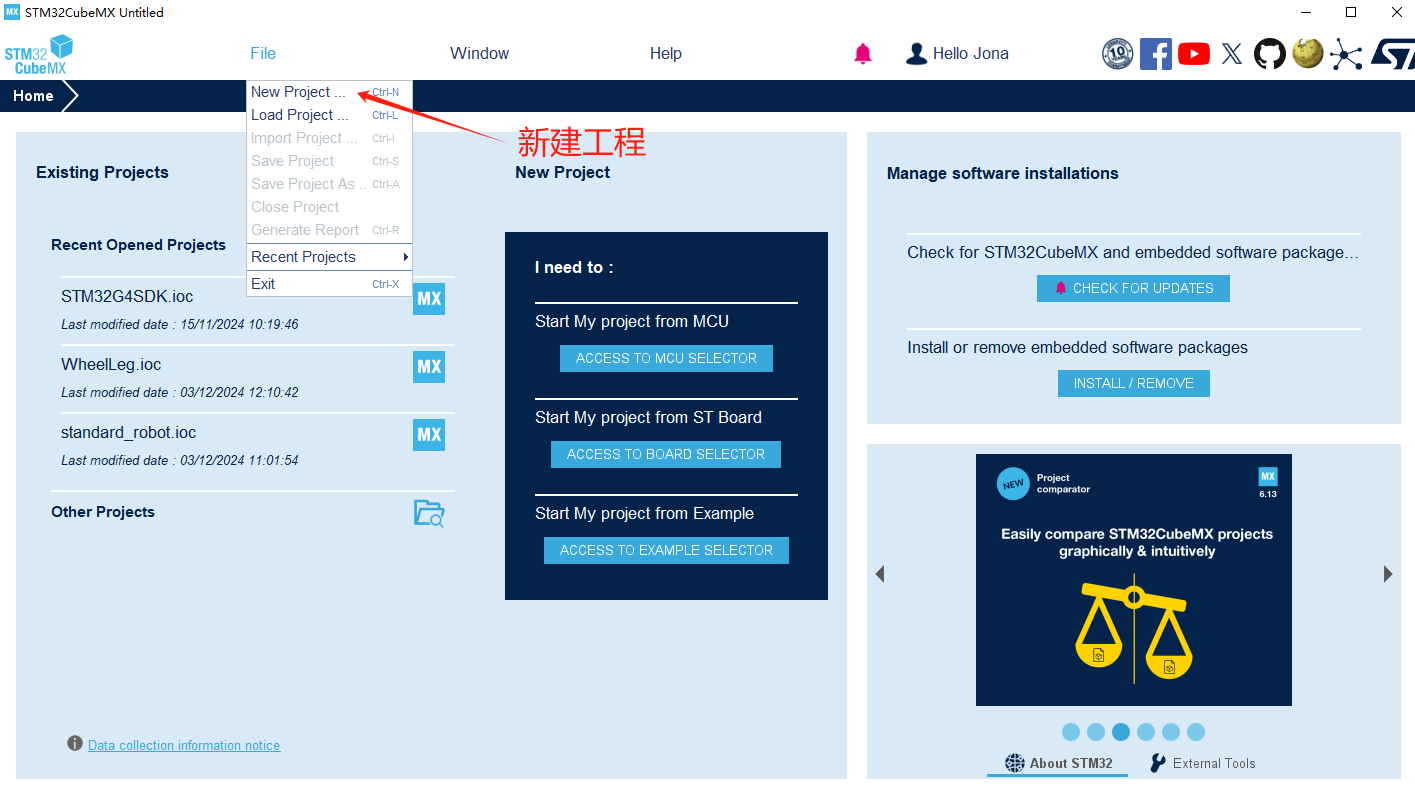
**守护兽驱动SDK用户手册**

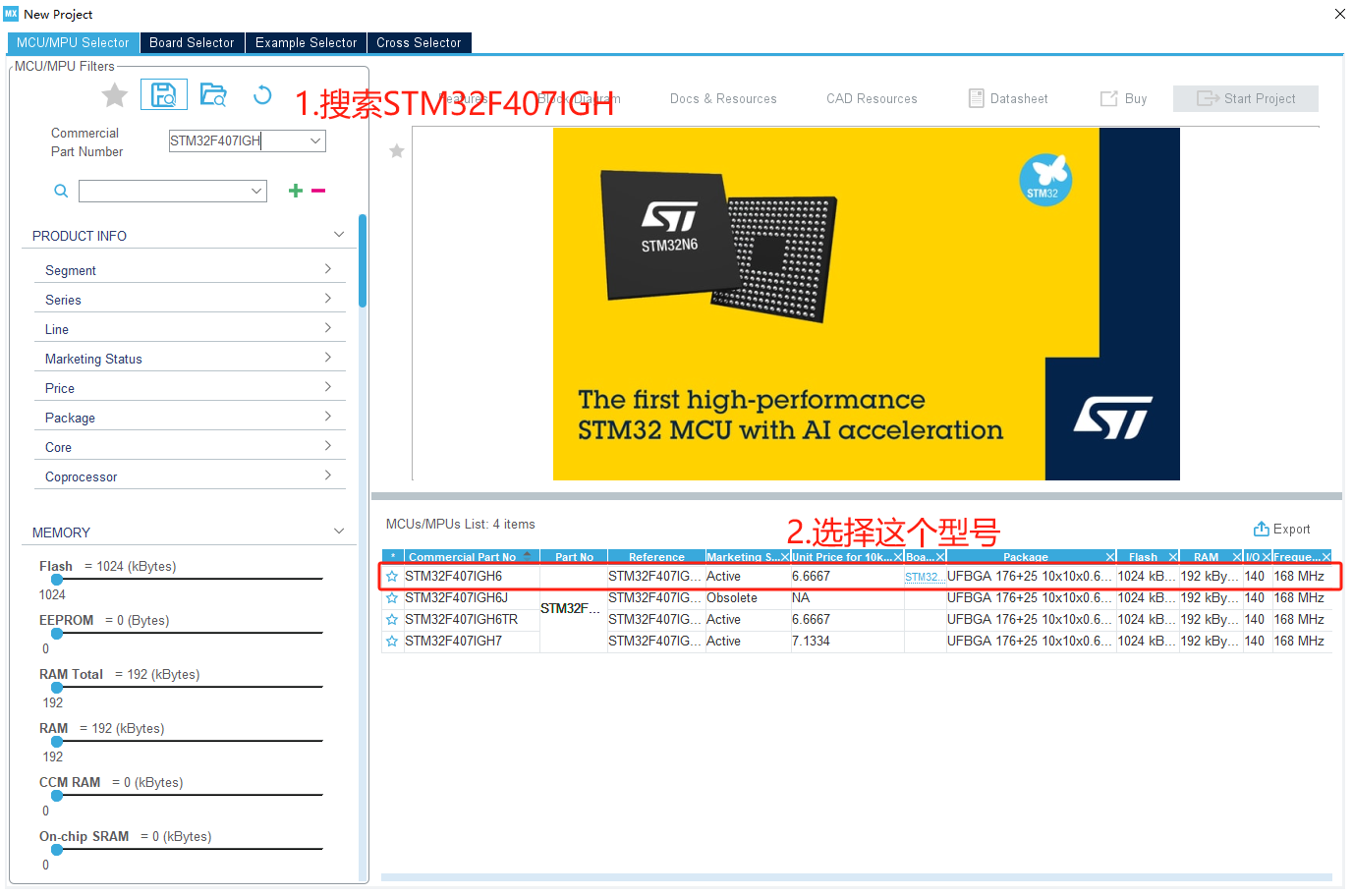
|  |
| --- |
| 本说明以大疆C板STM32F407IG为例，对SDK的部分功能进行说明，其中使用的MWMotor.c .h在任何支持C语言编译环境下的系统都可以使用，不限制于STM32，代码仓库：[北京守护兽科技有限公司/MWMotorSDK: MW电机软件开发工具包(SDK)](https://gitee.com/cyberbeast/mwmotorsdk) |

