 KB  |
| codescriptor.dmr   | 2023/9/2 20:54 | DMR 文件         | 753 KB |
| codeInfo.mat       | 2023/9/2 20:54 | MAT 文件         | 5 KB   |
| compileInfo.mat    | 2023/9/2 20:54 | MAT 文件         | 3 KB   |
| currloop.bat       | 2023/9/2 20:54 | Windows 批处理... | 1 KB   |
| currloop.c         | 2023/9/2 20:54 | C 源文件          | 22 KB  |
| currloop.h         | 2023/9/2 20:54 | C Header 源文件   | 14 KB  |
| currloop.mk        | 2023/9/2 20:54 | Makefile 源文件   | 12 KB  |
| currloop.rsp       | 2023/9/2 20:54 | RSP 文件         | 1 KB   |
| currloop_comp.rsp  | 2023/9/2 20:54 | RSP 文件         | 1 KB   |
| currloop_ref.rsp   | 2023/9/2 20:54 | RSP 文件         | 0 KB   |
| defines.txt        | 2023/9/2 20:54 | 文本文档           | 1 KB   |
| ert_main.c         | 2023/9/2 20:54 | C 源文件          | 3 KB   |
| modelsources.txt   | 2023/9/2 20:53 | 文本文档           | 1 KB   |
| rtw_proj.tmw       | 2023/9/2 20:54 | TMW 文件         | 1 KB   |
| rtwtypes.h         | 2023/9/2 20:54 | C Header 源文件   | 4 KB   |
| rtwtypeschksum.mat | 2023/9/2 20:53 | MAT 文件         | 2 KB   |
| setup_mingw.bat    | 2023/9/2 20:54 | Windows 批处理... | 1 KB   |



④ 打开 Keil 工程，移除 VF.c，新增 currloop.c;



⑤ Main.c 中包含 currloop.h 文件;

```

29  /* Private includes -----
30  /* USER CODE BEGIN Includes */
31  #include <stdio.h>
32  #include <string.h>
33  #include "currloop.h"
34  /* USER CODE END Includes */
35

```

⑥ 在按键 3 的中断回调函数中增加电机启动和暂停；

```

210
211 /* USER CODE BEGIN 4 */
212 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
213 {
214     /* Prevent unused argument(s) compilation warning */
215     UNUSED(GPIO_Pin);
216     if(Button3_Pin == GPIO_Pin)
217     {
218         Motor_state = ~Motor_state;
219         if(0 == Motor_state)
220         {
221             rtU.Motor_OnOff = 0;
222             HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1);
223             HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_2);
224             HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_3);
225             HAL_TIMEx_PWMN_Stop(&htim1, TIM_CHANNEL_1);
226             HAL_TIMEx_PWMN_Stop(&htim1, TIM_CHANNEL_2);
227             HAL_TIMEx_PWMN_Stop(&htim1, TIM_CHANNEL_3);
228         }
229         else
230         {
231             rtU.Motor_OnOff = 1;
232             HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
233             HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
234             HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
235             HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);
236             HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
237             HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_3);
238         }
239     }
240     /* NOTE: This function should not be modified, when the callback is needed,
241     the HAL_GPIO_EXTI_Callback could be implemented in the user file
242     */
243 }
244

```

⑦ 在 adc 中断回调函数中增加三相电流输入，执行 currloop 算法；UART 上传

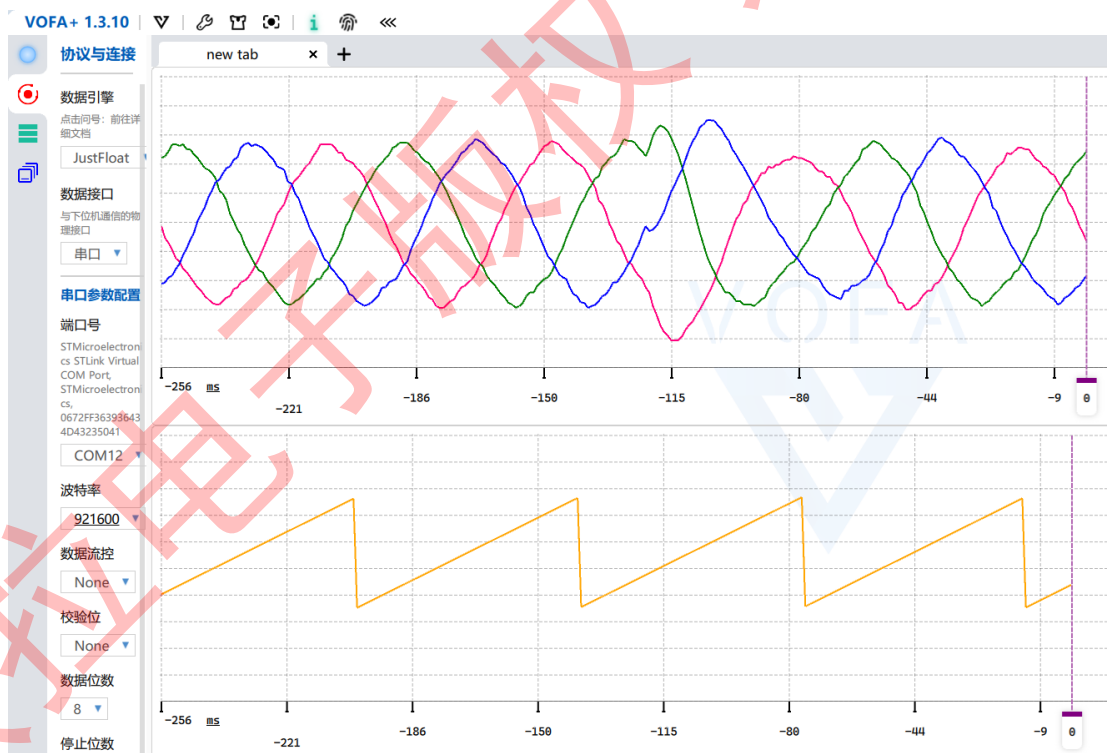
三相电流及开环 theta 角度；

```

324     else
325     {
326         adc1_in1 = hadc1.Instance->JDR1;
327         adc1_in3 = hadc1.Instance->JDR2;
328         adc1_in2 = hadc2.Instance->JDR1;
329         rtU.ia = (adc1_in1 - IA_Offset)*0.02197f;
330         rtU.ib = (adc1_in2 - IB_Offset)*0.02197f;
331         rtU.ic = (adc1_in3 - IC_Offset)*0.02197f;
332         currloop_step();
333         TIM1->CCR1 = rtY.tABC[0];
334         TIM1->CCR2 = rtY.tABC[1];
335         TIM1->CCR3 = rtY.tABC[2];
336         load_data[0] = rtU.ia;
337         load_data[1] = rtU.ib;
338         load_data[2] = rtU.ic;
339         load_data[3] = ThetaOpen;
340         load_data[4] = 0;
341         memcpy(tempData, (uint8_t *)&load_data, sizeof(load_data));
342         HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 6*4);
343     }
344 }
345

```

⑧ 编译并下载程序；打开 VOFA 上位机，按下按键 3，查看上传的波形数据；



⑨ 可以看到三相电流波形与仿真一致，会存在一些抖动，后续我们增加了速度环之后该抖动就可以被抑制掉。



## 第十七章：速度环闭环

### 1. 速度环闭环建模

上一章我们完成电流环闭环建模及 IF 电流闭环速度开环的方式强托启动电机，这章我们模型中增加速度环闭环。

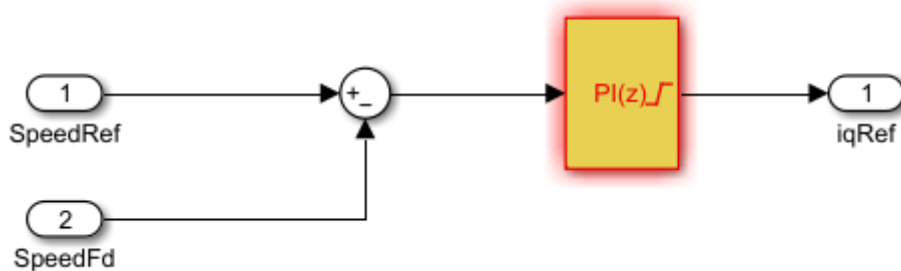
① 基于电流环闭环 simulink 工程，插入速度环子系统；



② 速度环输入为参考转速和反馈转速；输出为 q 轴参考电流；



③ 增加 PI 控制器；

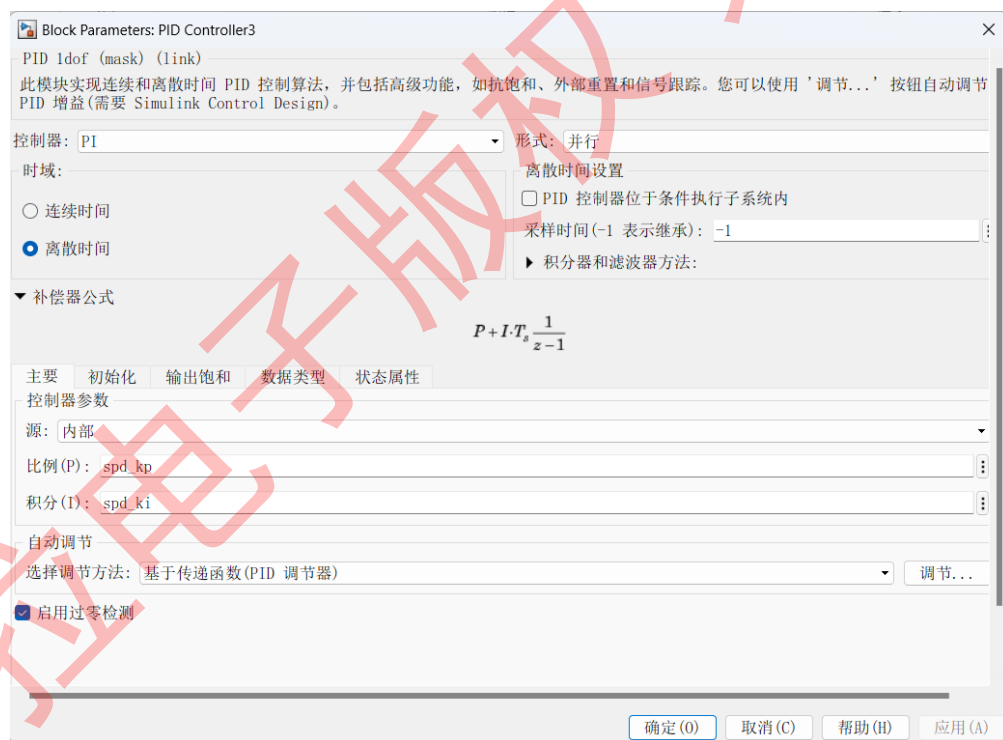




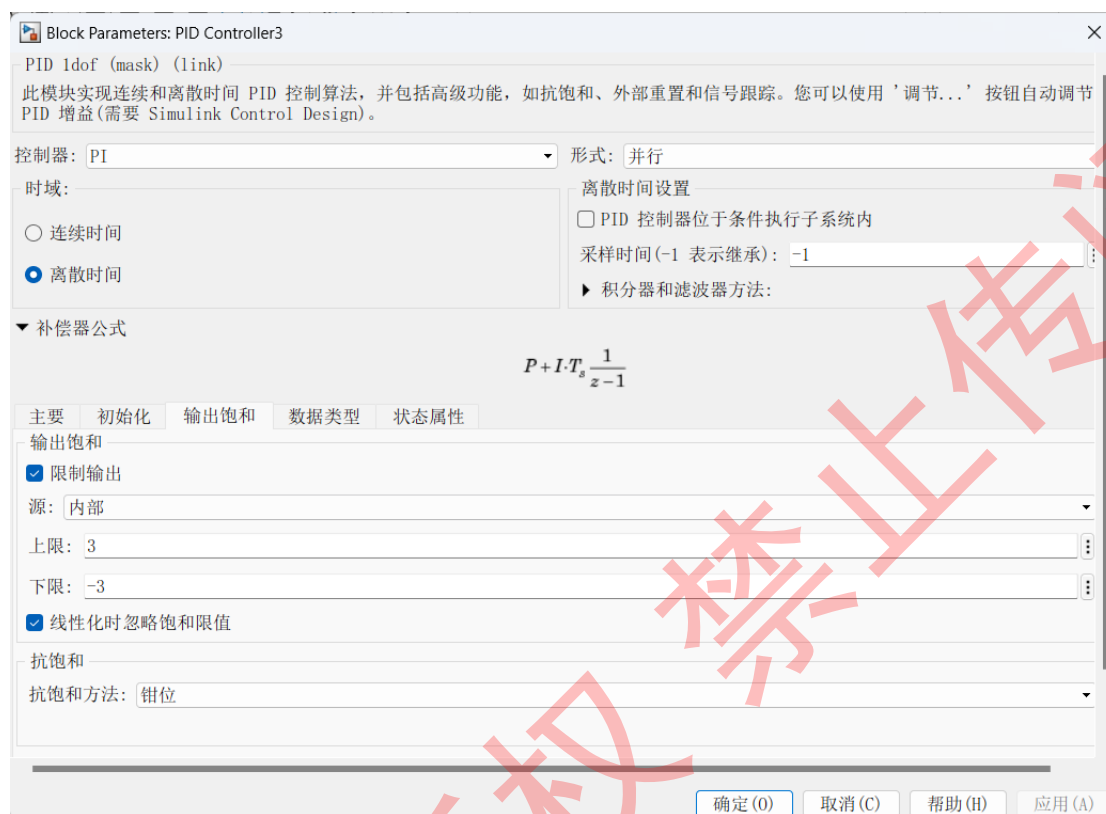
- ④ 速度环  $K_p K_i$  参数可参看如下公式设计；其中  $J$  为转动惯量， $\beta$  为闭环带宽；  
通道电机的转动惯量都很难得到一个较准确的值，一些电机出厂铭牌中也少有提到，根据经验值可以自己判断；此处我们设置的  $K_p$ : 0.3389,  $K_i$ : 0.0144;

$$\begin{cases} K_{p\omega} = \frac{\beta J}{1.5 p_n \psi_f} \\ K_{i\omega} = \beta K_{p\omega} \end{cases}$$

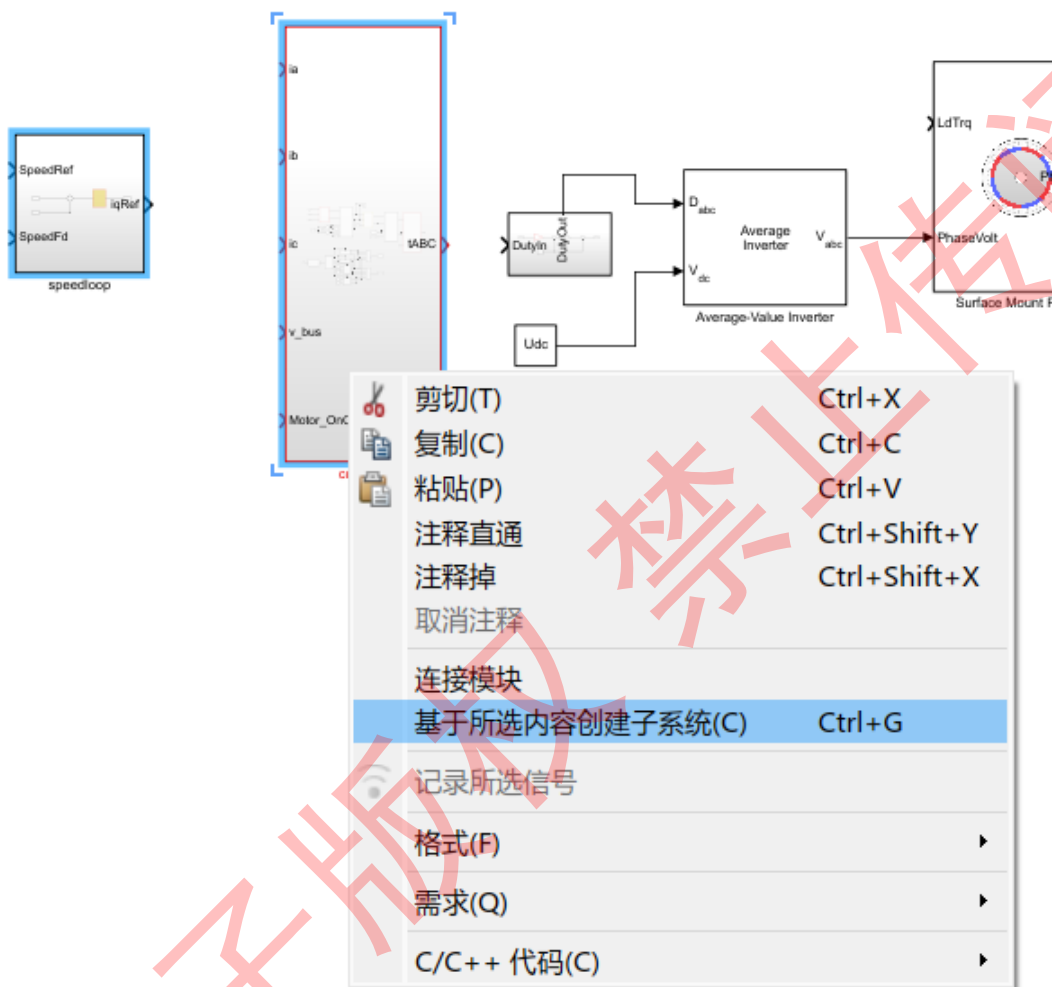
|                                                                                   |        |     |        |        |       |      |     |     |        |
|-----------------------------------------------------------------------------------|--------|-----|--------|--------|-------|------|-----|-----|--------|
|  | spd_kp | New | 0.3389 | single | [1 1] | real | [ ] | [ ] | Struct |
|  | spd_ki | New | 0.0144 | single | [1 1] | real | [ ] | [ ] | Struct |

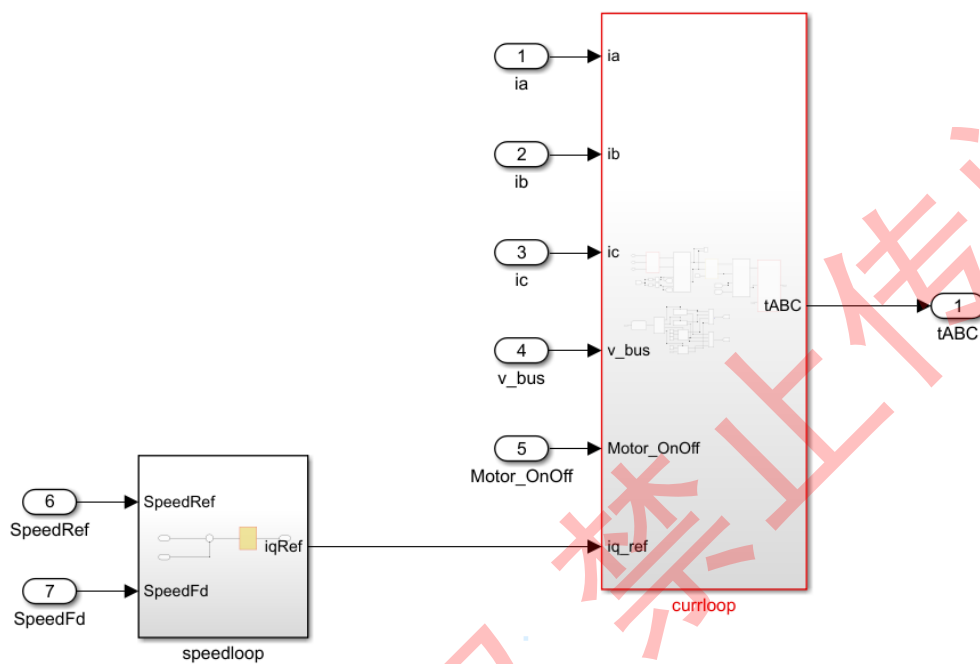


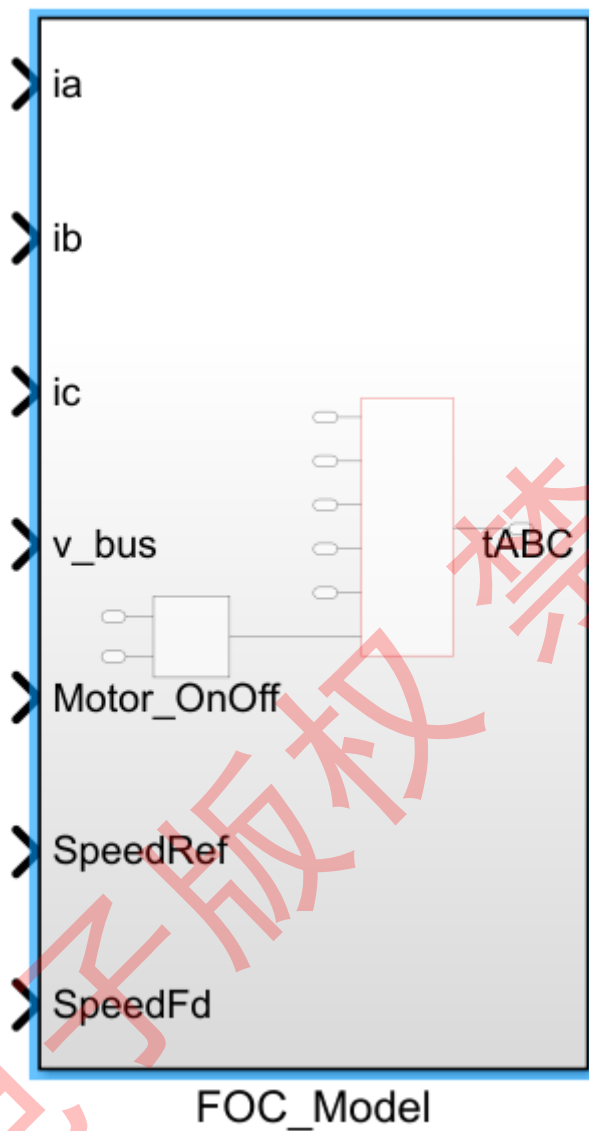
- ⑤ 速度环输出的  $q$  轴参考电流与开发板 DC 输入的母线电流能力相关，此处我们设置为  $\pm 3$ ;



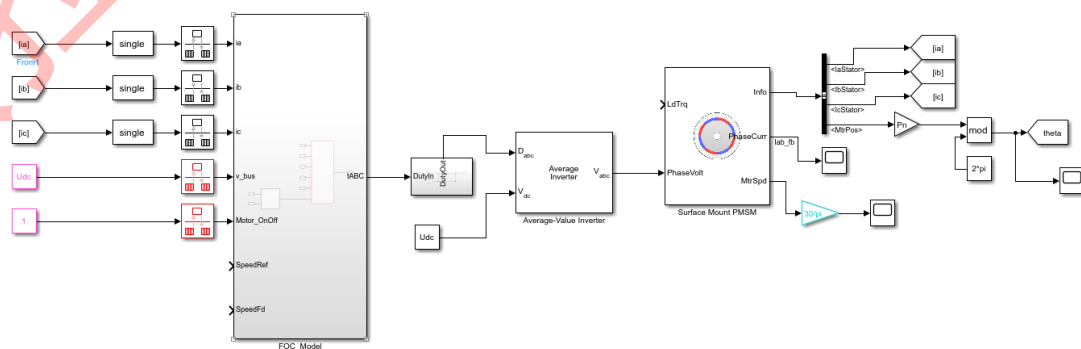
⑥ 将速度环与电流环子系统合在一起建立新的子系统 FOC\_Model;



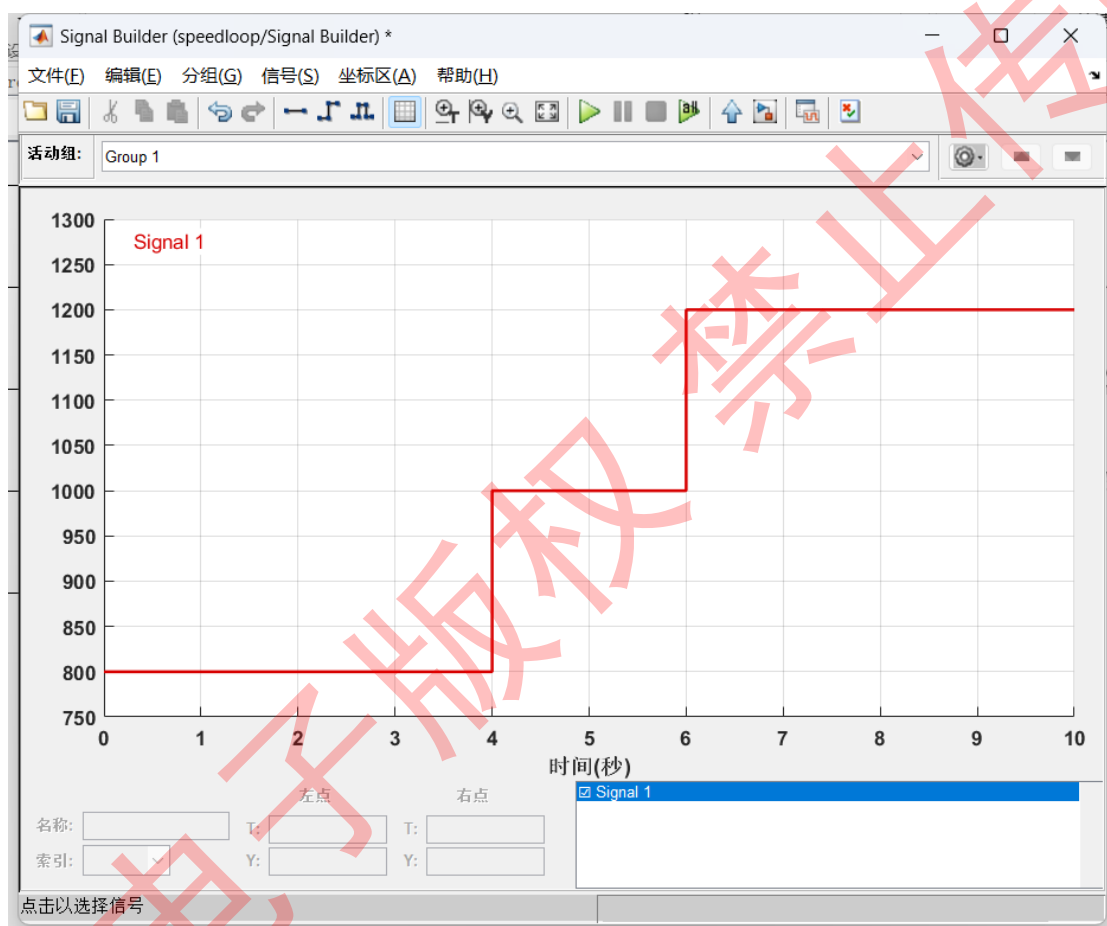




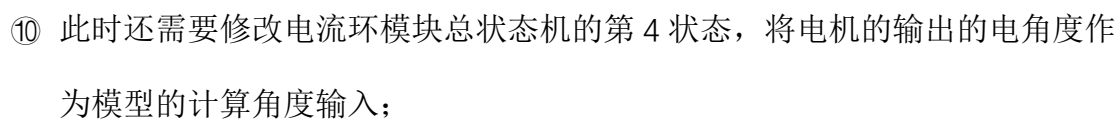
⑦ 连接模型的输入及输出；

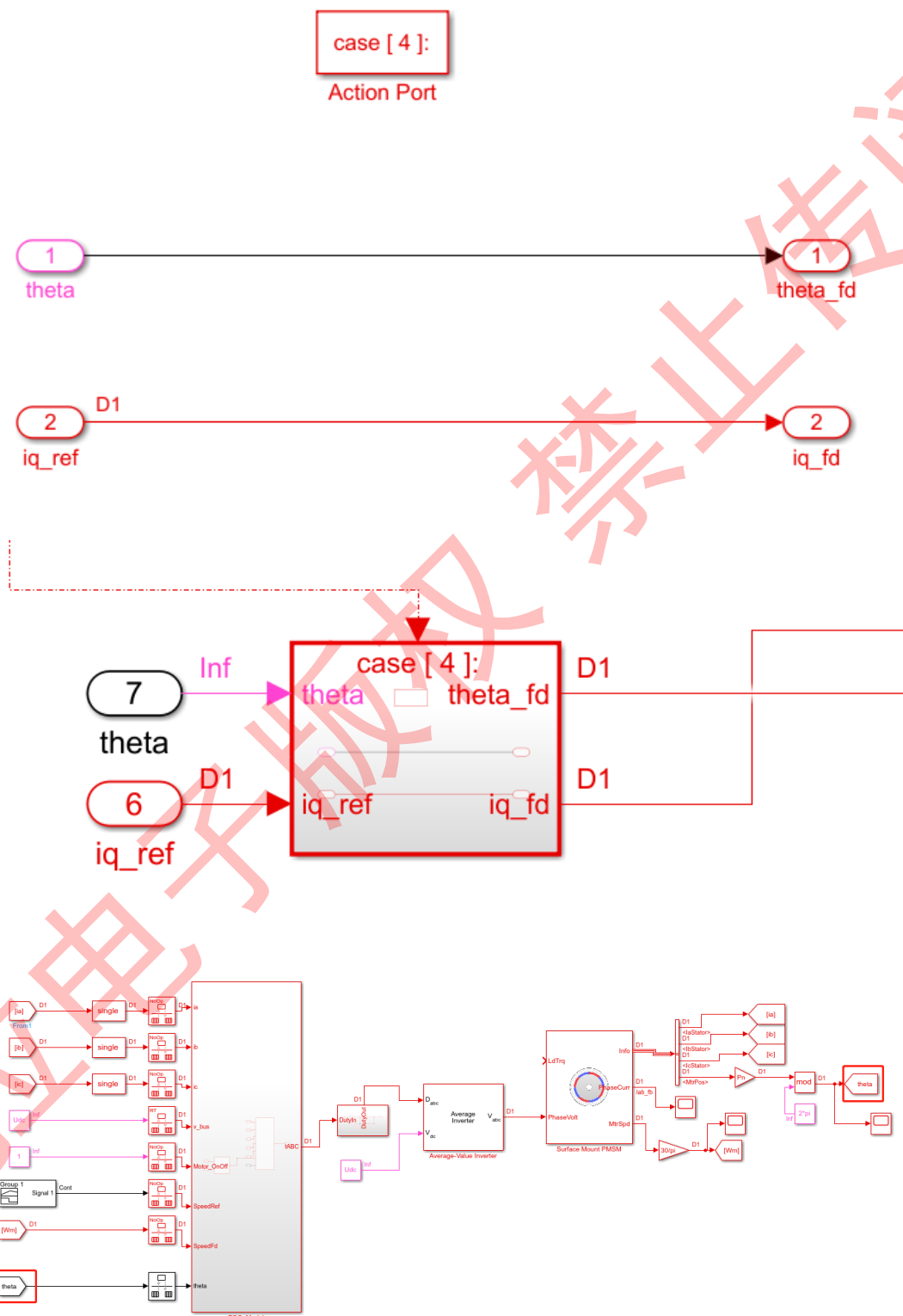


⑧ 使用 Signal Builder 模块来设置我们参考速度；



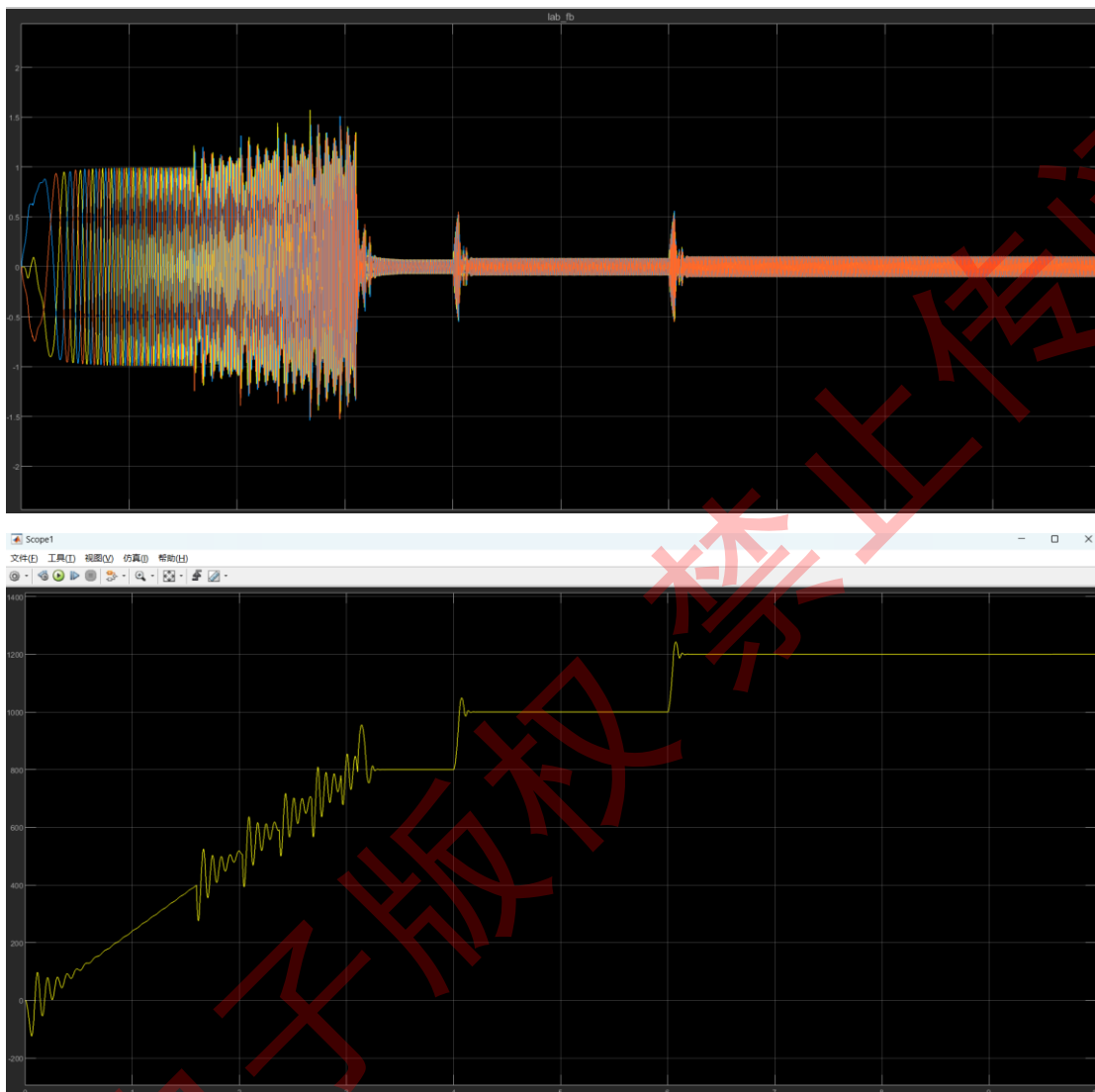
⑨ SpeedFd 的输入使用电机输出的机械速度；





- ⑪ 此时，FOC 的控制算法模型已经搭建完成，点击仿真按钮，查看三相电流波形及点输出速度；





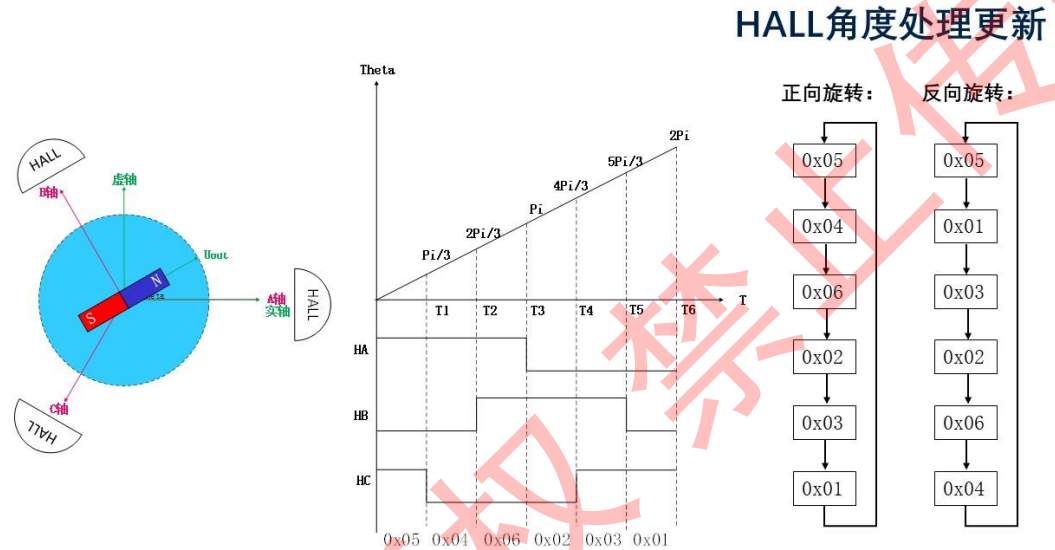
- ⑫ 可以看到三相电流速度闭环后，不再出现抖动，点击输出速度也完美收敛至我们期望的转速。

## 2. HALL 有感运行

配套的 BLDC 在定子侧以互差  $120^\circ$  电角度的位置安装 3 个霍尔元件 Ha, Hb, Hc。当转子转动时，霍尔元件会产生 3 个相位差  $120^\circ$  电角度的高低电平信号。

霍尔信号会将一个电周期划分为 6 个扇区，每个扇区  $60^\circ$  电角度，通过单片机的定时器捕获可以获得每个扇区运行的时间  $t$ 。为了获得准确的转子角度，电

机绕组 A 相接电源正极，BC 两相接电源负极，定子磁场与转子磁场相互作用，最终定位到转子零位点，也就是 A 相绕组的轴线位置。确定零位点后，根据霍尔的信号顺序，可以得到霍尔信号与转子位置的对应关系，如下表所示：

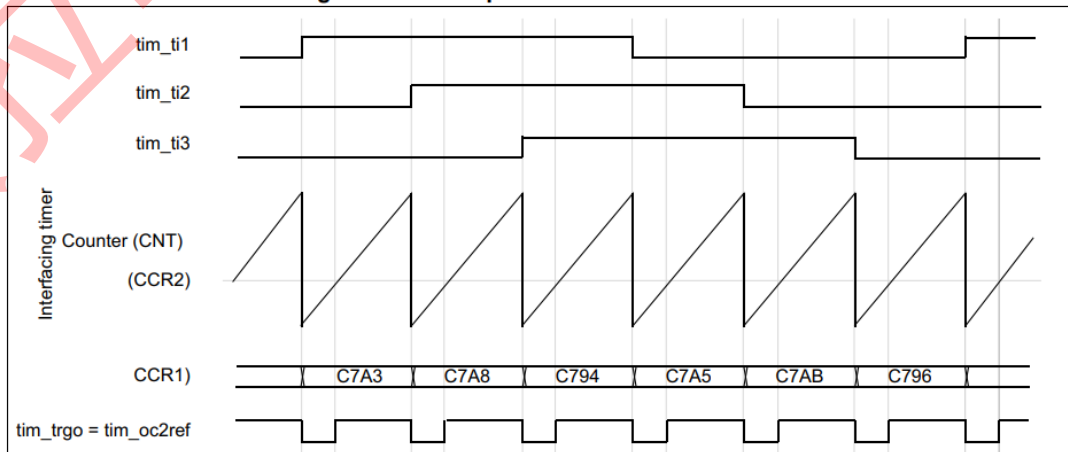


$$HallSpeed = \frac{PI}{3 * \triangle t}$$

$$\triangle t = HallTemp$$

$$HallTheta = \int_0^{\triangle t} HallSpeed$$

Figure 353. Example of Hall sensor interface

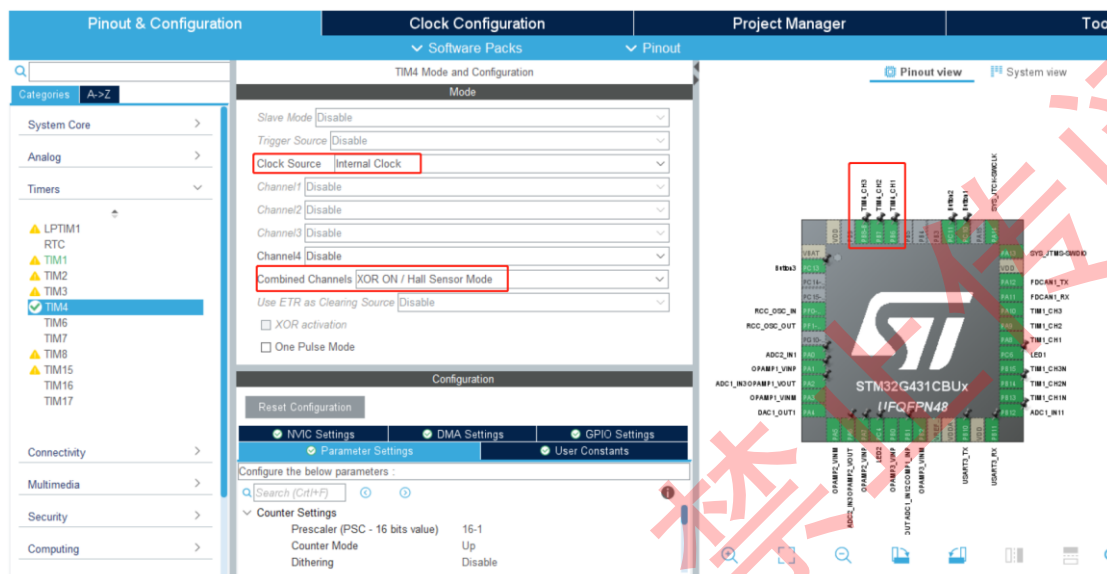


① 首先配置 STM32G4 的 HALL 接口；打开 STM32CUBEMX；使能 TIM4，选择

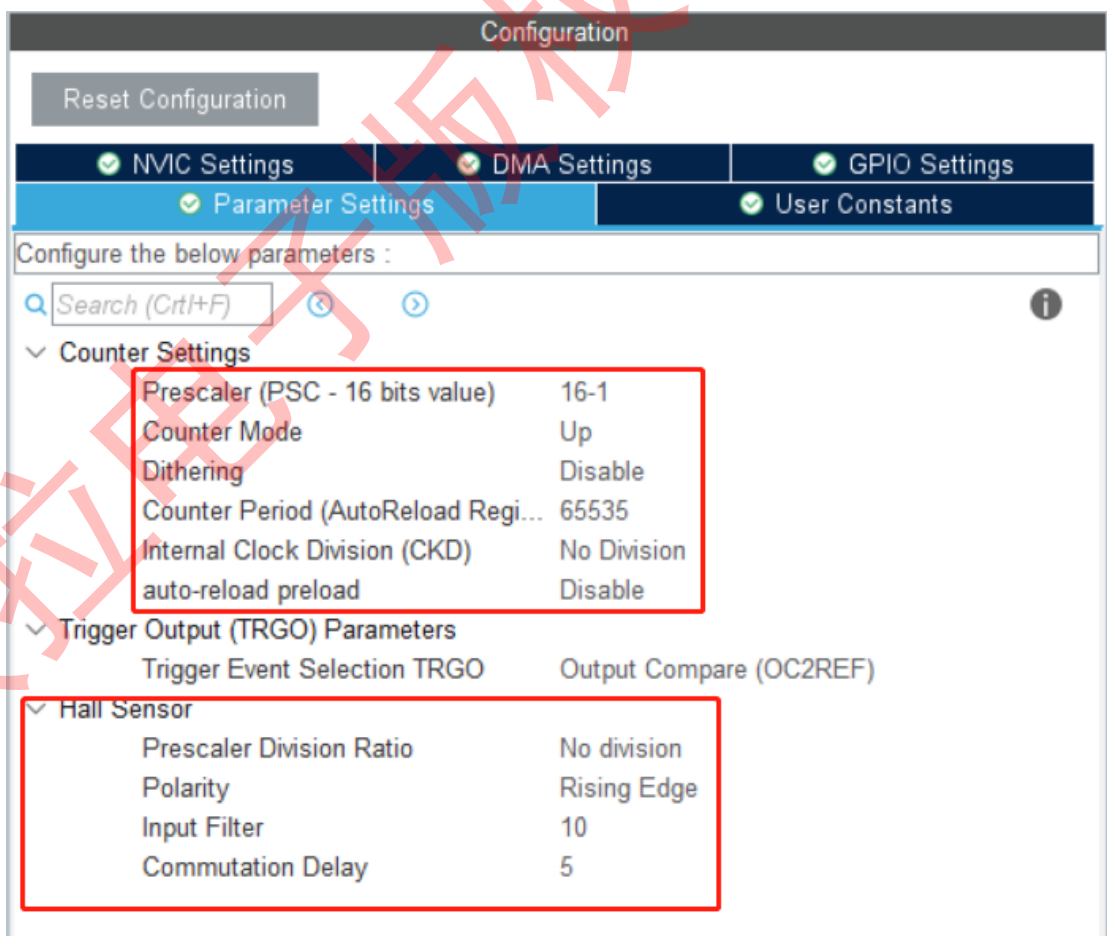
版本：V13

B 站 up 主：欧拉电子

内部时钟，在组合通道中选择 XOR ON/ HALL Sensor mode；



② 配置定时器参数：16 分频后，TIM4 的时钟为 10M；



③ 生成代码并打开 Keil 工程；

版本：V13

B 站 up 主：欧拉电子

④ 在 TIM 输入捕获回调函数中增加以下代码：

```

353 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
354 {
355     /* Prevent unused argument(s) compilation warning */
356     UNUSED(htim);
357     if(htim == &htim4)
358     {
359         HallTemp = HAL_TIM_ReadCapturedValue(&htim4, TIM_CHANNEL_1);
360         HallThetaAdd = (PI/3)/(HallTemp/1000000)/10000;
361         HallSpeed = (PI/3)/(HallTemp/1000000)*30/PI → 2PI
362         HallReadTemp = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_8);
363         HallReadTemp |= HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_7) << 1;
364         HallReadTemp |= HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_6) << 2;
365         if(HallReadTemp==0x05)
366         {
367             HallTheta = 0.0f+PHASE_SHIFT_ANGLE;
368         }
369         else if(HallReadTemp==0x04)
370         {
371             HallTheta = (PI/3.0f)+PHASE_SHIFT_ANGLE;
372         }
373         else if(HallReadTemp==0x06)
374         {
375             HallTheta = (PI*2.0f/3.0f)+PHASE_SHIFT_ANGLE;
376         }
377         else if(HallReadTemp==0x02)
378         {
379             HallTheta = PI+PHASE_SHIFT_ANGLE;
380         }
381         else if(HallReadTemp==0x03)
382         {
383             HallTheta = (PI*4.0f/3.0f)+PHASE_SHIFT_ANGLE;
384         }
385         else if(HallReadTemp==0x01)
386         {
387             HallTheta = (PI*5.0f/3.0f)+PHASE_SHIFT_ANGLE;
388         }
389         if(HallTheta<0.0f)
390         {
391             HallTheta += 2.0f*PI;
392         }
393         else if(HallTheta>(2.0f*PI))
394         {
395             HallTheta -= 2.0f*PI;
396         }
397     }
398 }

```

将此处PI改为2PI

⑤ 其中 HALL 速度角度计算计算公式：

$$HallSpeed = \frac{PI}{3 * \triangle t}$$

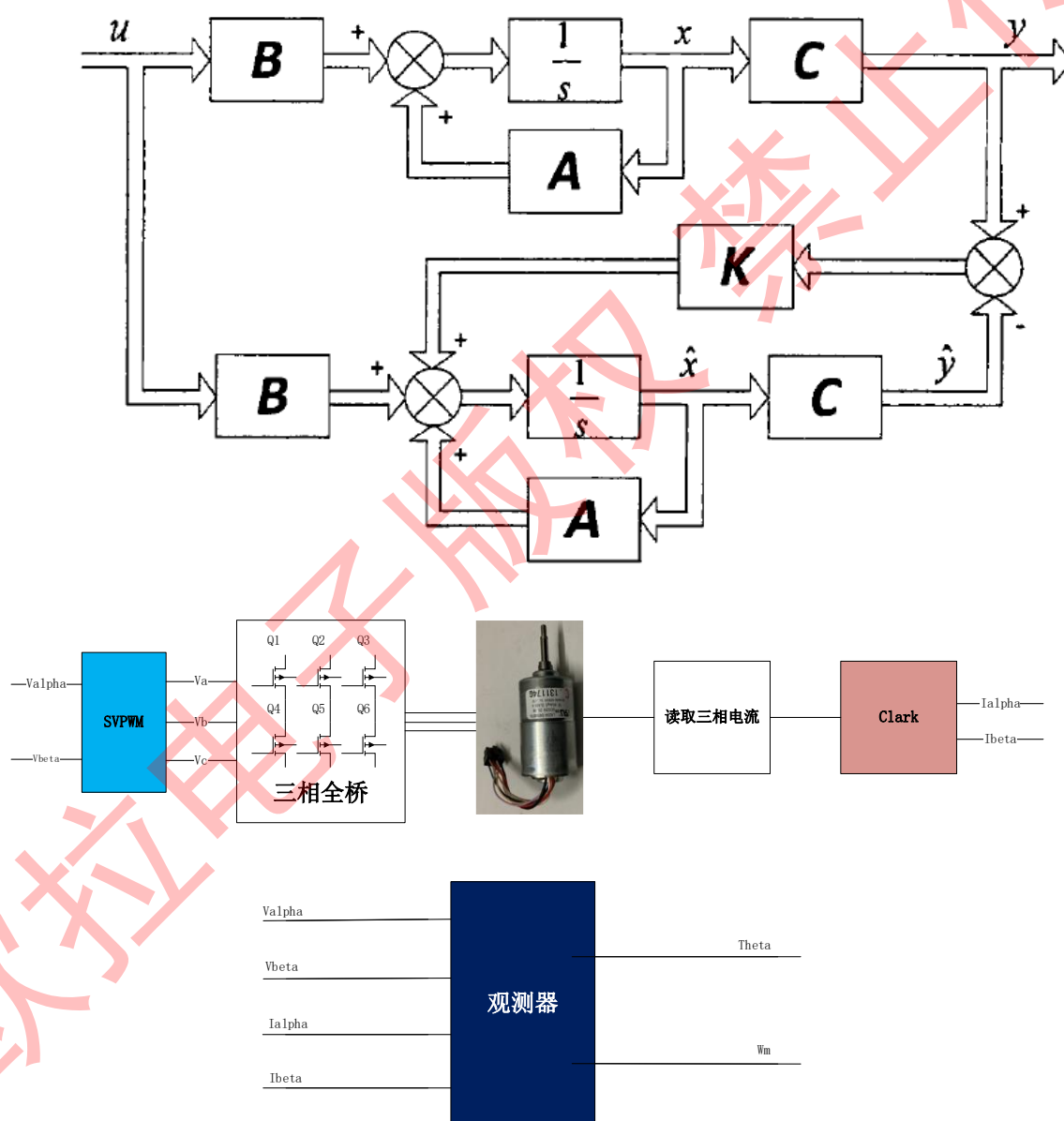
$$\triangle t = HallTemp$$

$$HallTheta = \int_0^{\triangle t} HallSpeed$$

- ⑥ 在 ADC 注入组中断回调函数中输入，HALL 传感器得到的速度和角度；

## 第十八章：龙伯格观测器

状态观测器框图：



$$\begin{cases} u_\alpha = R i_\alpha + \frac{di_\alpha}{dt} L + e_\alpha \\ u_\beta = R i_\beta + \frac{di_\beta}{dt} L + e_\beta \end{cases}$$

状态观测器方程：

$$\begin{aligned}\dot{\hat{x}} &= \mathbf{A}\hat{x} + \mathbf{B}u + \mathbf{K}(y - \hat{y}) = (\mathbf{A} - \mathbf{K}\mathbf{C})\hat{x} + \mathbf{K}y + \mathbf{B}u \\ \hat{y} &= \mathbf{C}\hat{x}\end{aligned}$$

## 1. 龙伯格观测器及 PLL 建模

①  $\alpha$   $\beta$  轴电压方程：

$$\begin{bmatrix} \frac{di_{\alpha}}{dt} \\ \frac{di_{\beta}}{dt} \\ \frac{de_{\alpha}}{dt} \\ \frac{de_{\beta}}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} \\ 0 & 0 & 0 & -p_n\omega_r \\ 0 & 0 & p_n\omega_r & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \\ e_{\alpha} \\ e_{\beta} \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{\alpha} \\ u_{\beta} \end{bmatrix}$$

$$\begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \\ e_{\alpha} \\ e_{\beta} \end{bmatrix}$$

② A、B、C 分别为系统的状态矩阵、输入矩阵与输出矩阵；

$$\mathbf{A} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} \\ 0 & 0 & 0 & -p_n\omega_r \\ 0 & 0 & p_n\omega_r & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

③ 系统的状态矢量  $x$ ，输入矢量  $u$  与输出矢量  $y$  为

$$x = [i_\alpha \quad i_\beta \quad e_\alpha \quad e_\beta]^T$$

$$u = [u_\alpha \quad u_\beta]^T$$

$$y = [i_\alpha \quad i_\beta]^T$$

④ 龙伯格观测器模型如下：

$$\begin{bmatrix} \frac{di_\alpha}{dt} \\ \frac{di_\beta}{dt} \\ \frac{d\hat{e}_\alpha}{dt} \\ \frac{d\hat{e}_\beta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} \\ 0 & 0 & 0 & -\omega_e \\ 0 & 0 & \omega_e & 0 \end{bmatrix} \begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \\ \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} + \begin{bmatrix} K1 & 0 \\ 0 & K1 \\ K2 & 0 \\ 0 & K2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} i_\alpha - \hat{i}_\alpha \\ i_\beta - \hat{i}_\beta \\ e_\alpha - \hat{e}_\alpha \\ e_\beta - \hat{e}_\beta \end{bmatrix}$$

⑤ 系统增益  $K$  矩阵如下：

$$\mathbf{K} = \begin{bmatrix} K_1 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & K_2 \end{bmatrix}$$

$$\begin{bmatrix} \frac{di_\alpha}{dt} \\ \frac{di_\beta}{dt} \\ \frac{d\hat{e}_\alpha}{dt} \\ \frac{d\hat{e}_\beta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} \\ 0 & 0 & 0 & -\omega_e \\ 0 & 0 & \omega_e & 0 \end{bmatrix} \begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \\ \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} + \begin{bmatrix} K_1 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & K_2 \end{bmatrix} \begin{bmatrix} i_\alpha - \hat{i}_\alpha \\ i_\beta - \hat{i}_\beta \\ e_\alpha - \hat{e}_\alpha \\ e_\beta - \hat{e}_\beta \end{bmatrix}$$

⑥ 将龙伯格观测器模型离散化：

$$\frac{di_\alpha}{dt} \rightarrow \frac{I_{ahat}(k+1) - I_{ahat}(k)}{T}$$

$$\hat{i}_\alpha(k+1) = \left(1 - \frac{R_s * T}{L_s}\right) * \hat{i}_\alpha(k) - \left(\frac{T}{L_s}\right) * \hat{e}_\alpha(k) + \left(\frac{T}{L_s}\right) * u_\alpha(k) + (G1 * T) * (\hat{i}_\alpha(k) - i_\alpha(k))$$

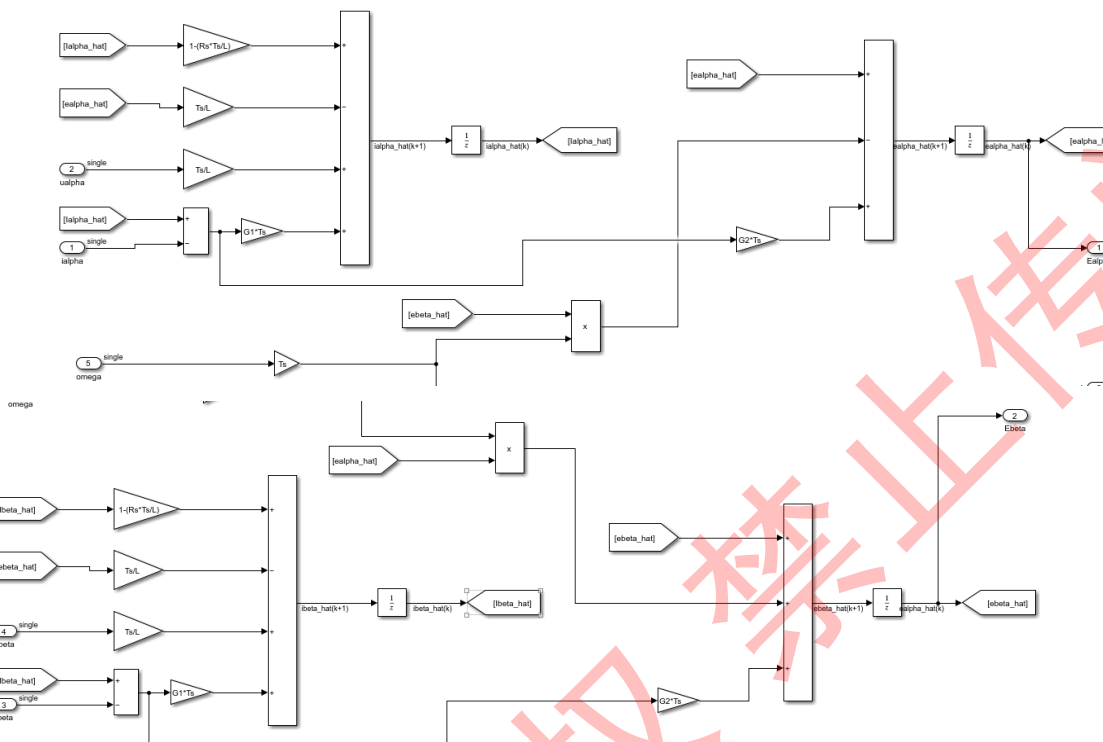
$$\hat{i}_\beta(k+1) = \left(1 - \frac{R_s * T}{L_s}\right) * \hat{i}_\beta(k) - \left(\frac{T}{L_s}\right) * \hat{e}_\beta(k) + \left(\frac{T}{L_s}\right) * u_\beta(k) + (G1 * T) * (\hat{i}_\beta(k) - i_\beta(k))$$

$$\hat{e}_\alpha(k+1) = \hat{e}_\alpha(k) - we * T * \hat{e}_\beta(k) + (G2 * T) * (\hat{i}_\alpha(k) - i_\alpha(k))$$

$$\hat{e}_\beta(k+1) = \hat{e}_\beta(k) + we * T * \hat{e}_\alpha(k) + (G2 * T) * (\hat{i}_\beta(k) - i_\beta(k))$$

⑦ 基于离散化后的龙伯格观测器在 simulink 中建立模型：

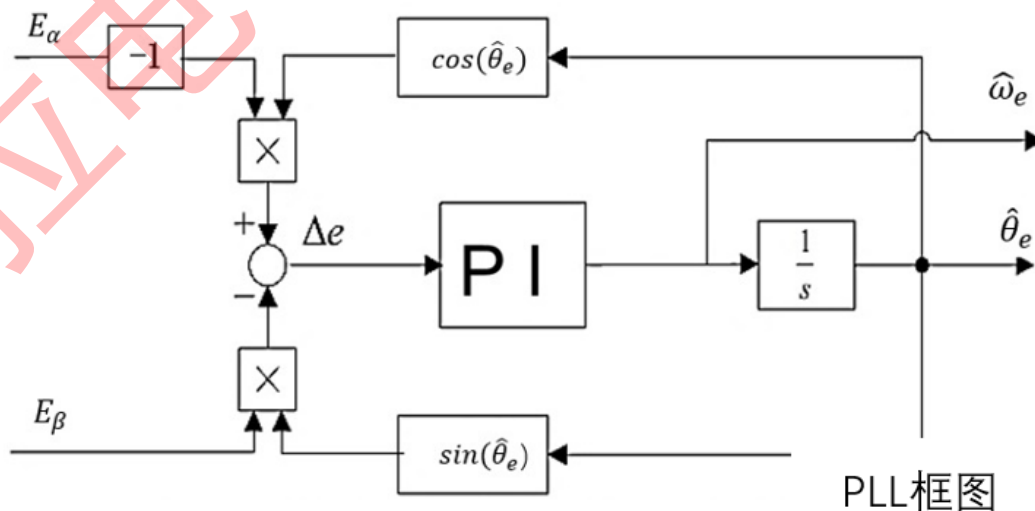




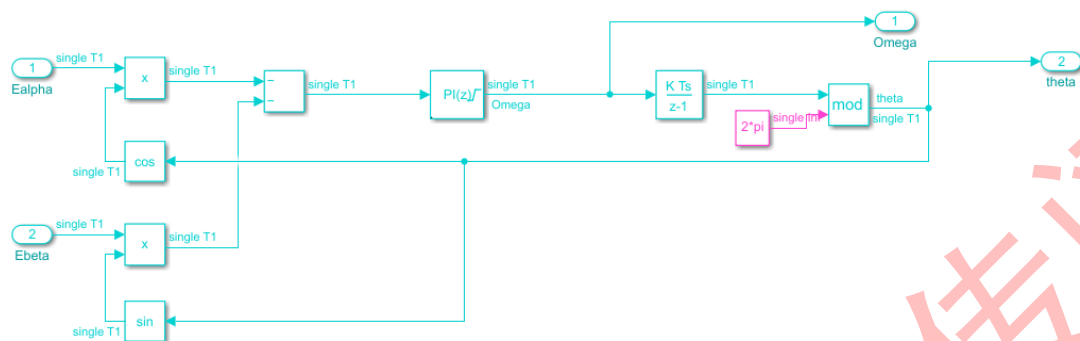
⑧ 经多次实验，获得一组较好的仿真增益参数为

$$\begin{cases} K1 = -4000 \\ K2 = 14000 \end{cases}$$

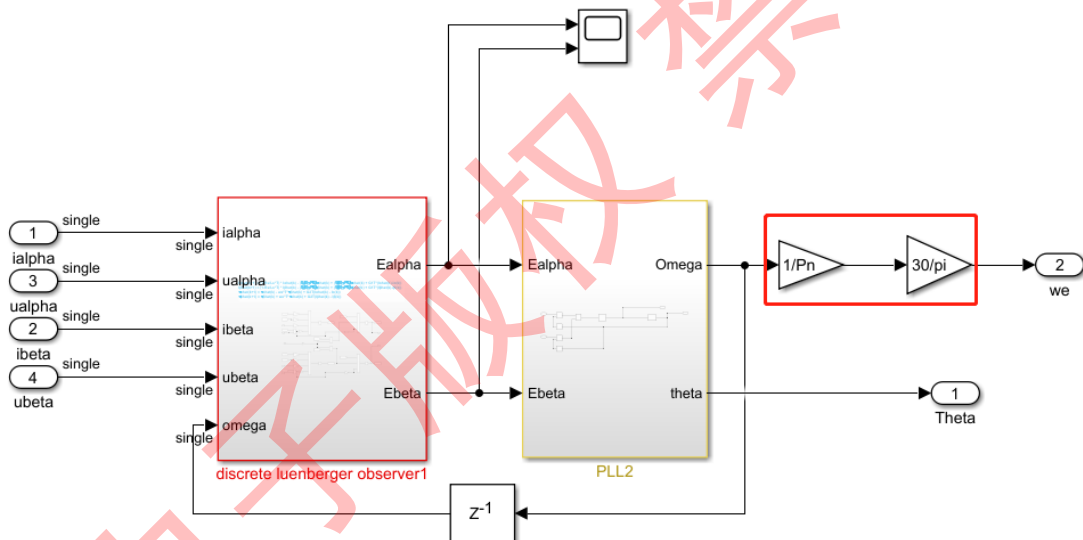
⑨ 建立锁相环模型：



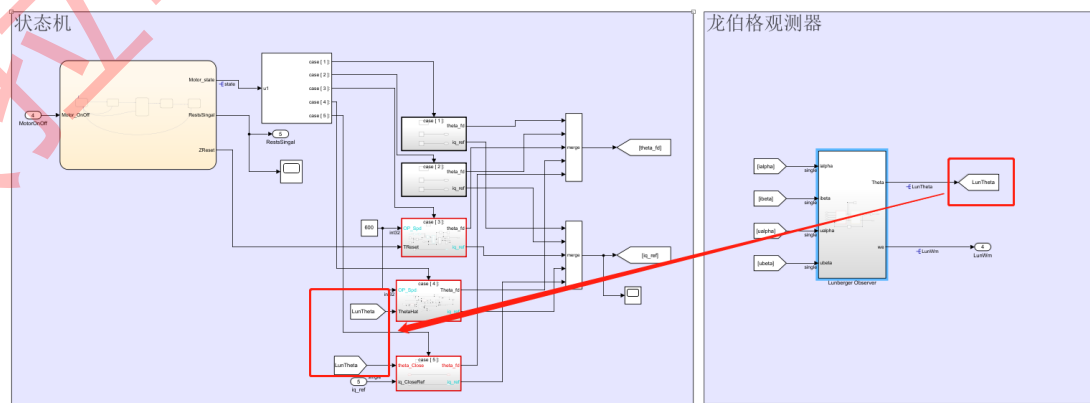
PLL框图

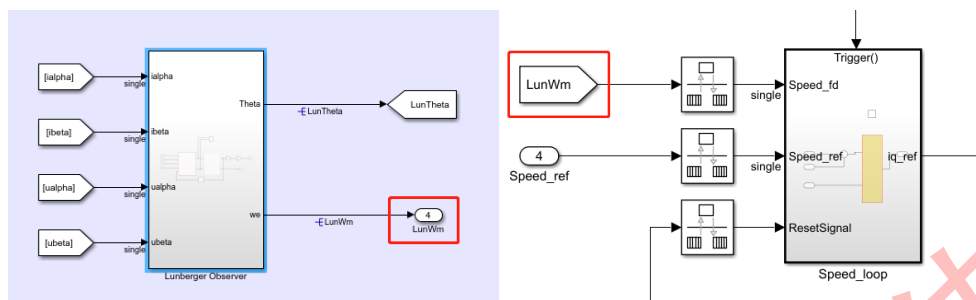


⑩ 观测到的转速单位 rad/s，需转换为 rpm/min；

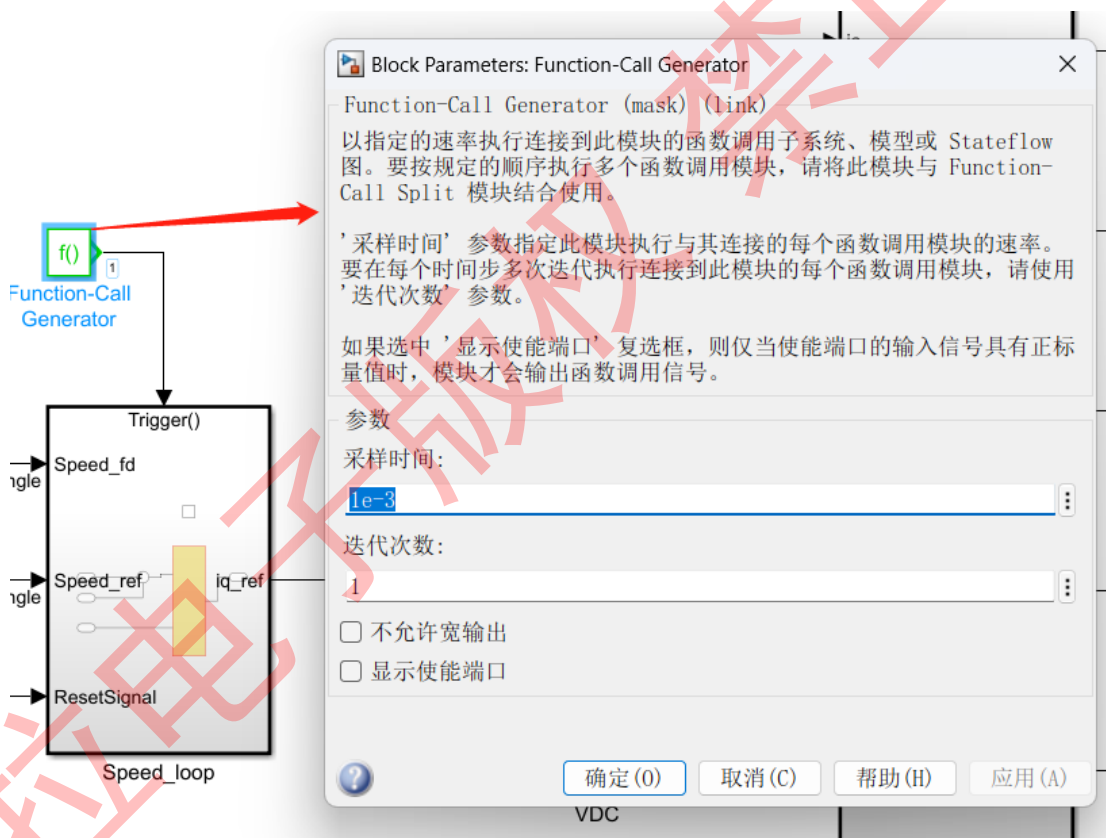


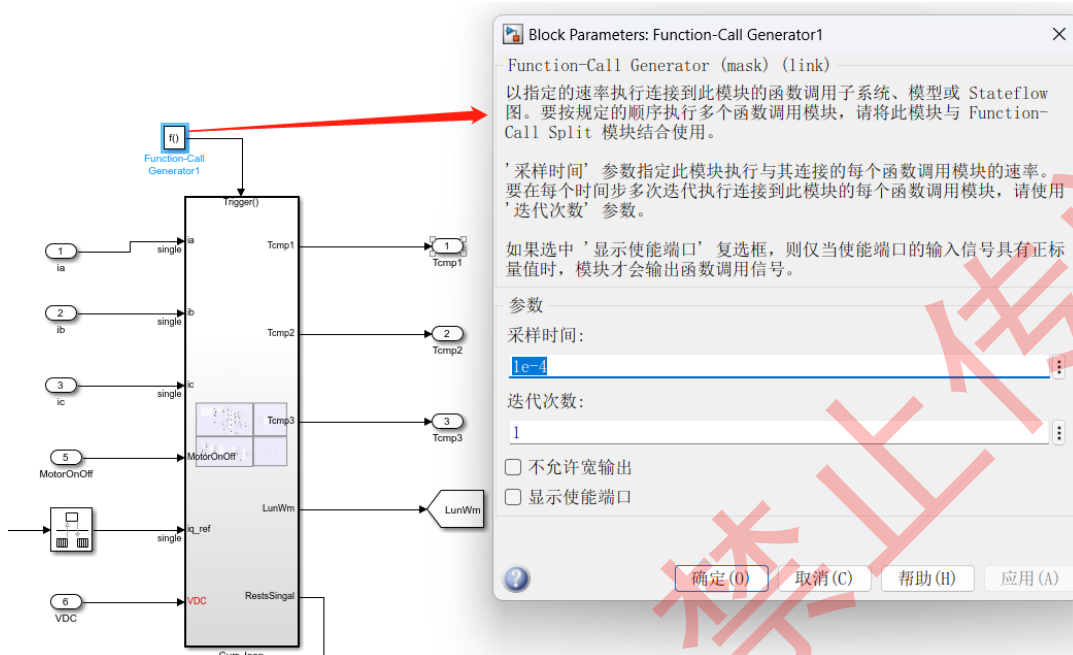
⑪ 将观测器输出的角度及转速作为闭环的角度和参考速度输入；



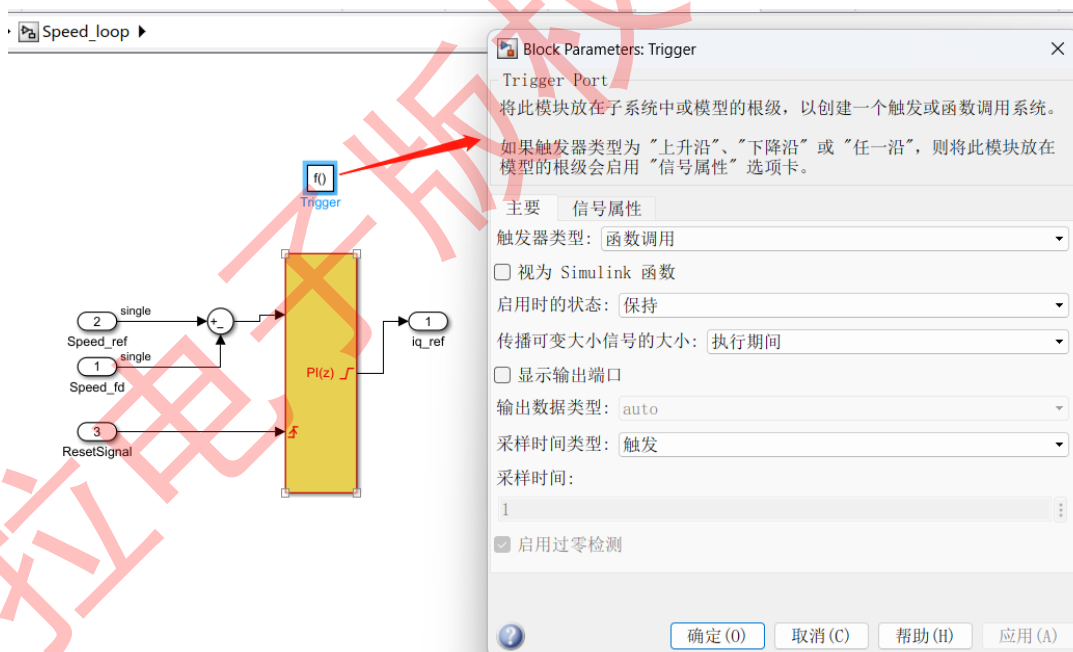


- ⑫ 由于速度环不需要太高的执行频率，故将速度环的采样时间改为  $1e-3$ ；电流环的采样时间保持  $1e-4$ ；





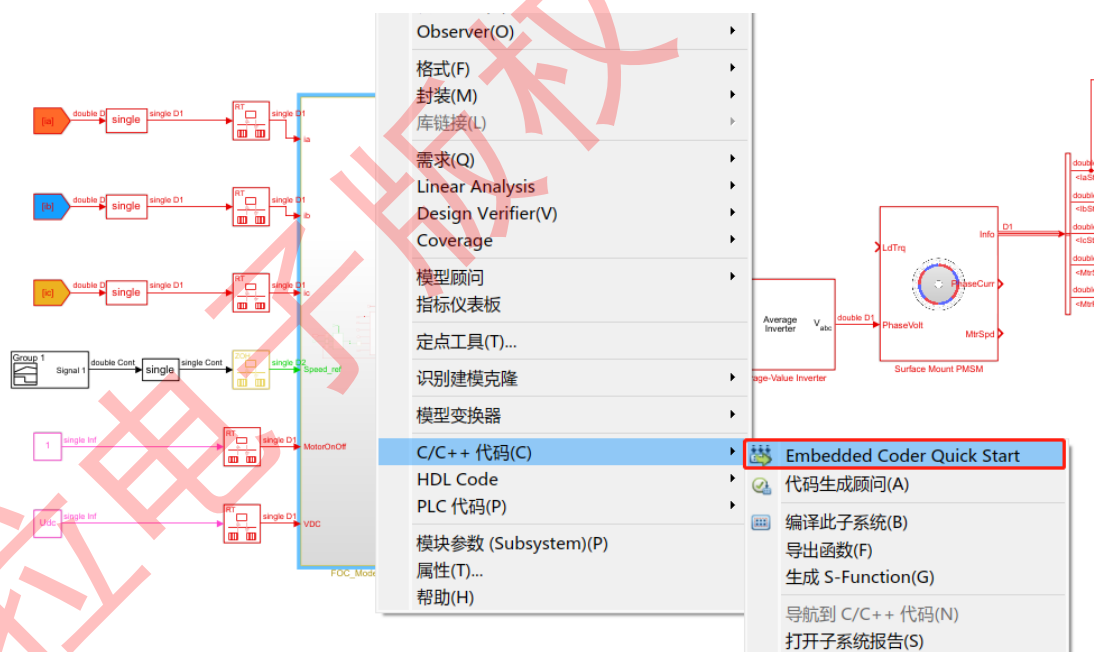
⑬ 在速度环及电流环的内部增加 trigger 模块;



⑭ 模型搭建完成后, 电机仿真按钮, 对比观测速度与电机实际转速;



- ⑮ 速度收敛较为完美，证明所搭建的观测器没有问题；右键点击 FOC\_Model 模块，点击生成代码；



- ⑯ 生成代码的流程如第十章流程；

- ⑰ 将生成的 FOC 代码放进 Keil 工程中 MATLAB 文件夹；

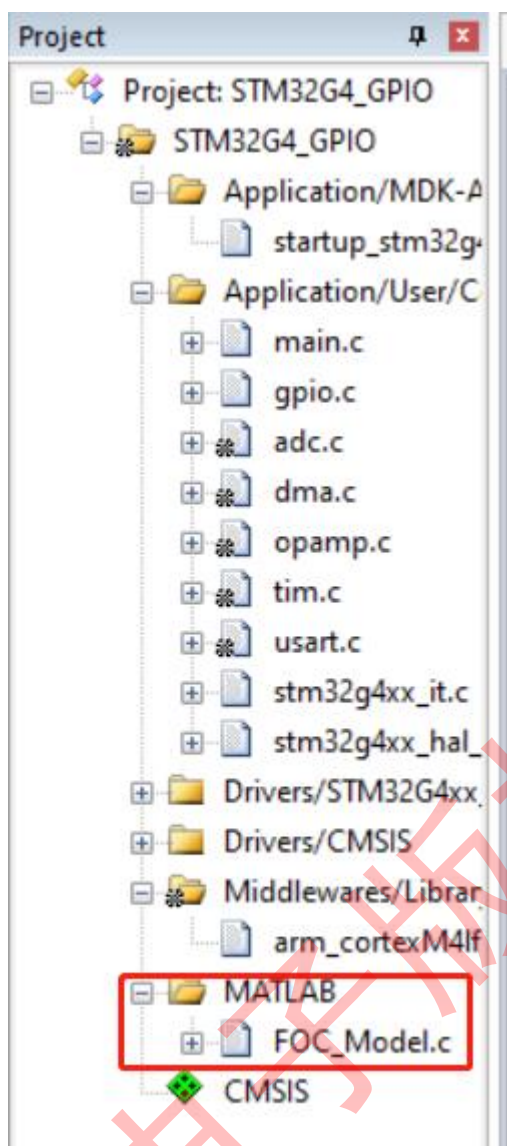
.Lunberger > BLDC > BLDC\_Lun\_MODEL > FOC\_Model\_ert\_rtw > 在 FOC\_Model\_ert\_rt...

| 名称                 | 修改日期            | 类型             | 大小    |
|--------------------|-----------------|----------------|-------|
| defines.txt        | 2023/8/22 21:11 | 文本文档           | 1 KB  |
| ert_main.c         | 2023/8/22 21:11 | C 源文件          | 4 KB  |
| FOC_Model.bat      | 2023/8/14 20:39 | Windows 批处理... | 1 KB  |
| FOC_Model.c        | 2023/8/22 21:11 | C 源文件          | 48 KB |
| FOC_Model.h        | 2023/8/22 21:11 | C Header 源文件   | 32 KB |
| FOC_Model.mk       | 2023/8/22 21:11 | Makefile 源文件   | 11 KB |
| FOC_Model.rsp      | 2023/8/20 12:23 | RSP 文件         | 1 KB  |
| FOC_Model_ref.rsp  | 2023/8/14 20:39 | RSP 文件         | 0 KB  |
| modelsources.txt   | 2023/8/22 21:11 | 文本文档           | 1 KB  |
| rtw_proj.tmw       | 2023/8/14 20:39 | TMW 文件         | 1 KB  |
| rtwtypes.h         | 2023/8/22 21:11 | C Header 源文件   | 4 KB  |
| rtwtypeschksum.mat | 2023/8/22 21:11 | MAT 文件         | 2 KB  |

> BLDC > BLDC\_Lun\_Code > STM32G4\_Lun > MATLAB 在 MATLAB 中搜索

| 名称          | 修改日期            | 类型           | 大小    |
|-------------|-----------------|--------------|-------|
| FOC_Model.c | 2023/8/22 21:11 | C 源文件        | 48 KB |
| FOC_Model.h | 2023/8/22 21:11 | C Header 源文件 | 32 KB |
| rtwtypes.h  | 2023/8/22 21:11 | C Header 源文件 | 4 KB  |

⑱ 打开 Keil 工程，在工程目录中 MATLAB 文件夹增加 FOC\_MODEL.C 文件；



⑲ Main.c 中包含 FOC\_Model.h:

```

/* Private includes -----
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
#include "FOC_Model.h"
/* USER CODE END Includes */

```

⑳ 通过电位器旋钮调节算法模型输入的参考转速，转速在 300-1200 之间；

ADC 采样的母线电压作为模型 Vbus 输入；

```

7   {
8       HAL_ADC_Start(&hadc1);
9       HAL_ADC_Start(&hadc2);
0       adc1_in4 = HAL_ADC_GetValue(&hadc1);
1       adc_vbus = HAL_ADC_GetValue(&hadc2);
2       rtU.VDC = adc_vbus*3.3f/4096*26;
3       finalspeed = (adc1_in4>>4) * 15;
4       if(finalspeed >=0 && finalspeed <300)
5       {
6           finalspeed = 300;
7       }
8       if(finalspeed > 1200)
9       {
0           finalspeed = 1200;
1       }
2       rtU.Speed_ref = finalspeed;
3       HAL_Delay(10);

```

- 21 在 ADC 中断中读取三相电流值输入算法模型，执行算法代码；将算法模型输出作为 TIM1 的比较值写入寄存器；同时通过 UART 上传数据至 VOFA 上位机显示波形；

```

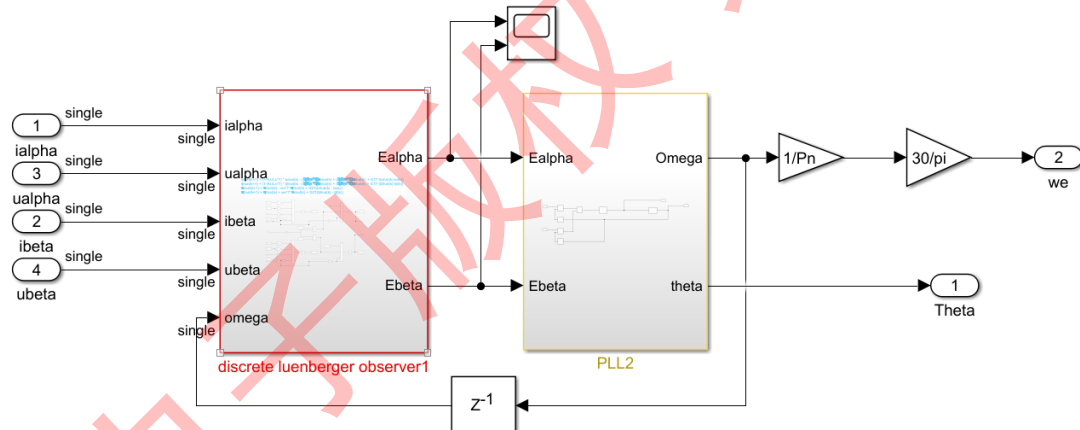
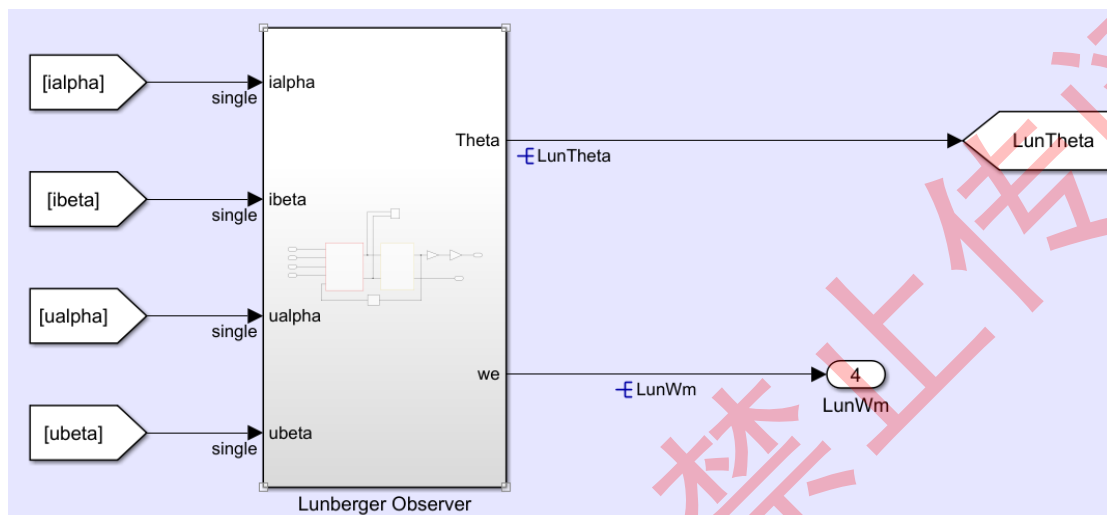
338   {
339       adc1_in1 = hadc1.Instance->JDR1;
340       adc1_in3 = hadc1.Instance->JDR2;
341       adc1_in2 = hadc2.Instance->JDR1;
342       rtU.ia = (adc1_in1 - IA_Offset)*0.02197f;
343       rtU.ib = (adc1_in2 - IB_Offset)*0.02197f;
344       rtU.ic = (adc1_in3 - IC_Offset)*0.02197f;
345       FOC_Model_step();
346       TIM1->CCR1 = rtY.Tcmp1;
347       TIM1->CCR2 = rtY.Tcmp2;
348       TIM1->CCR3 = rtY.Tcmp3;
349       load_data[0] = rtU.ia;
350       load_data[1] = LunTheta;
351       load_data[2] = LunWm;
352       load_data[3] = rtU.Speed_ref;
353       load_data[4] = 0;
354       memcpy(tempData, (uint8_t *)&load_data, sizeof(load_data));
355       HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 6*4);
356   }