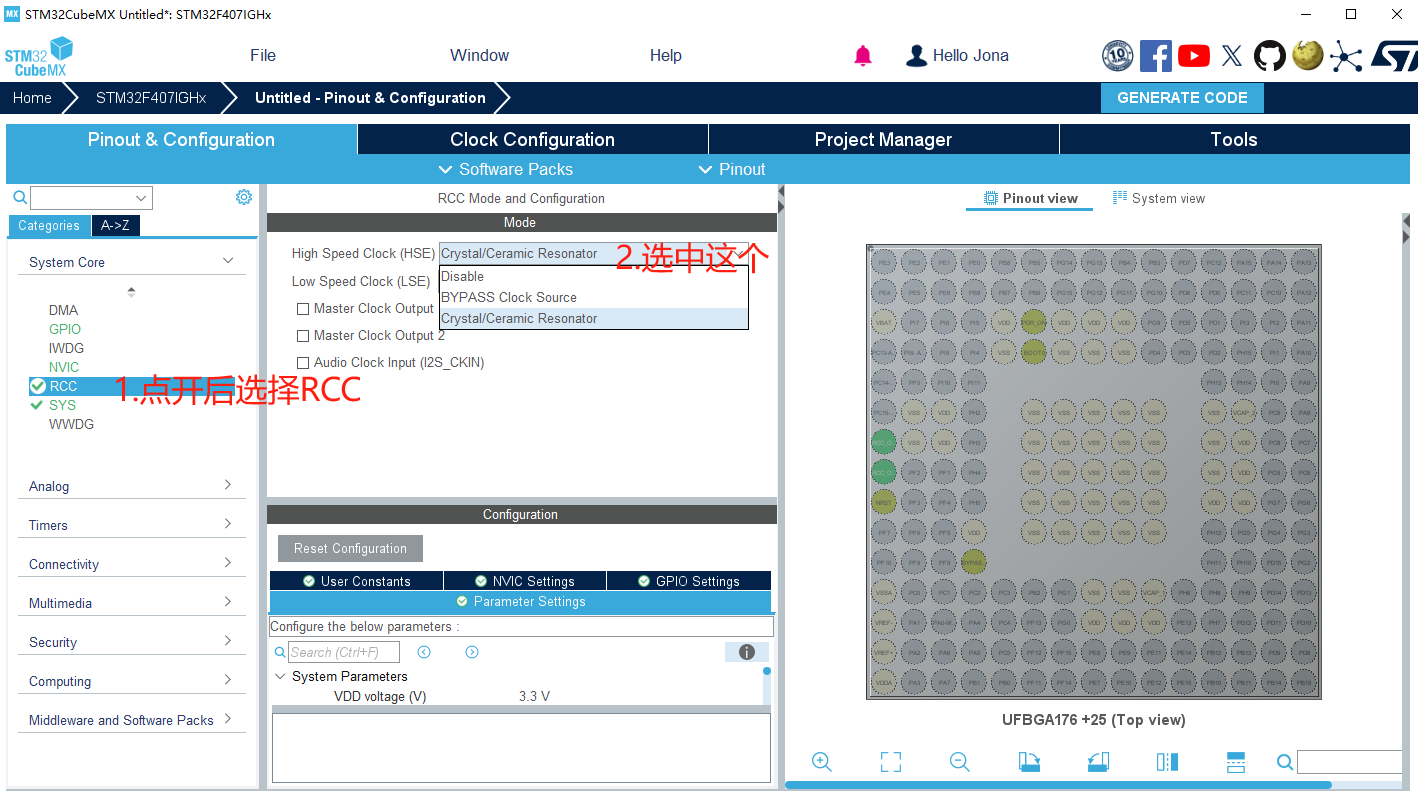
1. **STM32CUBEMX和Keil5工程创建**

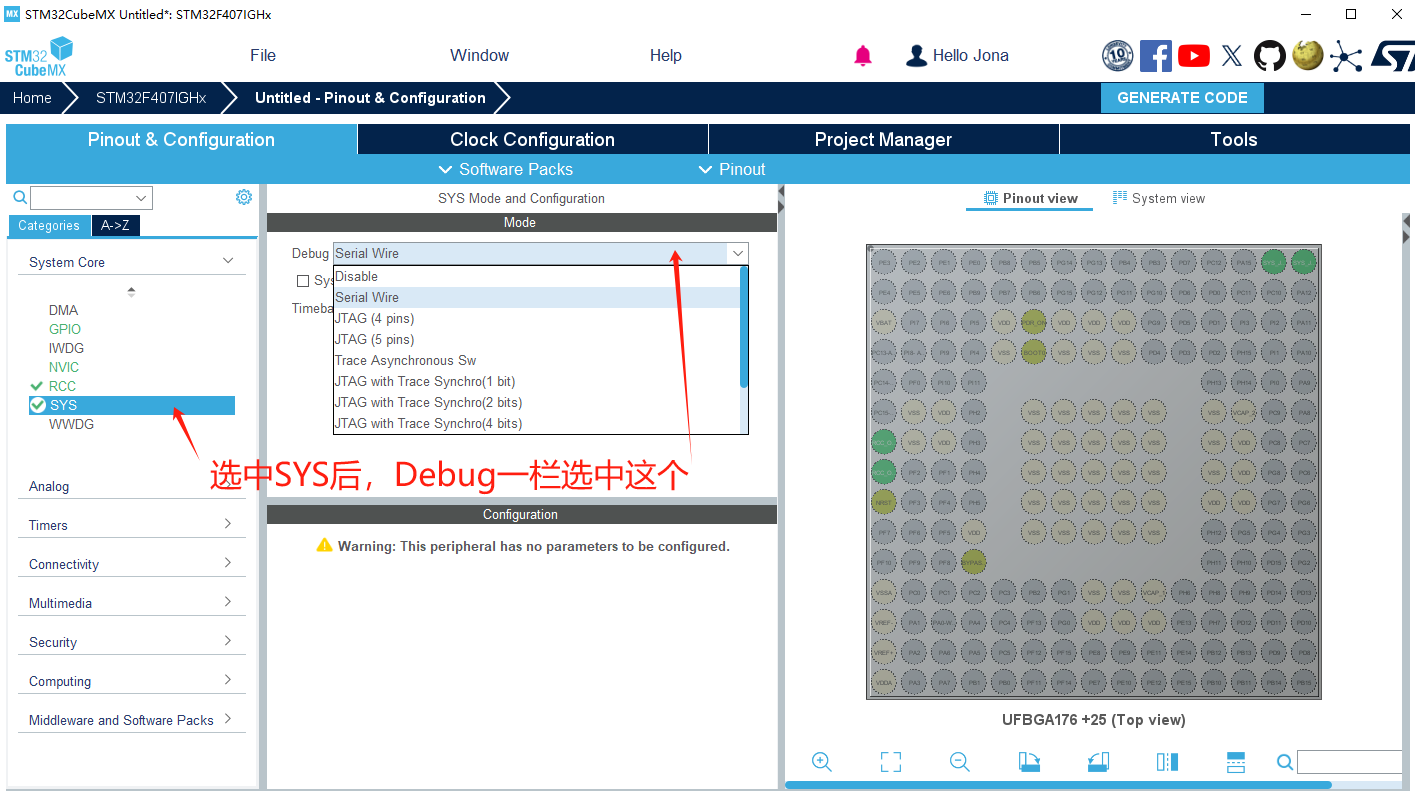
如果仅进行移植，只需将MWMotor.c .h复制到自己的工程目录下进行编译，编译通过后，直接看[第二节](https://bcnyljrhe70u.feishu.cn/docx/KRJcdmc3yovKxsxNnXOcNmppnwf#doxcn3BTRjCjAQqFO18RHBStHie)即可。