```

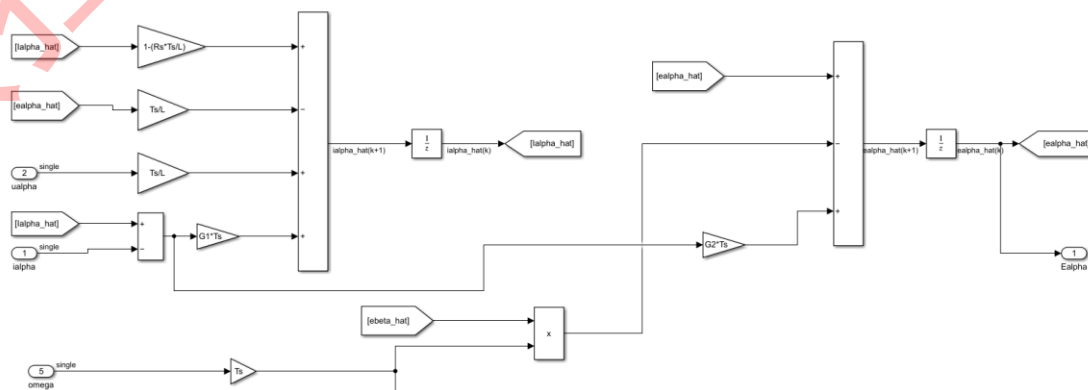
- 22 编译代码；

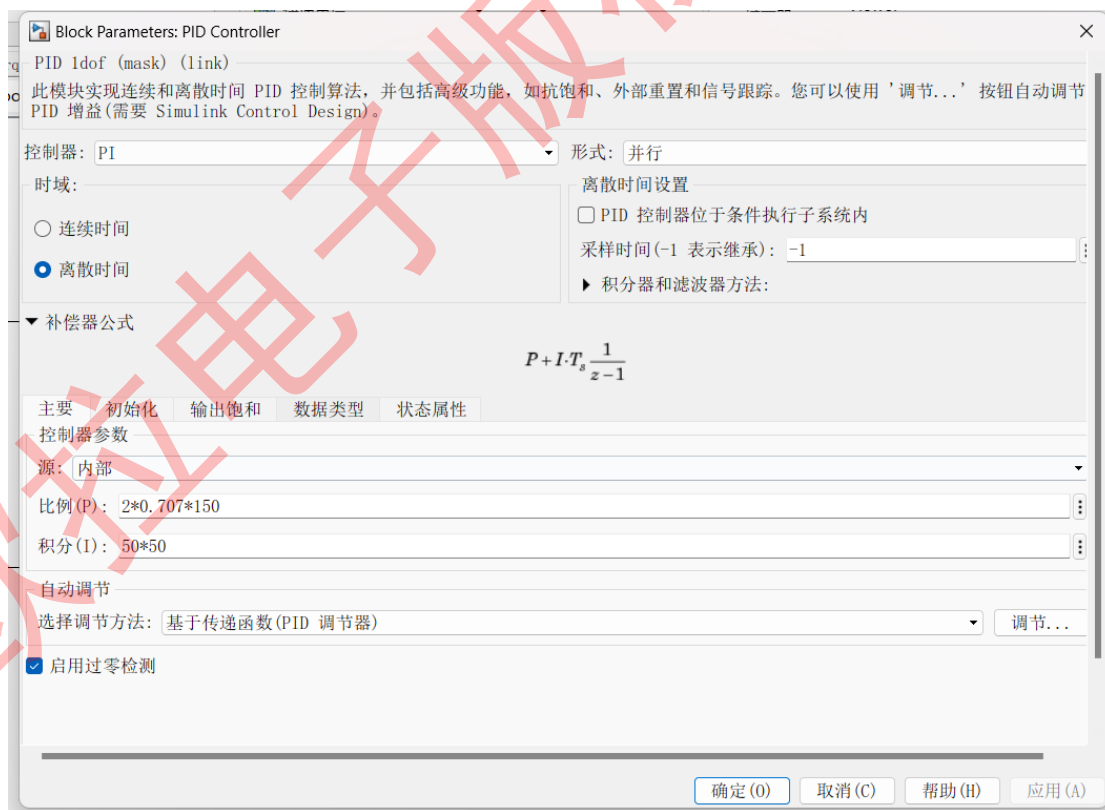
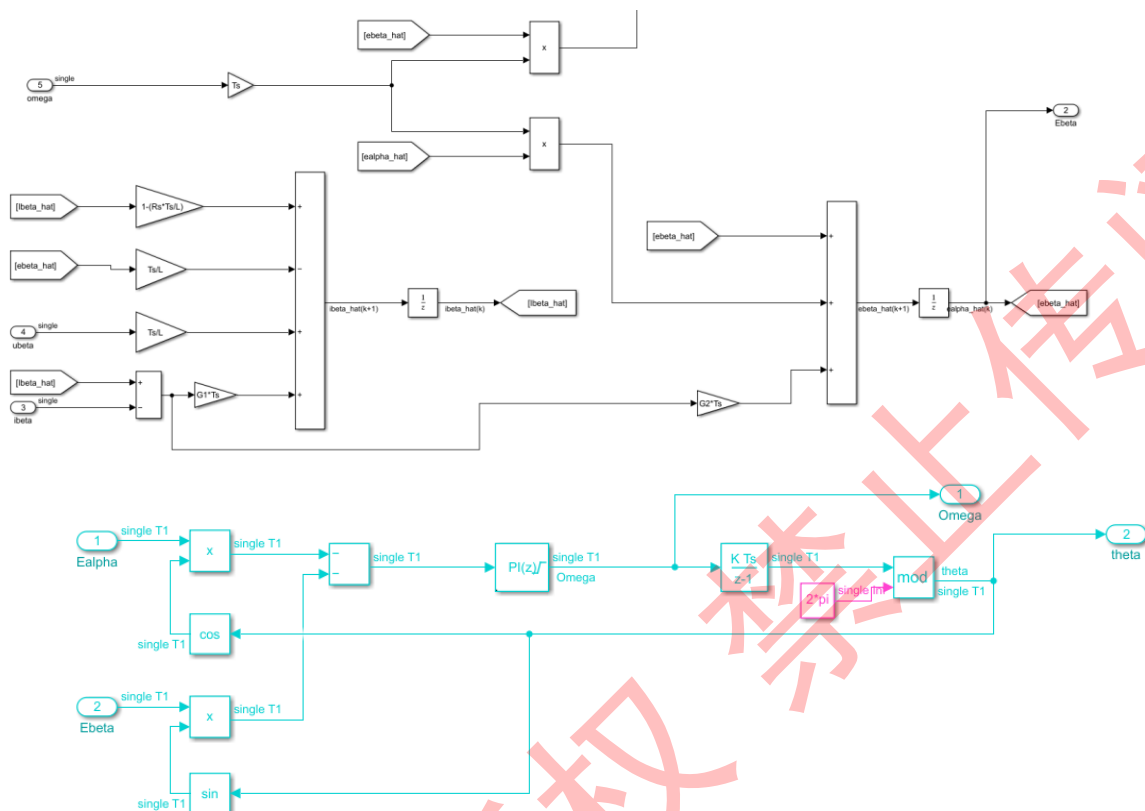


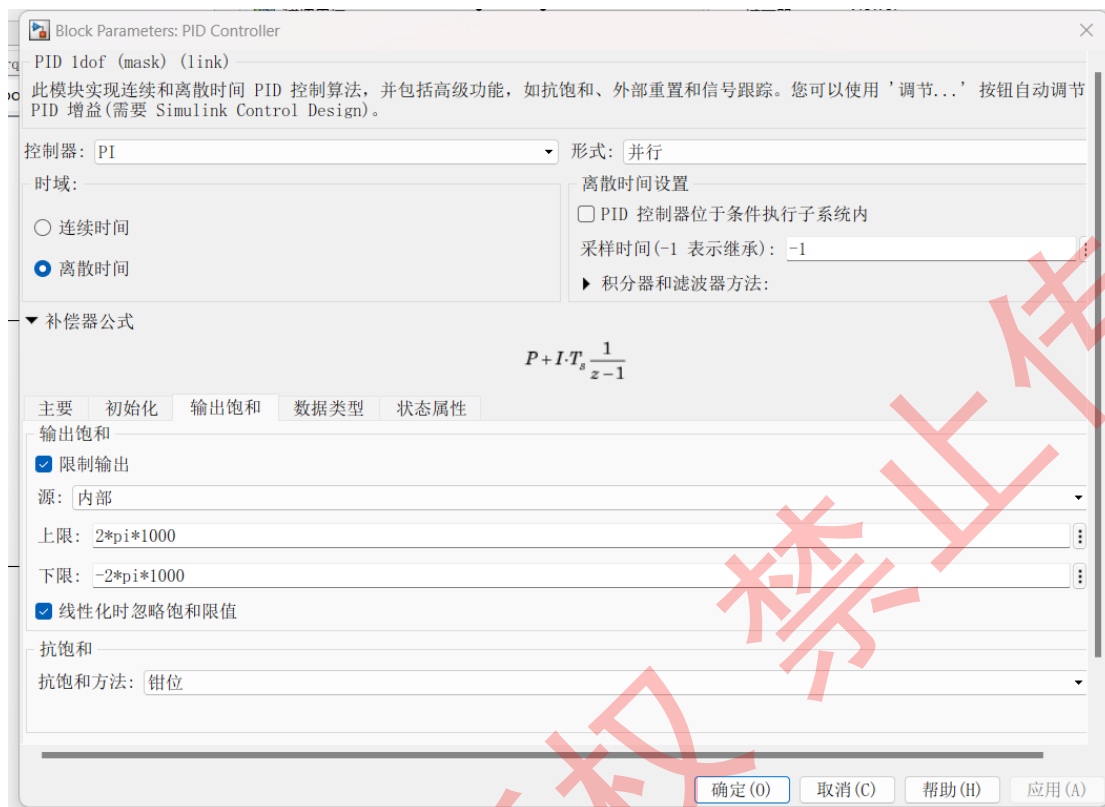
## 2. 龙伯格观测器模块内部及数据字典截图



$$\begin{aligned} \text{la} \hat{\text{h}} \text{at}(k+1) &= (1 - R_s / L_s * T) * \text{la} \hat{\text{h}} \text{at}(k) - (1 / L_s * T) \text{ea} \hat{\text{h}} \text{at}(k) + (1 / L_s * T) \text{ua} \hat{\text{h}} \text{at}(k) + G1T * (\text{la} \hat{\text{h}} \text{at}(k) - \text{la}(k)) \\ \text{lb} \hat{\text{h}} \text{at}(k+1) &= (1 - R_s / L_s * T) * \text{lb} \hat{\text{h}} \text{at}(k) - (1 / L_s * T) \text{eb} \hat{\text{h}} \text{at}(k) + (1 / L_s * T) \text{ub} \hat{\text{h}} \text{at}(k) + G1T * (\text{lb} \hat{\text{h}} \text{at}(k) - \text{lb}(k)) \\ \text{ea} \hat{\text{h}} \text{at}(k+1) &= \text{ea} \hat{\text{h}} \text{at}(k) - w_e * T * \text{eb} \hat{\text{h}} \text{at}(k) + G2T(\text{la} \hat{\text{h}} \text{at}(k) - \text{la}(k)) \\ \text{eb} \hat{\text{h}} \text{at}(k+1) &= \text{eb} \hat{\text{h}} \text{at}(k) + w_e * T * \text{ea} \hat{\text{h}} \text{at}(k) + G2T(\text{lb} \hat{\text{h}} \text{at}(k) - \text{lb}(k)) \end{aligned}$$







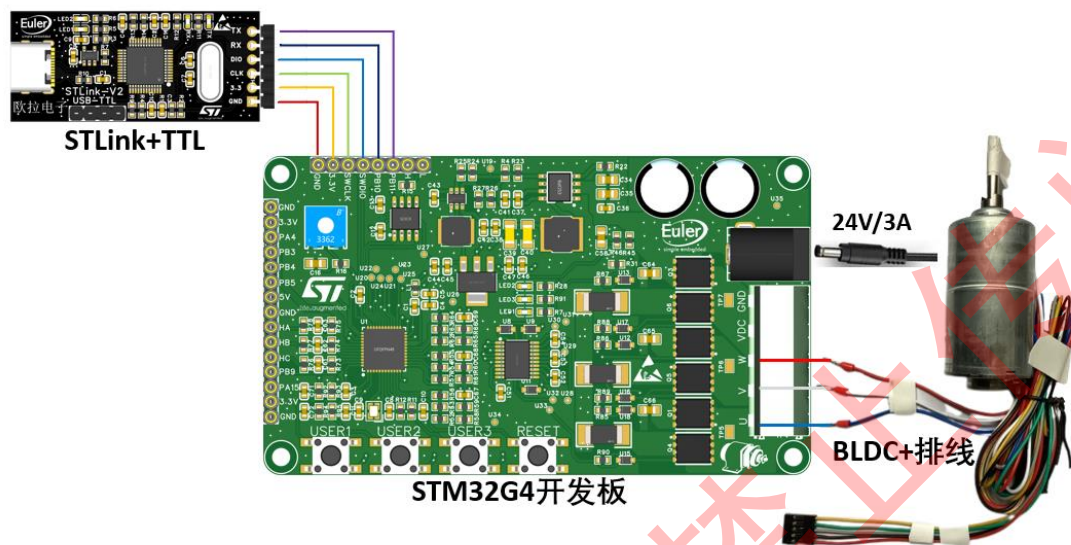
内容: ... C:\Users\Jay\Desktop\G4\Simulink\6.Lunberger\BLDC\BLDC\_Lun\_MODEL\BLDC\_Lun.sldd' (仅限此项) 筛选内容

列视图: Dictionary Objects 显示详细信息 23 个对象

|  | Name      | Status | Value    | DataType | Dimensions | Complexity | Min | Max | Unit | Argument | StorageClass   | DataSource    | L   |
|--|-----------|--------|----------|----------|------------|------------|-----|-----|------|----------|----------------|---------------|-----|
|  | Tpwm      |        | 16000    | uint16   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_Lun.sldd | 202 |
|  | ialpha    |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | ibeta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | ualpha    |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | ubeta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | Pn        |        | 4        | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | Ts        |        | 0.0001   | single   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_Lun.sldd | 202 |
|  | Udc       |        | 24       | single   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_Lun.sldd | 202 |
|  | state     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | theta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | G1        |        | -4000    | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | G2        |        | 14000    | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | LunTheta  |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | LunWm     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | L         |        | 0.00535  | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | Rs        |        | 6.97     | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | flux      |        | 0.016884 | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | curr_d_ki |        | 35       | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | curr_d_kp |        | 0.17     | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | spd_kp    |        | 0.3389   | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | spd_ki    |        | 0.0144   | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
|  | id        |        |          | auto     | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
|  | iq        |        |          | auto     | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |

### 3. 无感龙伯格观测器实验现象

① 按以下图片接线，并讲 USB 转 UART 模块连接至电脑：



② 下载代码，并打开 VOFA 上位机；

## 第十九章：滑膜观测器

### 1. 滑膜观测器及 PLL 建模

①  $\alpha$   $\beta$  轴电压方程：

$$\begin{bmatrix} \frac{di_{\alpha}}{dt} \\ \frac{di_{\beta}}{dt} \\ \frac{de_{\alpha}}{dt} \\ \frac{de_{\beta}}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} \\ 0 & 0 & 0 & -\omega_e \\ 0 & 0 & \omega_e & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \\ e_{\alpha} \\ e_{\beta} \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{\alpha} \\ u_{\beta} \end{bmatrix}$$

$$\begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \\ e_{\alpha} \\ e_{\beta} \end{bmatrix}$$

② A、B、C 分别为系统的状态矩阵、输入矩阵与输出矩阵；

$$A = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} \\ 0 & 0 & 0 & -\omega_e \\ 0 & 0 & \omega_e & 0 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

③ 系统的状态矢量  $x$ , 输入矢量  $u$  与输出矢量  $y$  为

$$x = \begin{bmatrix} i_\alpha \\ i_\beta \\ e_\alpha \\ e_\beta \end{bmatrix}, u = \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix}, y = \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}$$

④ 龙伯格观测器模型:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y}) \\ \hat{y} = C\hat{x} \end{cases}$$

$$\begin{bmatrix} \frac{di_\alpha}{dt} \\ \frac{di_\beta}{dt} \\ \frac{de_\alpha}{dt} \\ \frac{de_\beta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} & 0 \\ 0 & -\frac{R_s}{L_s} & 0 & -\frac{1}{L_s} \\ 0 & 0 & 0 & -\omega_e \\ 0 & 0 & \omega_e & 0 \end{bmatrix} \begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \\ \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} + \begin{bmatrix} K1 & 0 \\ 0 & K1 \\ K2 & 0 \\ 0 & K2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \\ \hat{e}_\alpha - e_\alpha \\ \hat{e}_\beta - e_\beta \end{bmatrix}$$

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} R_s + \frac{d}{dt}Ld & we(Ld - Lq) \\ -we(Ld - Lq) & R_s + \frac{d}{dt}Ld \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \left[ (Ld - Lq) \left( we * id - \frac{d}{dt}iq \right) + we\varphi_f \right] \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \varphi_f \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} = Ke * we \begin{bmatrix} -\sin(we * t) \\ \cos(we * t) \end{bmatrix}$$

$$\begin{bmatrix} \frac{d}{dt}e_\alpha \\ \frac{d}{dt}e_\beta \end{bmatrix} = Ke * we \begin{bmatrix} -we * \cos(we * t) \\ -we * \sin(we * t) \end{bmatrix} = we \begin{bmatrix} -e_\beta \\ e_\alpha \end{bmatrix}$$

⑤ 滑膜观测器误差补偿为:

$$\begin{bmatrix} K & 0 \\ 0 & K \\ M & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \text{sign}(\hat{i}_\alpha - i_\alpha) \\ \text{sign}(\hat{i}_\beta - i_\beta) \\ \text{sign}(\hat{e}_\alpha - e_\alpha) \\ \text{sign}(\hat{e}_\beta - e_\beta) \end{bmatrix}$$

其中 sign 是符号函数，正数取 1，负数取-1

⑥ 定义滑模面：

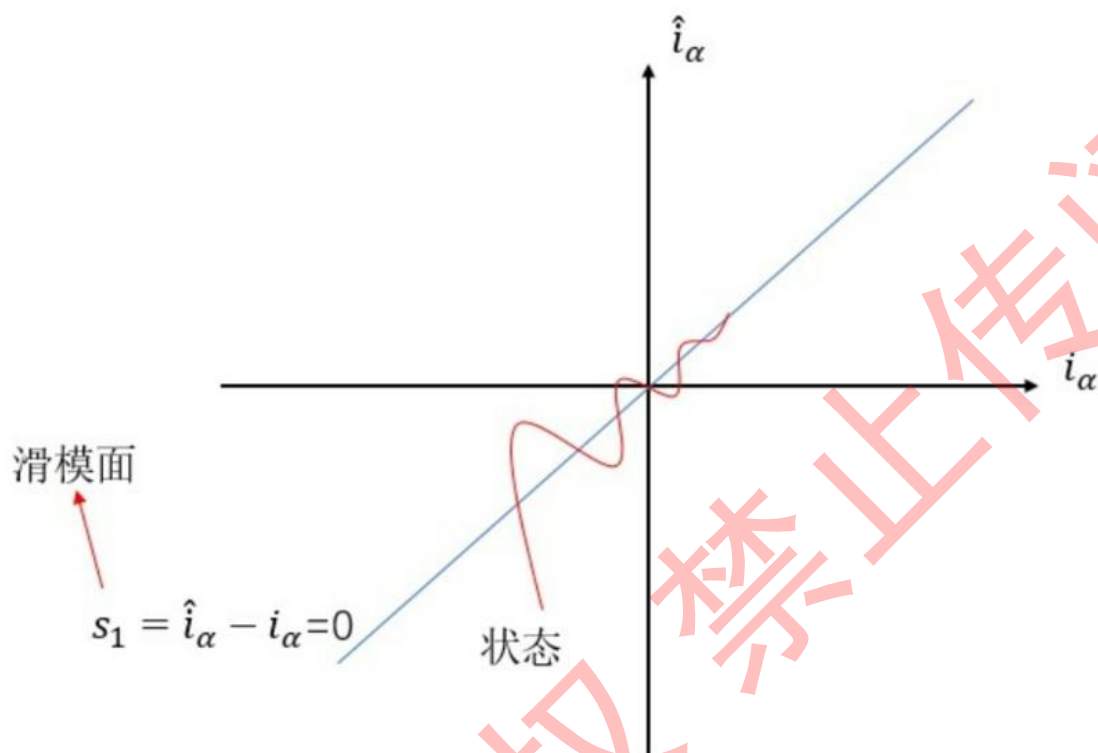
$$\begin{bmatrix} s1 \\ s2 \\ s3 \\ s4 \end{bmatrix} = \begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \\ \hat{e}_\alpha - e_\alpha \\ \hat{e}_\beta - e_\beta \end{bmatrix}$$

如果  $s * \dot{s} < 0$  满足（ $\dot{s}$  是  $s$  的时间导数），那么  $s$  将会收敛到 0。分析如下：如果  $s > 0$  且  $\dot{s} < 0$ ， $s$  将会从大于 0 的正值下降到 0，因为  $\dot{s} < 0$  表示时间导数为负，随着时间的推移的  $s$  值会减小。同理如果  $s < 0$  且  $\dot{s} > 0$ ，那么  $s$  将会从负值上升到 0。

⑦ 离散化后滑膜观测器模型，其中 K, M 为观测器增益：

$$\begin{aligned} \frac{d\hat{i}_\alpha}{dt} &\rightarrow \frac{\hat{i}_\alpha(k+1) - \hat{i}_\alpha(k)}{T} \\ \hat{i}_\alpha(k+1) &= \left(1 - \frac{Rs * T}{Ls}\right) * \hat{i}_\alpha(k) - \left(\frac{T}{Ls}\right) * \hat{e}_\alpha(k) + \left(\frac{T}{Ls}\right) * u_\alpha(k) - (K * T) * \left(\text{sign}(\hat{i}_\alpha(k) - i_\alpha(k))\right) \\ \hat{i}_\beta(k+1) &= \left(1 - \frac{Rs * T}{Ls}\right) * \hat{i}_\beta(k) - \left(\frac{T}{Ls}\right) * \hat{e}_\beta(k) + \left(\frac{T}{Ls}\right) * u_\beta(k) - (K * T) * \left(\text{sign}(\hat{i}_\beta(k) - i_\beta(k))\right) \\ \hat{e}_\alpha(k+1) &= \hat{e}_\alpha(k) - we * T * \hat{e}_\beta(k) + (M * T) * \left(\text{sign}(\hat{i}_\alpha(k) - i_\alpha(k))\right) \\ \hat{e}_\beta(k+1) &= \hat{e}_\beta(k) + we * T * \hat{e}_\alpha(k) + (M * T) * \left(\text{sign}(\hat{i}_\beta(k) - i_\beta(k))\right) \end{aligned}$$

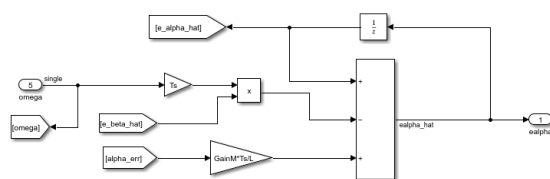
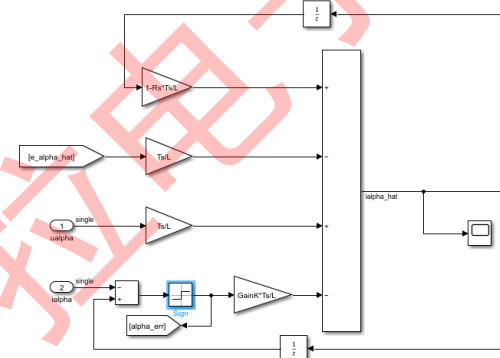
通过上式可以看出，即使  $\hat{i}_\alpha$  无限接近于  $i_\alpha$  或者  $\hat{i}_\alpha$  远大于及远小于  $i_\alpha$ ，误差补偿一直都是对应的增益值 K(M)，故误差补偿并不是一个连续性的补偿；所以滑膜观测器是一个非连续性的系统。同时无论增益值 K(M) 取值大小， $\hat{i}_\alpha$  都将在  $i_\alpha$  附近抖动；在系统稳定前抖动幅值较大，稳定后在该滑模面附近小幅波动（波动幅度很小，我们认为该状态到达并呆在滑模面上了）。



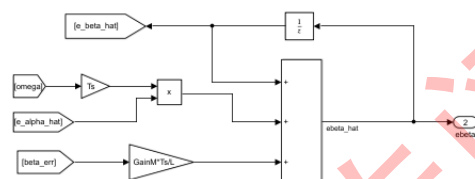
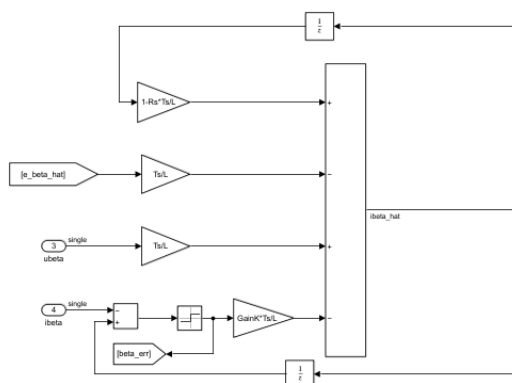
⑧ 经过不断的调试仿真，确定以下参数为适合参数：

|  |       |     |        |       |      |    |    |      |
|--|-------|-----|--------|-------|------|----|----|------|
|  | GainK | 2   | single | [1 1] | real | [] | [] | Auto |
|  | GainM | 100 | single | [1 1] | real | [] | [] | Auto |

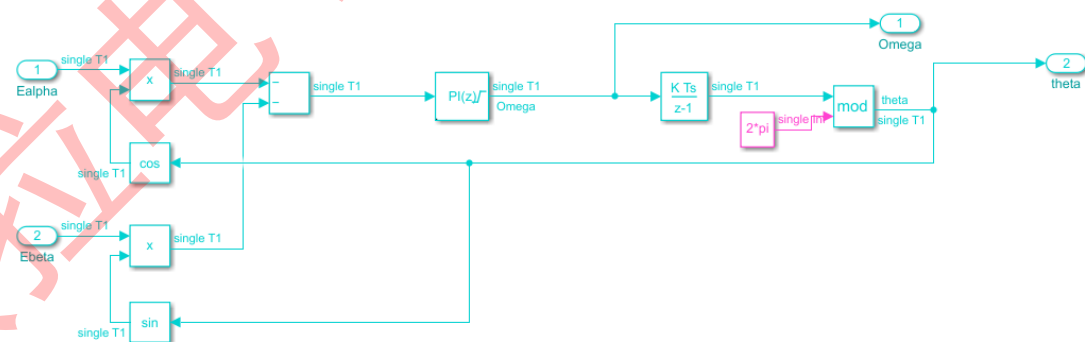
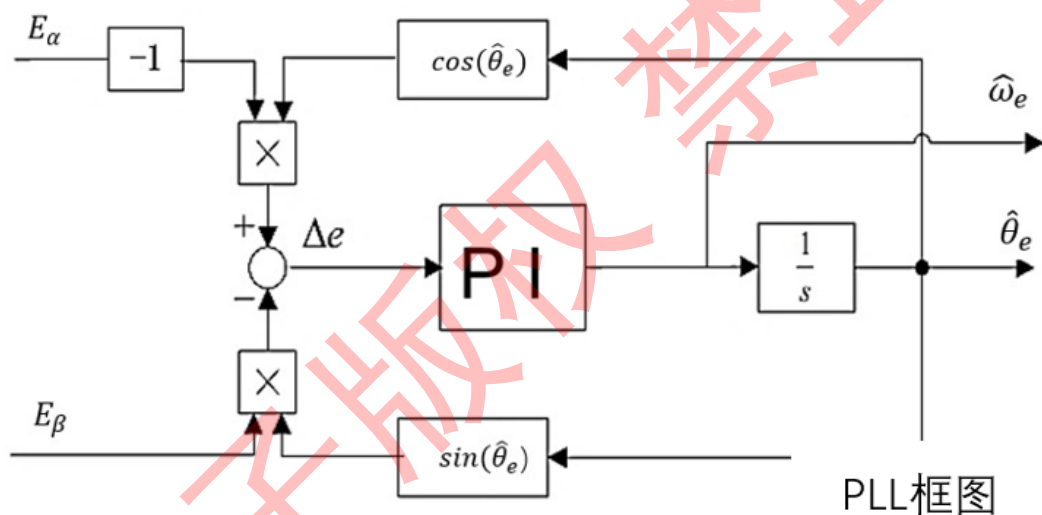
⑨ 基于离散化后滑膜观测器方程建模：



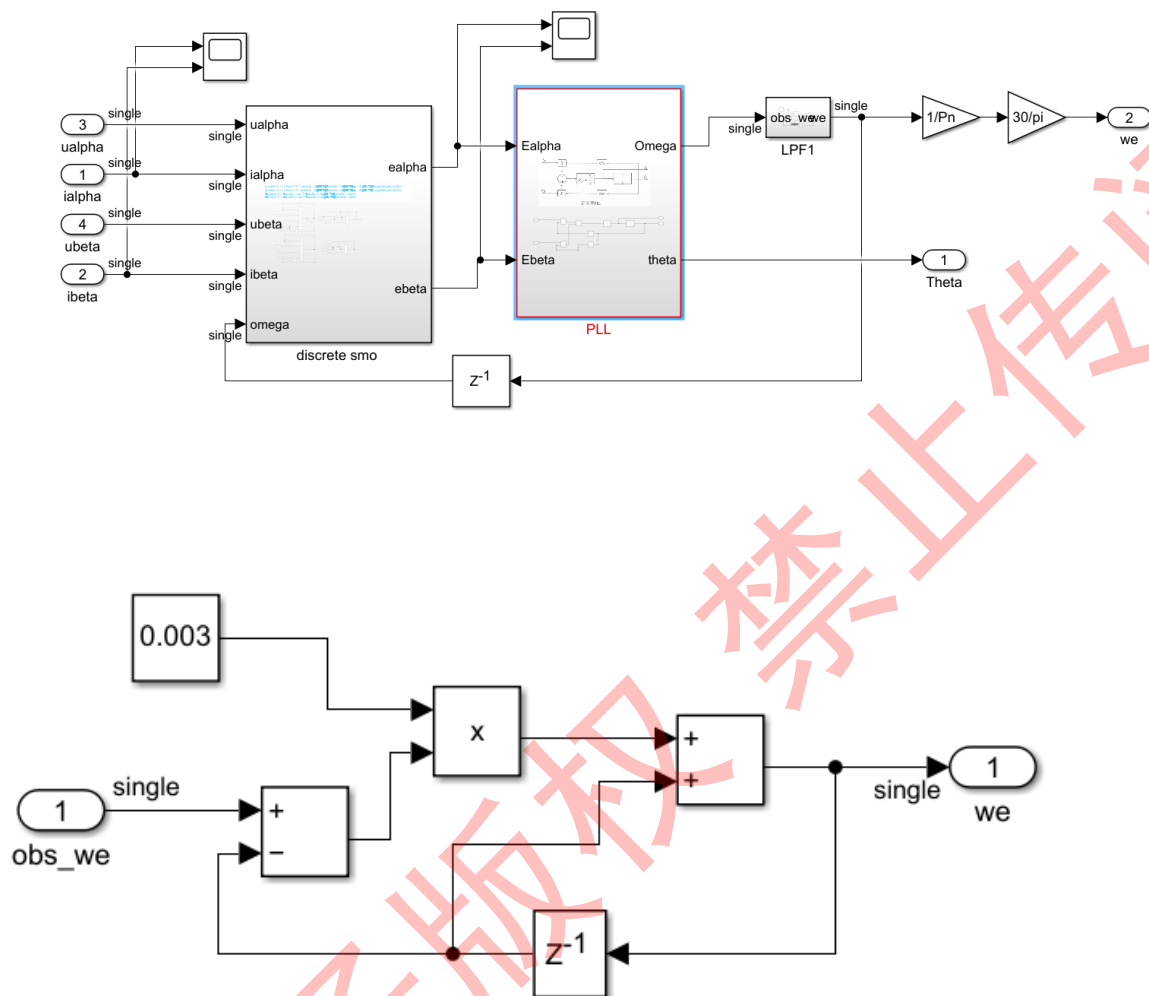




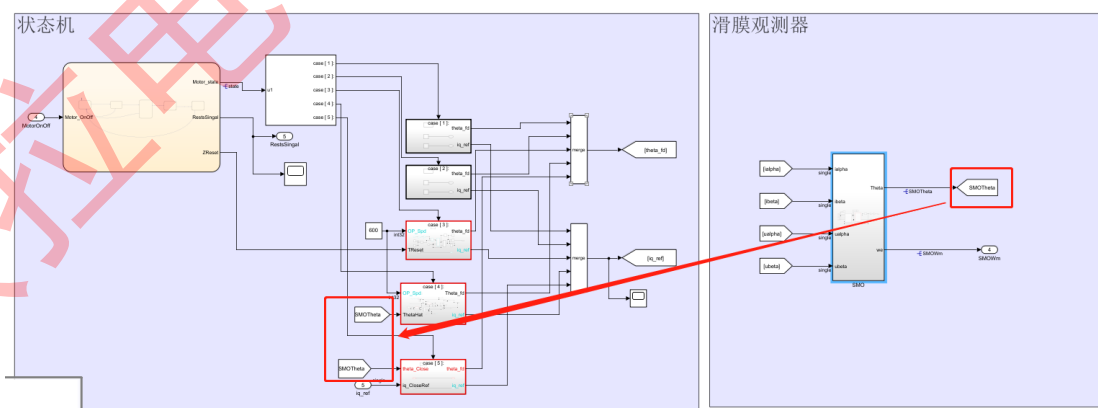
⑩ 建立锁相环模型：

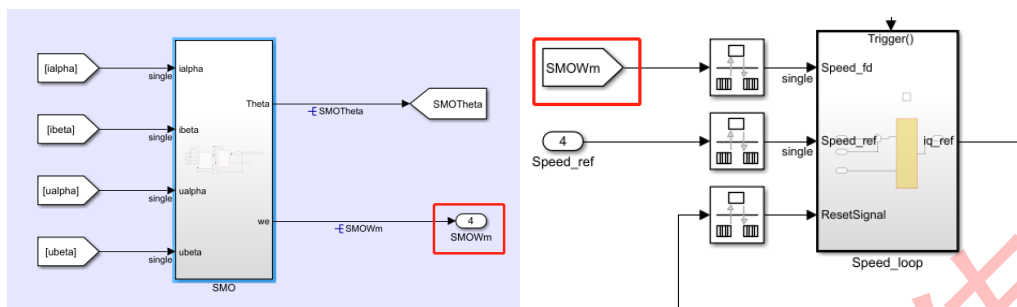


⑪ 由于滑膜观测器在滑模面存在高速抖动，故需要增加低通滤波器滤掉高频杂波；观测到的转速单位 rad/s，需转换为 rpm/min；

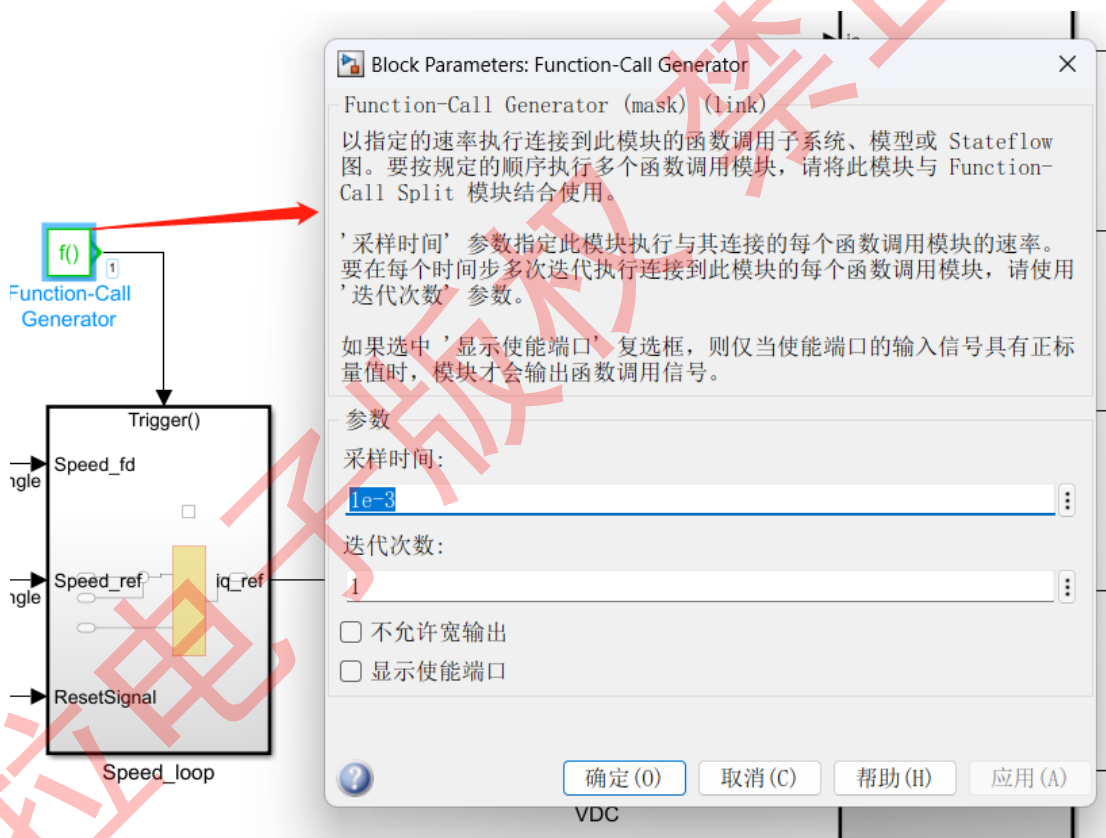


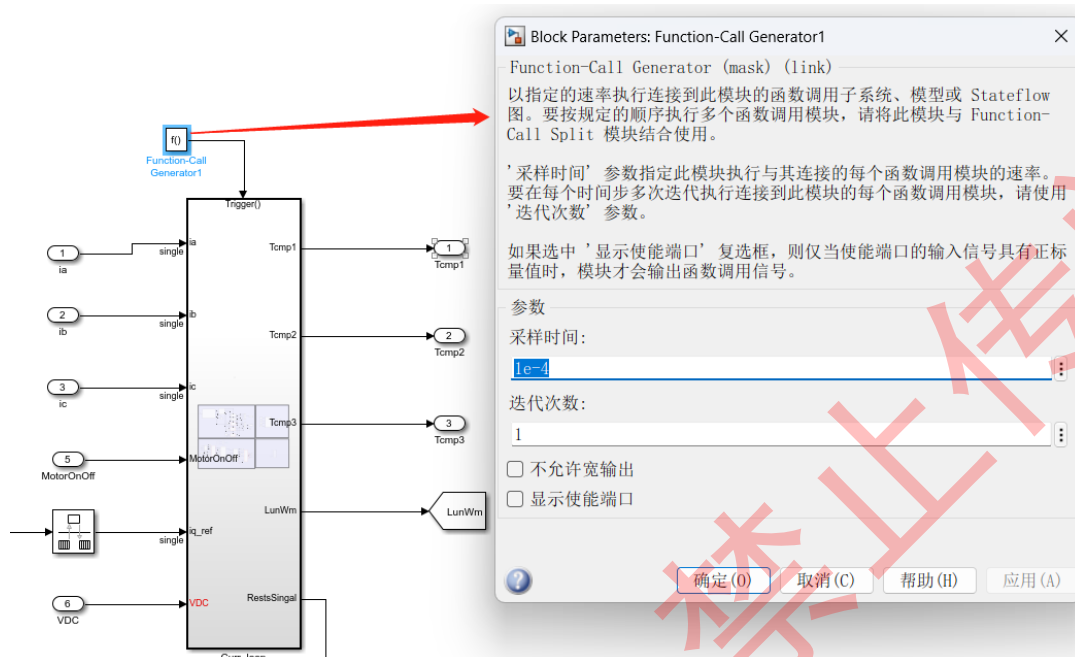
⑫ 将观测器输出的角度及转速作为闭环的角度和参考速度输入；





23 由于速度环不需要太高的执行频率，故将速度环的采样时间改为  $1e-3$ ；电流环的采样时间保持  $1e-4$ ；





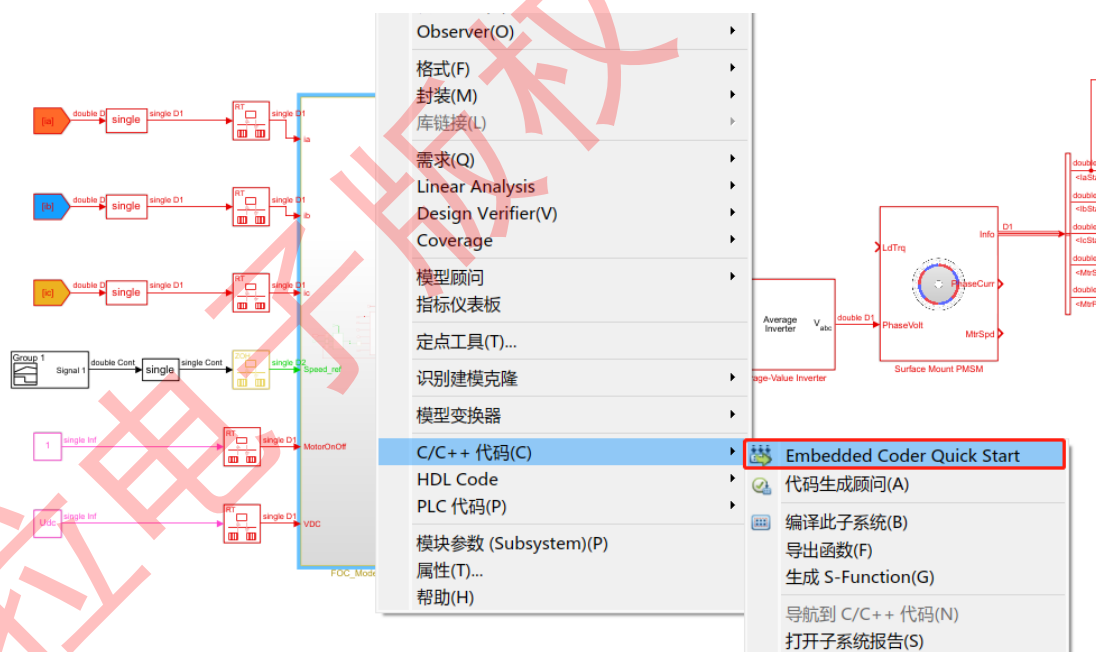
⑬ 在速度环及电流环的内部增加 trigger 模块;