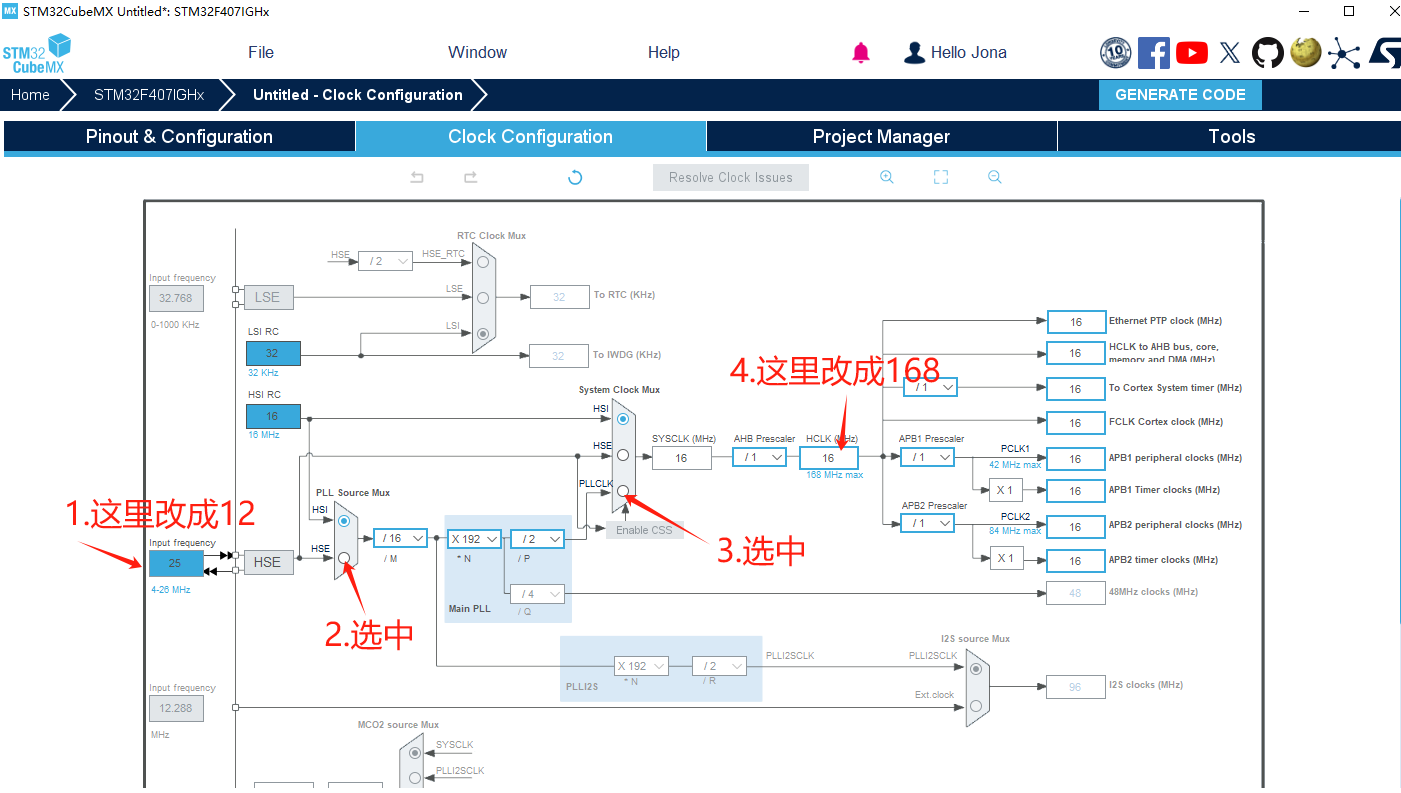
1.1 **工程创建**

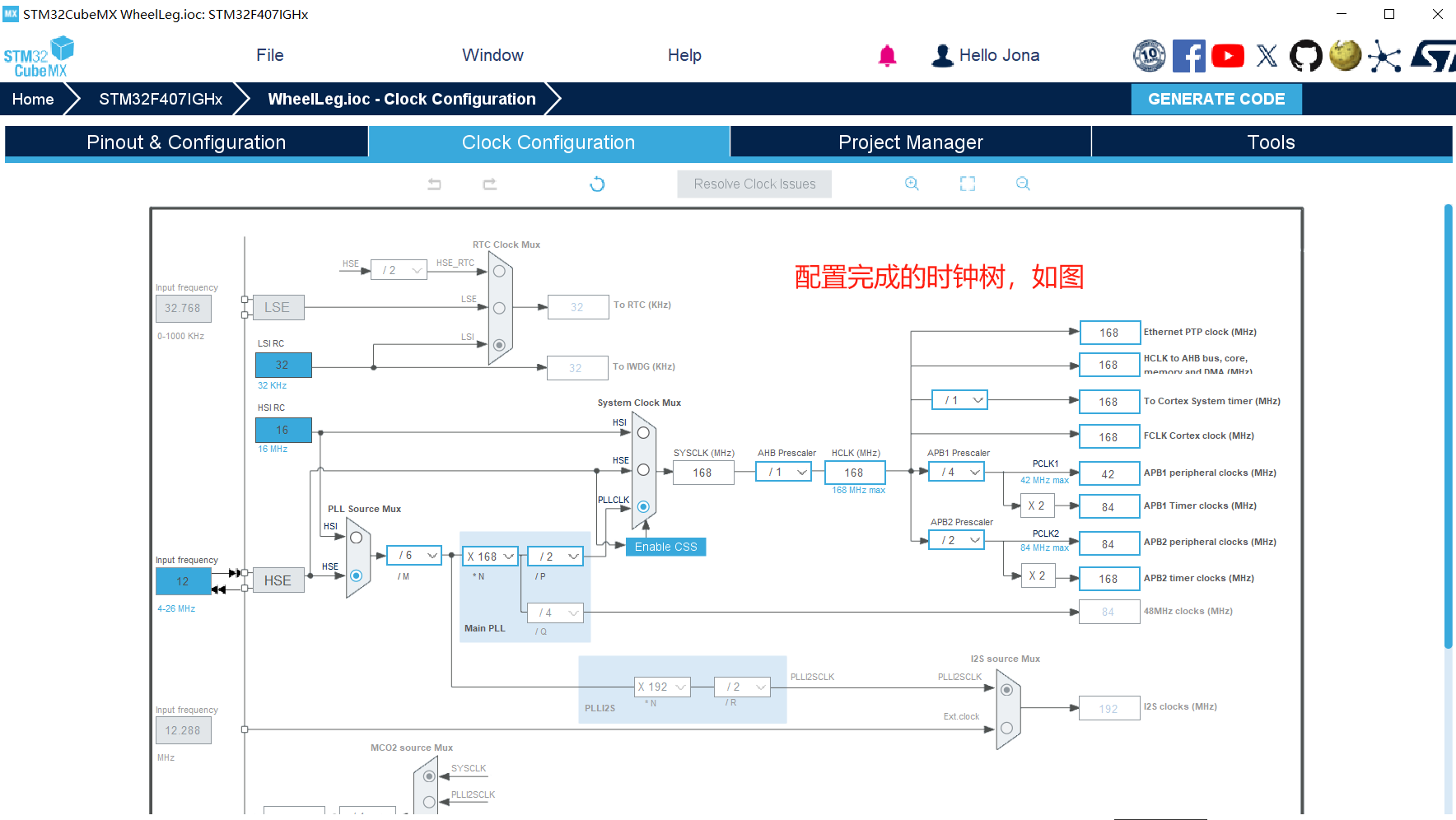


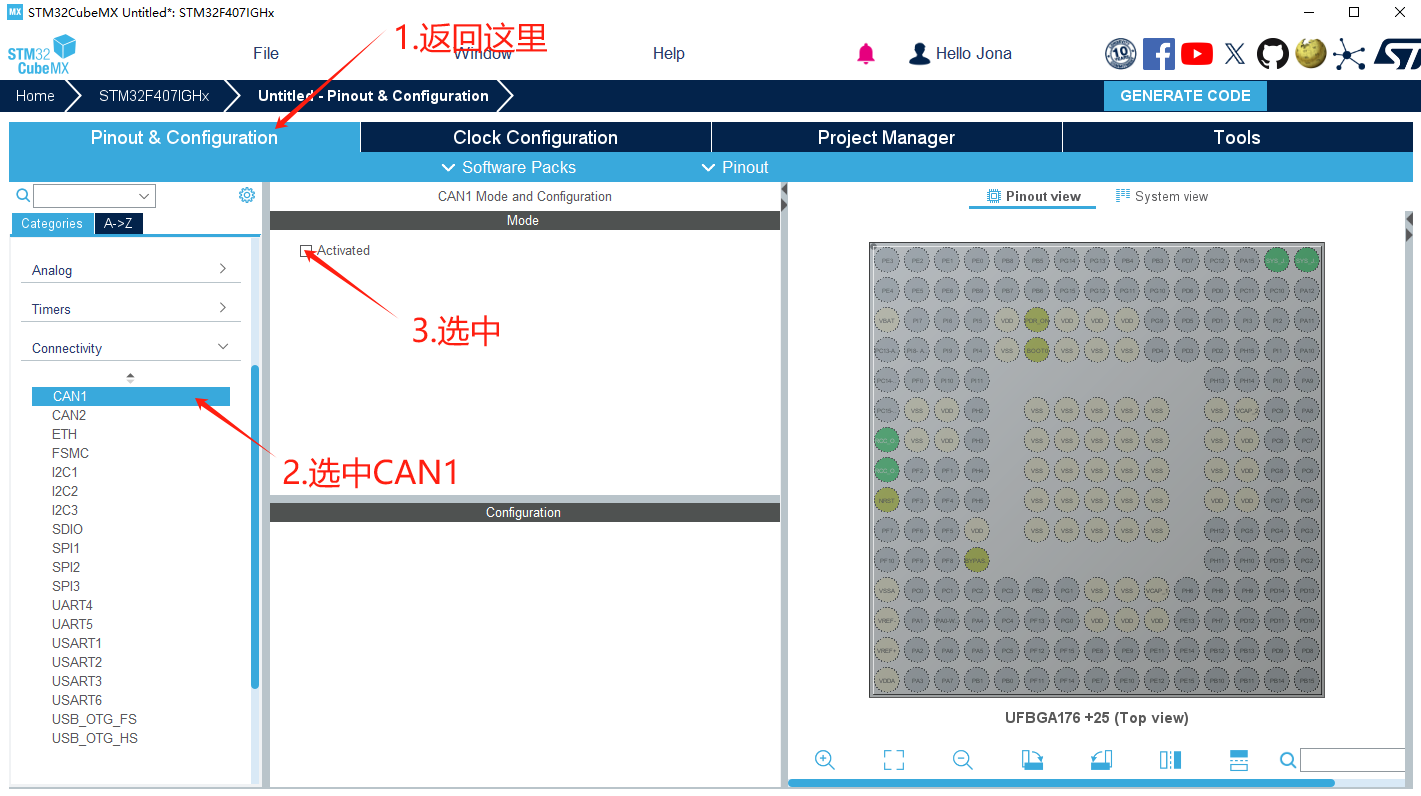


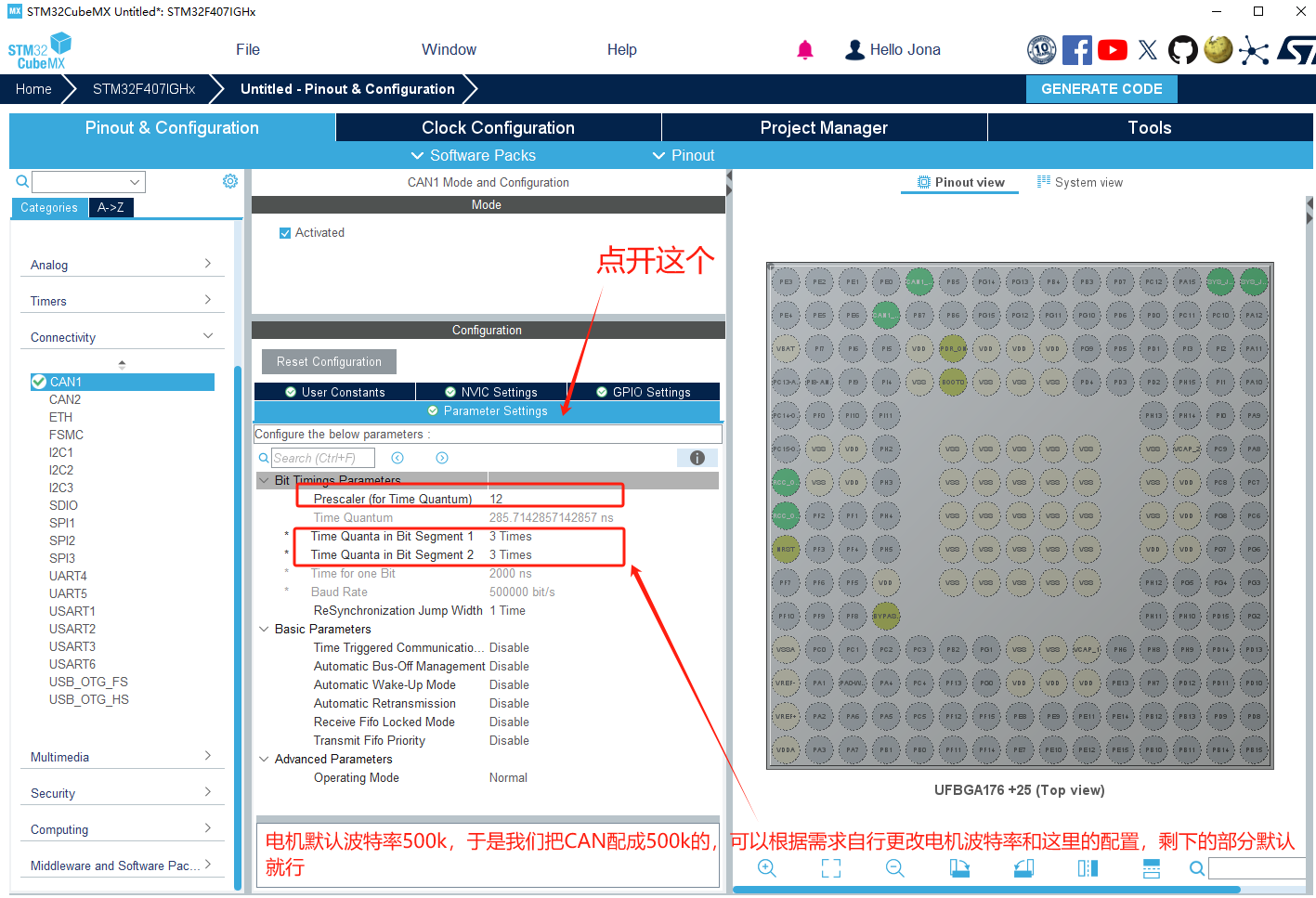


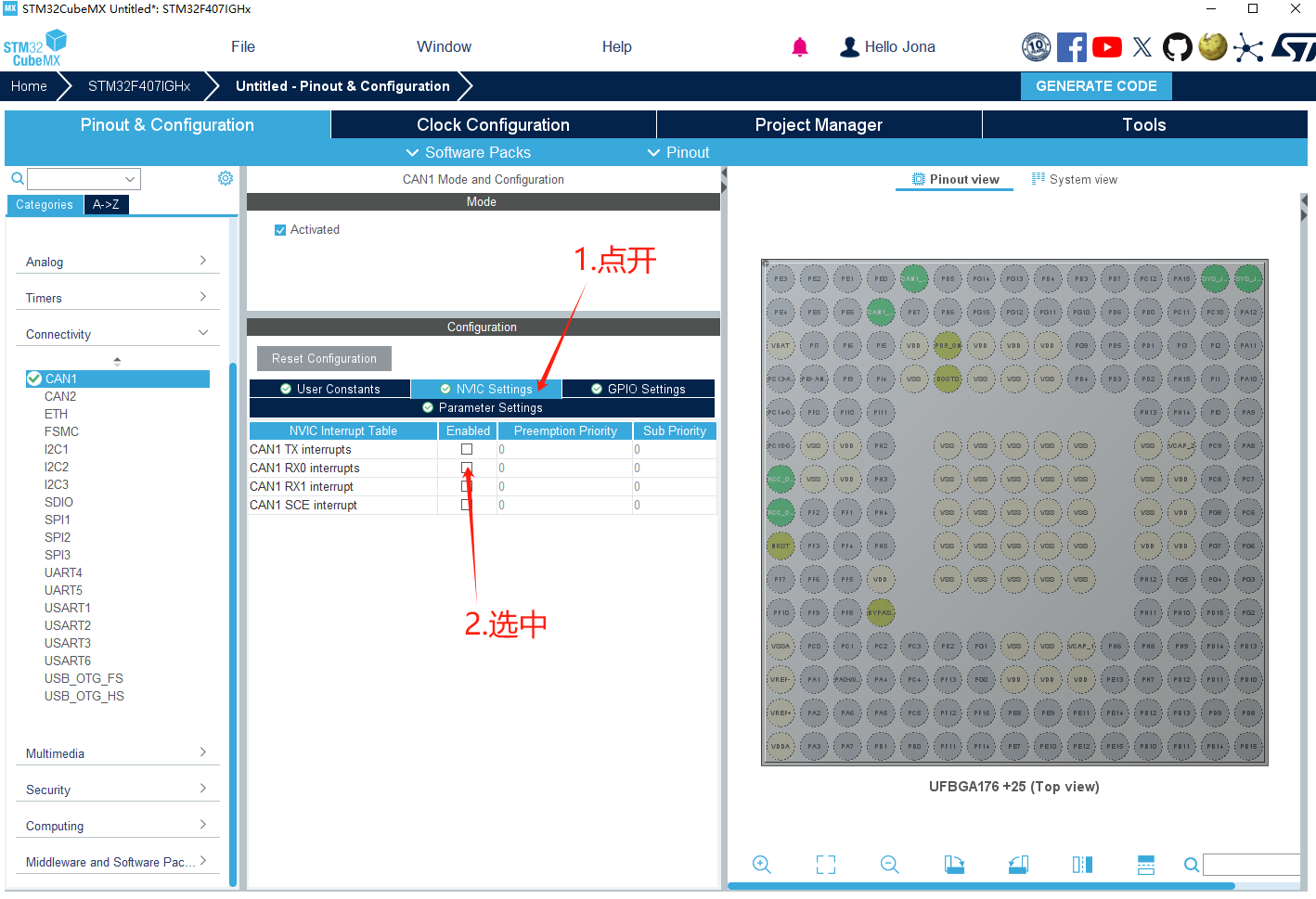


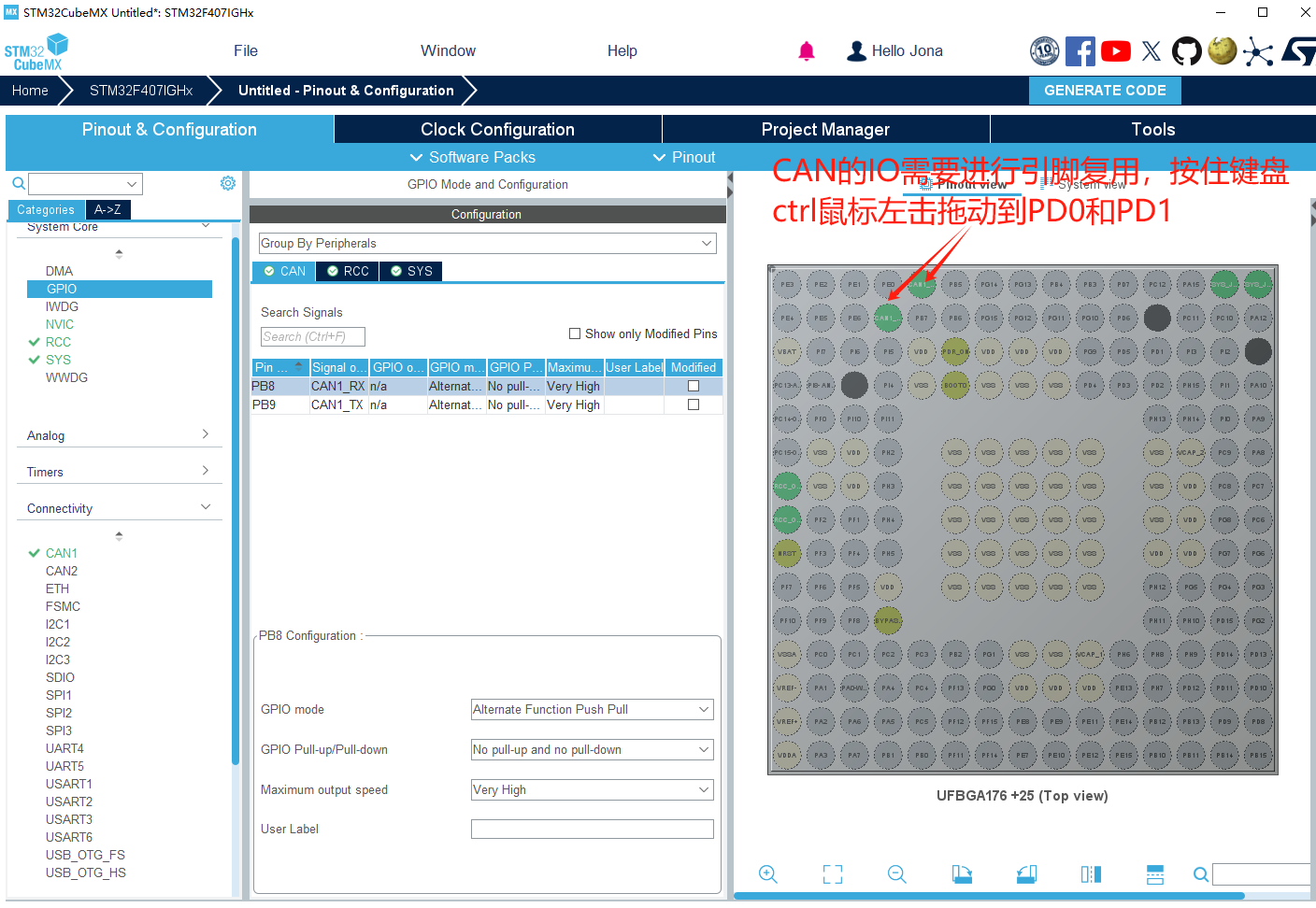