⑭ 模型搭建完成后, 电机仿真按钮, 对比观测速度与电机实际转速;



- ⑮ 速度收敛较为完美，证明所搭建的观测器没有问题；但存在高频抖动，这与滑膜观测器本身的性质有关；右键点击 FOC\_Model 模块，点击生成代码；



- ⑯ 生成代码的流程如第十章流程；

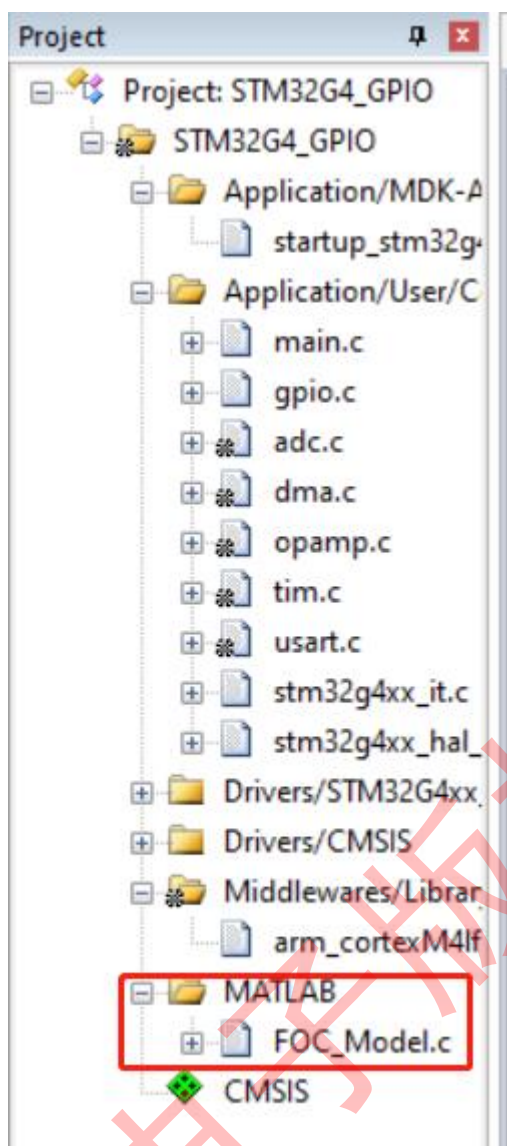
- ⑰ 将生成的 FOC 代码放进 Keil 工程中 MATLAB 文件夹；

| 名称                 | 修改日期            | 类型             | 大小    |
|--------------------|-----------------|----------------|-------|
| defines.txt        | 2023/8/22 21:11 | 文本文档           | 1 KB  |
| ert_main.c         | 2023/8/22 21:11 | C 源文件          | 4 KB  |
| FOC_Model.bat      | 2023/8/14 20:39 | Windows 批处理... | 1 KB  |
| FOC_Model.c        | 2023/8/22 21:11 | C 源文件          | 48 KB |
| FOC_Model.h        | 2023/8/22 21:11 | C Header 源文件   | 32 KB |
| FOC_Model.mk       | 2023/8/22 21:11 | Makefile 源文件   | 11 KB |
| FOC_Model.rsp      | 2023/8/20 12:23 | RSP 文件         | 1 KB  |
| FOC_Model_ref.rsp  | 2023/8/14 20:39 | RSP 文件         | 0 KB  |
| modelsources.txt   | 2023/8/22 21:11 | 文本文档           | 1 KB  |
| rtw_proj.tmw       | 2023/8/14 20:39 | TMW 文件         | 1 KB  |
| rtwtypes.h         | 2023/8/22 21:11 | C Header 源文件   | 4 KB  |
| rtwtypeschksum.mat | 2023/8/22 21:11 | MAT 文件         | 2 KB  |

| 名称          | 修改日期            | 类型           | 大小    |
|-------------|-----------------|--------------|-------|
| FOC_Model.c | 2023/8/22 21:11 | C 源文件        | 48 KB |
| FOC_Model.h | 2023/8/22 21:11 | C Header 源文件 | 32 KB |
| rtwtypes.h  | 2023/8/22 21:11 | C Header 源文件 | 4 KB  |

⑱ 打开 Keil 工程，在工程目录中 MATLAB 文件夹增加 FOC\_MODEL.C 文件；



⑲ Main.c 中包含 FOC\_Model.h:

```

/* Private includes -----
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
#include "FOC_Model.h"
/* USER CODE END Includes */

```

⑳ 通过电位器旋钮调节算法模型输入的参考转速，转速在 300-1200 之间；

ADC 采样的母线电压作为模型 Vbus 输入；

```

7 | {
8 |     HAL_ADC_Start(&hadc1);
9 |     HAL_ADC_Start(&hadc2);
0 |     adc1_in4 = HAL_ADC_GetValue(&hadc1);
1 |     adc_vbus = HAL_ADC_GetValue(&hadc2);
2 |     rtU.VDC = adc_vbus*3.3f/4096*26;
3 |     finalspeed = (adc1_in4>>4) * 15;
4 |     if(finalspeed >=0 && finalspeed <300)
5 |     {
6 |         finalspeed = 300;
7 |     }
8 |     if(finalspeed > 1200)
9 |     {
0 |         finalspeed = 1200;
1 |     }
2 |     rtU.Speed_ref = finalspeed;
3 |     HAL_Delay(10);

```

- 21 在 ADC 中断中读取三相电流值输入算法模型，执行算法代码；将算法模型输出作为 TIM1 的比较值写入寄存器；同时通过 UART 上传数据至 VOFA 上位机显示波形；

```

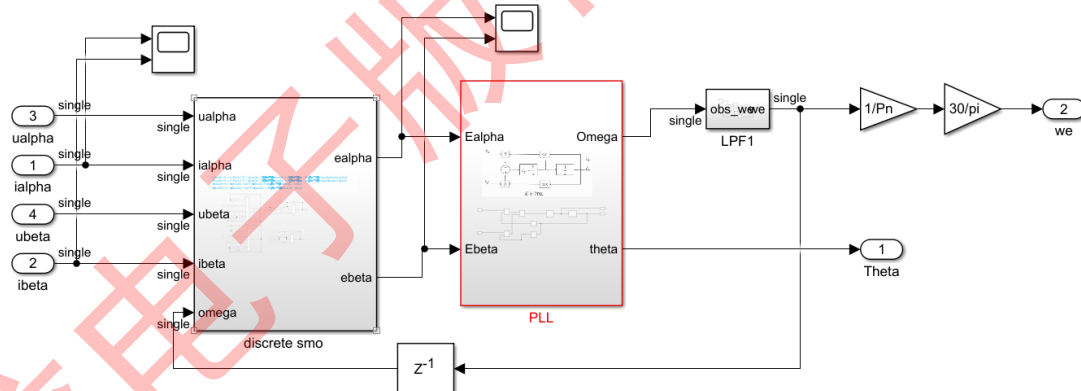
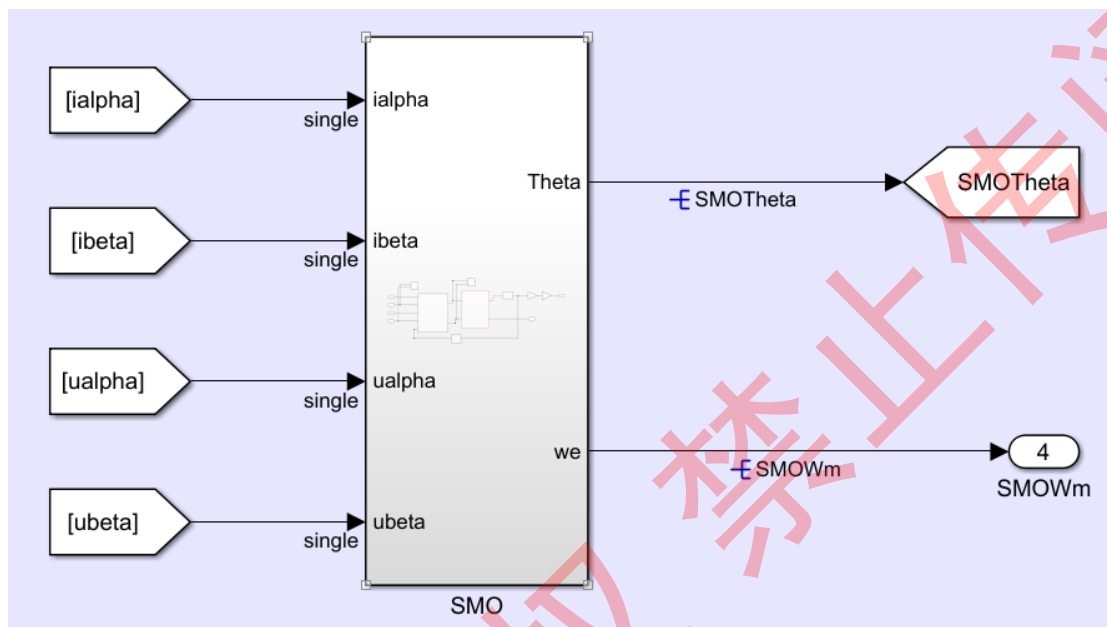
337 | else
338 | {
339 |     adc1_in1 = hadc1.Instance->JDR1;
340 |     adc1_in3 = hadc1.Instance->JDR2;
341 |     adc1_in2 = hadc2.Instance->JDR1;
342 |     rtU.ia = (adc1_in1 - IA_Offset)*0.02197f;
343 |     rtU.ib = (adc1_in2 - IB_Offset)*0.02197f;
344 |     rtU.ic = (adc1_in3 - IC_Offset)*0.02197f;
345 |     FOC_Model_step();
346 |     TIM1->CCR1 = rtY.Tcmp1;
347 |     TIM1->CCR2 = rtY.Tcmp2;
348 |     TIM1->CCR3 = rtY.Tcmp3;
349 |     load_data[0] = rtU.ia;
350 |     load_data[1] = SMTheta;
351 |     load_data[2] = SMOWm;
352 |     load_data[3] = rtU.Speed_ref;
353 |     load_data[4] = 0;
354 |     memcpy(tempData, (uint8_t *)&load_data, sizeof(load_data));
355 |     HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 6*4);
356 | }
357 |
358 |

```

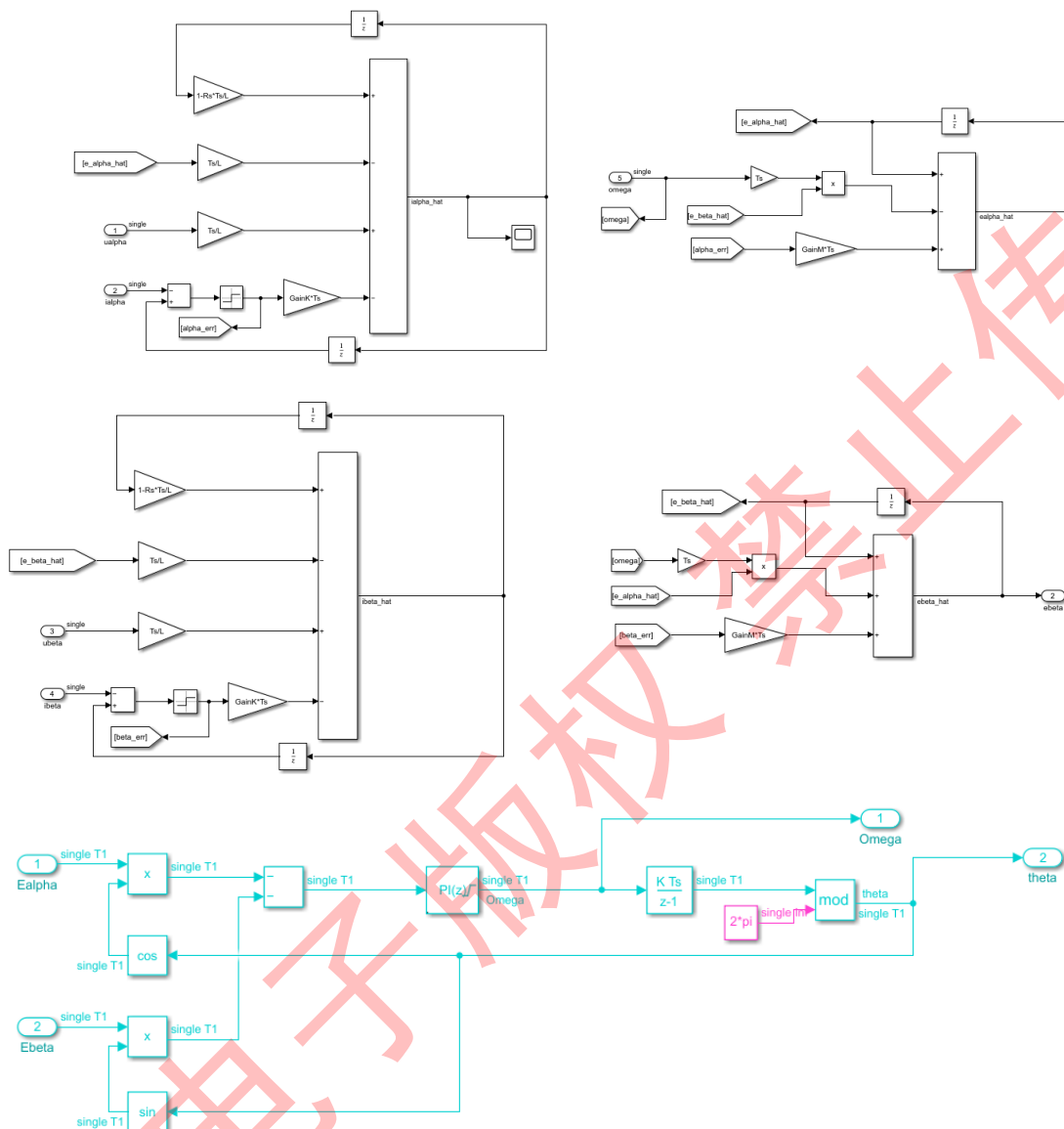
- 22 编译代码；



## 2. 滑膜观测器模块内部及数据字典截图



$$\begin{aligned} \hat{\alpha}(k+1) &= (1-R_s/L_s \cdot T) \cdot \hat{\alpha}(k) - (1/L_s \cdot T) e_{\alpha}(k) + (1/L_s \cdot T) u_{\alpha} - (K/L_s \cdot T) \cdot (\text{sign}(\hat{\alpha}(k)) - \alpha(k)) \\ \hat{\beta}(k+1) &= (1-R_s/L_s \cdot T) \cdot \hat{\beta}(k) - (1/L_s \cdot T) e_{\beta}(k) + (1/L_s \cdot T) u_{\beta} - (K/L_s \cdot T) \cdot (\text{sign}(\hat{\beta}(k)) - \beta(k)) \\ \hat{e}_{\alpha}(k+1) &= \hat{e}_{\alpha}(k) - \omega \cdot T \cdot \hat{e}_{\beta}(k) + (M/L_s \cdot T) (\hat{\alpha}(k) - \alpha(k)) \\ \hat{e}_{\beta}(k+1) &= \hat{e}_{\beta}(k) + \omega \cdot T \cdot \hat{e}_{\alpha}(k) + (M/L_s \cdot T) (\hat{\beta}(k) - \beta(k)) \end{aligned}$$



Block Parameters: PID Controller

PID 1dof (mask) (link)

此模块实现连续和离散时间 PID 控制算法，并包括高级功能，如抗饱和、外部重置和信号跟踪。您可以使用 '调节...' 按钮自动调节 PID 增益(需要 Simulink Control Design)。

控制器: PI 形式: 并行

时域:

☐ 连续时间

☒ 离散时间

离散时间设置

☐ PID 控制器位于条件执行子系统内

采样时间(-1 表示继承): -1

积分器和滤波器方法:

补偿器公式

$$P + I \cdot T_s \frac{1}{z-1}$$

主要 初始化 输出饱和 数据类型 状态属性

控制器参数

源: 内部

比例(P): 2\*0.707\*150

积分(I): 50\*50

自动调节

选择调节方法: 基于传递函数(PID 调节器) 调节...

☒ 启用过零检测

确定(O) 取消(C) 帮助(H) 应用(A)

Block Parameters: PID Controller

PID 1dof (mask) (link)

此模块实现连续和离散时间 PID 控制算法，并包括高级功能，如抗饱和、外部重置和信号跟踪。您可以使用 '调节...' 按钮自动调节 PID 增益(需要 Simulink Control Design)。

控制器: PI 形式: 并行

时域:

☐ 连续时间

☒ 离散时间

离散时间设置

☐ PID 控制器位于条件执行子系统内

采样时间(-1 表示继承): -1

积分器和滤波器方法:

补偿器公式

$$P + I \cdot T_s \frac{1}{z-1}$$

主要 初始化 输出饱和 数据类型 状态属性

输出饱和

☒ 限制输出

源: 内部

上限: 2\*pi\*1000

下限: -2\*pi\*1000

☒ 线性化时忽略饱和和限值

抗饱和

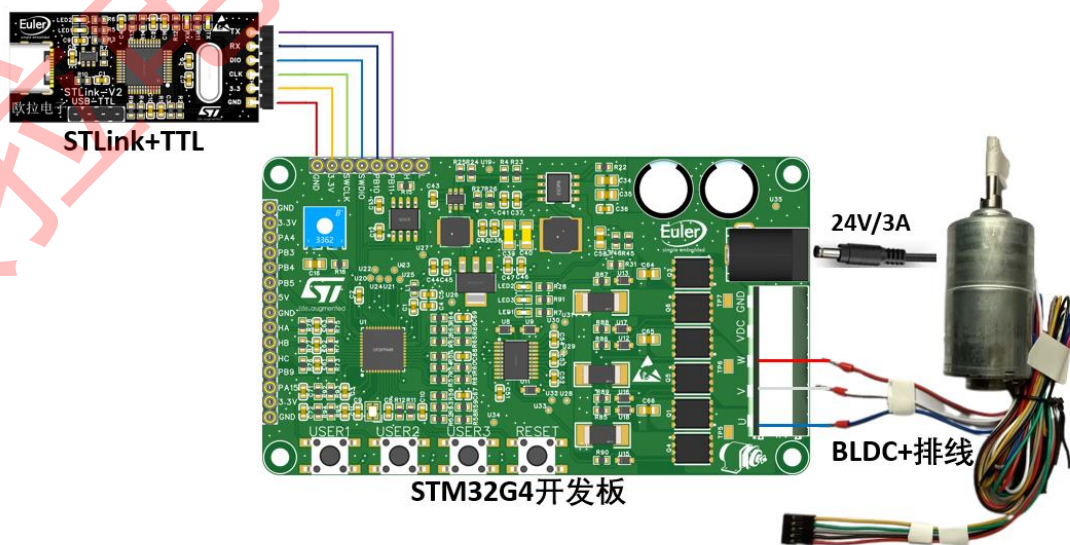
抗饱和方法: 钳位

确定(O) 取消(C) 帮助(H) 应用(A)

|           | Name      | Status | Value    | DataType | Dimensions | Complexity | Min | Max | Unit | Argument | StorageClass   | DataSource    | L   |
|-----------|-----------|--------|----------|----------|------------|------------|-----|-----|------|----------|----------------|---------------|-----|
| [101 010] | Tpwm      |        | 16000    | uint16   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_Lun.sldd | 202 |
| [101 010] | ialpha    |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | ibeta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | ualpha    |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | ubeta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | Pn        |        | 4        | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | Ts        |        | 0.0001   | single   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_Lun.sldd | 202 |
| [101 010] | Udc       |        | 24       | single   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_Lun.sldd | 202 |
| [101 010] | state     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | theta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | G1        |        | -4000    | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | G2        |        | 14000    | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | LunTheta  |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | LunWm     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | L         |        | 0.00535  | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | Rs        |        | 6.97     | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | flux      |        | 0.016884 | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | curr_d_ki |        | 35       | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | curr_d_kp |        | 0.17     | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | spd_kp    |        | 0.3389   | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | spd_ki    |        | 0.0144   | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_Lun.sldd | 202 |
| [101 010] | id        |        |          | auto     | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |
| [101 010] | iq        |        |          | auto     | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_Lun.sldd | 202 |

### 3. 无感滑膜观测器实验现象

① 按以下图片接线，并讲 USB 转 UART 模块连接至电脑：



② 下载代码，并打开 VOFA 上位机；

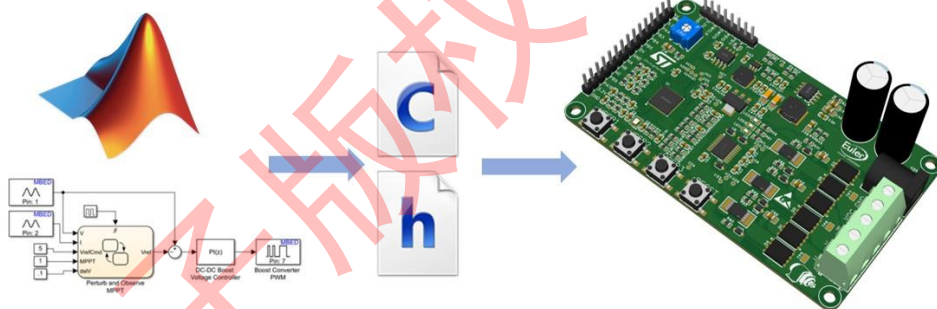
## 第二十章：拓展卡尔曼观测器

### 1. 卡尔曼观测器理论引入

# STM32G4 FOC

1

拓展卡尔曼观测器EKF理论推导  
Q、R矩阵参数



主题概要

- Part 1 方差、标准差、协方差、六西格玛
- Part 2 卡尔曼核心公式推导
- Part 3 卡尔曼观测器递推流程
- Part 4 拓展卡尔曼观测器模型推导

## 拓展卡尔曼观测器建模

离散化:

$$\begin{bmatrix} \widehat{i_{\alpha k}} \\ \widehat{i_{\beta k}} \\ \widehat{we_k} \\ \widehat{\theta_k} \end{bmatrix} = \begin{bmatrix} \widehat{i_{\alpha k-1}} \\ \widehat{i_{\beta k-1}} \\ \widehat{we_{k-1}} \\ \widehat{\theta_{k-1}} \end{bmatrix} + Ts \left\{ \begin{bmatrix} -\frac{Rs}{Ls} & 0 & \frac{\varphi_f}{Ls} \sin \theta & \frac{we\varphi_f}{Ls} \cos \theta \\ 0 & -\frac{Rs}{Ls} & -\frac{\varphi_f}{Ls} \cos \theta & \frac{we\varphi_f}{Ls} \sin \theta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \widehat{i_{\alpha k-1}} \\ \widehat{i_{\beta k-1}} \\ \widehat{we_{k-1}} \\ \widehat{\theta_{k-1}} \end{bmatrix} + \begin{bmatrix} \frac{1}{Ls} & 0 \\ 0 & \frac{1}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{\alpha-1} \\ u_{\beta-1} \end{bmatrix} \right\}$$

$$\begin{bmatrix} i_{\alpha k} \\ i_{\beta k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha k} \\ i_{\beta k} \\ we_k \\ \theta_k \end{bmatrix}$$

$$\begin{bmatrix} \widehat{i_{\alpha k}} \\ \widehat{i_{\beta k}} \\ \widehat{we_k} \\ \widehat{\theta_k} \end{bmatrix} = \left\{ I + Ts \begin{bmatrix} -\frac{Rs}{Ls} & 0 & \frac{\varphi_f}{Ls} \sin \theta & \frac{we\varphi_f}{Ls} \cos \theta \\ 0 & -\frac{Rs}{Ls} & -\frac{\varphi_f}{Ls} \cos \theta & \frac{we\varphi_f}{Ls} \sin \theta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \right\} \begin{bmatrix} \widehat{i_{\alpha k-1}} \\ \widehat{i_{\beta k-1}} \\ \widehat{we_{k-1}} \\ \widehat{\theta_{k-1}} \end{bmatrix} + Ts \begin{bmatrix} \frac{1}{Ls} & 0 \\ 0 & \frac{1}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{\alpha-1} \\ u_{\beta-1} \end{bmatrix}$$

## 拓展卡尔曼观测器建模

$$\begin{bmatrix} \frac{di_{\alpha}}{dt} \\ \frac{di_{\beta}}{dt} \\ \frac{dwe}{dt} \\ \frac{d\theta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{Rs}{Ls} & 0 & \frac{we\varphi_f}{Ls} \sin \theta & 0 \\ 0 & -\frac{Rs}{Ls} & -\frac{we\varphi_f}{Ls} \cos \theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \\ we \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{Ls} & 0 \\ 0 & \frac{1}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{\alpha} \\ u_{\beta} \end{bmatrix}$$

$$\begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \\ we_k \\ \theta_k \end{bmatrix}$$

线性化:

$$\begin{bmatrix} \frac{di_{\alpha}}{dt} \\ \frac{di_{\beta}}{dt} \\ \frac{dwe}{dt} \\ \frac{d\theta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{Rs}{Ls} & 0 & \frac{\varphi_f}{Ls} \sin \theta & \frac{we\varphi_f}{Ls} \cos \theta \\ 0 & -\frac{Rs}{Ls} & -\frac{\varphi_f}{Ls} \cos \theta & \frac{we\varphi_f}{Ls} \sin \theta \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \\ we \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{Ls} & 0 \\ 0 & \frac{1}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{\alpha} \\ u_{\beta} \end{bmatrix}$$

$$\begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \\ we_k \\ \theta_k \end{bmatrix}$$

## 拓展卡尔曼观测器建模

$$\begin{bmatrix} \frac{di_\alpha}{dt} \\ \frac{di_\beta}{dt} \\ \frac{dwe}{dt} \\ \frac{d\theta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{Rs}{Ls} & 0 & \frac{we\varphi_f}{Ls} \sin \theta & 0 \\ 0 & -\frac{Rs}{Ls} & -\frac{we\varphi_f}{Ls} \cos \theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ we \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{Ls} & 0 \\ 0 & \frac{1}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} \quad \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ we_k \\ \theta_k \end{bmatrix}$$

$$\begin{bmatrix} \frac{di_\alpha}{dt} \\ \frac{di_\beta}{dt} \\ \frac{dwe}{dt} \\ \frac{d\theta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{Rs}{Ls} & \frac{we\varphi_f}{Ls} \sin \theta - \frac{u_\alpha}{Ls} \\ -\frac{Rs}{Ls} & -\frac{we\varphi_f}{Ls} \cos \theta - \frac{u_\beta}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ we \\ \theta \end{bmatrix}$$

离散化:

$$\begin{bmatrix} \hat{i}_{\alpha k} \\ \hat{i}_{\beta k} \\ \hat{we}_k \\ \hat{\theta}_k \end{bmatrix} = \begin{bmatrix} \hat{i}_{\alpha k-1} \\ \hat{i}_{\beta k-1} \\ \hat{we}_{k-1} \\ \hat{\theta}_{k-1} \end{bmatrix} + Ts \left\{ \begin{bmatrix} -\frac{Rs}{Ls} & 0 & \frac{we\varphi_f}{Ls} \sin \theta & 0 \\ 0 & -\frac{Rs}{Ls} & -\frac{we\varphi_f}{Ls} \cos \theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{i}_{\alpha k-1} \\ \hat{i}_{\beta k-1} \\ \hat{we}_{k-1} \\ \hat{\theta}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{Ls} & 0 \\ 0 & \frac{1}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{\alpha k-1} \\ u_{\beta k-1} \end{bmatrix} \right\}$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ we_k \\ \theta_k \end{bmatrix}$$

## 拓展卡尔曼观测器

由于卡尔曼滤波（观测器）算法都是需要线性、离散的系统模型，但是在实际应用中常见的系统大多都是非线性的，包括无刷和永磁同步电机系统。此时就需要扩展卡尔曼滤波器（Extended Kalman Filter，简称EKF）的理论来把其推广到更大的应用范围。在利用扩展卡尔曼滤波器时，需要首先对线性系统的模型方程进行线性化和离散化的处理，再对应实际情况去应用卡尔曼滤波器。

一元系统线性化:

泰勒公式:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \dots + \frac{f^n(x_0)}{n!}(x-x_0)^n$$

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0), \quad x-x_0 \rightarrow 0$$

多元系统线性化:

$$\begin{aligned} \dot{X}_1 &= f_1(X_1, X_2) \\ \dot{X}_2 &= f_2(X_1, X_2) \end{aligned}$$

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} \\ \frac{\partial f_2}{\partial X_1} & \frac{\partial f_2}{\partial X_2} \end{bmatrix}_{X=X_0} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$



## 卡尔曼五大核心公式

### 预估

先验状态值

$$\widehat{X}_k^- = A\widehat{X}_{k-1} + Bu_{k-1}$$

先验误差协方差

$$P_k^- = AP_{k-1}A^T + Q$$

### 校正

卡尔曼增益

$$K_k = \frac{P_k^- H^T}{(HP_k^- H^T + R)}$$

做优估计值

$$\widehat{X}_k = \widehat{X}_k^- + K_k(Z_k - H\widehat{X}_k^-)$$

更新做优协方差

$$P_k = (I - K_k H)P_k^-$$

## 卡尔曼核心公式-误差协方差 $P_k$

由于 $P_k^-$ 中存在上次一次的误差协方差 $P_{k-1}$ ，故我们需要计算出 $P_{k-1}$

$$P_k = P_k^- - K_k H P_k^- - P_k^- H^T K_k^T + K_k H P_k^- H^T K_k^T + K_k R K_k^T$$

将 $K_k$ 带入上式中：

$$P_k = P_k^- - K_k H P_k^- - P_k^- H^T K_k^T + K_k (H P_k^- H^T + R) K_k^T$$

$$P_k = P_k^- - K_k H P_k^- - P_k^- H^T K_k^T + K_k H P_k^- + \frac{P_k^- H^T}{(H P_k^- H^T + R)} (H P_k^- H^T + R) K_k^T$$

$$P_k = P_k^- - K_k H P_k^-$$

$$P_k = (I - K_k H)P_k^-$$



## 卡尔曼核心公式-误差协方差先验 $P_k^-$

$$P^- = E[e_k^- e_k^{-T}]$$

$$e_k^- = X_k - \widehat{X}_k$$

基于状态空间方程及先验估计公式：

$$\begin{aligned} X_k &= AX_{k-1} + Bu_{k-1} + W_{k-1} \\ \widehat{X}_k &= A\widehat{X}_{k-1} + Bu_{k-1} \\ e_k^- &= X_k - \widehat{X}_k = AX_{k-1} + Bu_{k-1} + W_{k-1} - A\widehat{X}_{k-1} - Bu_{k-1} \\ &= A(X_{k-1} - \widehat{X}_{k-1}) + W_{k-1} \\ &= Ae_{k-1} + W_{k-1} \end{aligned}$$

将 $e_k^-$ 带入先验误差协方差计算公式：

$$\begin{aligned} P_k^- &= E[e_k^- e_k^{-T}] = E[(Ae_{k-1} + W_{k-1})(Ae_{k-1} + W_{k-1})^T] \\ &= E[(Ae_{k-1} + W_{k-1})(e_{k-1}^T A^T + W_{k-1}^T)] \\ &= E[Ae_{k-1}e_{k-1}^T A^T + Ae_{k-1}W_{k-1}^T + W_{k-1}e_{k-1}^T A^T + W_{k-1}W_{k-1}^T] \\ &= E(Ae_{k-1}e_{k-1}^T A^T) + E(Ae_{k-1}W_{k-1}^T) + E(W_{k-1}e_{k-1}^T A^T) + E(W_{k-1}W_{k-1}^T) \end{aligned}$$

由于 $e_{k-1}$ 与 $W_{k-1}$ 相互独立，并且 $e_{k-1}$ 的期望值是0，故

$$\begin{aligned} P_k^- &= AE(e_{k-1}e_{k-1}^T)A^T + E(W_{k-1}W_{k-1}^T) \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned}$$

## 卡尔曼核心公式-增益 $K_k$

由于 $V_k$ 与 $e_k^-$ 相互独立，且 $e_k^-$ 的期望为0，故：

$$\begin{aligned} (I - K_k H)E(e_k^- V_k^T)K_k^T &= 0, \quad K_k E(V_k e_k^{-T})(I - K_k H)^T = 0 \\ P_k &= (I - K_k H)E(e_k^- e_k^{-T})(I - K_k H)^T + K_k E(V_k V_k^T)K_k^T \\ P_k &= (I - K_k H)P_k^- (I - K_k H)^T + K_k R K_k^T \\ &= P_k^- - K_k H P_k^- - P_k^- H^T K_k^T + K_k H P_k^- H^T K_k^T + K_k R K_k^T \\ \text{tr}(P_k) &= \text{tr}(P_k^-) - 2\text{tr}(K_k H P_k^-) + \text{tr}(K_k H P_k^- H^T K_k^T) + \text{tr}(K_k R K_k^T) \\ \frac{d\text{tr}(P_k)}{dK_k} &= 0 \\ -P_k^- H^T + K_k (H P_k^- H^T + R) &= 0 \end{aligned}$$

卡尔曼增益计算公式：

$$K_k = \frac{P_k^- H^T}{(H P_k^- H^T + R)}$$

## 卡尔曼核心公式-增益 $K_k$

我们的目标（期望）是 $e_{k1}, e_{k2}, e_{k3}, e_{k4}$ 尽可能的小，也就是对应的方差最小，即协方差矩阵的迹最小：

$$\begin{aligned} \text{tr}(P) &= \sigma_{e1}^2 + \sigma_{e2}^2 + \sigma_{e3}^2 + \sigma_{e4}^2 \\ e_k &= X_k - \hat{X}_k = X_k - [\hat{X}_k + K_k H (H^- Z_k - \hat{X}_k)] \\ &= X_k - \hat{X}_k - K_k Z_k + K_k H \hat{X}_k \\ &= X_k - \hat{X}_k - K_k H X_k - K_k V_k + K_k H \hat{X}_k \\ &= (X_k - \hat{X}_k) - K_k H (X_k - \hat{X}_k) - K_k V_k \\ &= (I - K_k H) (X_k - \hat{X}_k) - K_k V_k \\ &= (I - K_k H) (X_k - \hat{X}_k) - K_k V_k \\ &= (I - K_k H) e_k^- - K_k V_k \\ P_k &= \text{cov}(X, Y) = E[(X - E(X))(Y - E(Y))] \\ &= E[(X_k - \hat{X}_k)(X_k - \hat{X}_k)] = E[e_k e_k^T] \\ &= E\left[\left((I - K_k H) e_k^- - K_k V_k\right)\left((I - K_k H) e_k^- - K_k V_k\right)^T\right] \\ &= E\left[\left((I - K_k H) e_k^- - K_k V_k\right)(e_k^{-T} (I - K_k H)^T - V_k^T K_k^T)\right] \\ &= E\left[(I - K_k H) e_k^- e_k^{-T} (I - K_k H)^T - K_k V_k e_k^{-T} (I - K_k H)^T - (I - K_k H) e_k^- V_k^T K_k^T + K_k V_k V_k^T K_k^T\right] \\ &= (I - K_k H) E(e_k^- e_k^{-T}) (I - K_k H)^T - K_k E(V_k e_k^{-T}) (I - K_k H)^T - (I - K_k H) E(e_k^- V_k^T) K_k^T + K_k E(V_k V_k^T) K_k^T \end{aligned}$$

## 卡尔曼核心公式-最优估计值 $\hat{X}_k$

先验估计值  $\hat{X}_k^-$ ：

$$\begin{aligned} \hat{X}_k^- &= A \hat{X}_{k-1} + B u_{k-1} \\ Z_k &= H X_k \\ X_k &= H^- Z_k \end{aligned}$$

后验估计值  $\hat{X}_k$ ：

$$\hat{X}_k = \hat{X}_k^- + G (H^- Z_k - \hat{X}_k^-)$$

其中  $G = K_k H$

$$\hat{X}_k = \hat{X}_k^- + K_k H (H^- Z_k - \hat{X}_k^-)$$

$\hat{X}_k = \hat{X}_k^- + K_k (Z_k - H \hat{X}_k^-)$ ,  $K_k \in [0, H^-]$ , 其中  $K_k$  为卡尔曼增益

当  $K_k = 0$  时：

$$\hat{X}_k = \hat{X}_k^-$$

当  $K_k = H^-$  时：

$$\hat{X}_k = H^- Z_k = X_k$$

我们的目标是  $\hat{X}_k = X_k$ ，即：

$$e_k = X_k - \hat{X}_k = 0$$

观测误差的协方差矩阵：

$$P = \begin{bmatrix} \sigma_{e1}^2 & 0 & 0 & 0 \\ 0 & \sigma_{e2}^2 & 0 & 0 \\ 0 & 0 & \sigma_{e3}^2 & 0 \\ 0 & 0 & 0 & \sigma_{e4}^2 \end{bmatrix}$$

## 卡尔曼观测器模型引入

$$\begin{aligned}\widehat{X}_k &= A\widehat{X}_{k-1} + Bu_k + W_k \\ Z_k &= HX_k + V_k\end{aligned}$$

$$\begin{bmatrix} \frac{di_\alpha}{dt} \\ \frac{di_\beta}{dt} \\ \frac{d\omega}{dt} \\ \frac{d\theta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{Rs}{Ls} & 0 & \frac{\varphi_f}{Ls} \sin \theta & 0 \\ 0 & -\frac{Rs}{Ls} & -\frac{we\varphi_f}{Ls} \cos \theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ we \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{Ls} & 0 \\ 0 & \frac{1}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} + \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix}$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ e_\alpha \\ e_\beta \end{bmatrix} + \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

$W_k, V_k$  满足正态分布:

$$\begin{aligned}P(W_k) &\sim N(0, Q) \\ P(V_k) &\sim N(0, R)\end{aligned}$$

$$Q = \begin{bmatrix} \sigma_{w1}^2 & \sigma_{w1w2} & \sigma_{w1w3} & \sigma_{w1w4} \\ \sigma_{w1w2} & \sigma_{w2}^2 & \sigma_{w2w3} & \sigma_{w2w4} \\ \sigma_{w1w3} & \sigma_{w2w3} & \sigma_{w3}^2 & \sigma_{w3w4} \\ \sigma_{w1w4} & \sigma_{w2w4} & \sigma_{w3w4} & \sigma_{w4}^2 \end{bmatrix}$$

$$R = \begin{bmatrix} \sigma_{v1}^2 & \sigma_{v1v2} \\ \sigma_{v1v2} & \sigma_{v2}^2 \end{bmatrix}$$

## 卡尔曼观测器模型引入

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} Rs + \frac{d}{dt}Ls & 0 \\ 0 & Rs + \frac{d}{dt}Ls \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + [we\varphi_f] \begin{bmatrix} -\sin \theta \\ \cos \theta \end{bmatrix}$$

$$\begin{aligned}\frac{di_\alpha}{dt} &= -\frac{Rs}{Ls} + \frac{we\varphi_f}{Ls} \sin \theta - \frac{u_\alpha}{Ls} \\ \frac{di_\beta}{dt} &= -\frac{Rs}{Ls} - \frac{we\varphi_f}{Ls} \cos \theta - \frac{u_\beta}{Ls}\end{aligned}$$

由于电机控制环路的采样时间非常短, 我们可以近似的认为每个时刻的转速不变, 进而简化卡尔曼观测器模型;

$$\begin{bmatrix} \frac{di_\alpha}{dt} \\ \frac{di_\beta}{dt} \\ \frac{d\omega}{dt} \\ \frac{d\theta}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{Rs}{Ls} & 0 & \frac{\varphi_f}{Ls} \sin \theta & 0 \\ 0 & -\frac{Rs}{Ls} & -\frac{we\varphi_f}{Ls} \cos \theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ we \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{Ls} & 0 \\ 0 & \frac{1}{Ls} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix}$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ e_\alpha \\ e_\beta \end{bmatrix}$$

## 状态空间方程引入

状态空间方程如下：

$$\begin{aligned}\widehat{X}_t &= A\widehat{X}_t + Bu_t \\ Z_t &= HX_t\end{aligned}$$

当t为1时离散化后：

$$\begin{aligned}\widehat{X}_k &= A\widehat{X}_{k-1} + Bu_{k-1} \\ Z_k &= HX_k\end{aligned}$$

$$\begin{aligned}\widehat{X}_k &= A\widehat{X}_{k-1} + Bu_k + W_k \\ Z_k &= HX_k + V_k\end{aligned}$$

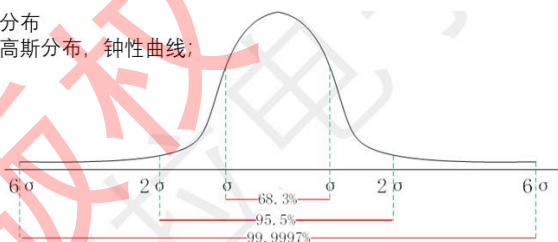
其中 $W_k$ 为过程噪音， $V_k$ 为测量噪音；

卡尔曼观测器的作用就是通过递归的思想，找到最准确的 $W_k$ 和 $V_k$ ，使得观测器计算的结果无限接近于实际系统。卡尔曼滤波器是对一系列随机信号的估算方法，所以与其说它属于滤波器，不如称它属于最优控制。

## 方差、标准差、协方差、六西格玛

| Iq电流 | Wm电机转速 |
|------|--------|
| 0.3  | 1000   |
| 0.4  | 1200   |
| 0.5  | 1300   |

正态分布  
又称高斯分布，钟形曲线；



协方差：

$$\sigma_{iq}\sigma_{wm} = \frac{(0.3-0.4)(1000-1198.57) + (0.4-0.4)(1200-1198.57) + (0.5-0.4)(990-1198.57)}{3} = 15$$

协方差矩阵：

$$P = \begin{bmatrix} \sigma_{iq}^2 & \sigma_{iq}\sigma_{wm} \\ \sigma_{iq}\sigma_{wm} & \sigma_{wm}^2 \end{bmatrix} = \begin{bmatrix} 0.00667 & 15 \\ 15 & 15555.56 \end{bmatrix}$$

|        | Iq电流    | Wm电机转速   |
|--------|---------|----------|
| Iq电流   | 0.00667 | 15       |
| Wm电机转速 | 15      | 15555.56 |