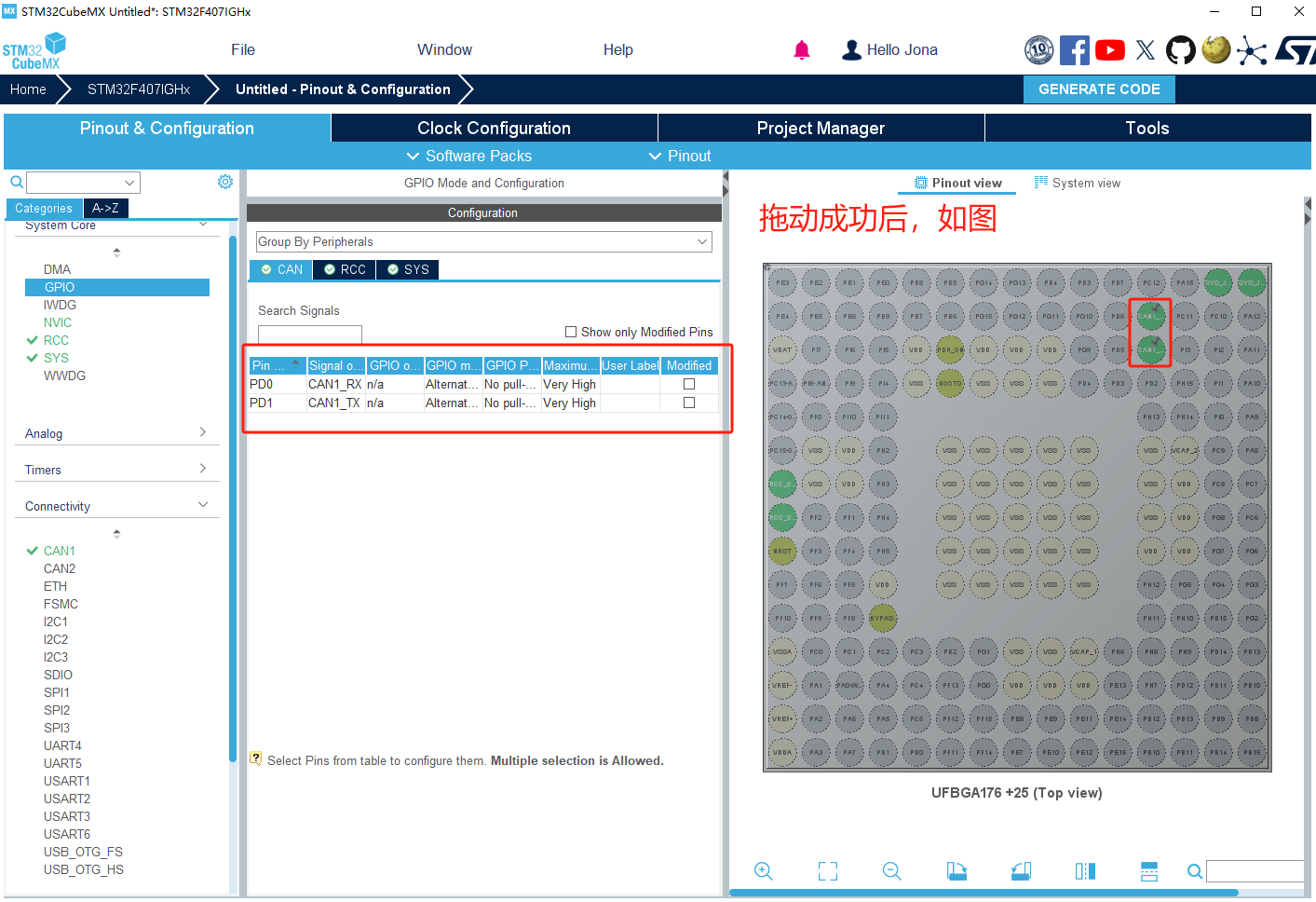


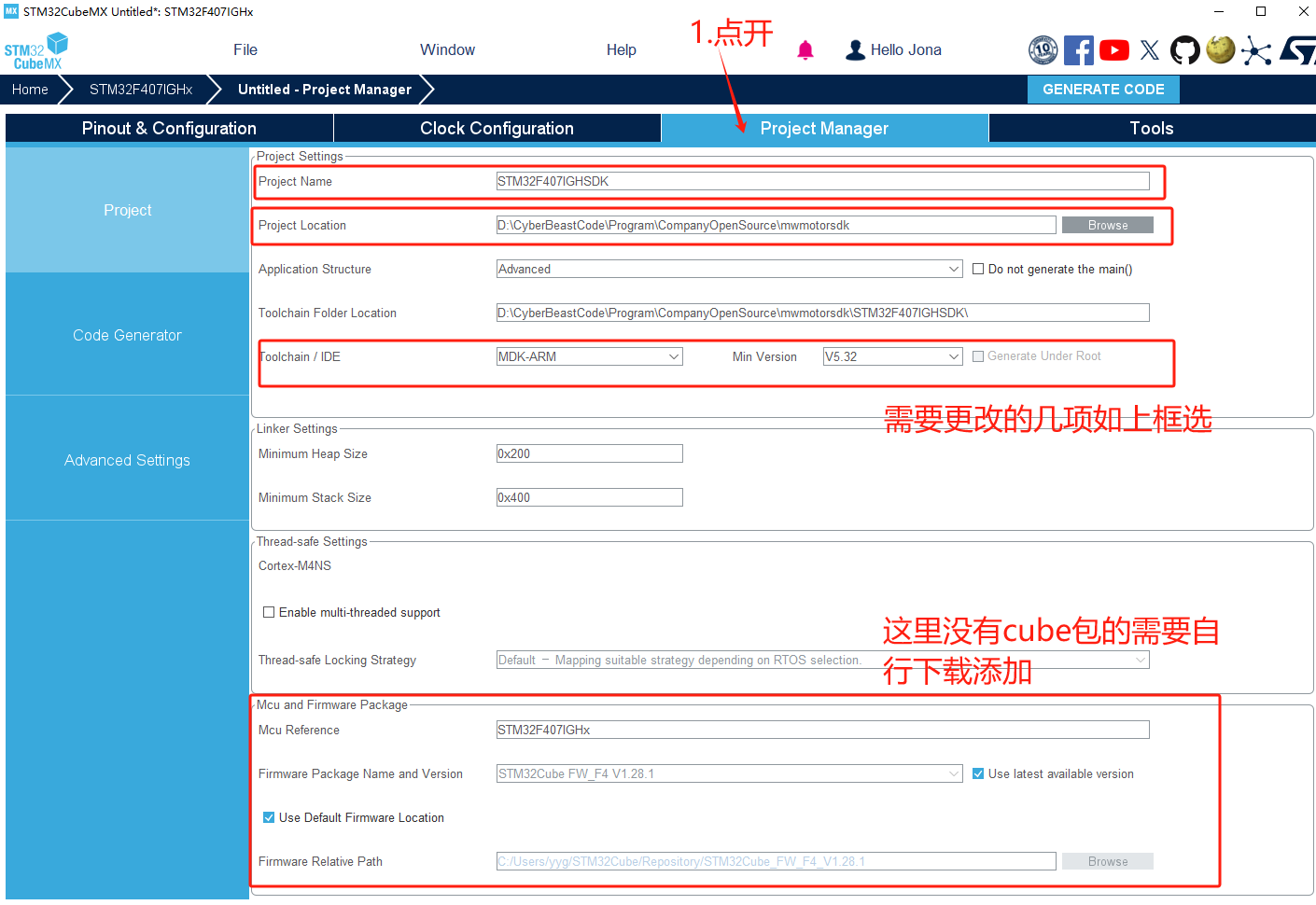


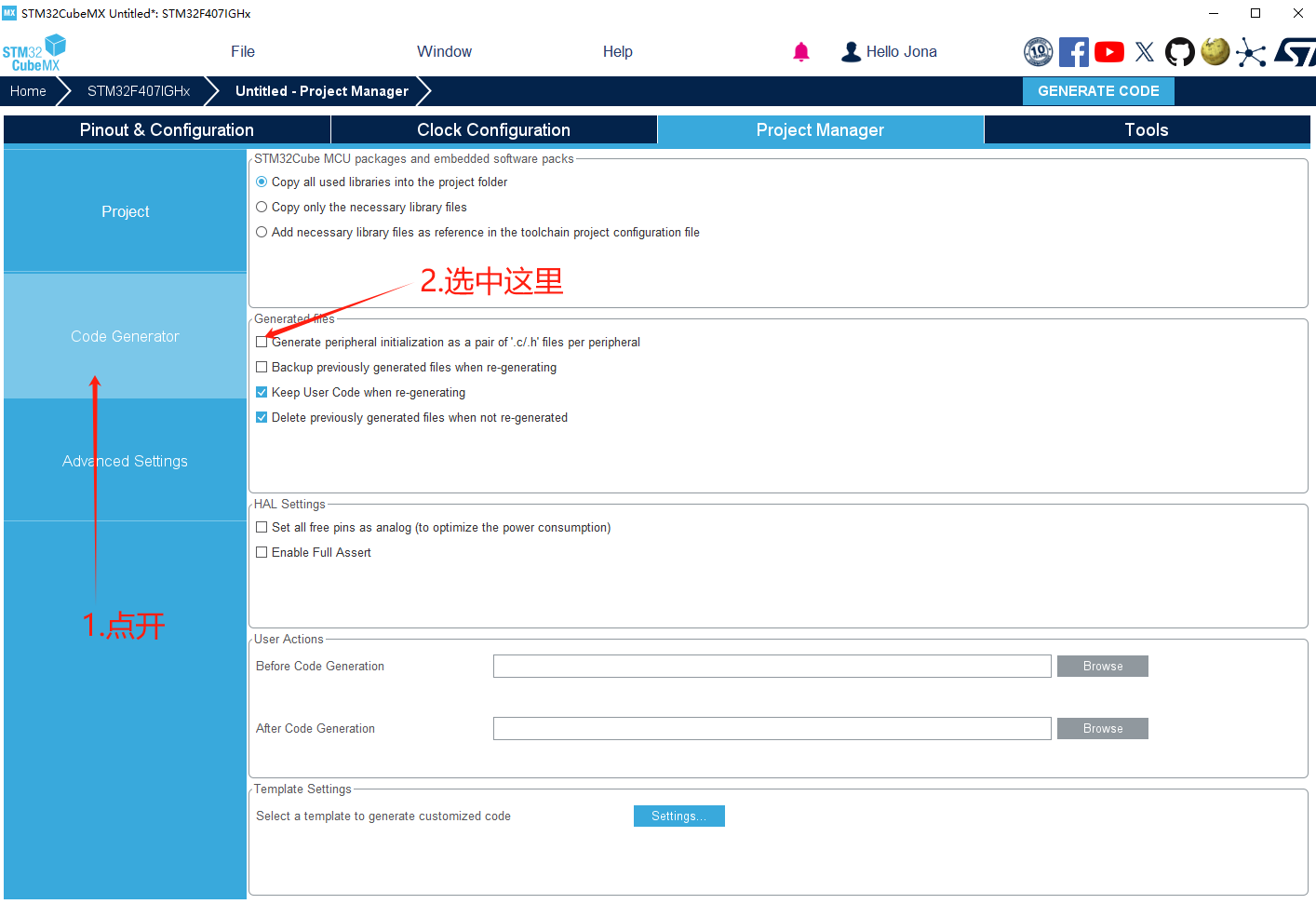


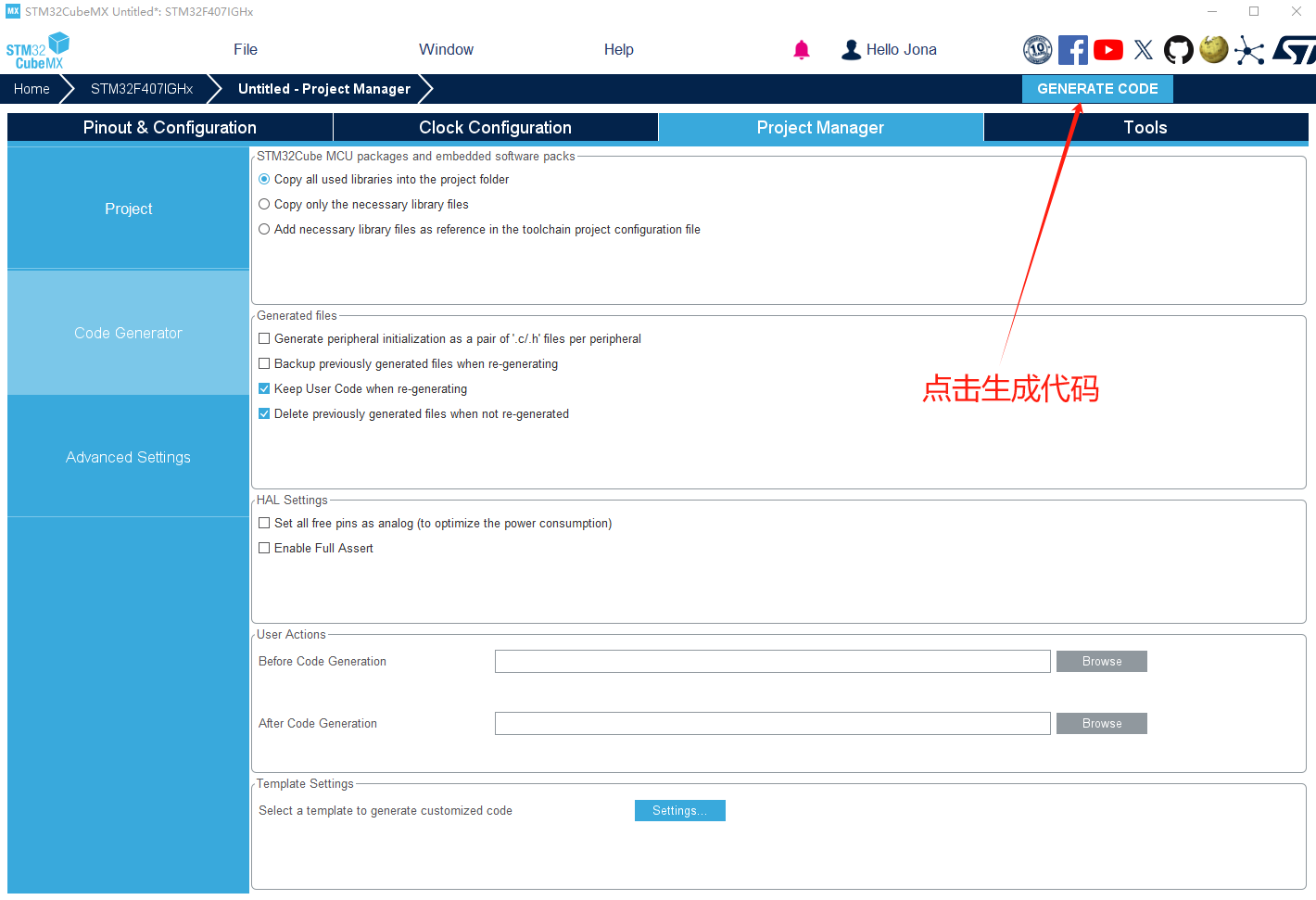


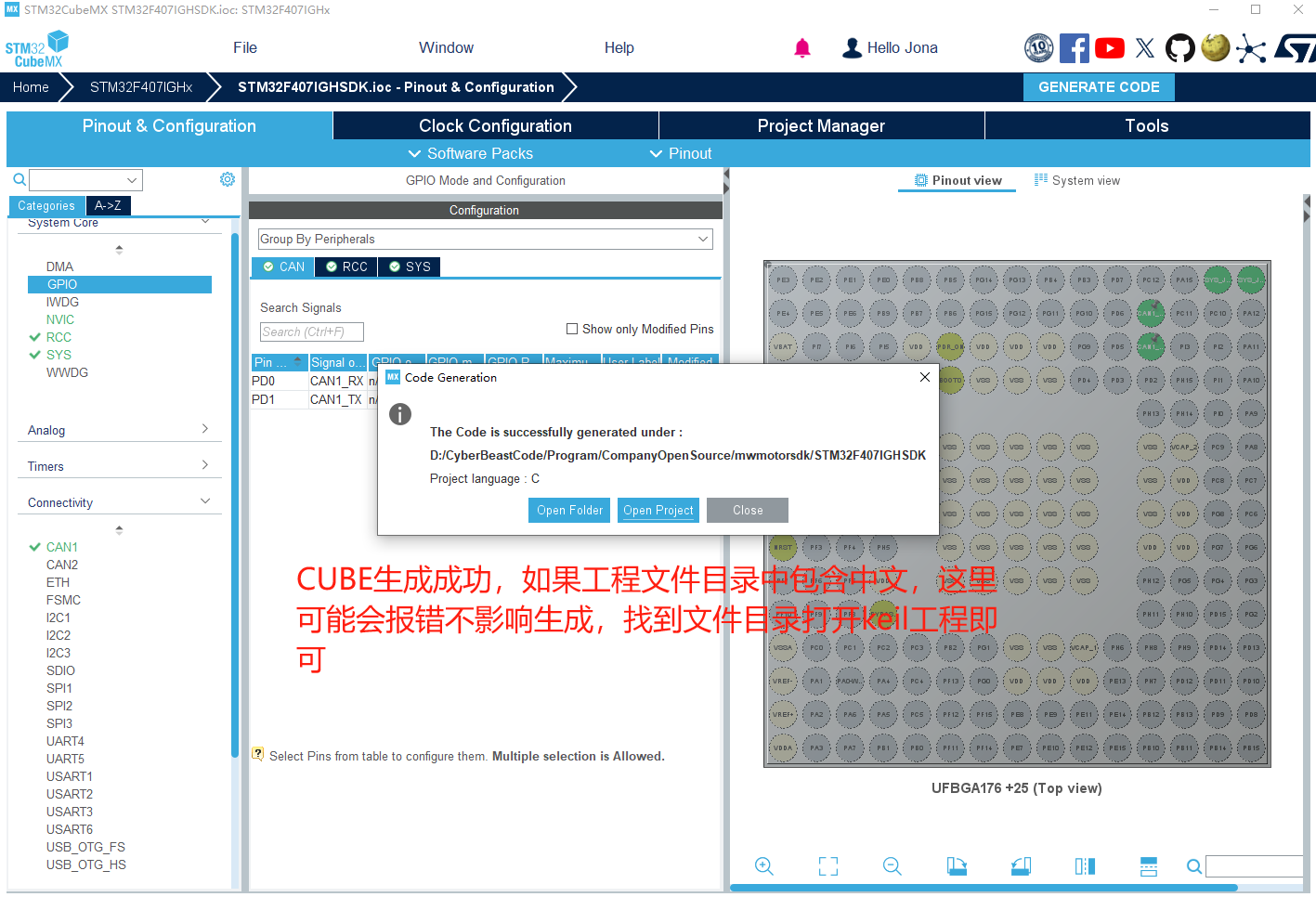




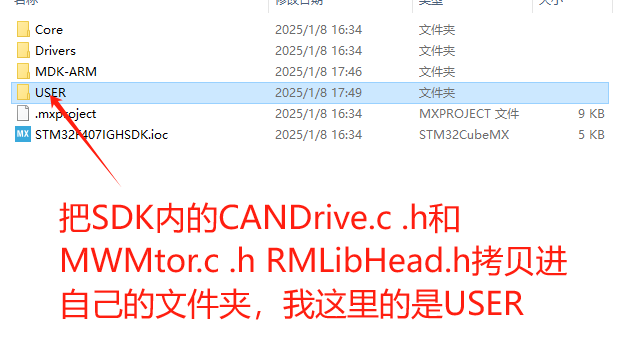