## 方差、标准差、协方差、六西格玛

| Iq电流 | Wm电机转速 |
|------|--------|
| 0.3  | 1000   |
| 0.4  | 1200   |
| 0.5  | 1300   |

平均值计算：

$$Iq\_Average = \frac{0.3 + 0.4 + 0.5}{3} = 0.4$$

$$Wm\_Average = \frac{1000 + 1200 + 1300}{3} = 1166.67$$

方差计算计算：

$$\sigma_{iq}^2 = \frac{(0.3 - 0.4)^2 + (0.4 - 0.4)^2 + (0.5 - 0.4)^2}{3} = 0.00667$$

$$\sigma_{wm}^2 = \frac{(1000 - 1166.67)^2 + (1200 - 1166.67)^2 + (1300 - 1166.67)^2}{3} = 15555.56$$

状态变量的先验值：

$$\hat{X}_k = \hat{X}_{k-1} + Ts(F(\hat{X}_{k-1}) + Bu)$$

状态转移矩阵：

$$\Phi = A = I + TsF(\hat{X}_{k-1})$$

误差协方差矩阵先验：

$$P_k^- = \Phi P_{k-1} \Phi^T + Q$$

卡尔曼增益系数：

$$K_k = \frac{P_k^- H^T}{(H P_k^- H^T + R)}$$

最优估计值：

$$\hat{X}_k = \hat{X}_k^- + K_k(Z_k - H \hat{X}_k^-)$$

更新最新估计值的误差协方差矩阵：

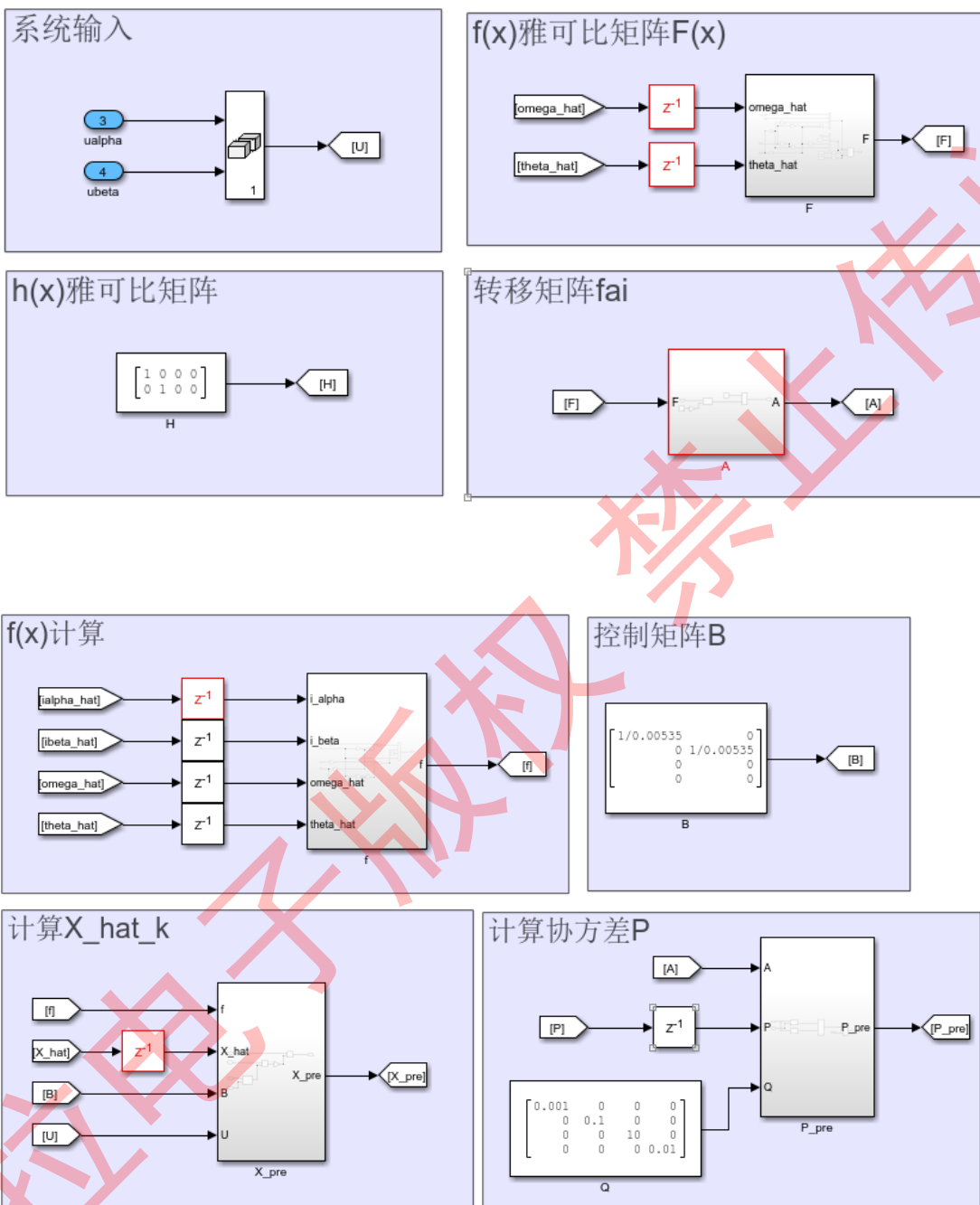
$$P_k = (I - K_k H) P_k^-$$

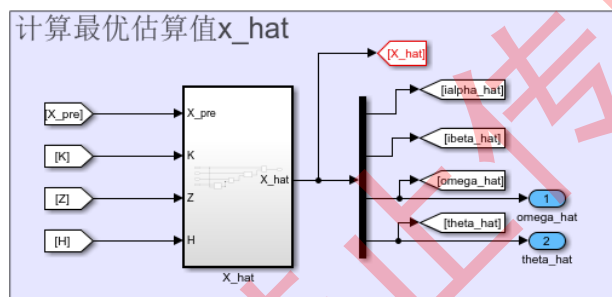
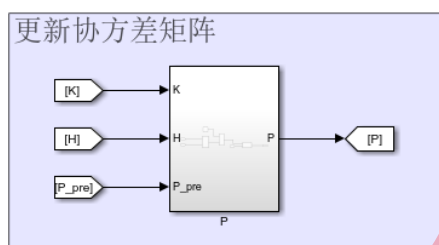
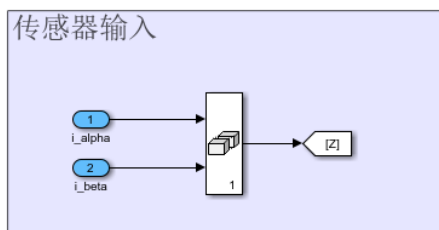
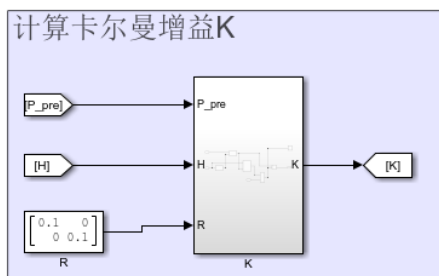
## 拓展卡尔曼观测器建模

其中，Q矩阵为过程噪声的协方差矩阵，R矩阵为测量噪声的协方差矩阵，对系统的收敛性有着决定性的影响，选取的不适当可能会导致收敛过慢、抖动过大甚至完全发散；

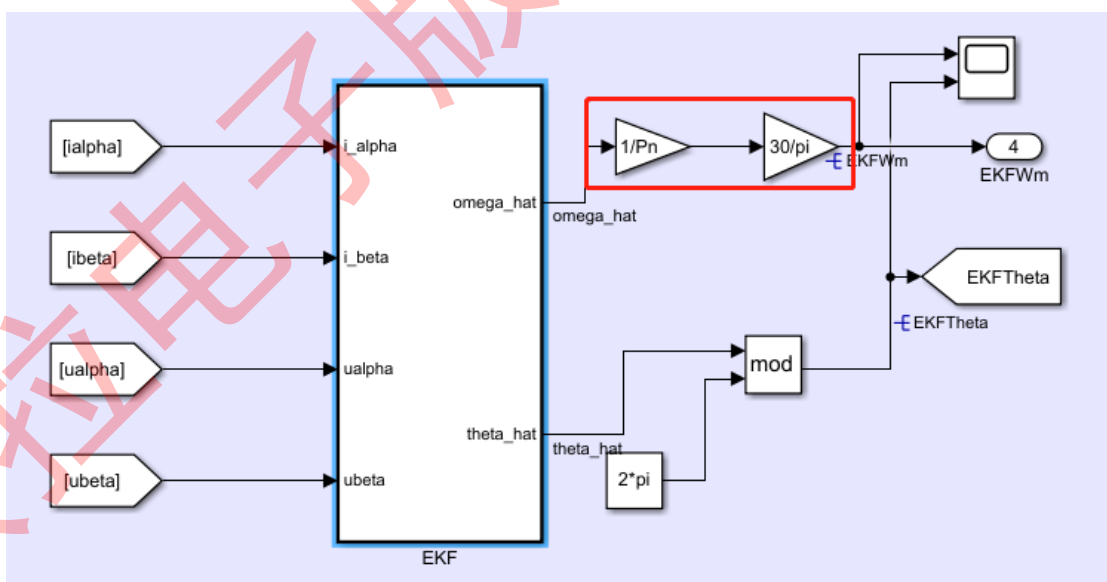
$$Q = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}, R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

① 基于以上数学模型，在 simulink 中建立模型；

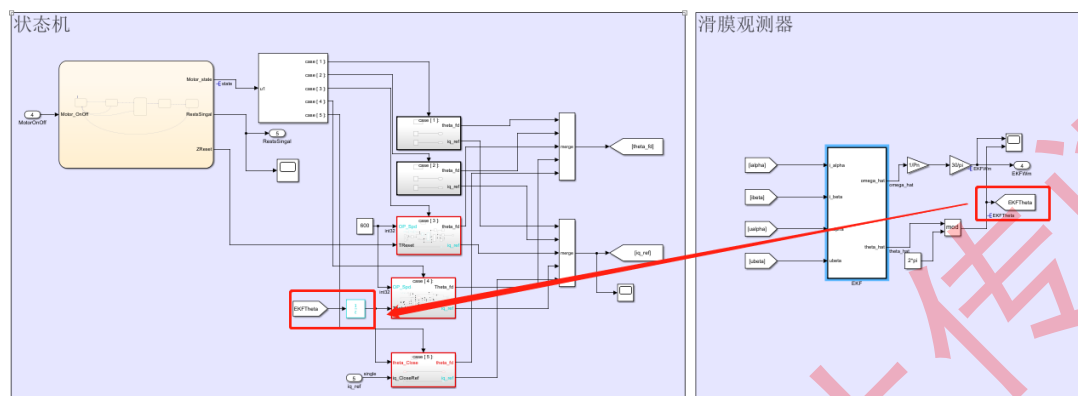




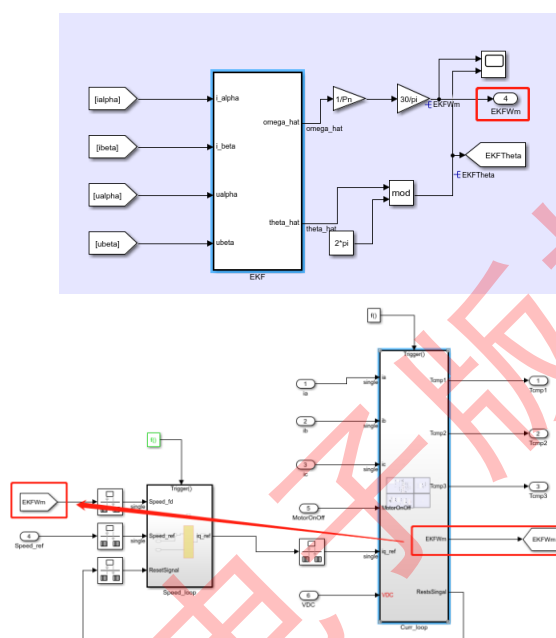
② 观测的转速从 rad/s 转为 rpm/min;



③ 观测到的 theta 角度最为闭环的 theta 输入;

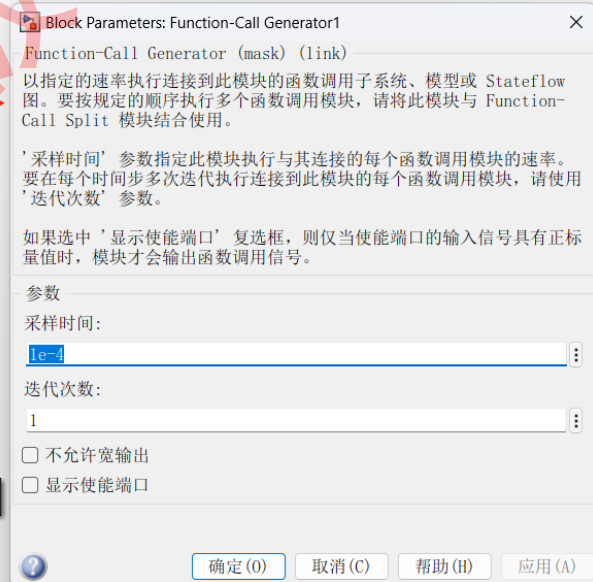
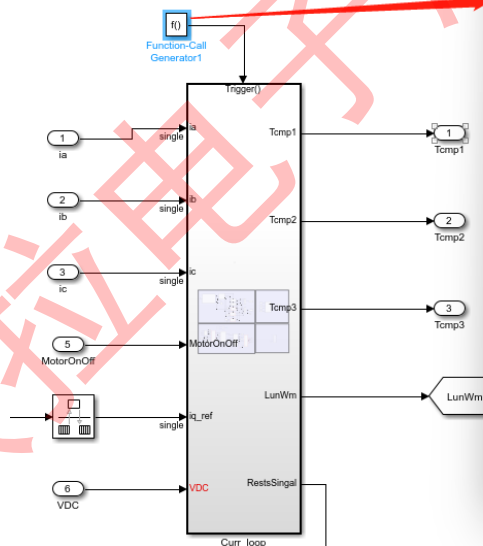
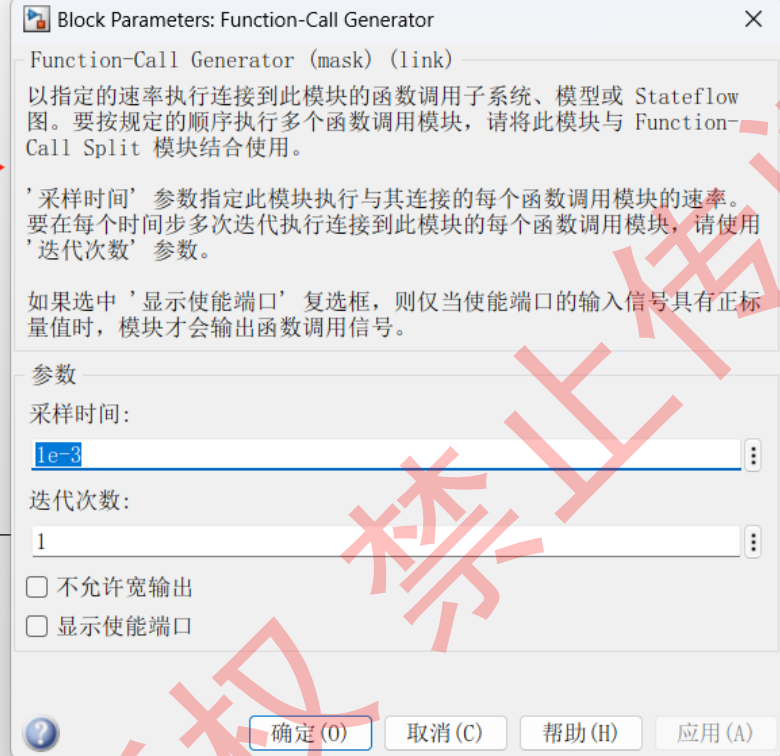
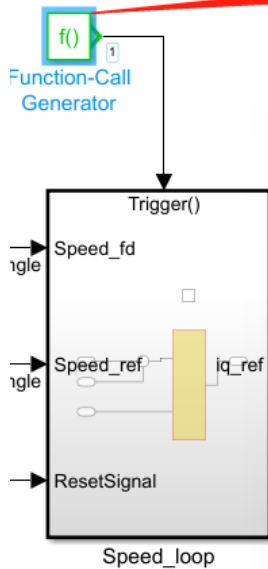


④ 观测到的  $\omega_m$  转速作为速度环闭环的实际转速输入：

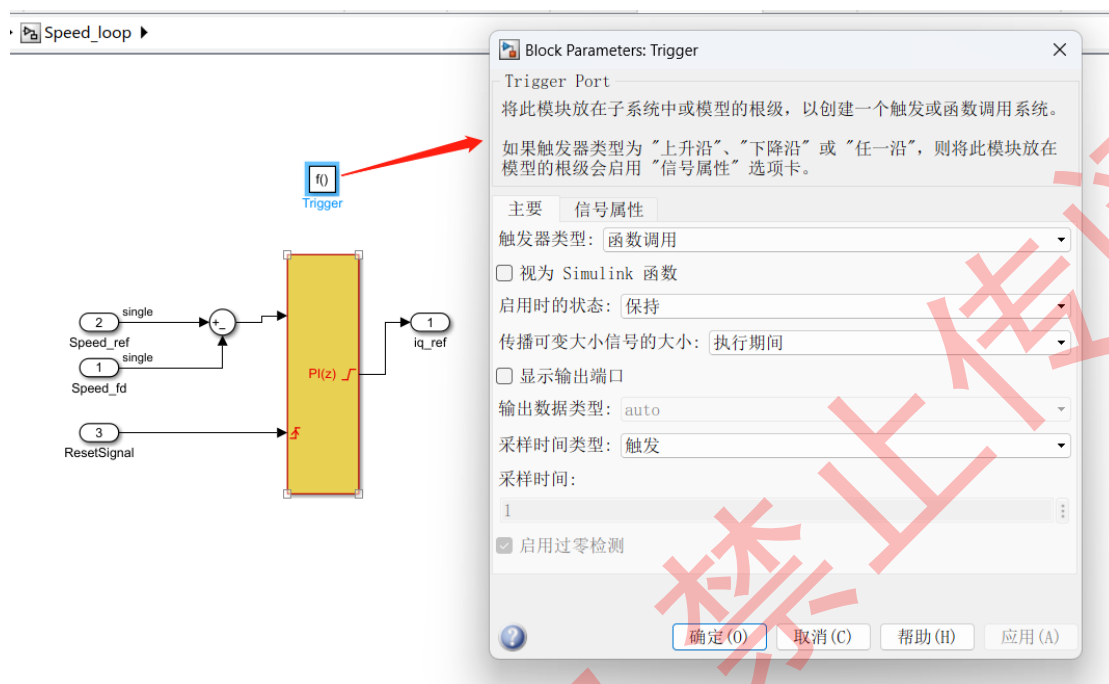


⑤ 由于速度环不需要太高的执行频率，故将速度环的采样时间改为  $1e-3$ ；电流环的采样时间保持  $1e-4$ ；

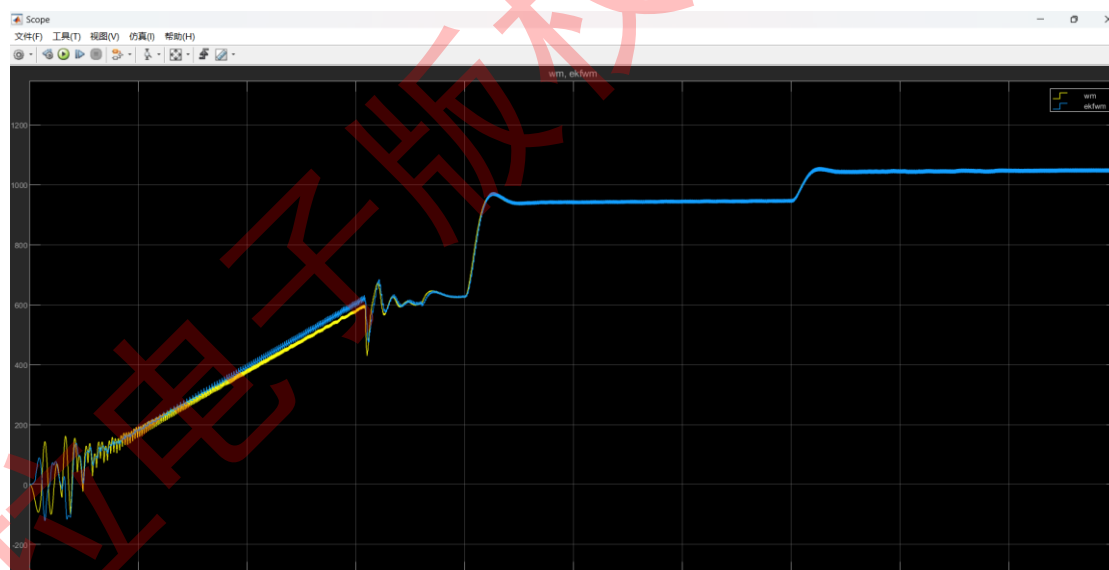




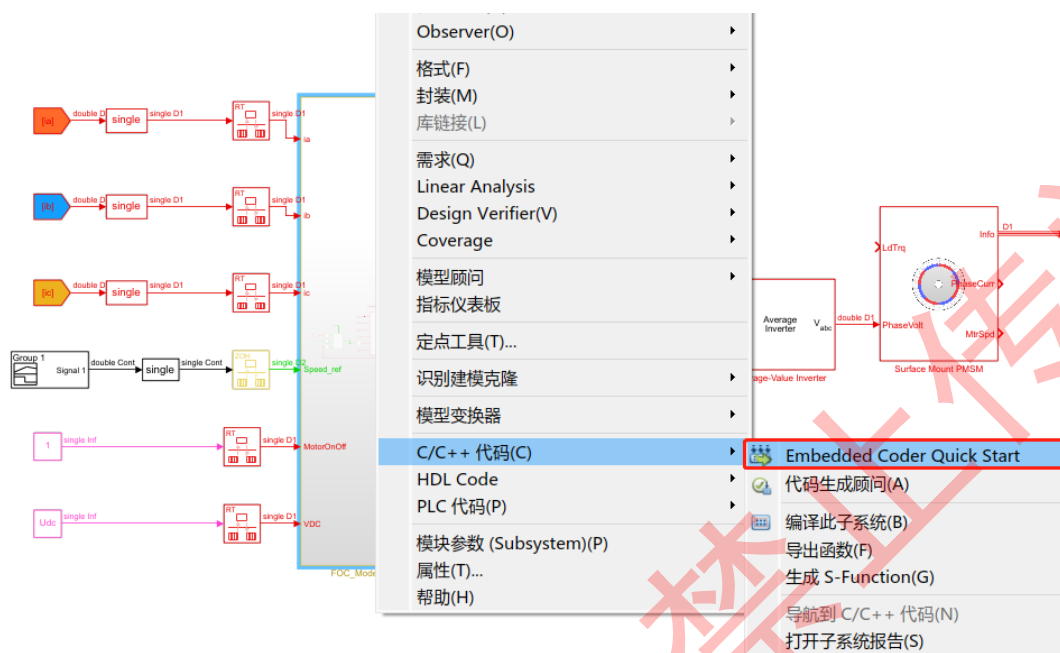
⑥ 在速度环及电流环的内部增加 trigger 模块;



⑦ 模型搭建完成后，电机仿真按钮，对比观测速度与电机实际转速；



⑧ 速度收敛较为完美，证明所搭建的观测器没有问题；右键点击 FOC\_Model 模块，点击生成代码；



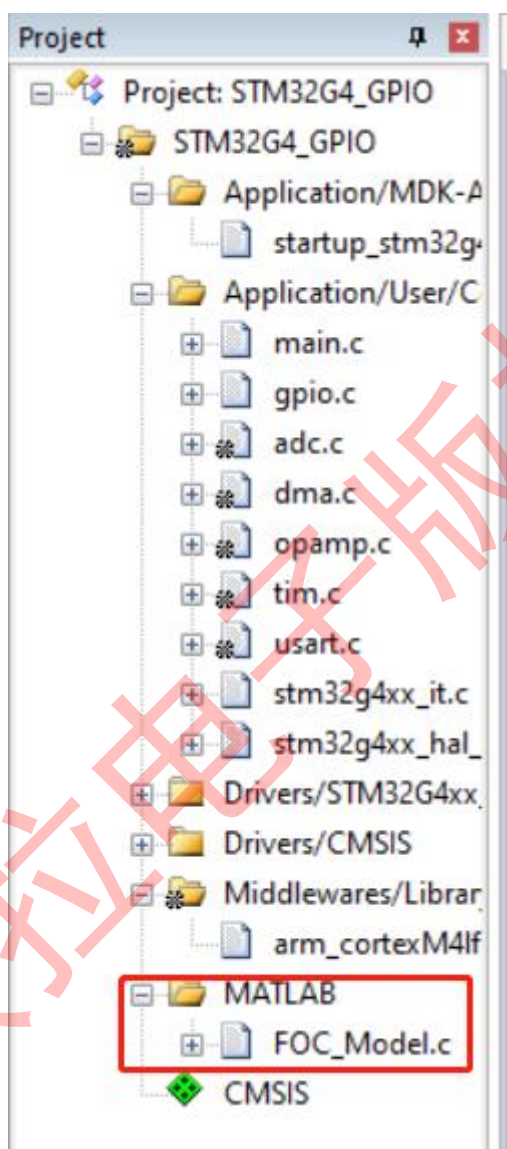
⑨ 生成代码的流程如第十章流程；

⑩ 将生成的 FOC 代码放进 Keil 工程中 MATLAB 文件夹；

| 名称                 | 修改日期            | 类型             | 大小    |
|--------------------|-----------------|----------------|-------|
| defines.txt        | 2023/8/22 21:11 | 文本文档           | 1 KB  |
| ert_main.c         | 2023/8/22 21:11 | C 源文件          | 4 KB  |
| FOC_Model.bat      | 2023/8/14 20:39 | Windows 批处理... | 1 KB  |
| FOC_Model.c        | 2023/8/22 21:11 | C 源文件          | 48 KB |
| FOC_Model.h        | 2023/8/22 21:11 | C Header 源文件   | 32 KB |
| FOC_Model.mk       | 2023/8/22 21:11 | Makefile 源文件   | 11 KB |
| FOC_Model.rsp      | 2023/8/20 12:23 | RSP 文件         | 1 KB  |
| FOC_Model_ref.rsp  | 2023/8/14 20:39 | RSP 文件         | 0 KB  |
| modelsources.txt   | 2023/8/22 21:11 | 文本文档           | 1 KB  |
| rtw_proj.tmw       | 2023/8/14 20:39 | TMW 文件         | 1 KB  |
| rtwtypes.h         | 2023/8/22 21:11 | C Header 源文件   | 4 KB  |
| rtwtypeschksum.mat | 2023/8/22 21:11 | MAT 文件         | 2 KB  |

| 名称          | 修改日期            | 类型           | 大小    |
|-------------|-----------------|--------------|-------|
| FOC_Model.c | 2023/8/22 21:11 | C 源文件        | 48 KB |
| FOC_Model.h | 2023/8/22 21:11 | C Header 源文件 | 32 KB |
| rtwtypes.h  | 2023/8/22 21:11 | C Header 源文件 | 4 KB  |

⑪ 打开 Keil 工程，在工程目录中 MATLAB 文件夹增加 FOC\_MODEL.C 文件；



⑫ Main.c 中包含 FOC\_Model.h；

```

1  /* Private includes -----
2  /* USER CODE BEGIN Includes */
3  #include <stdio.h>
4  #include <string.h>
5  #include "FOC_Model.h"
6  /* USER CODE END Includes */

```

- ⑬ 通过电位器旋钮调节算法模型输入的参考转速，转速在 300-1200 之间；

ADC 采样的母线电压作为模型 Vbus 输入；

```

7  {
8      HAL_ADC_Start(&hadc1);
9      HAL_ADC_Start(&hadc2);
10     adc1_in4 = HAL_ADC_GetValue(&hadc1);
11     adc_vbus = HAL_ADC_GetValue(&hadc2);
12     rtU.VDC = adc_vbus*3.3f/4096*26;
13     finalspped = (adc1_in4>>4) * 15;
14     if(finalspped >=0 && finalspped <300)
15     {
16         finalspped = 300;
17     }
18     if(finalspped > 1200)
19     {
20         finalspped = 1200;
21     }
22     rtU.Speed_ref = finalspped;
23     HAL_Delay(10);

```

- ⑭ 在 ADC 中断中读取三相电流值输入算法模型，执行算法代码；将算法模型输出作为 TIM1 的比较值写入寄存器；同时通过 UART 上传数据至 VOFA 上位机显示波形；

```

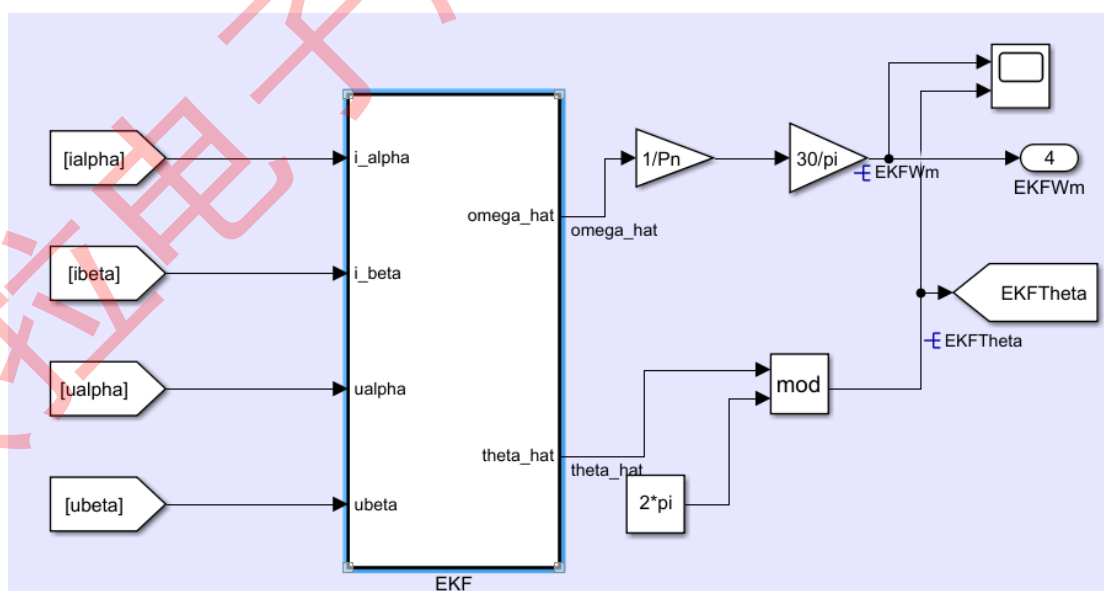
337     else
338     {
339         adc1_in1 = hadc1.Instance->JDR1;
340         adc1_in3 = hadc1.Instance->JDR2;
341         adc1_in2 = hadc2.Instance->JDR1;
342         rtU.ia = (adc1_in1 - IA_Offset)*0.02197f;
343         rtU.ib = (adc1_in2 - IB_Offset)*0.02197f;
344         rtU.ic = (adc1_in3 - IC_Offset)*0.02197f;
345         FOC_Model_step();
346         TIM1->CCR1 = rtY.Tcmp1;
347         TIM1->CCR2 = rtY.Tcmp2;
348         TIM1->CCR3 = rtY.Tcmp3;
349         load_data[0] = rtU.ia;
350         load_data[1] = EKFTheta;
351         load_data[2] = EKFWm;
352         load_data[3] = rtU.Speed_ref;
353         load_data[4] = 0;
354         memcpy(tempData, (uint8_t *)&load_data, sizeof(load_data));
355         HAL_UART_Transmit_DMA(&huart3, (uint8_t *)tempData, 6*4);
356     }
357 }

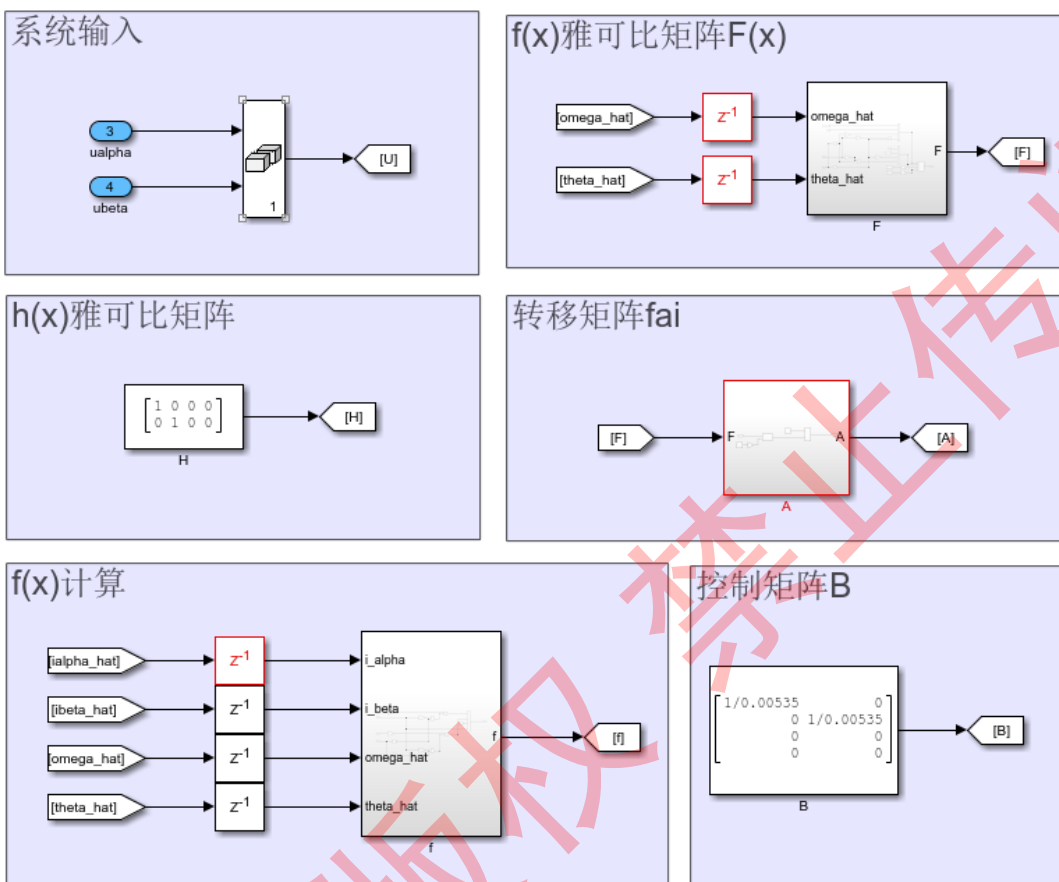
```

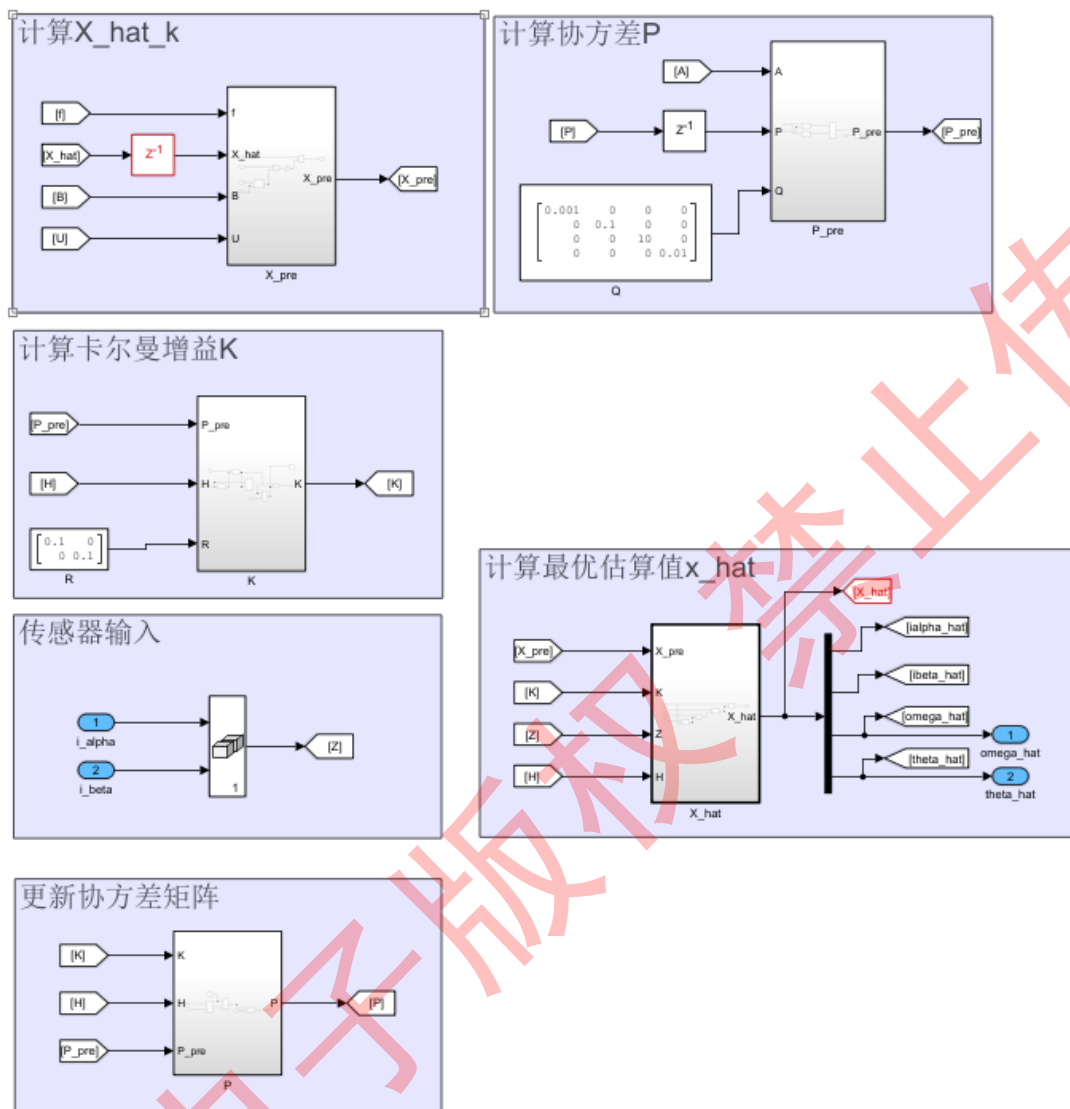
⑮ 编译代码：

## 2. 拓展卡尔曼观测器模块内部及数据字典截图

① 模型总览：



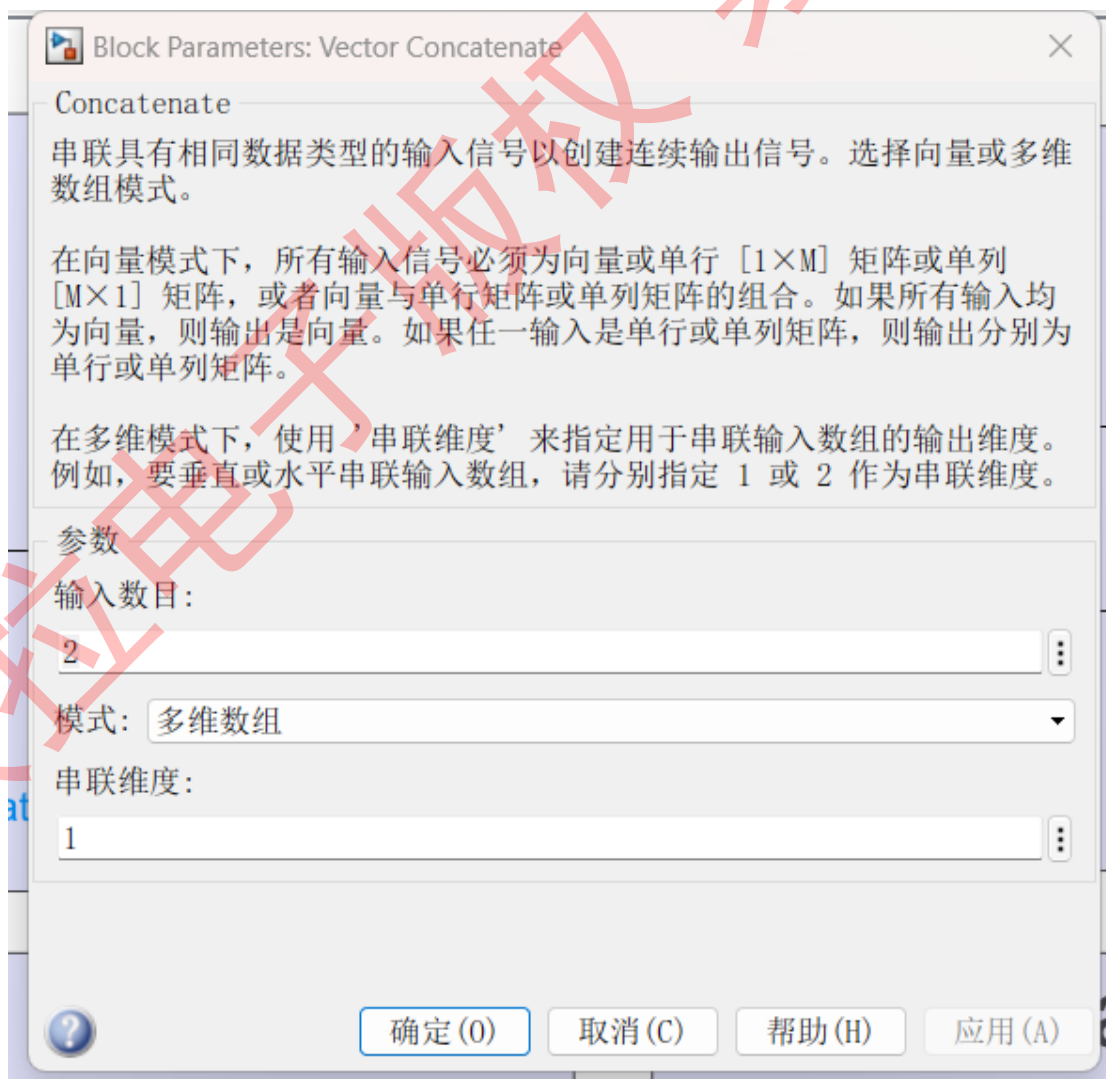
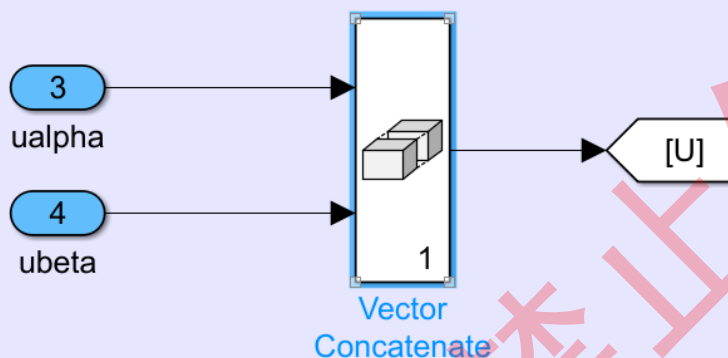




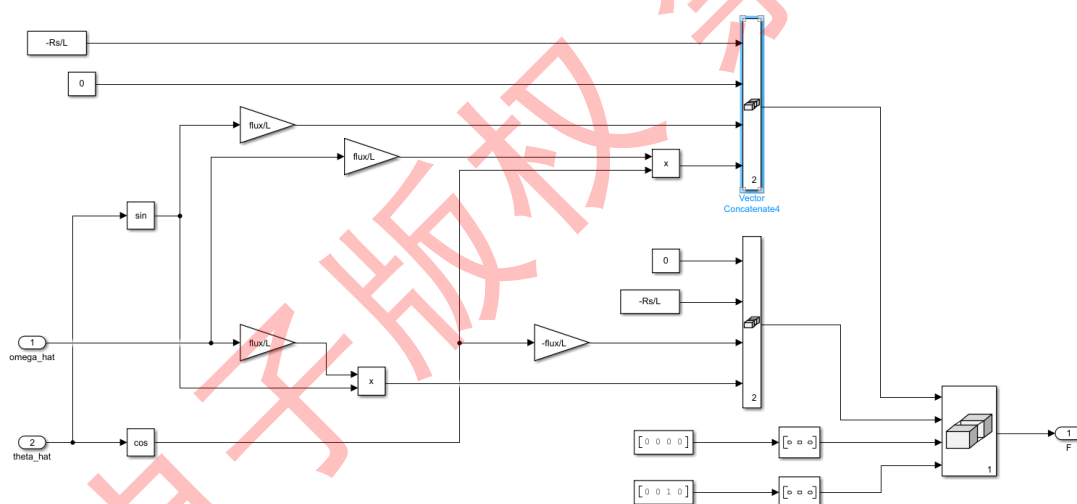
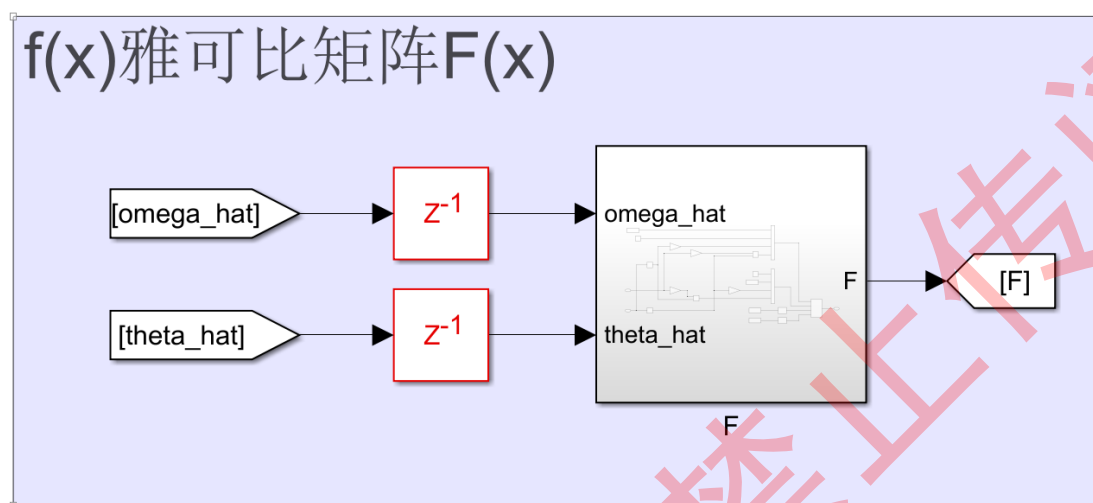
② 系统输入模块：



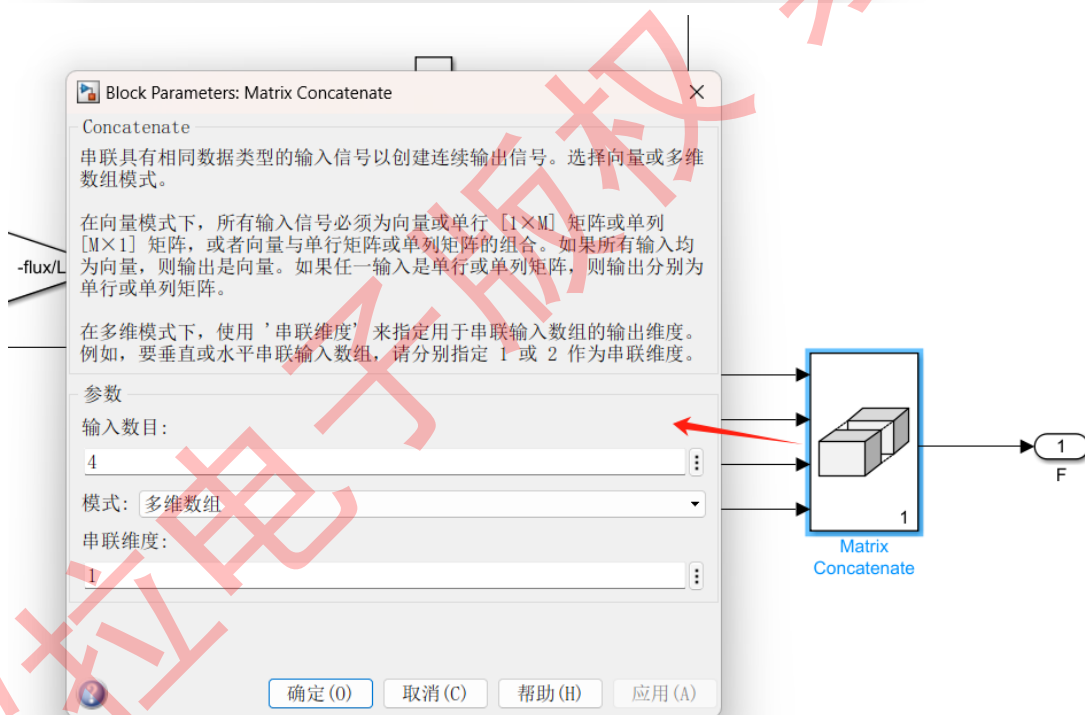
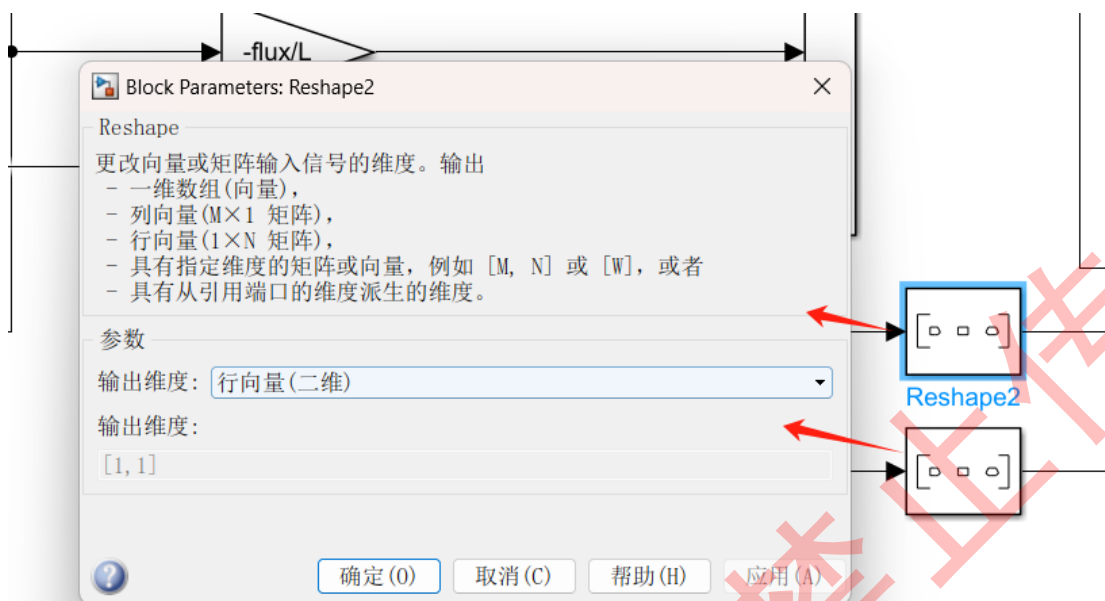
## 系统输入



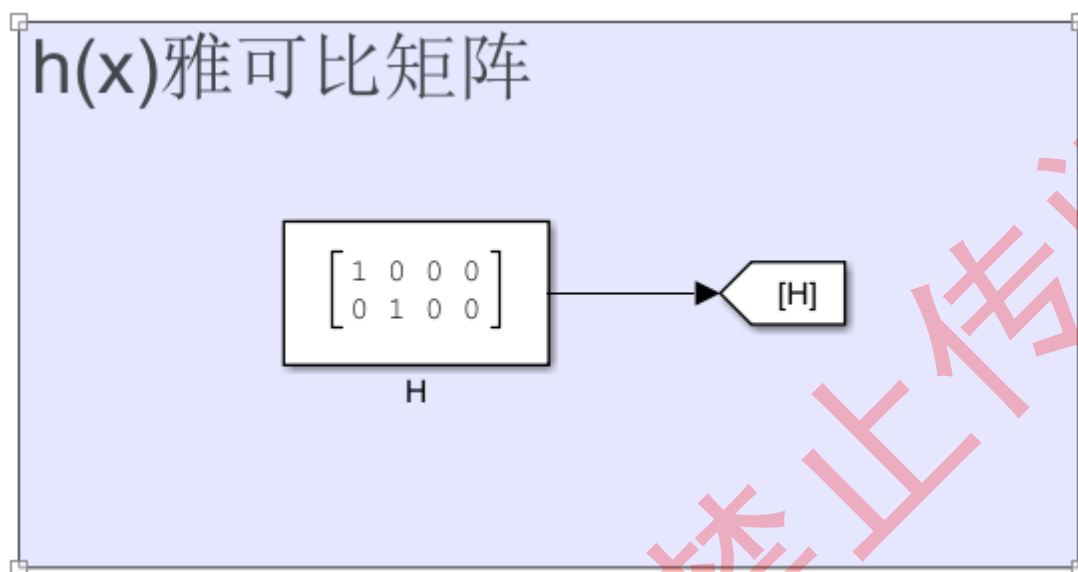
③  $f(x)$ 雅可比矩阵:



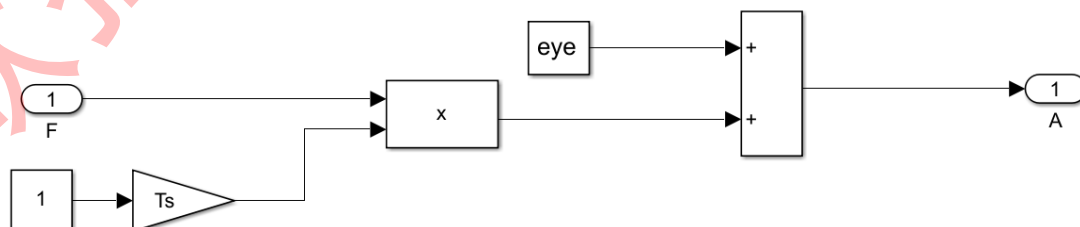
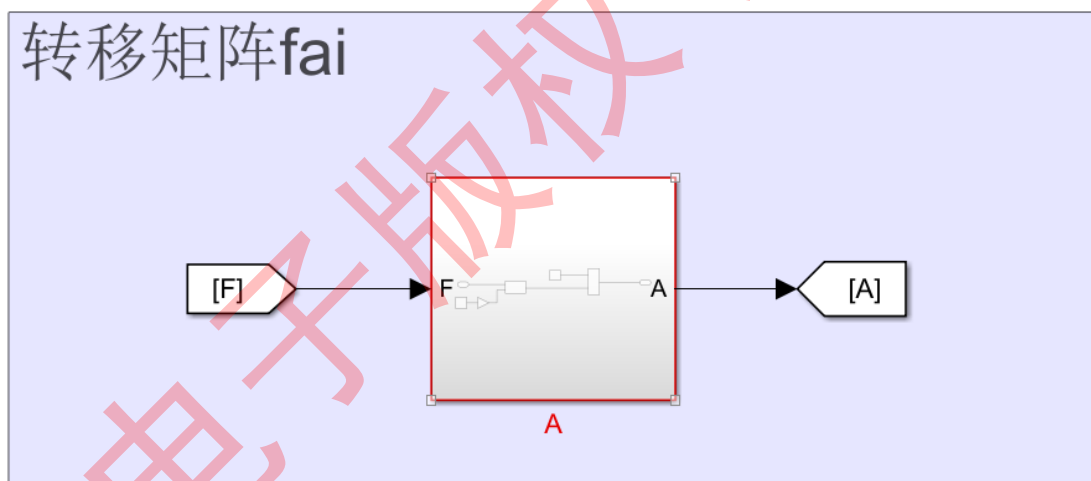


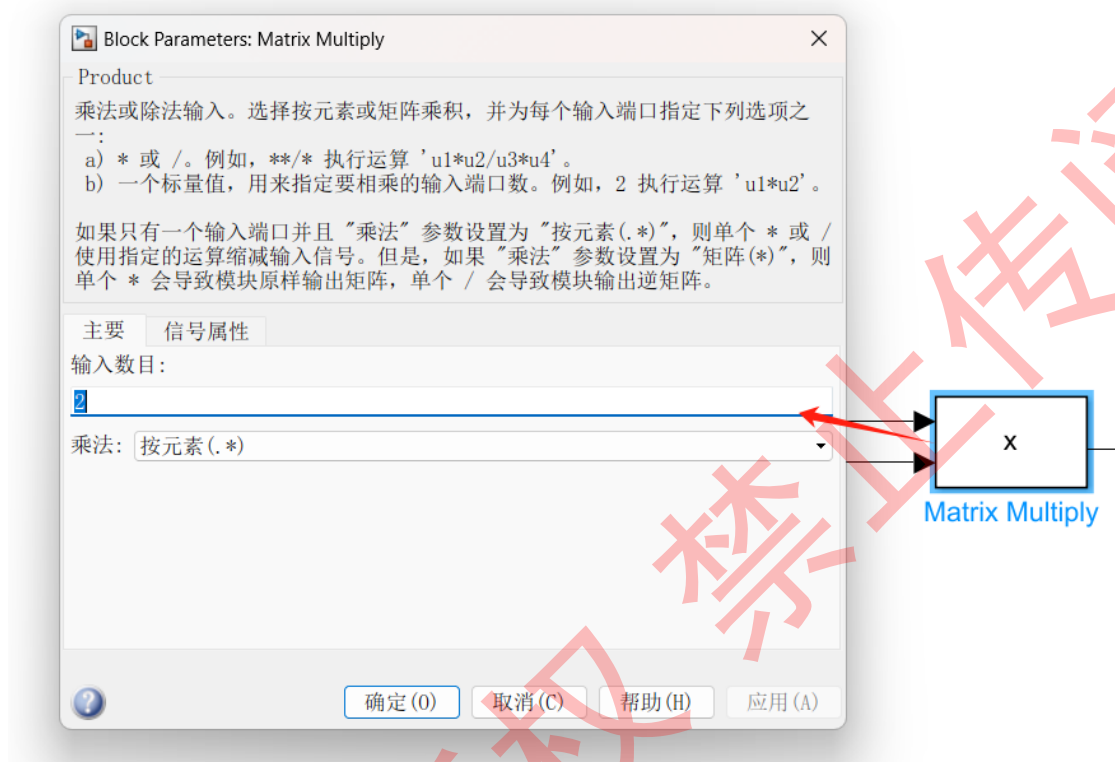


#### ④ $h(x)$ 雅可比矩阵



⑤ 转移矩阵 fai

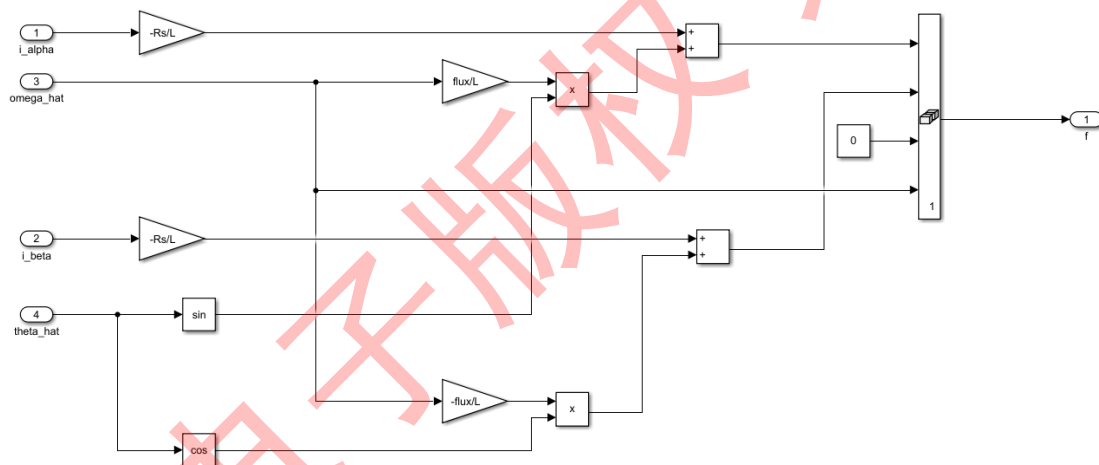
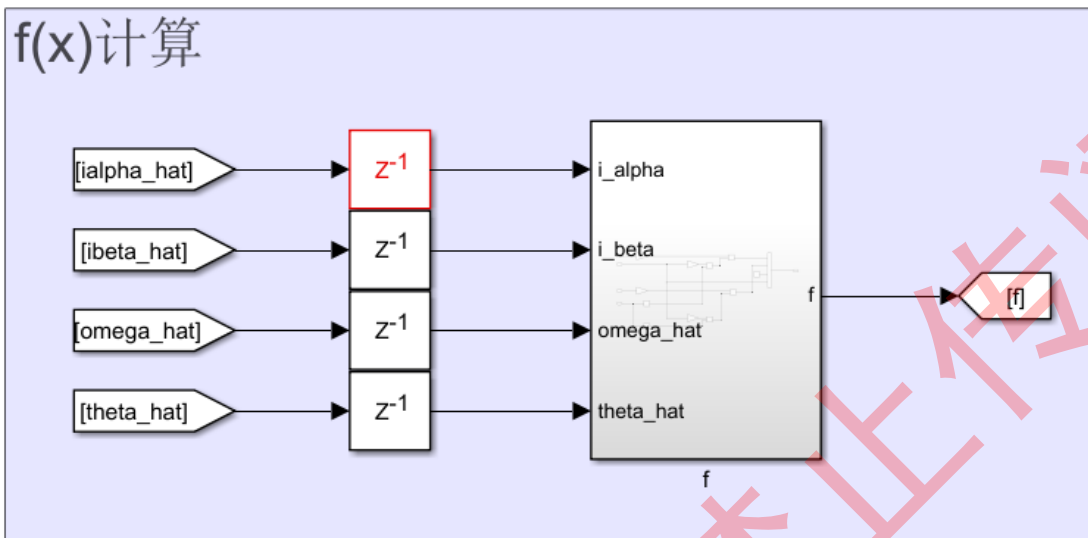




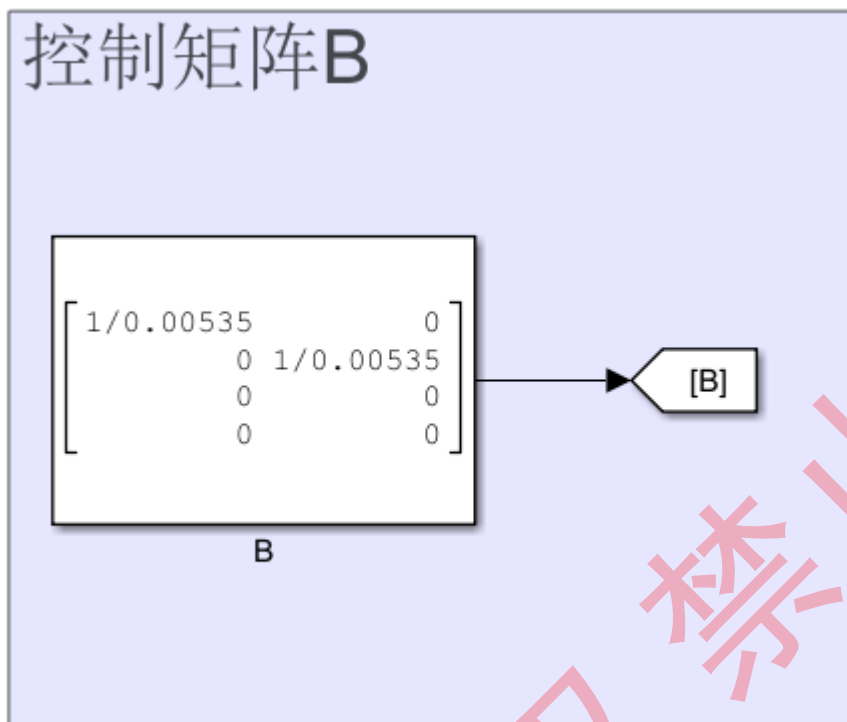
## ⑥ f(x)计算

版本: V13

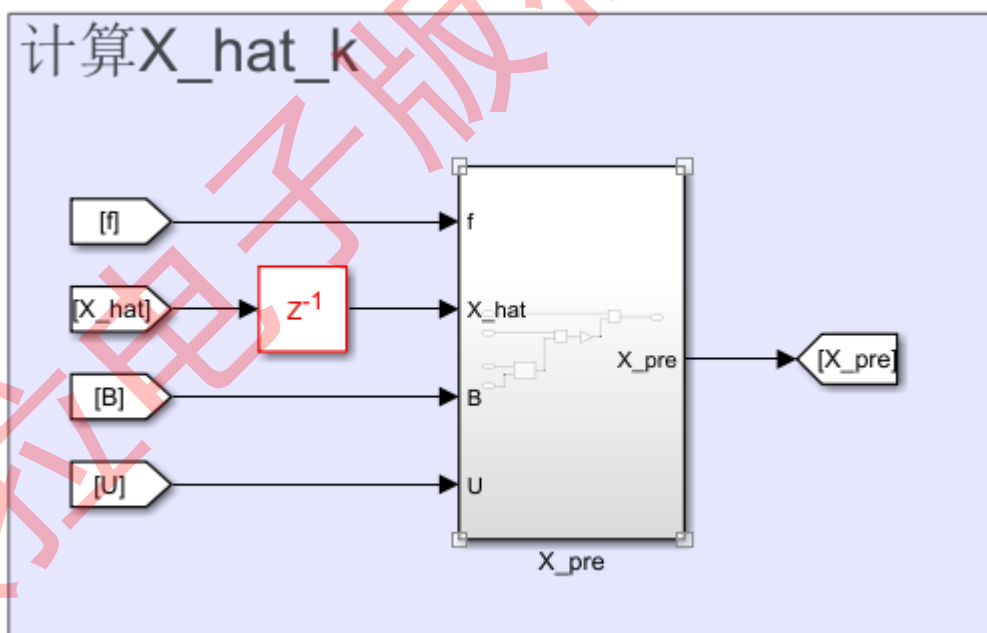
B 站 up 主: 欧拉电子



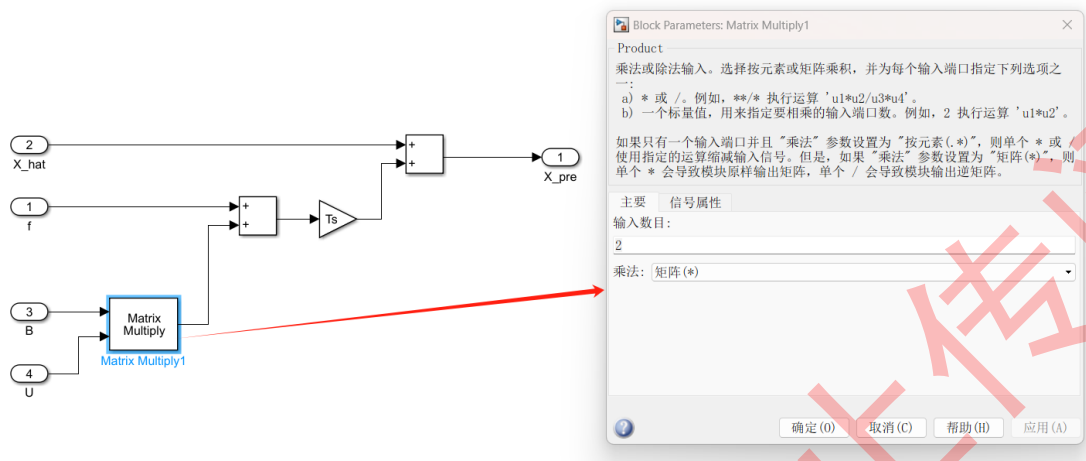
⑦ 控制矩阵 B



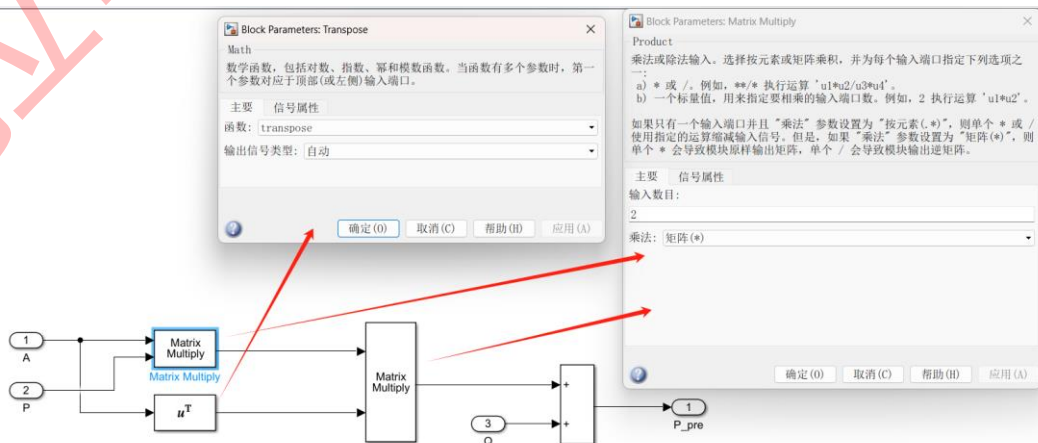
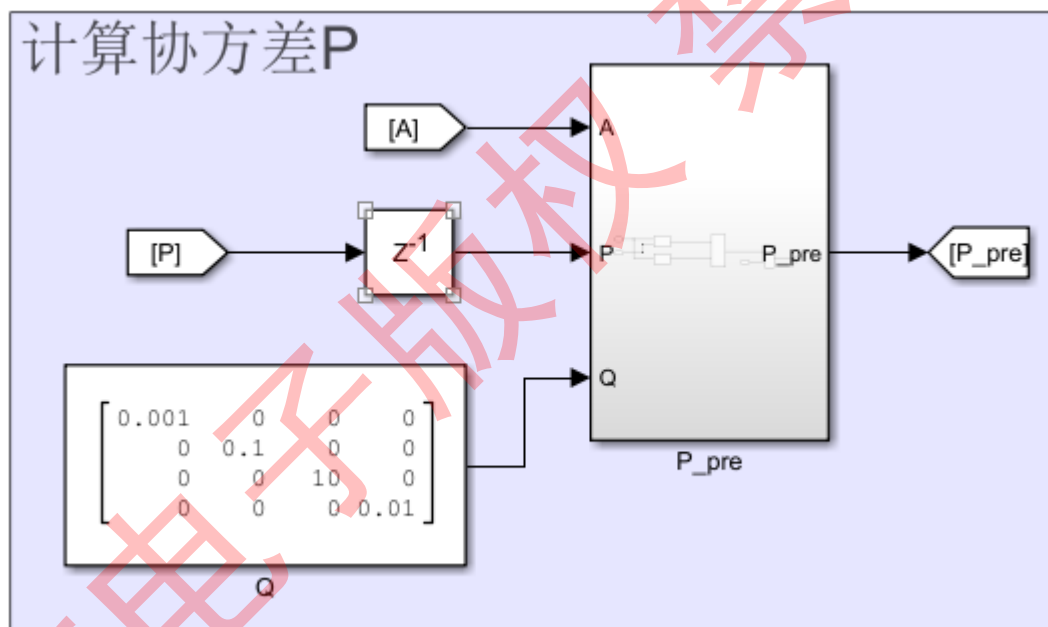
⑧ 计算  $X_{\text{hat}_k}$



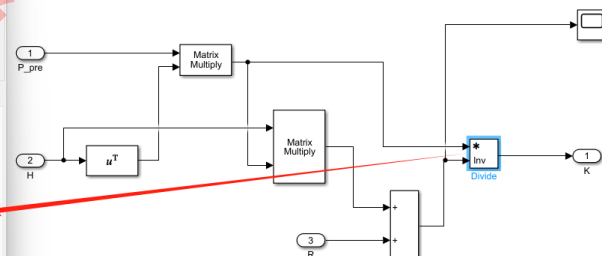
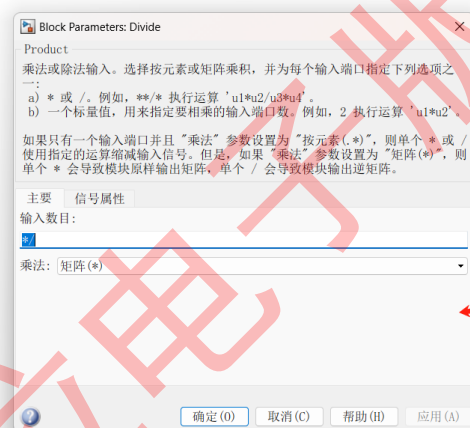
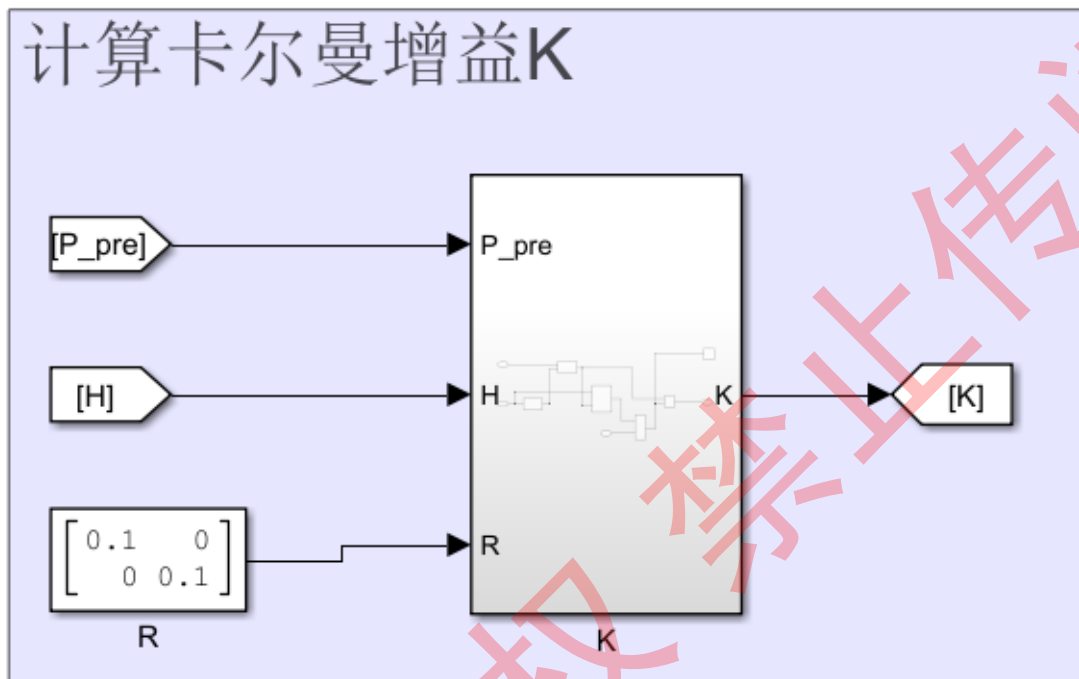




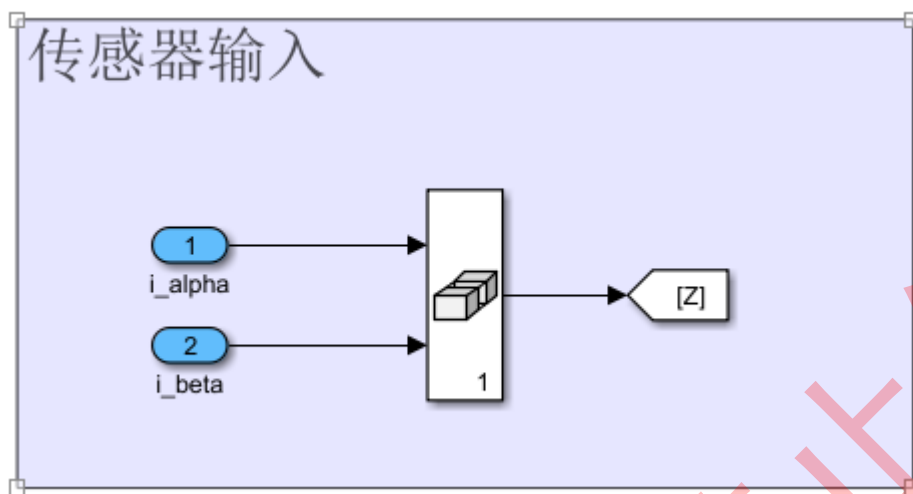
### ⑨ 计算协方差 P



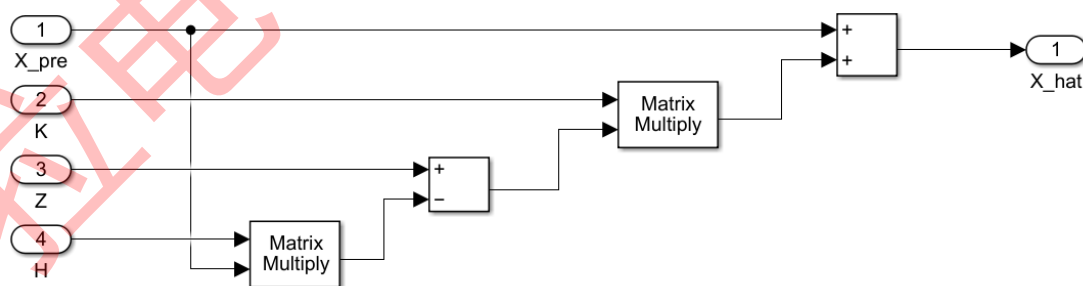
# ⑩ 计算卡尔曼增益 K



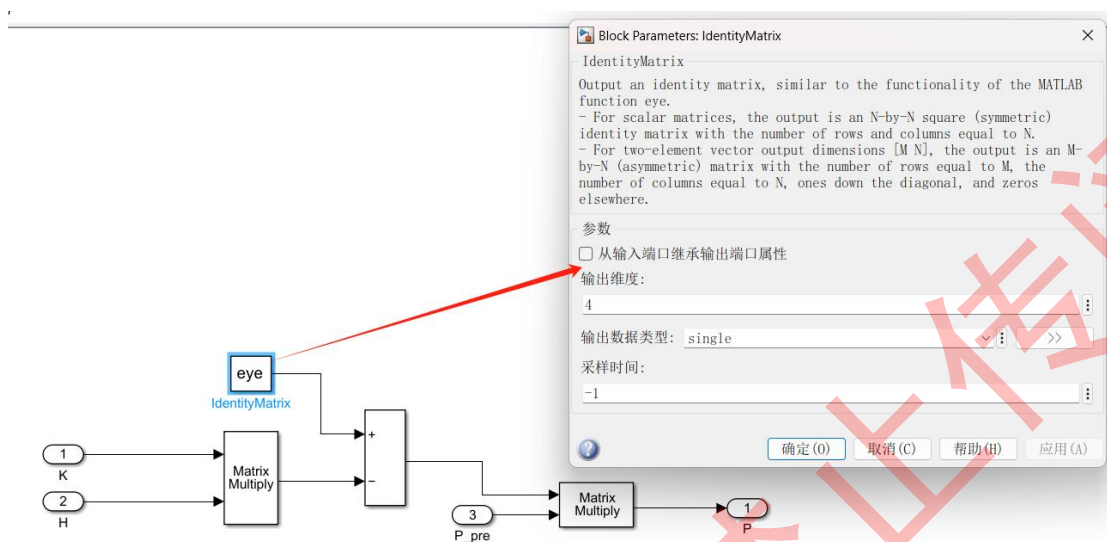
# ⑪ 传感器输入



### ⑫ 计算最优估算值 $\hat{x}$



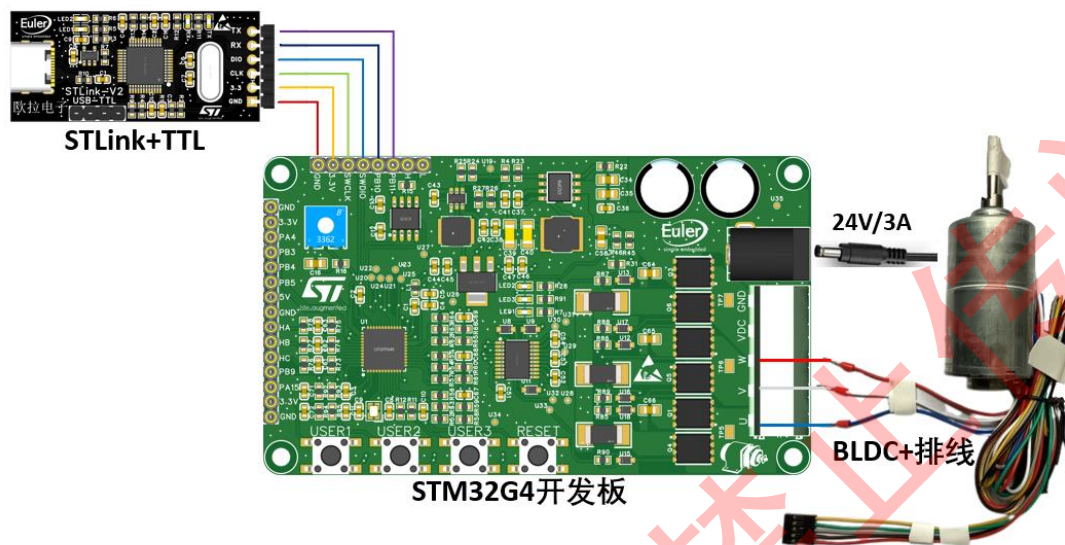
### ⑬ 更新协方差矩阵



|           | Name      | Status | Value        | DataType | Dimensions | Complexity | Min | Max | Unit | Argument | StorageClass   | DataSource    |
|-----------|-----------|--------|--------------|----------|------------|------------|-----|-----|------|----------|----------------|---------------|
| [101 010] | Tpwm      |        | 16000        | uint16   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_EKF.sldc |
| [101 010] | ialpha    |        |              | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_EKF.sldc |
| [101 010] | ibeta     |        |              | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_EKF.sldc |
| [101 010] | ualpha    |        |              | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_EKF.sldc |
| [101 010] | ubeta     |        |              | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_EKF.sldc |
| [101 010] | Pn        |        | 4            | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_EKF.sldc |
| [101 010] | Ts        |        | 0.0001       | single   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_EKF.sldc |
| [101 010] | Udc       |        | 24           | single   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_EKF.sldc |
| [101 010] | state     |        |              | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_EKF.sldc |
| [101 010] | L         |        | 0.00535      | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_EKF.sldc |
| [101 010] | Rs        |        | 6.97         | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_EKF.sldc |
| [101 010] | flux      |        | 0.016884     | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_EKF.sldc |
| [101 010] | spd_kp    |        | 0.0031514    | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_EKF.sldc |
| [101 010] | spd_ki    |        | 0.0001229046 | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_EKF.sldc |
| [101 010] | curr_d_ki |        | 35           | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_EKF.sldc |
| [101 010] | curr_d_kp |        | 0.025        | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_EKF.sldc |
| [101 010] | EKFTheta  |        |              | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_EKF.sldc |
| [101 010] | EKFWm     |        |              | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_EKF.sldc |

### 3. 无感拓展卡尔曼实验现象

① 按以下图片接线，并讲 USB 转 UART 模块连接至电脑：

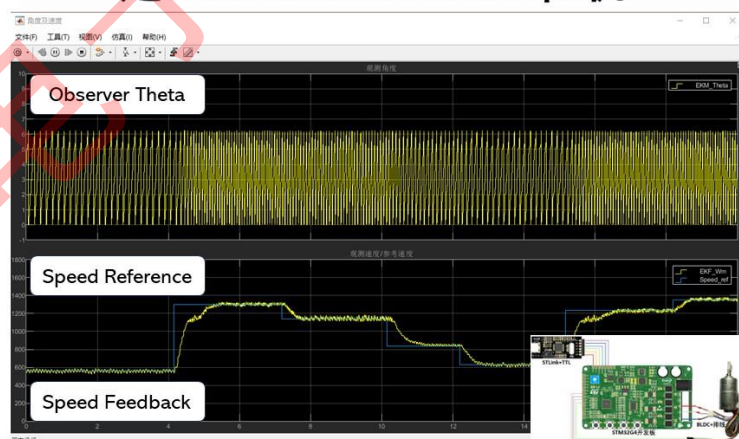


② 下载代码，并打开 VOFA 上位机；

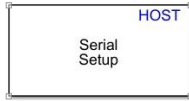
## 第二十一章：Simulink 上位机

### STM32G4 FOC

建立Simulink FOC上位机



## 1.Host Serial Setup



**Block Parameters: Host Serial Setup**

Host Serial Setup  
Configure the host-side serial communications interface.

| Serial 1                 | Serial 2 | Serial 3 | Serial 4 |
|--------------------------|----------|----------|----------|
| Port name: COM3          |          |          |          |
| Baud rate: 2000000       |          |          |          |
| Number of stop bits: 1   |          |          |          |
| Parity mode: none        |          |          |          |
| Timeout: 1.0             |          |          |          |
| Byte order: LittleEndian |          |          |          |

确定(O) 取消(C) 帮助(H) 应用(A)

1. 连接USB转UART模块后，找到对应的端口

2. 选择波特率与下位机保持一致

3. 停止位1与下位机保持一致

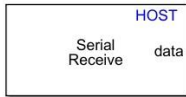
4. 无校验位与下位机保持一致

5. 超时时间1s

6. 字节命令选择小端模式

### 下位机UART配置

```
huart3.Instance = USART3;
huart3.Init.BaudRate = 2000000;
huart3.Init.WordLength = UART_WORDLENGTH_8B;
huart3.Init.StopBits = UART_STOPBITS_1;
huart3.Init.Parity = UART_PARITY_NONE;
huart3.Init.Mode = UART_MODE_TX_RX;
huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart3.Init.OverSampling = UART_OVERSAMPLING_16;
huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
huart3.Init.ClockPrescaler = UART_PRESCALER_DIV1;
huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
```



**Block Parameters: Host Serial Receive**

Host Serial Receive  
Configure the host-side serial communications interface to receive data from serial port.  
The data port outputs the received data.