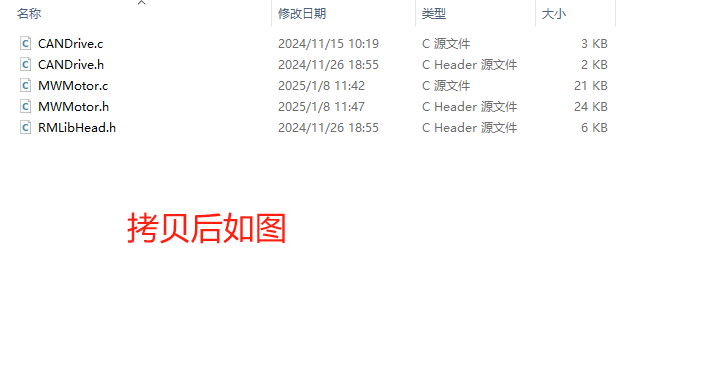




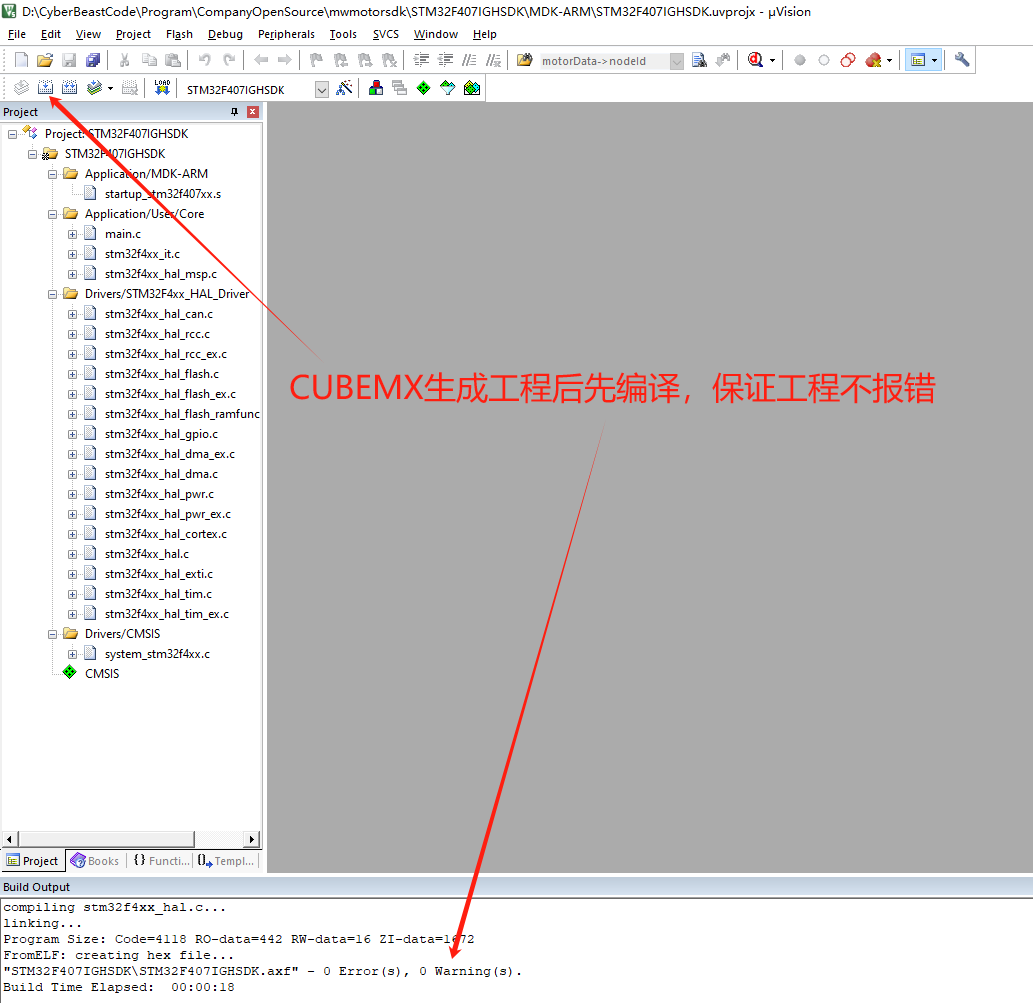


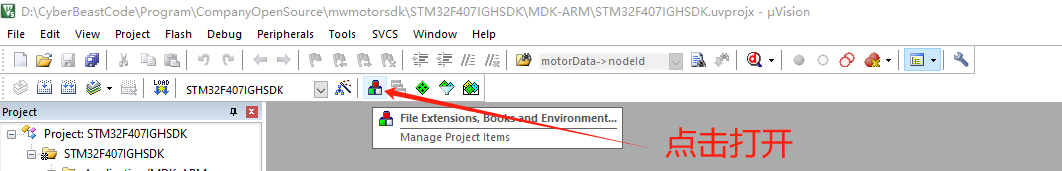