参数

Serial connection: Serial 1

Additional package header: "SSSS"

Additional package terminator: "EEEE"

Data type: single

Data size: 1000 6

Initial output: 0

Action taken when connection times out: Output custom value

Output value when connection times out: 0

Sample time: 1

☐ Output receiving status

☒ Enable blocking mode

确定(O) 取消(C) 帮助(H) 应用(A)

1. 选择对应serial 1

2. 数据帧头

3. 数据帧尾

4. 上传浮点数据

5. 一帧数据长度

6. 初始输出0

7. 接收超时输出自定义值

8. 自定义值0

9. 采样时间1s

## 2.Host Serial Receive

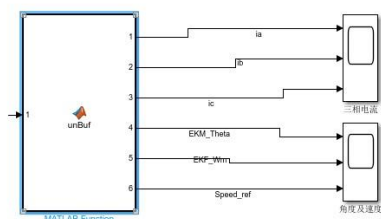
### 下位机上传数据帧函数

```
void UART_UPLOAD(void)
{
    if(Loadcnt == 0)
    {
        Loadcnt++;
        load_data[0] = "(float)" "SSSS";
        load_data[1] = rtU.ia;
        load_data[2] = rtU.ib;
        load_data[3] = rtU.ic;
        load_data[4] = EKFFtheta;
        load_data[5] = EKFFw;
        load_data[6] = rtU.Speed_ref;
        _HAL_UNLOCK(huart3.hdmatrix);
        huart3->State = HAL_UART_STATE_READY;
        huart3.hdmatrix->State = HAL_DMA_STATE_READY;
        HAL_UART_Transmit_DMA(huart3, (uint8_t *)load_data, 7*4);
    }
    else if(Loadcnt == 999)
    {
        Loadcnt = 0;
        load_data[0] = rtU.ia;
        load_data[1] = rtU.ib;
        load_data[2] = rtU.ic;
        load_data[3] = EKFFtheta;
        load_data[4] = EKFFw;
        load_data[5] = rtU.Speed_ref;
        load_data[6] = "(float)" "EEEE";
        _HAL_UNLOCK(huart3.hdmatrix);
        huart3->State = HAL_UART_STATE_READY;
        huart3.hdmatrix->State = HAL_DMA_STATE_READY;
        HAL_UART_Transmit_DMA(huart3, (uint8_t *)load_data, 7*4);
    }
    else
    {
        Loadcnt++;
        load_data[0] = rtU.ia;
        load_data[1] = rtU.ib;
        load_data[2] = rtU.ic;
        load_data[3] = EKFFtheta;
        load_data[4] = EKFFw;
        load_data[5] = rtU.Speed_ref;
        _HAL_UNLOCK(huart3.hdmatrix);
        huart3->State = HAL_UART_STATE_READY;
        huart3.hdmatrix->State = HAL_DMA_STATE_READY;
        HAL_UART_Transmit_DMA(huart3, (uint8_t *)load_data, 6*4);
    }
}
```



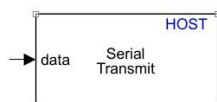
### 3.数据转化

$$\begin{matrix} k1 \\ k2 \\ k3 \\ \dots \\ k1000 \end{matrix} \begin{bmatrix} ia & ib1 & ic1 & EKM\_Theta & EKF\_Wm & Speed\_ref \\ ia & ib2 & ic2 & EKM\_Theta & EKF\_Wm & Speed\_ref \\ ia & ib3 & ic3 & EKM\_Theta & EKF\_Wm & Speed\_ref \\ \dots & \dots & \dots & \dots & \dots & \dots \\ ia & ib & ic & EKM\_Theta & EKF\_Wm & Speed\_ref \end{bmatrix}$$

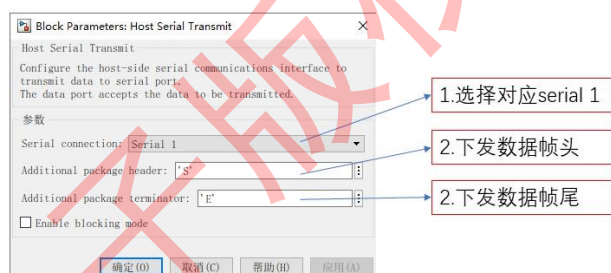
$$\begin{bmatrix} ia1 & ia2 & \dots & ia1000 \\ ib1 & ib2 & \dots & ib1000 \\ ic1 & ic2 & \dots & ic1000 \\ EKM\_Theta1 & EKM\_Theta2 & \dots & EKM\_Theta1000 \\ EKF\_Wm1 & EKF\_Wm2 & \dots & EKF\_Wm1000 \\ Speed\_ref1 & Speed\_ref2 & \dots & Speed\_ref1000 \end{bmatrix}$$


```
function [Output1,Output2,Output3,Output4,Output5,Output6] = unBuf(Input)

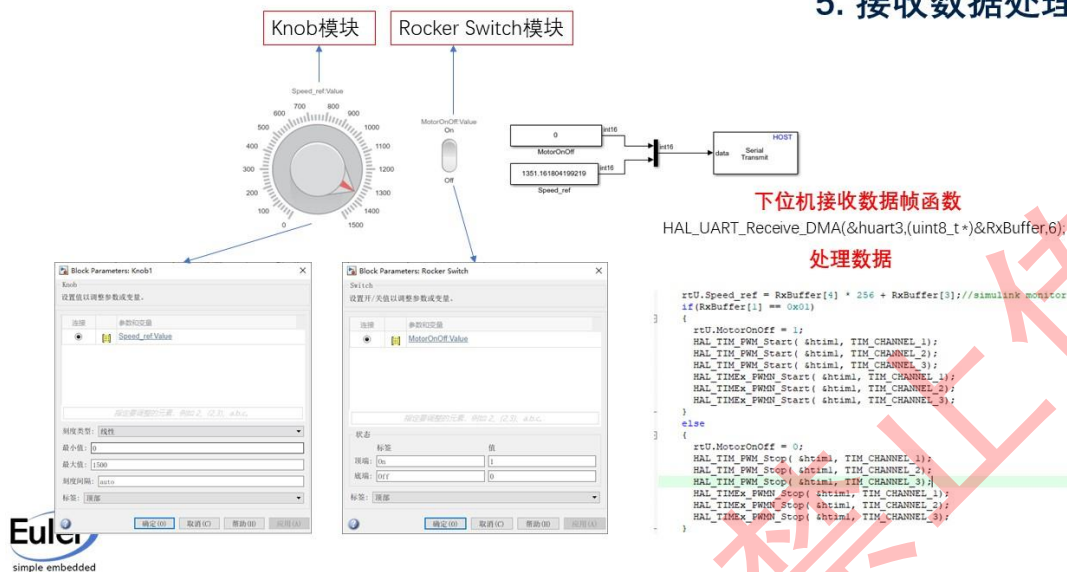
Output1 = Input(:, 1);
Output2 = Input(:, 2);
Output3 = Input(:, 3);
Output4 = Input(:, 4);
Output5 = Input(:, 5);
Output6 = Input(:, 6);
end
```



### 4. Host Serial Transmit

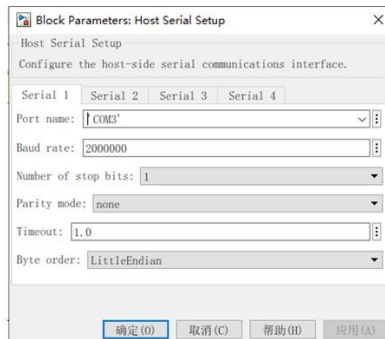
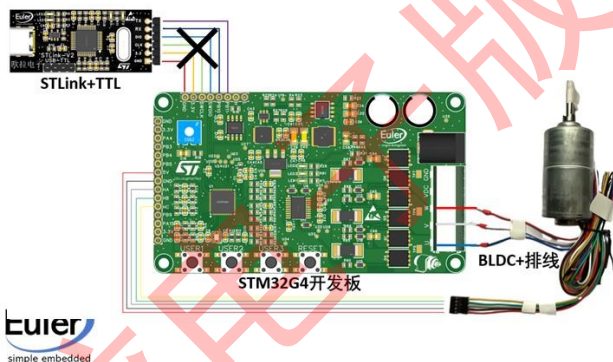


## 5. 接收数据处理



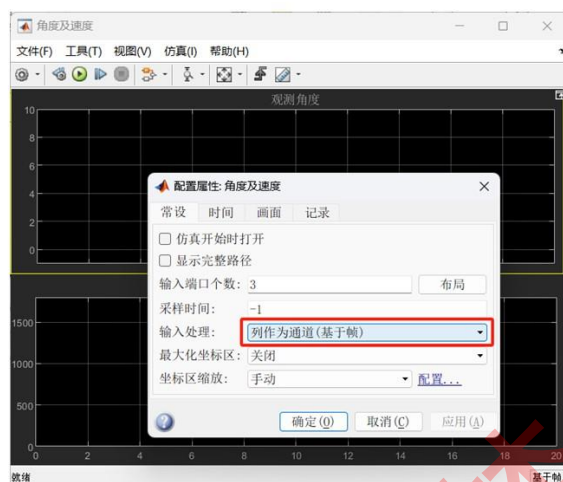
## 6. 接线流程

- 1.使用USB Type C将STLink+TTL工具连接至电脑, 此时先不要连接至G4开发板;
- 2.在Host Serial Setup中找到对应的COM口, 再将STLink+TTL工具连接至G4开发板;
- 3.运行上位机仿真;
- 4.下载程序至G4开发板并运行程序;





## 7. 示波器输入处理设置为基于帧



## 第二十二章：电机参数转换说明

电机参数更新说明：Pn 极对数由 4 更新为 2，反电动势常数由 12.25 更新为 6.13，两组电机参数都可用于无感控制。由于之前厂家提供数据存在错误，后由厂家从新测试后提供了新的参数。

| 教程22章之前使用下组参数   |                                                              |          |
|-----------------|--------------------------------------------------------------|----------|
| 电机参数            | 参数值                                                          | 参数描述     |
| Lq、Ld(H)        | 0.00535(+ -20%)                                              | q轴和d轴的电感 |
| R (ohm)         | 6.97(+ -10%)                                                 | 定子绕组的电阻  |
| Pn              | 4                                                            | 电机极对数    |
| Ke(Vpk LL/krpm) | 12.25(+ -10%)                                                | 反电动势常数   |
| Ψpm (Wb)        | 0.016884(+ -10%)                                             | 永磁体磁链    |
| 磁链与反电动势         | $\Psi_{pm} = 1 * K_e * 60 / (\sqrt{3}) * 1000 * P * 2 * \pi$ |          |
| 最高转速(rpm)       | 1200                                                         | 最高转速     |

| 教程22章之后使用改组参数    |                                                             |          |
|------------------|-------------------------------------------------------------|----------|
| 电机参数             | 参数值                                                         | 参数描述     |
| Lq、Ld(H)         | 0.00535(+/-20%)                                             | q轴和d轴的电感 |
| R (ohm)          | 6.97(+/-10%)                                                | 定子绕组的电阻  |
| Pn               | 2                                                           | 电机极对数    |
| Ke(Vpk LL/krpm)  | 6.13(+/-10%)                                                | 反电动势常数   |
| $\Psi_{pm}$ (Wb) | 0.016884(+/-10%)                                            | 永磁体磁链    |
| 磁链与反电动势          | $\Psi_{pm} = 1 * Ke * 60 / (\sqrt{3} * 1000 * P * 2 * \pi)$ |          |
| 最高转速(rpm)        | 2400                                                        | 最高转速     |

## 第二十三章：非线性磁链观测器

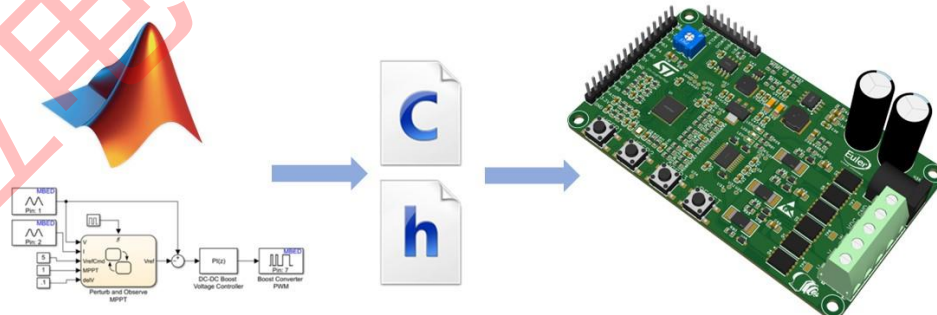
### 1. 非线性磁链观测器理论引入

# STM32G4 FOC

## 非线性磁链观测器

正反转切换，0速速度环闭环

1



## 主题概要

Part 1 非线性磁链观测器效果演示

Part 2 非线性磁链观测器数学建模

Part 3 PLL



## 非线性磁链观测器数学建模

参考文献：

Sensorless Control of Surface-Mount  
Permanent-Magnet Synchronous Motors  
Based on a Nonlinear Observer

非线性磁链观测器顾名思义观测器的状态变量为磁链值，观测的磁链值收敛于电机实际磁链值，观测器收敛。非线性是由于观测器存在sin和cos项，所以是非线性观测器。  
表贴式永磁同步电机 $\alpha\beta$ 轴电压方程：

$$\begin{bmatrix} u_{\alpha} \\ u_{\beta} \end{bmatrix} = \begin{bmatrix} R_s + \frac{d}{dt}L & 0 \\ 0 & R_s + \frac{d}{dt}L \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} + [\omega\varphi_m] \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$



## 非线性磁链观测器数学建模

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} R_s + \frac{d}{dt}L & 0 \\ 0 & R_s + \frac{d}{dt}L \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + [\omega \varphi_m] \begin{bmatrix} -\sin \theta \\ \cos \theta \end{bmatrix}$$

将公式变换：

$$L \begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} = -R_s \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \omega \varphi_m \begin{bmatrix} \sin \theta \\ -\cos \theta \end{bmatrix} + \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix}$$

定义状态变量X:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = L \begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} - \omega \varphi_m \begin{bmatrix} \sin \theta \\ -\cos \theta \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\sin \omega t' = \omega \cos \omega t$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = L \begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} + \varphi_m \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}$$

状态空间方程：

$$\begin{aligned} \dot{\hat{X}} &= A\hat{X} + Bu \\ Z &= CX \end{aligned}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -R_s \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix}$$

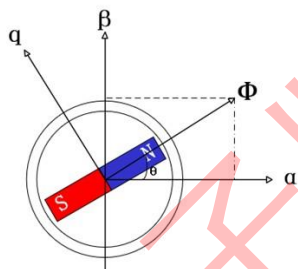
其中y不包含任何未知量，可作为一个已知量；

定义空间向量eta(x):

$$\text{eta}(x) = \begin{bmatrix} \text{eta}(x_1) \\ \text{eta}(x_2) \end{bmatrix} = x - L \begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} = \varphi_m \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}$$

## 非线性磁链观测器数学建模

欧式范数也称之为欧式距离，||x||表示向量的长度，计算方法依然是向量各个元素模的平方之和再开方；



$$\|\varphi_m\|^2 = |\varphi_m \cos \theta|^2 + |\varphi_m \sin \theta|^2$$

$$\|\text{eta}(x)\|^2 = |\text{eta}(x_1)|^2 + |\text{eta}(x_2)|^2$$

观测器模型：

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y}) \\ \hat{y} = C\hat{x} \end{cases}$$

$$\text{eta}(x) = \begin{bmatrix} \text{eta}(x_1) \\ \text{eta}(x_2) \end{bmatrix} = x - L \begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} = \varphi_m \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}$$

非线性磁链观测器模型：

$$\begin{bmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \frac{\gamma}{2} \begin{bmatrix} \text{eta}(x_1) \\ \text{eta}(x_2) \end{bmatrix} \{\varphi_m^2 - \|\text{eta}(x)\|^2\}$$

收敛：

$$\|\text{eta}(x)\|^2 = \varphi_m^2$$

## 非线性磁链观测器数学建模

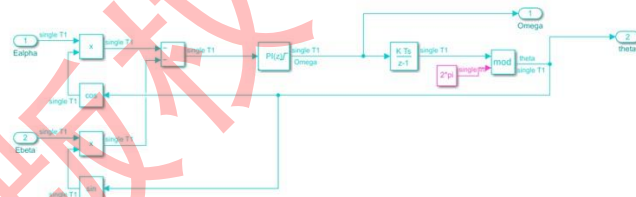
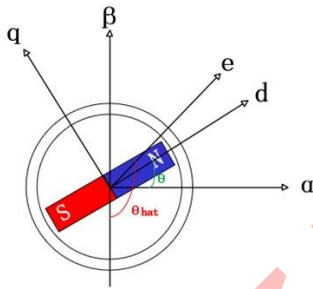
$$\begin{bmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \frac{\gamma}{2} \begin{bmatrix} \text{eta}(x_1) \\ \text{eta}(x_2) \end{bmatrix} \{ \varphi_m^2 - \|\text{eta}(x)\|^2 \}$$

离散化:

$$\dot{\hat{x}} = \frac{\hat{x}_{(k)} - \hat{x}_{(k-1)}}{T_s}$$

$$\begin{aligned} \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \end{bmatrix} &= T_s \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} + \frac{\gamma}{2} \begin{bmatrix} \text{eta}x_1(k-1) \\ \text{eta}x_2(k-1) \end{bmatrix} \{ \varphi_m^2 - \|\text{eta}(x)\|^2 \} + \begin{bmatrix} \hat{x}_1(k-1) \\ \hat{x}_2(k-1) \end{bmatrix} \\ \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} &= \frac{1}{\varphi_m} \left( \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \end{bmatrix} - L \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \right) \\ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} &= -R_s \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} \quad \begin{bmatrix} \text{eta}(x_1) \\ \text{eta}(x_2) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - L \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \end{aligned}$$

## 基于反电动势PLL

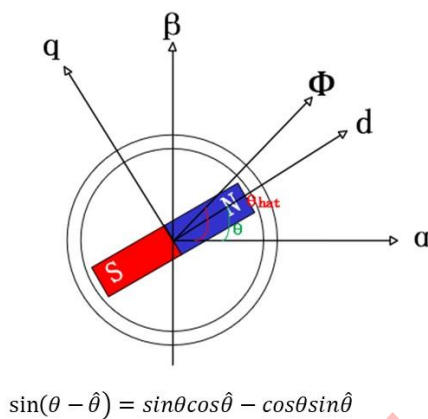


基于反电动势观测器系统观测的是反电动势电压，而反电动势电压又是基于ADC采样到的三相电流；由于电压经过电感相位相对电流超前90度，故观测得到的角度需要将90度补偿回来：

$$\hat{\theta} = \hat{\theta} + \frac{\pi}{2}$$

$$\begin{aligned} \sin(\theta - \hat{\theta}) &= \sin\theta\cos\hat{\theta} - \cos\theta\sin\hat{\theta} = \sin\theta\cos(\hat{\theta} + \frac{\pi}{2}) - \cos\theta\sin(\hat{\theta} + \frac{\pi}{2}) \\ &= -\sin\theta\sin\hat{\theta} - \cos\theta\cos\hat{\theta} \end{aligned}$$

## 磁链观测器PLL

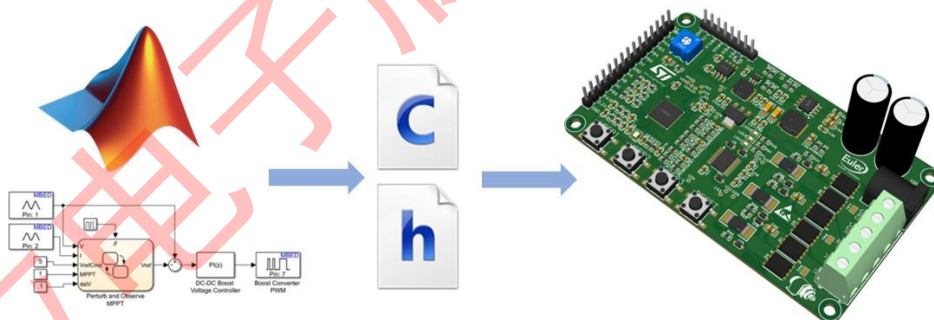


# STM32G4 FOC

## 非线性磁链观测器

### Simulink建模及代码生成

2





## 主题概要

- Part 1 非线性磁链观测器Simulink建模及代码生成
- Part 2 程序架构及HALL角度处理错误更正
- Part 3 Simulink代码及STM32G4底层代码结合

| NVIC Interrupt Table                                                  | Enabled                             | Preemption Priority |
|-----------------------------------------------------------------------|-------------------------------------|---------------------|
| Non maskable interrupt                                                | <input checked="" type="checkbox"/> | 0                   |
| Hard fault interrupt                                                  | <input checked="" type="checkbox"/> | 0                   |
| Memory management fault                                               | <input checked="" type="checkbox"/> | 0                   |
| Prefetch fault, memory access fault                                   | <input checked="" type="checkbox"/> | 0                   |
| Undefined instruction or illegal state                                | <input checked="" type="checkbox"/> | 0                   |
| System service call via SWI instruction                               | <input checked="" type="checkbox"/> | 0                   |
| Debug monitor                                                         | <input checked="" type="checkbox"/> | 0                   |
| Pendable request for system service                                   | <input checked="" type="checkbox"/> | 0                   |
| Time base: System tick timer                                          | <input checked="" type="checkbox"/> | 15                  |
| PVD/PVM1/PVM2/PVM3/PVM4 interrupts through EXTI lines 16/38/...       | <input type="checkbox"/>            | 0                   |
| Flash global interrupt                                                | <input type="checkbox"/>            | 0                   |
| RCC global interrupt                                                  | <input type="checkbox"/>            | 0                   |
| DMA1 channel1 global interrupt                                        | <input checked="" type="checkbox"/> | 0                   |
| DMA1 channel2 global interrupt                                        | <input checked="" type="checkbox"/> | 0                   |
| ADC1 and ADC2 global interrupt                                        | <input checked="" type="checkbox"/> | 0                   |
| TIM1 break interrupt and TIM15 global interrupt                       | <input checked="" type="checkbox"/> | 1                   |
| TIM1 update interrupt and TIM16 global interrupt                      | <input checked="" type="checkbox"/> | 1                   |
| TIM1 trigger and commutation interrupts and TIM17 global interrupt    | <input type="checkbox"/>            | 0                   |
| TIM1 capture compare interrupt                                        | <input checked="" type="checkbox"/> | 1                   |
| TIM2 global interrupt                                                 | <input checked="" type="checkbox"/> | 3                   |
| TIM4 global interrupt                                                 | <input checked="" type="checkbox"/> | 0                   |
| USART3 global interrupt / USART3 wake-up interrupt through EXTI li... | <input checked="" type="checkbox"/> | 2                   |
| EXTI line[15:10] interrupts                                           | <input checked="" type="checkbox"/> | 4                   |
| FPU global interrupt                                                  | <input type="checkbox"/>            | 0                   |

### 高频任务：

ADC ISR: 10K

读取三相电流

速度环：1K

电流环：1K

更新占空比

END ISR

### 中频任务：

TIM2 ISR: 2K

GUI数据上传

END ISR

TIM4 ISR

HALL角度获取

END ISR

### 低频任务：

MianLoop:1K

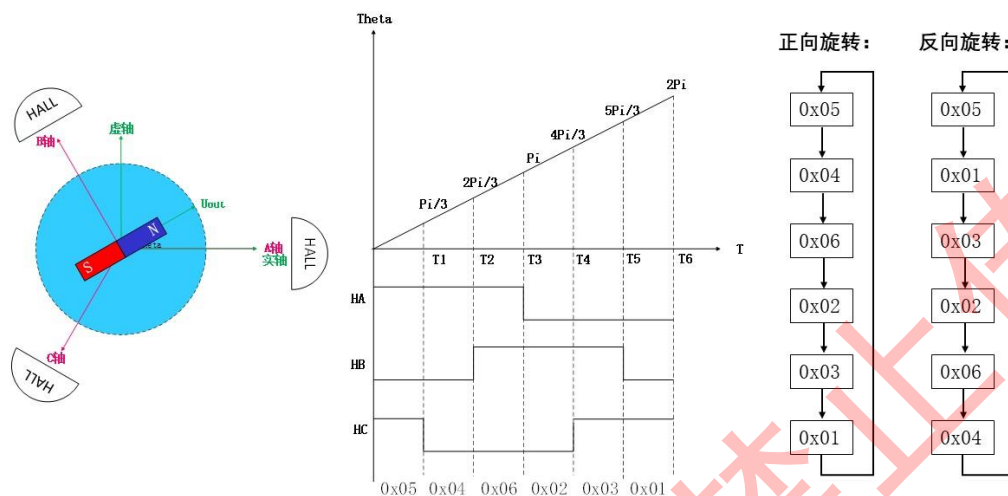
VBUS,电位器电压读取

GUI接收数据处理

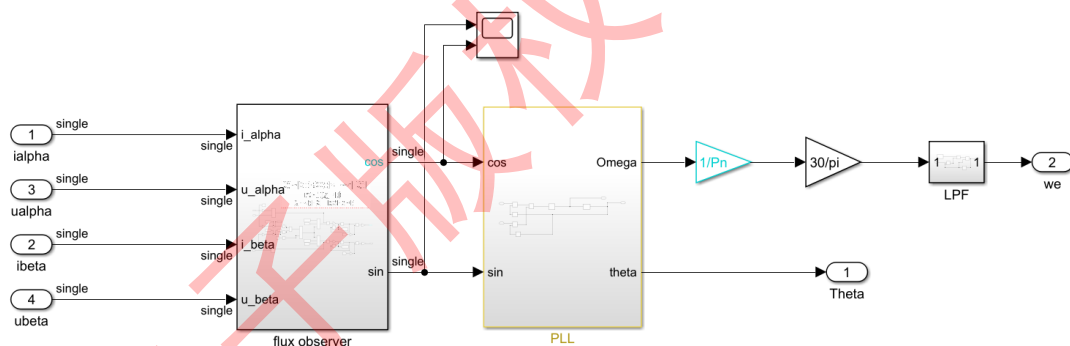
END

## 程序结构

## HALL角度处理更新



## 非线性磁链观测器内部截图：



$$\begin{bmatrix} \widehat{x1}_{(k)} \\ \widehat{x2}_{(k)} \end{bmatrix} = Ts \left[ \begin{bmatrix} y1_{(k-1)} \\ y2_{(k-1)} \end{bmatrix} + \frac{\gamma}{2} \begin{bmatrix} etax1_{(k-1)} \\ etax2_{(k-1)} \end{bmatrix} \{ \varphi_m^2 - \|eta(x)\|^2 \} \right] + \begin{bmatrix} \widehat{x1}_{(k-1)} \\ \widehat{x2}_{(k-1)} \end{bmatrix}$$

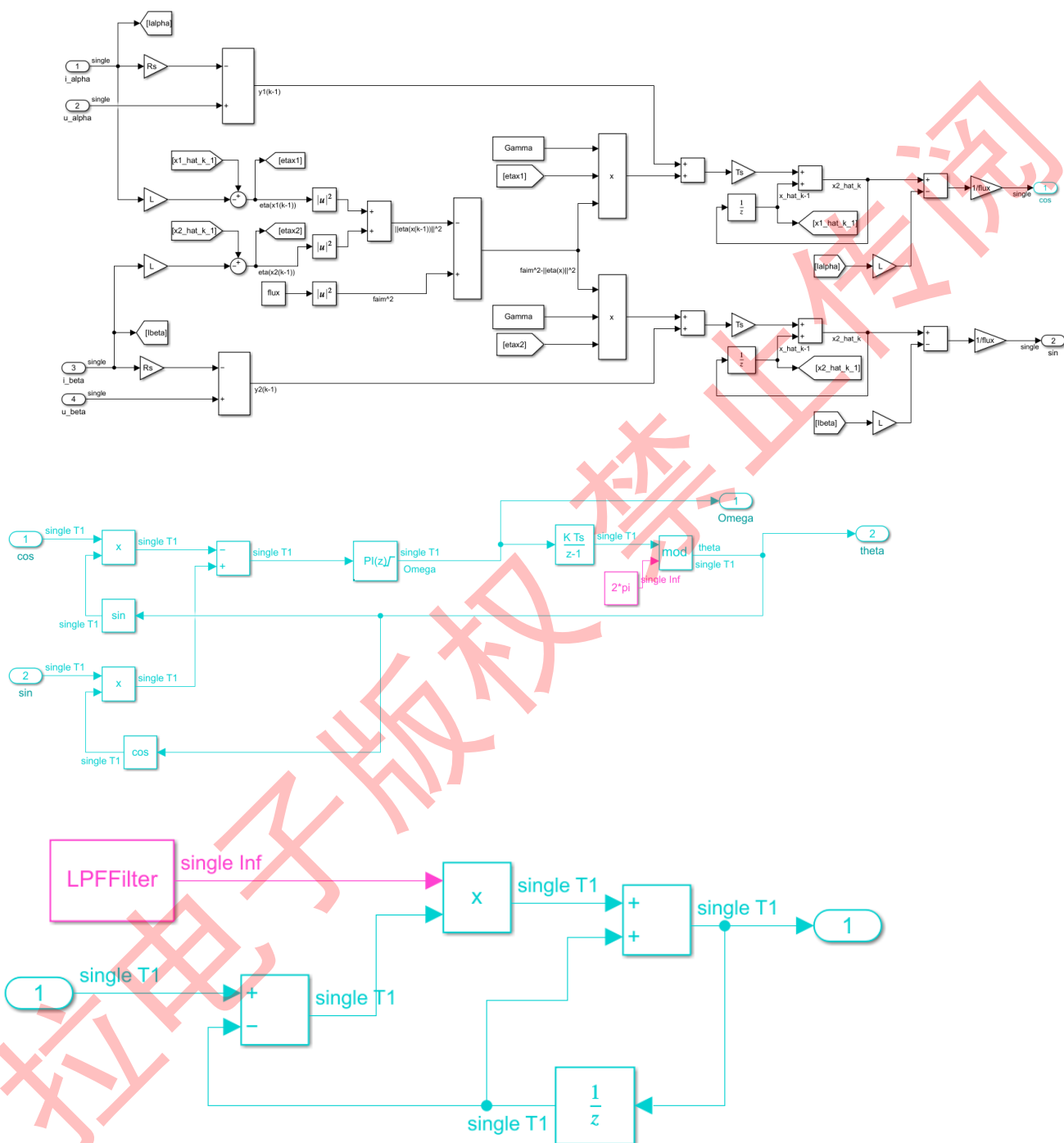
$$\begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} = \frac{1}{\varphi_m} \left( \begin{bmatrix} \widehat{x1}_{(k)} \\ \widehat{x2}_{(k)} \end{bmatrix} - L \begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} \right)$$

$$\begin{bmatrix} y1 \\ y2 \end{bmatrix} = -R_s \begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} + \begin{bmatrix} u_{\alpha} \\ u_{\beta} \end{bmatrix} \quad \begin{bmatrix} eta(x1) \\ eta(x2) \end{bmatrix} = \begin{bmatrix} x1 \\ x2 \end{bmatrix} - L \begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix}$$

## 2. 非线性磁链观测器模块内部及数据字典截图

观测器外部模块直接使用 Lun 或者 SMO 模型即可。



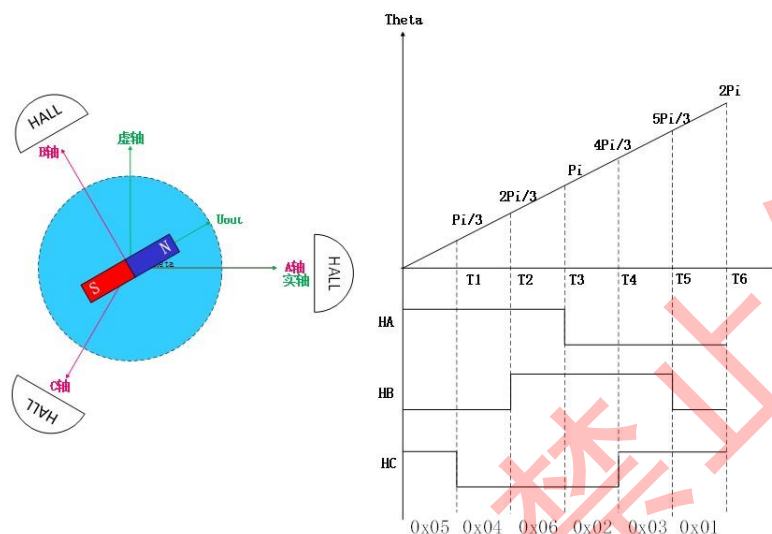


|                                                                                     | Name      | Status | Value    | DataType | Dimensions | Complexity | Min | Max | Unit | Argument | StorageClass   | DataSource          | La |
|-------------------------------------------------------------------------------------|-----------|--------|----------|----------|------------|------------|-----|-----|------|----------|----------------|---------------------|----|
|    | Tpwm      |        | 16000    | uint16   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_flux.sldd 2023 |    |
|    | ialpha    |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|    | ibeta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|    | ualpha    |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|    | ubeta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|    | Pn        |        | 2        | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|    | Ts        |        | 0.0001   | single   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_flux.sldd 2023 |    |
|    | Udc       |        | 24       | single   | [1 1]      | real       | []  | []  |      |          | Auto           | BLDC_flux.sldd 2023 |    |
|    | state     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|    | theta     |        |          | single   | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|    | L         |        | 0.00535  | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|    | Rs        |        | 6.97     | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|    | flux      |        | 0.008442 | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|    | curr_d_ki |        | 35       | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|    | curr_d_kp |        | 0.026    | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|    | spd_kp    |        | 0.03389  | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|    | spd_ki    |        | 0.00144  | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|    | id        |        |          | auto     | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|  | iq        |        |          | auto     | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|  | Gamma     |        | 100000   | auto     | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|  | iqref     |        |          | auto     | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|  | err       |        |          | auto     | -1         | real       | []  | []  |      |          | ExportedGlobal | BLDC_flux.sldd 2023 |    |
|  | PLLkP     |        | 1131.2   | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|  | PLLKi     |        | 640000   | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |
|  | LPFFilter |        | 0.3      | single   | [1 1]      | real       | []  | []  |      |          | Struct         | BLDC_flux.sldd 2023 |    |

## 第二十四章：HALL 传感器 0 速闭环正反转

无刷电机的转子上通常有若干个磁极，而霍尔传感器能够检测这些磁极的位置。在电机的定子上有霍尔传感器，它们与转子的磁极相对应。信号输出：每个霍尔传感器输出一个二进制信号（0 或 1），通常 N 为 1，S 为 0。表示其对应位置的磁极是否存在。

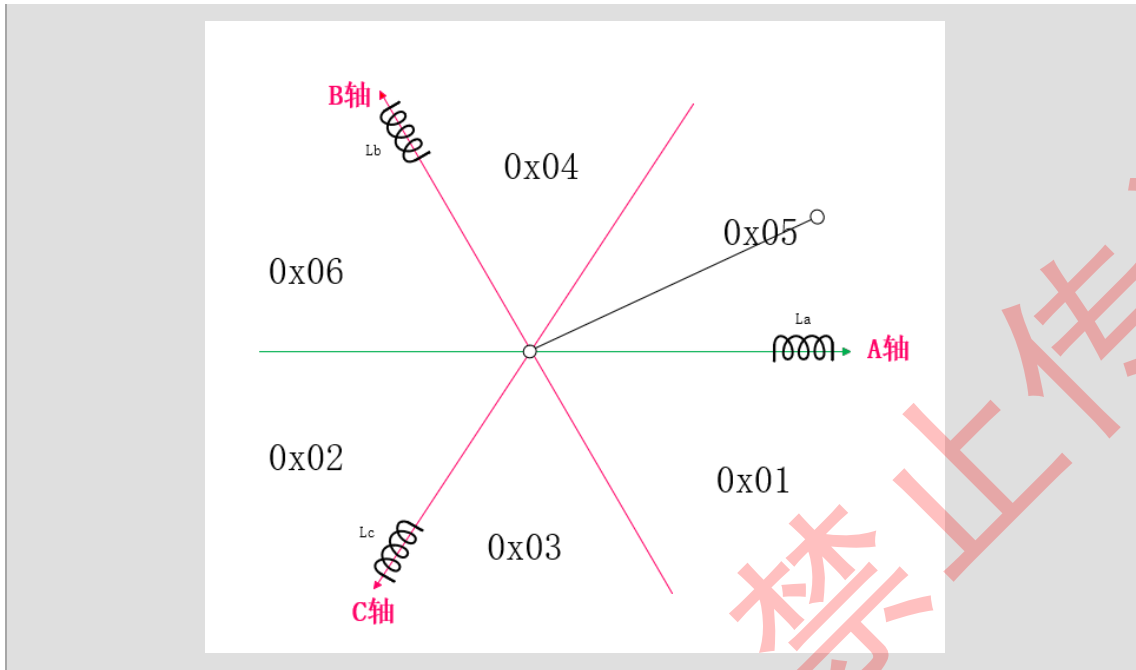
无刷电机通常都会安装 3 个 hall 元器件，分为 60 度安装和 120 度安装；我们所采用的为 120 度安装。



## 1. HALL 传感器 0 速角度判断

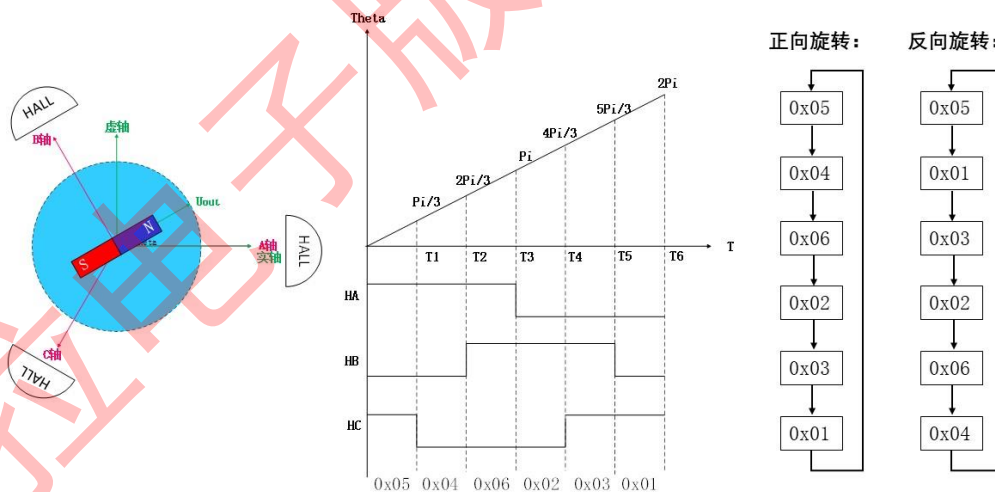
在电机未启动前，可以通过获取当前 HALL 状态，及转子所在扇区，我们将电机初始角度定义为该扇区的中点，即在当前扇区的角度上增加  $\pi/6$ 。

```
void HALL_Init_Electrical_Angle(void)
{
    HALL_Handle_t *phandle = &Motor_App.HALL_Handle;
    //读取 HALL 传感器 HA/HB/HC 三个 IO 捕获状态
    phandle->HallState = HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_8);
    phandle->HallState |= HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_7)<<1;
    phandle->HallState |= HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_6)<<2;
    switch (phandle->HallState)
    {
        case STATE_5:
        {
            //获取当前所在扇区，在当前扇区的角度上增加  $\pi/6$ 
            phandle->HallElAngle = PHASE_SHIFT_ANGLE +  $\pi/6$ ;
            break;
        }
    }
}
```



## 2. HALL 传感器正反转角度计算

### HALL 角度处理更新



电机正向旋转时，角度由 0 度增长为  $2\pi$ ，获取的三个 hall 状态从 0x05 至 0x01；反正则相关。

当电机正向旋转时，HALL 状态由 0x05-→0x04 时，由 HC 的下降沿触发进入中断，在中断中获取的 HALL 状态为 0x04，此时我们认为电机 hall 角度为  $\pi/3$ ；

而当电机反转时，当 HALL 状态由 0x06->0x04 时，由 HB 的上降沿触发进入中断，在中断中获取的 HALL 状态为 0x04，此时我们认为电机 hall 角度为  $2\pi/3$ ；同样读取 HALL 状态为 0x04，在反转时则需要增加  $\pi/3$  补偿。以此大家可以类推各个状态切换来验证这个结论。

```
void HALL_Get_Electrical_Angle(void *pHandleVoid)
{
    //函数传入 HALL 传感器的参数结构体指针
    HALL_Handle_t *phandle = (HALL_Handle_t *)pHandleVoid;
    //获取电机转子转动上一个 PI/3 所用时间 (TIM4 的计数值)
    phandle->hHighSpeedCapture = HAL_TIM_ReadCapturedValue(&htim4, TIM_CHANNEL_1);
    //记录上一次 HALL 状态，用于判断电机正反转
    phandle->bPrevHallState = phandle->HallState;
    //获取当前 HALL 状态
    phandle->HallState = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_8);
    phandle->HallState |= HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_7) << 1;
    phandle->HallState |= HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_6) << 2;
    switch (phandle->HallState)
    {
    case STATE_5:
    {
        //根据上图的 HALL 正反转所对应的 HALL 状态，来判断电机正反转
        if (STATE_1 == phandle->bPrevHallState)
        {
            phandle->MeasuredElAngle = PHASE_SHIFT_ANGLE;
            phandle->Direction = POSITIVE;
        }
        else if (STATE_4 == phandle->bPrevHallState)
        {
            phandle->MeasuredElAngle = PHASE_SHIFT_ANGLE + PI / 3.0f;
            phandle->Direction = NEGATIVE;
        }
        else
        {
            // nothing
        }
        break;
    }
    ...
    ...
    if (phandle->MeasuredElAngle < 0.0f)
```

```

{
    phandle->MeasuredElAngle += 2.0f * PI;
}
else if (phandle->MeasuredElAngle > (2.0f * PI))
{
    phandle->MeasuredElAngle -= 2.0f * PI;
}

// AvrElSpeedDpp 为平均电角速度，在电流环（10KHz）中累加就得到角度
phandle->AvrElSpeedDpp = (PI/3)/((phandle->hHighSpeedCapture/3200000)*10000);
// HallSpeed 为上一个 PI/3 的平均速度，3200000=160M/50，50 为 tim4 分频系数；30/PI 为转速单
// 位由 rad/s 转为 rpm/min；2 为极对数
phandle->HallSpeed = (PI/3)/((phandle->hHighSpeedCapture/3200000)*30/(2*PI));
//根据电机转动方向，判断转速及平均电角速度的正负
phandle->HallSpeed = phandle->HallSpeed * phandle->Direction;
phandle->AvrElSpeedDpp = phandle->AvrElSpeedDpp * phandle->Direction;
//HALL 处理是基于之前 60° 的 hall 时间处理的，不是当前的值，所以这么处理的前提是默认每个扇
// 区的运行时间是不突变的，hall 处理总归会存在误差，因此需要进行补偿
// 补偿方法：HALL 测量到角度减去电流环中累加得到角度，除以 10000 后在电流环中进行补偿
phandle->DeltaAngle = (phandle->MeasuredElAngle - phandle->HallElAngle) / 10000;
}

```

在 tim.c 文件中可以修改 tim4 分频系数：

```

void MX_TIM4_Init(void)
{
    /* USER CODE BEGIN TIM4_Init 0 */

    /* USER CODE END TIM4_Init 0 */

    TIM_ClockConfigTypeDef sClockSourceConfig = {0};
    TIM_HallSensor_InitTypeDef sConfig = {0};
    TIM_MasterConfigTypeDef sMasterConfig = {0};

    /* USER CODE BEGIN TIM4_Init 1 */

    /* USER CODE END TIM4_Init 1 */
    htim4.Instance = TIM4;
    //TIM4 分频系数
    htim4.Init.Prescaler = 50-1;
    htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim4.Init.Period = 65535;
    htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
}

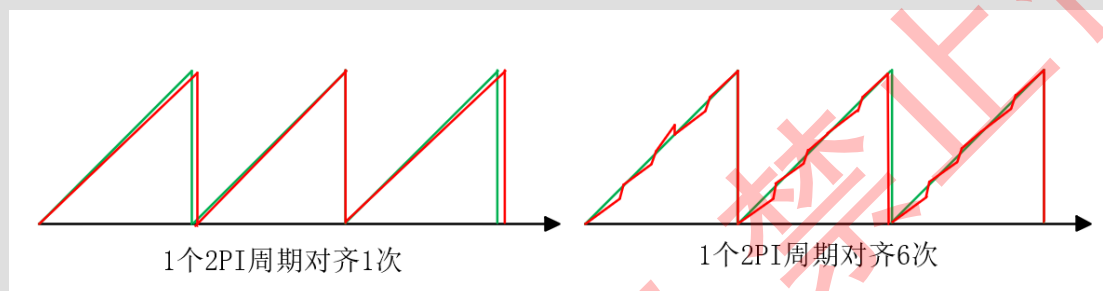
```

在 10kHz 电流环的任务中，对 tim4 中断获取的平均电角速度累加就可以得到实时电角度；

```
void task_isr_10kHz(void)
```

```
{
...
...
}
```

//当 HALL 测量的角度等于偏置角度时，需要对累加得到的电角度做一次校准，即每转一圈（电角度）校准一次



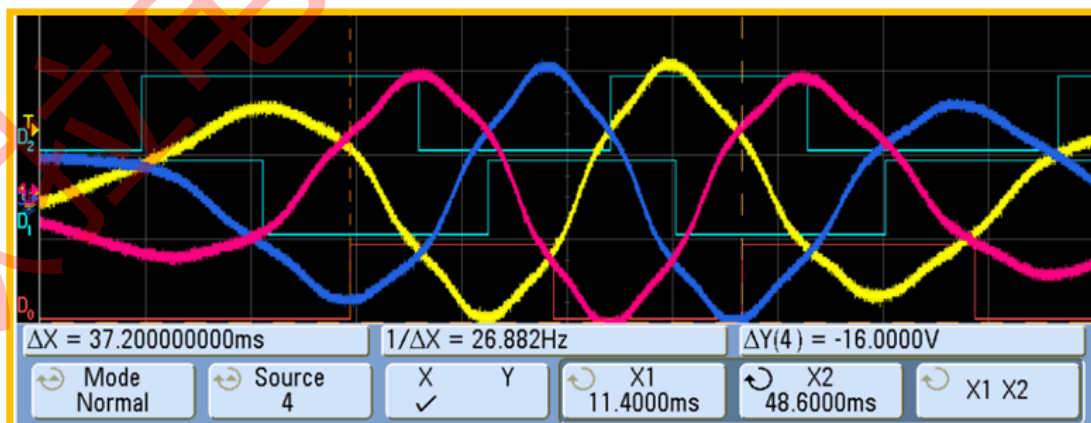
```
if (Motor_App.HALL_Handle.MeasuredElAngle == PHASE_SHIFT_ANGLE
&&Motor_App.HALL_Handle.Direction == 1)
{
    //正转校准，就是让 hall 获取的角度等于实际的电角度
    Motor_App.HALL_Handle.HallElAngle = Motor_App.HALL_Handle.MeasuredElAngle;
    Motor_App.HALL_Handle.MeasuredElAngle = Motor_App.HALL_Handle.MeasuredElAngle +
    Motor_App.HALL_Handle.AvrElSpeedDpp;
    Motor_App.HALL_Handle.HallElAngle = Motor_App.HALL_Handle.HallElAngle +
    Motor_App.HALL_Handle.AvrElSpeedDpp;
}
else if (Motor_App.HALL_Handle.MeasuredElAngle == PHASE_SHIFT_ANGLE + PI / 3 &&
Motor_App.HALL_Handle.Direction == -1)
{
    //反转校准
    Motor_App.HALL_Handle.HallElAngle = Motor_App.HALL_Handle.MeasuredElAngle;
    Motor_App.HALL_Handle.MeasuredElAngle = Motor_App.HALL_Handle.MeasuredElAngle +
    Motor_App.HALL_Handle.AvrElSpeedDpp;
    Motor_App.HALL_Handle.HallElAngle = Motor_App.HALL_Handle.HallElAngle +
    Motor_App.HALL_Handle.AvrElSpeedDpp;
}
else
{
    //非校准阶段，角度累加
    Motor_App.HALL_Handle.MeasuredElAngle = Motor_App.HALL_Handle.MeasuredElAngle +
    Motor_App.HALL_Handle.AvrElSpeedDpp;
    Motor_App.HALL_Handle.HallElAngle = Motor_App.HALL_Handle.HallElAngle +
```

```
Motor_App.HALL_Handle.AvrElSpeedDpp + Motor_App.HALL_Handle.DeltaAngle;//补偿的角度
}
//对 HallElAngle 取余，范围[0, 2PI]
if (Motor_App.HALL_Handle.HallElAngle < 0.0f)
{
    Motor_App.HALL_Handle.HallElAngle += 2.0f * PI;
}
else if (Motor_App.HALL_Handle.HallElAngle > (2.0f * PI))
{
    Motor_App.HALL_Handle.HallElAngle -= 2.0f * PI;
}
...
rtU.theta = Motor_App.HALL_Handle.HallElAngle;
rtU.Speed = Motor_App.HALL_Handle.HallSpeed;
...
}
}
```

### 3. HALL 传感器偏置角度测量计算

测量反电动势时，示波器 GND 需要连接电机中心点，电机测量前如果没有虚拟中点的话，需要连接三个相同阻值电阻到电机的三相接线上，电阻另外一端连接到一起作为虚拟中点。

测试一个电周期的时间，这个周期对应 360 度；



测试电机 A 相反向电动势最高点到 H1 的时间



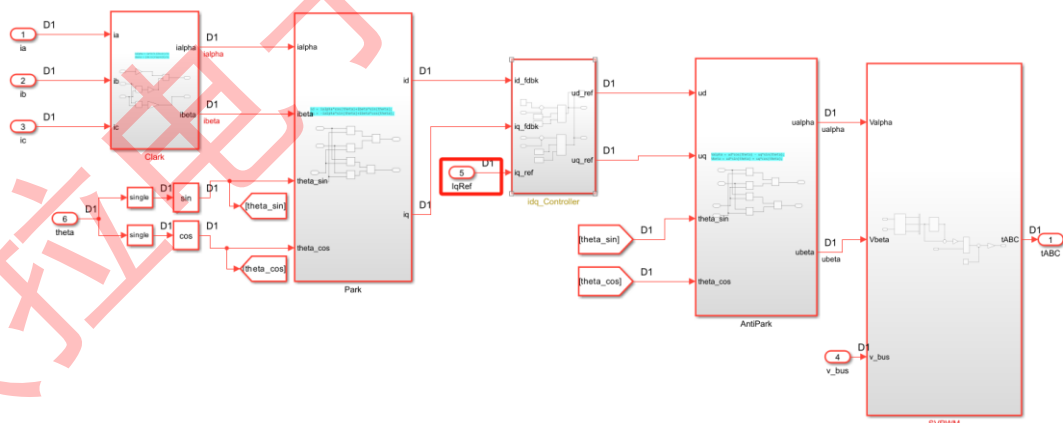


上图中粉色为电机 A 相反向电动势，红色数字端口 D0 为 H1 信号；

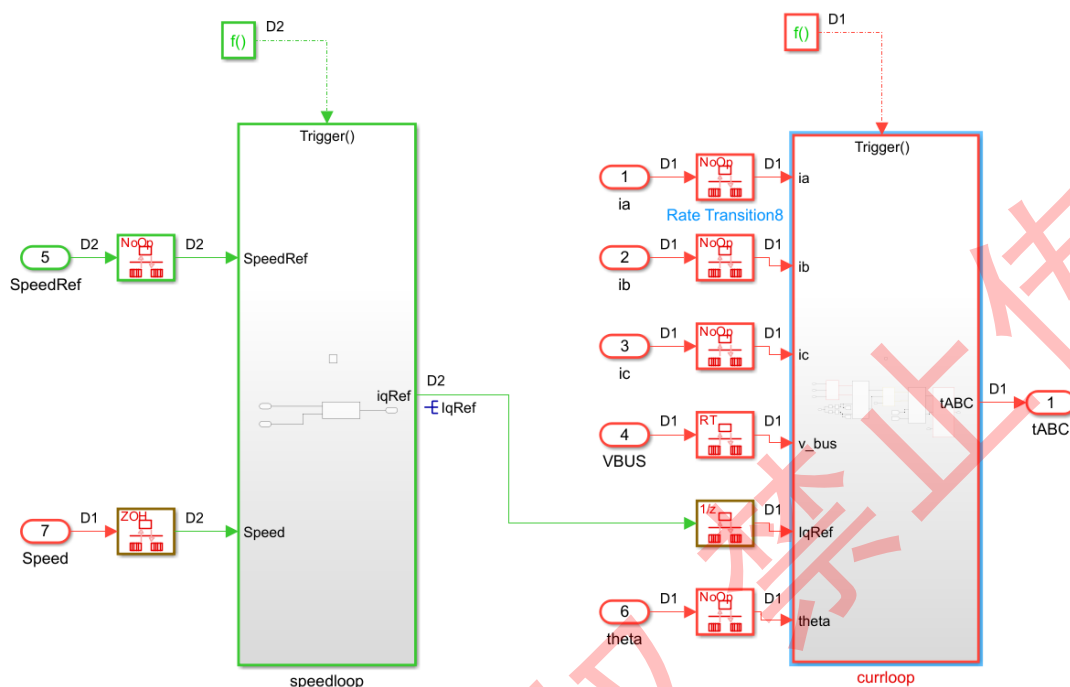
该电机同步电角度： $\theta = 37.2 - 6.6 / 37.2 * 360^\circ = 296^\circ$

#### 4. 速度环模型修改

本章我们基于 HALL 传感器实现速度环 0 速闭环，故我们基于第十七章的速度环模型，需要去掉开环强拖部分；将速度环的输出  $i_{q\_ref}$  直接给到电流环，无需强拖切换等环节。其他控制参数都无需修改。



电流环执行频率 10kHz，速度环执行频率 1kHz。



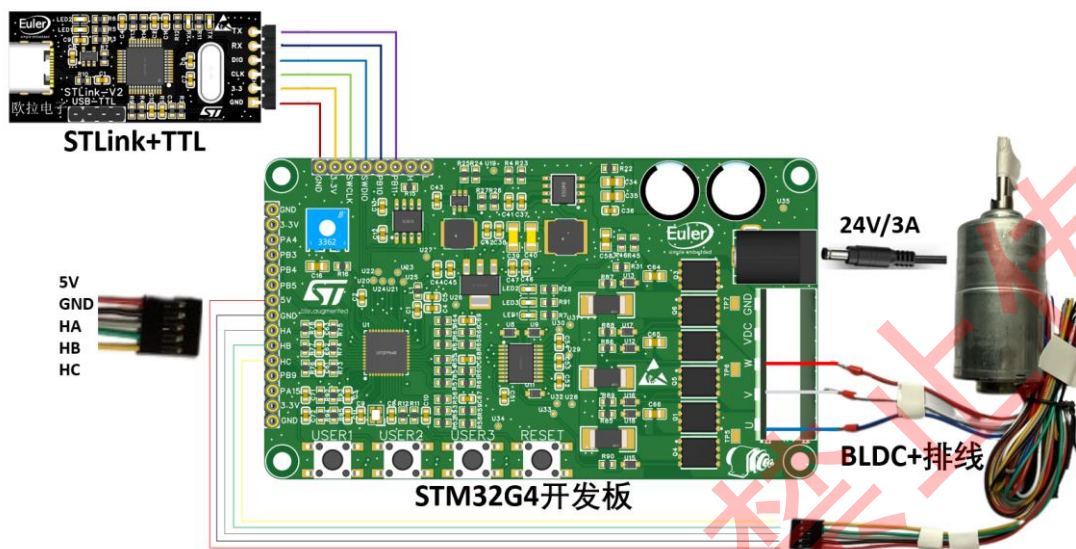
通过仿真查看模型速度输出是都能够完美跟随参考转速。



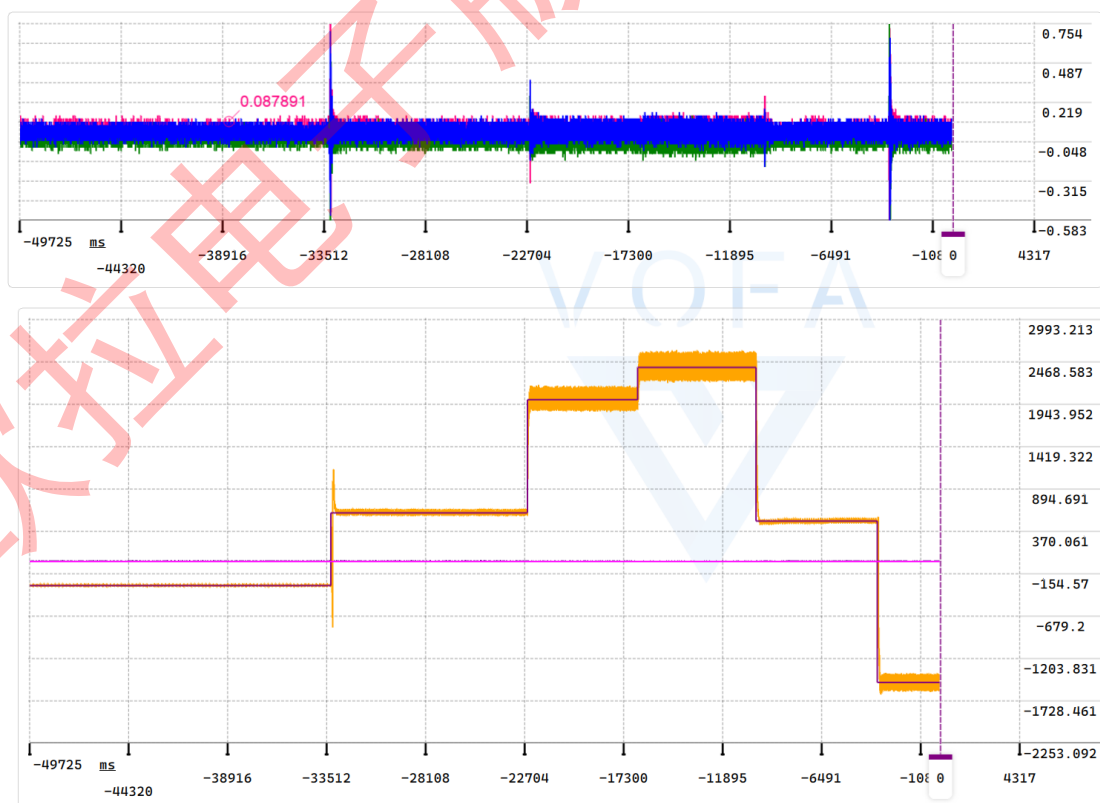
由上图可以发现，电机输出速度能够完美跟随参考转速，但超调量，响应时间及稳定时间依旧可以优化，大家可以根据实际的项目情况再做具体优化。

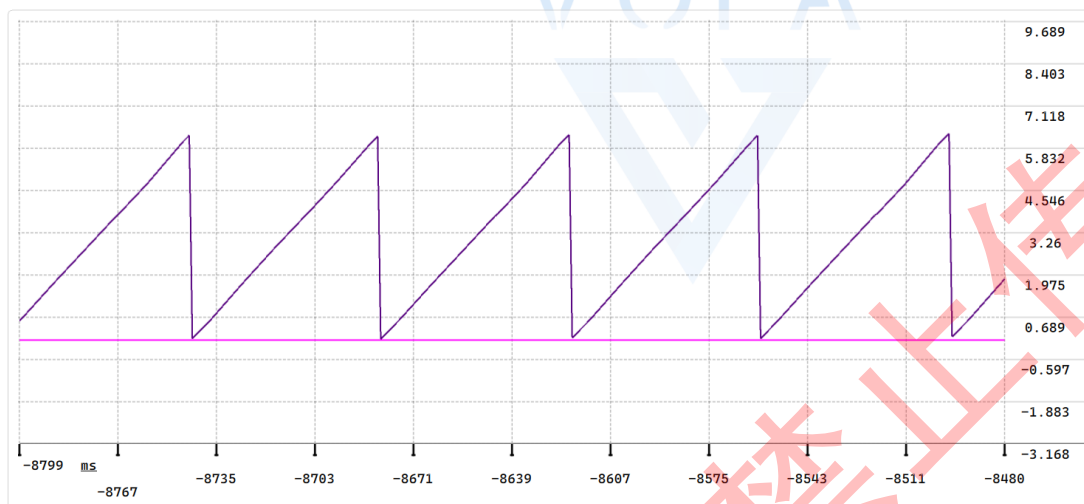
## 5. HALL 0 速闭环正反转实验现象

按以下图片接线，并将 USB 转 UART 模块连接至电脑：



下载代码，并打开 VOFA 上位机；可以看到电机在 0 速闭环时启动超调较大，这是由于 0 速时转子位置不是非常准确导致；电机可以实现无缝切换正反转，正反转切换时也会带来较大的电流抖动，同样是由于转速较低时候获取的角度不准确导致。这同样也会导致高速切换时切换失败的场景出现。





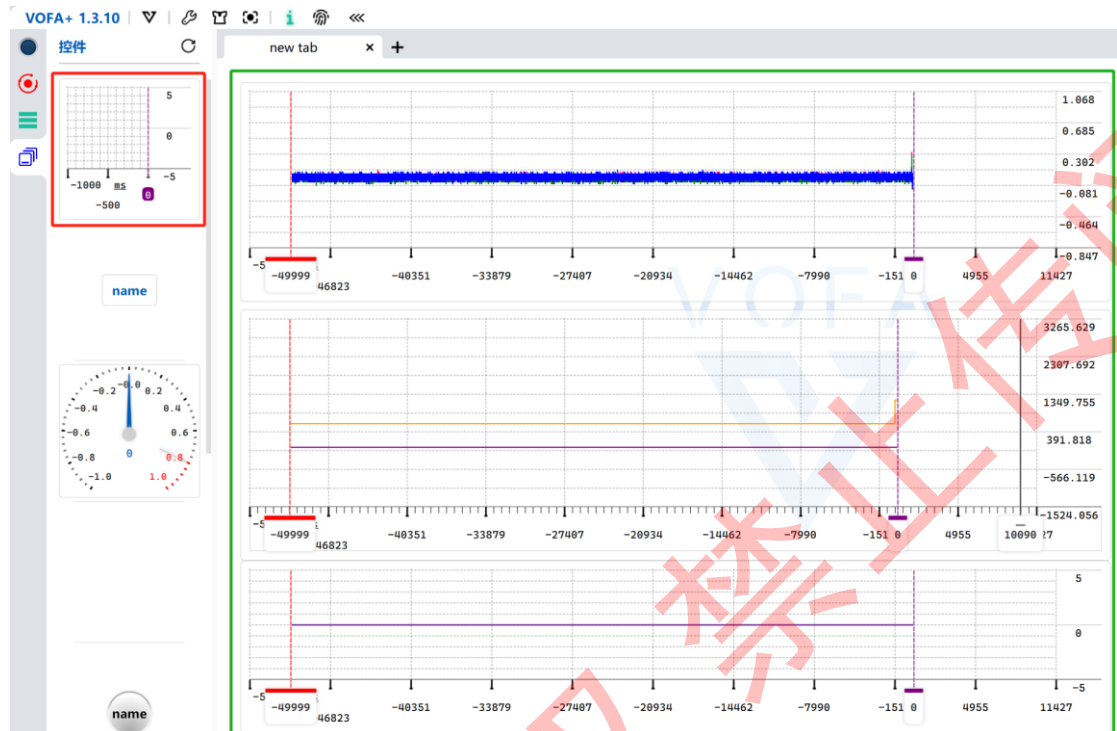
## 第二十五章：VOFA+ 控制参数下发

VOFA+是一款通用的 UART 上位机，在之前的课程里面我们使用 VOFA+显示电流，转速，电压等波形，主要是从 MCU 端向 VOFA+上位机端发送数据；本章为大家介绍如何使用 VOFA+上位机下发控制指令及参数下发值 MCU 端，实现 VOFA+实时控制电机运行状态。

### 1. 建立上位机控件及协议

VOFA+的基本使用教程大家可以参考附录文档--**附录三：vofa+上位机使用教程。**

首先，先建立 3 个波形图，分别用于显示三相电流，转速，电角度波形；当然此处大家也可以根据自己的实际需求进行建立；



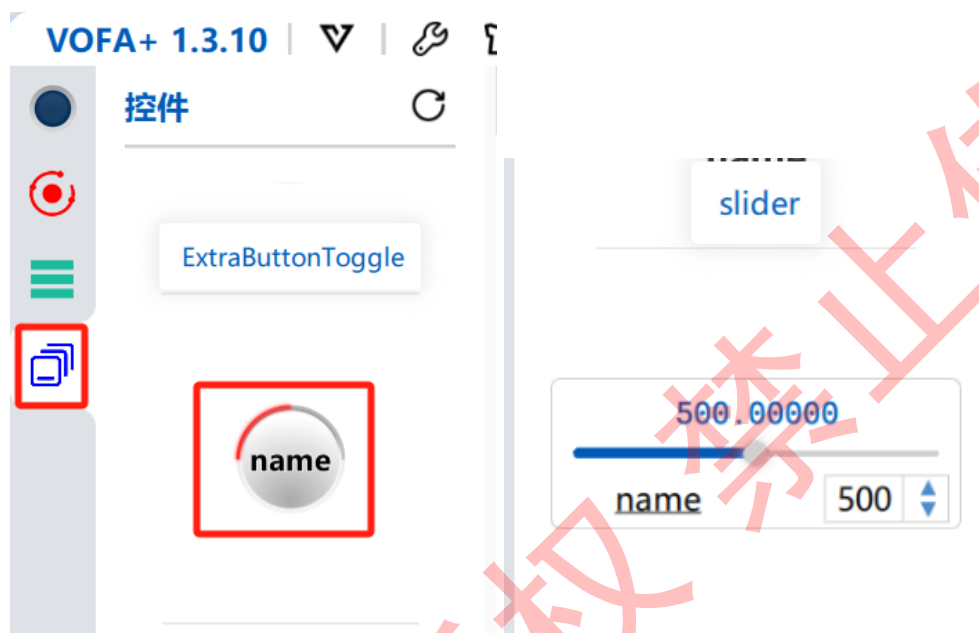
然后，需要在命令行建立相对应的控制命令，此处我们建立了 4 条控制指令，分别对应 OffOn->电机启动暂停；SpeedRef->参考转速；SpeedKP->速度环 Kp 参数；SpeedKI->速度环 Ki 参数；发送内容解析如下：

P 为帧头，P1 代表指令 1，以此类推；

支持转义符:\r,\n，命令内容中的%f 占位符(%f %.0f | %.2f | %.3f...)，小数点前面为整数占位，小数点后面为小数占位；



命令建立完成之后，需要拖动相对应的空间来绑定命令；在 VOFA+ 的控件栏分别选择 ExtraButtonToggle 和 slider 控件来分别绑定刚才设置好的命令；



拖拽 ExtraButtonToggle 控件，右键该控件点击绑定命令通道，绑定 OffOn 命令，该命令用于控制电机启动和暂停。





使用同样的流程和方法，拖拽 slider 控件来绑定其他几个命令，由于这几个命令是下发数字指令，故还需要设置数值范围和保留小数点个数：



此时，我们可以在电脑上接上 STlink ttl 模块，STlink 不需要接 G4 板子；通过操作控件，来检查我们配置的控件参数是否正确。



我们下发的参数必须满足控制所设置的参数，否则下发会报错，比如超过设置的范围和小数点，故在下发之前请检查您所输入的数据是否满足。

## 2. STM32G4 接收并解析协议

使用 UART3 用来接收 VOFA+ 上位机数据；

```
//定义 UART_RxBuff 结构体
typedef struct
{
    uint8_t uart_rx_buffer[UART_BUFFER_SIZE];
    uint8_t uart_rx_data;
    uint16_t uart_rx_index;
    uint8_t uart_rx_complete;
    uint8_t Data_Decoding_state;
}Motor_UART_RxBuff_t;

//初始化清除 UART3 空闲中断
__HAL_UART_CLEAR_FLAG(&huart3, UART_CLEAR_IDLEF);
//使能 UART3 DMA 空闲接收
HAL_UARTEx_ReceiveTxdle_DMA(&huart3, (uint8_t *)&Motor_App.Motor_UART_RxBuff, 20);

//UART 接收数据任务
void Motor_UART_Rx_Job(void)
{
    if (Motor_App.Motor_UART_RxBuff.uart_rx_data == 0x21)
    {
        Motor_App.Motor_UART_RxBuff.uart_rx_buffer[Motor_App.Motor_UART_RxBuff.uart_rx_index] = '\0'; //
        添加字符串结束符
        Motor_App.Motor_UART_RxBuff.uart_rx_complete = 1; // 标记接收完成
        Motor_App.Motor_UART_RxBuff.uart_rx_index = 0; // 重置索引
        Motor_App.Motor_UART_RxBuff.Data_Decoding_state = 1;
    }
    else
    {
        Motor_App.Motor_UART_RxBuff.uart_rx_buffer[Motor_App.Motor_UART_RxBuff.uart_rx_index++] =
        Motor_App.Motor_UART_RxBuff.uart_rx_data;
        if (Motor_App.Motor_UART_RxBuff.uart_rx_index >= UART_BUFFER_SIZE)
        {
            Motor_App.Motor_UART_RxBuff.uart_rx_index = 0; // 防止缓冲区溢出
        }
    }
}
```



```

    }
}

//寻找接收的字符串分隔符，分隔符“=”，并基于分隔符将字符串分成两段，=右侧为控制指令数据
uint8_t split_string(const char *str, char delimiter, char **left, char **right)
{
    *left = (char *)str;
    *right = NULL;

    for (; *str != '\0'; str++) {
        if (*str == delimiter) {
            *right = (char *)str + 1;
            // *((char *)str) = '\0'; // 将分隔符替换为字符串结束符
            return 0; // 成功找到分隔符
        }
    }

    return 1; // 没有找到分隔符
}

//将分割完成后的字符串转为 float 数据
float my_strtof(const char *str)
{
    float result = 0.0f;
    float sign = 1.0f;
    float fraction = 1.0f;
    int exponent = 0;
    int exp_sign = 1;
    int state = 0; // 0: integer part, 1: fractional part, 2: exponent part

    // 跳过前导空格
    while (*str == ' ' || *str == '\t' || *str == '\n' || *str == '\v' || *str == '\f' || *str == '\r') {
        str++;
    }

    // 处理符号
    if (*str == '-') {
        sign = -1.0f;
        str++;
    } else if (*str == '+') {
        str++;
    }
}

```

```
// 处理数字部分
while (*str != '\0') {
    if (*str >= '0' && *str <= '9') {
        if (state == 0) {
            result = result * 10.0f + (*str - '0');
        } else if (state == 1) {
            fraction /= 10.0f;
            result += (*str - '0') * fraction;
        } else if (state == 2) {
            exponent = exponent * 10 + (*str - '0');
        }
    } else if (*str == '.' && state == 0) {
        state = 1;
    } else if ((*str == 'e' || *str == 'E') && state < 2) {
        state = 2;
        str++;
        if (*str == '-') {
            exp_sign = -1;
            str++;
        } else if (*str == '+') {
            str++;
        }
        continue;
    } else {
        break;
    }
    str++;
}

// 计算最终结果
if (state == 2) {
    while (exponent > 0) {
        if (exp_sign > 0) {
            result *= 10.0f;
        } else {
            result /= 10.0f;
        }
        exponent--;
    }
}

return sign * result;
}
```

```
//对 UART 接收的数据做解耦处理
void Motor_UART_Rx_Data_Decoding(void)
{
    char *Frame_header;
    char *Frame_data;
    float strtotf;
    if(Motor_App.Motor_UART_RxBuff.Data_Decoding_state == 1)
    {
        //分割接收的字符串数据
        if(split_string((char
*)Motor_App.Motor_UART_RxBuff.uart_rx_buffer, '=', &Frame_header, &Frame_data) == 0x00)
        {
            //将接收的字符串数据转化为 float 类型
            strtotf = my_strtof(Frame_data);
            //判断接收的帧头对应的控制参数
            if(Motor_App.Motor_UART_RxBuff.uart_rx_buffer[0] == 'P' &&
Motor_App.Motor_UART_RxBuff.uart_rx_buffer[1] == 0x31)
            {
                if((uint8_t)strtotf == 0x01)
                {
                    Motor_App.Motor_UART_Control.Motor_UART_TurnOnOFF =
Motor_UART_Start;
                }
                else
                {
                    Motor_App.Motor_UART_Control.Motor_UART_TurnOnOFF =
Motor_UART_Stop;
                }
            }
            if(Motor_App.Motor_UART_RxBuff.uart_rx_buffer[0] == 'P' &&
Motor_App.Motor_UART_RxBuff.uart_rx_buffer[1] == 0x32)
            {
                Motor_App.Motor_UART_Control.Speed_ref = (int16_t)strtotf;
            }
            if(Motor_App.Motor_UART_RxBuff.uart_rx_buffer[0] == 'P' &&
Motor_App.Motor_UART_RxBuff.uart_rx_buffer[1] == 0x33)
            {
                Motor_App.Motor_UART_Control.Speed_KP = (float)strtotf;
            }
            if(Motor_App.Motor_UART_RxBuff.uart_rx_buffer[0] == 'P' &&
Motor_App.Motor_UART_RxBuff.uart_rx_buffer[1] == 0x34)
            {
                Motor_App.Motor_UART_Control.Speed_KI = (float)strtotf;
            }
        }
    }
}
```

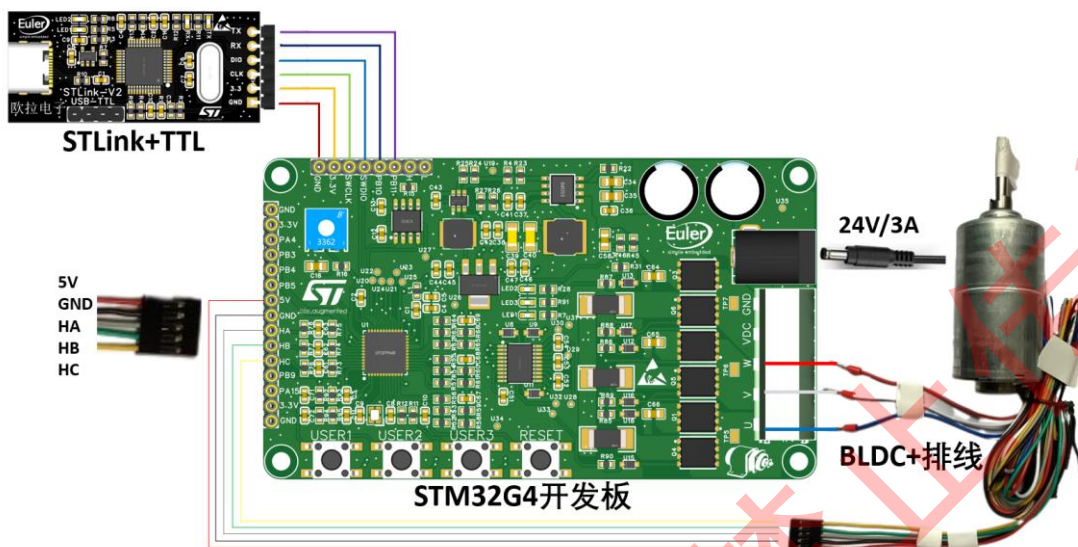
```

    }
    Motor_App.Motor_UART_RxBuff.Data_Decoding_state = 0;
    memset(Motor_App.Motor_UART_RxBuff.uart_rx_buffer, 0 , 256);
}
}
}
}
//在 1k 的任务中 UART 上传数据和解析接收的数据
void task_isr_1kHZ(void)
{
    static uint8_t motor_state = 0;
    Motor_Load_UART_Tx_Buff();
    Motor_UART_Rx_Data_Decoding();
    //在 1k 任务中实时更新上位机的下发的控制参数
    if(motor_state != Motor_App.Motor_UART_Control.Motor_UART_TurnOnOff)
    {
        if(Motor_App.Motor_UART_Control.Motor_UART_TurnOnOff == Motor_UART_Start)
        {
            motor_start();
        }
        else
        {
            motor_stop();
        }
    }
    motor_state = Motor_App.Motor_UART_Control.Motor_UART_TurnOnOff;
    rtU.SpeedRef = Motor_App.Motor_UART_Control.Speed_ref;
    spd_kpki.spd_kp = Motor_App.Motor_UART_Control.Speed_KP;
    spd_kpki.spd_ki = Motor_App.Motor_UART_Control.Speed_KI;
}

```

### 3. VOFA+上位机控制电机实验现象

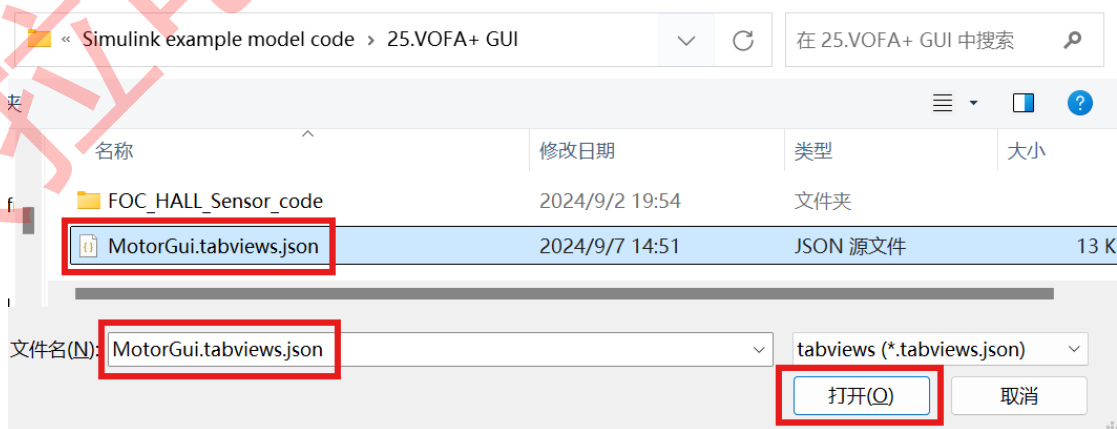
按以下图片接线，并讲 USB 转 UART 模块连接至电脑：

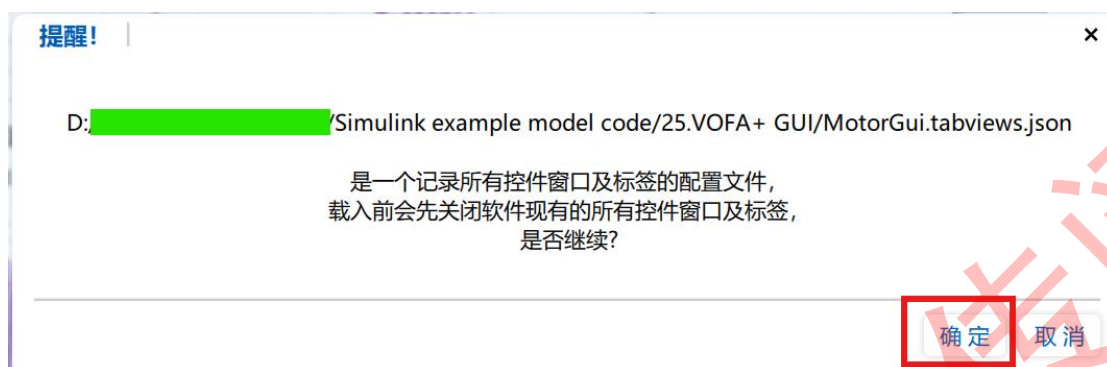


打开 VOFA+ 软件，载入我们已经配置好的控件窗口及标签；

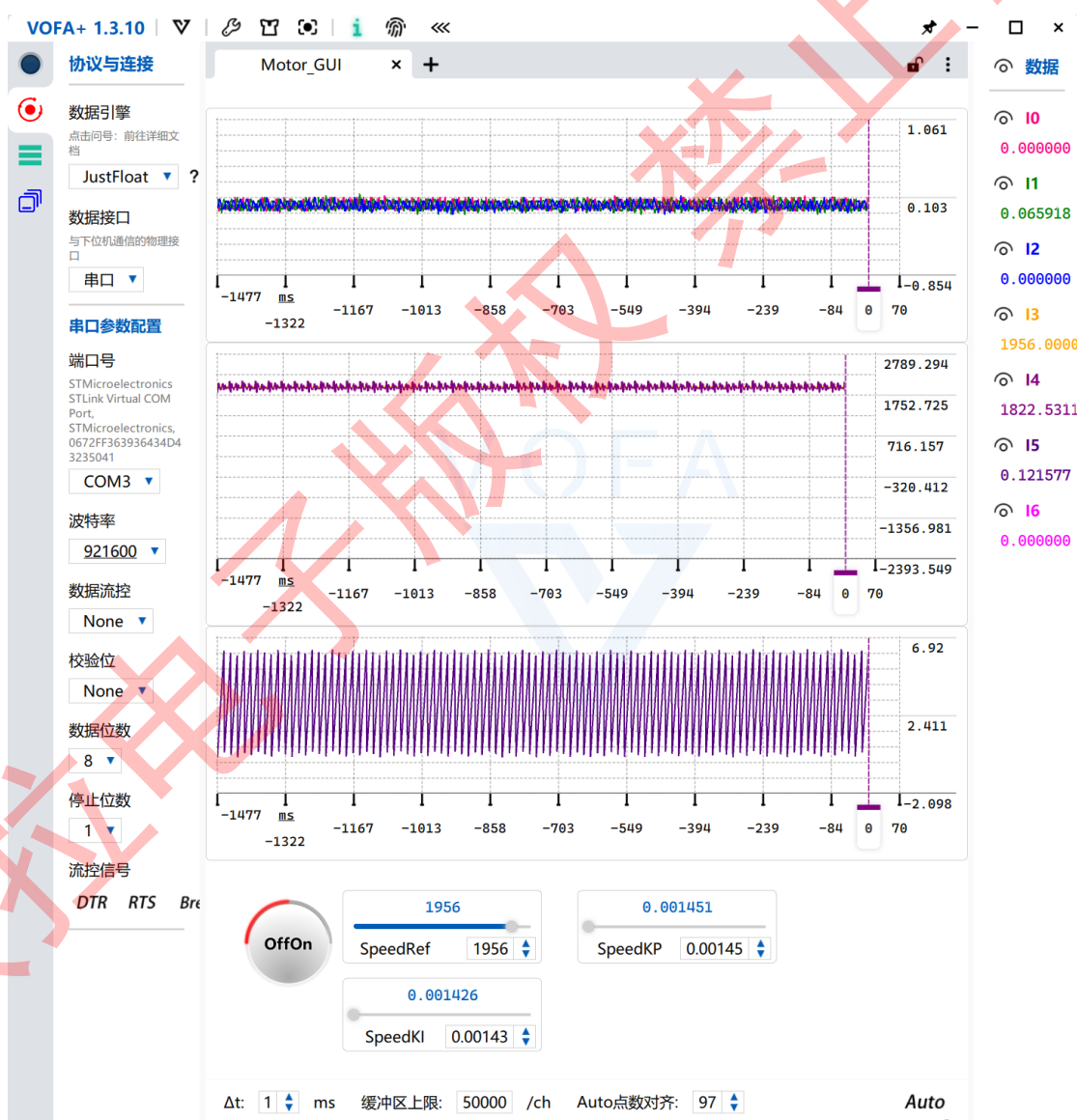


选择 MotorGui.tabviews.json 文件，并打开；

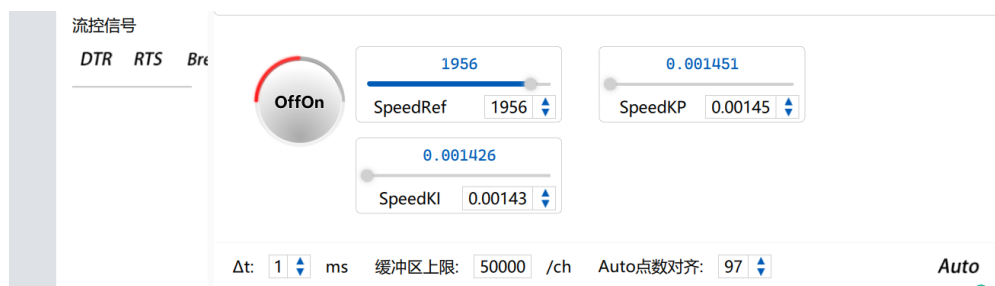




载入成功后，界面显示响应的控件：



选择对应的 COM 口，连接后即可下发数据指令控制电机运行及相应参数。



注意：控件下发的参数不能连续非常快速下发，可能会到 VOFA+软件卡死。