1.2 **Keil5导入文件并编译**

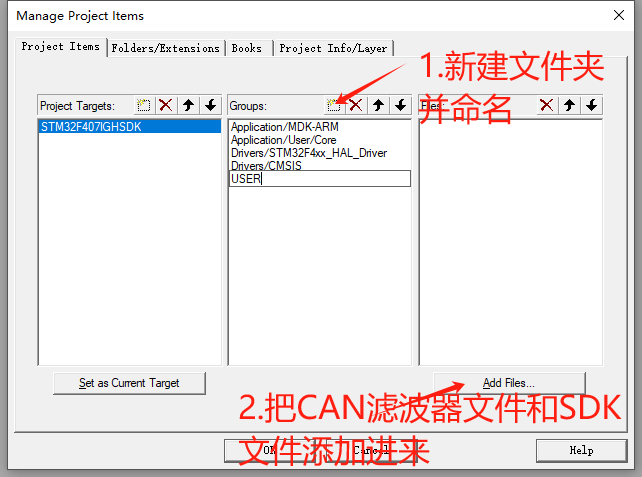


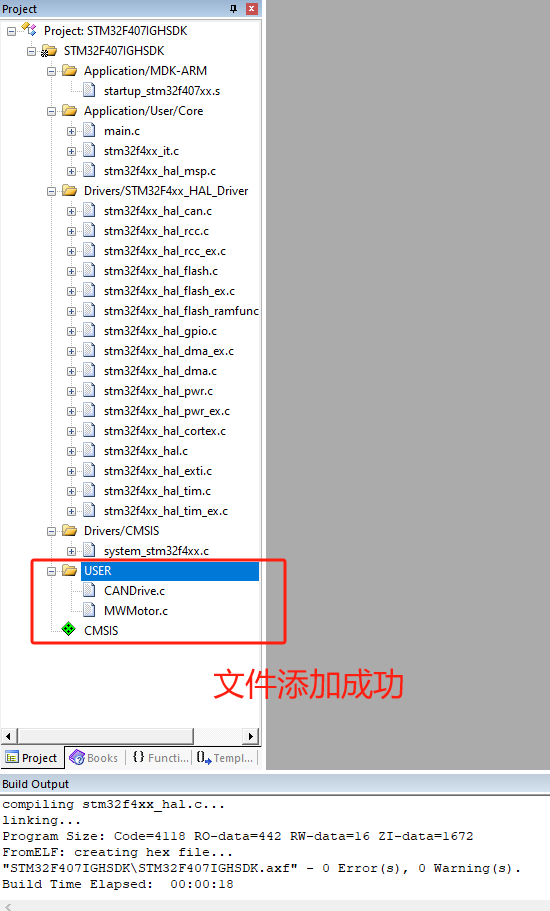


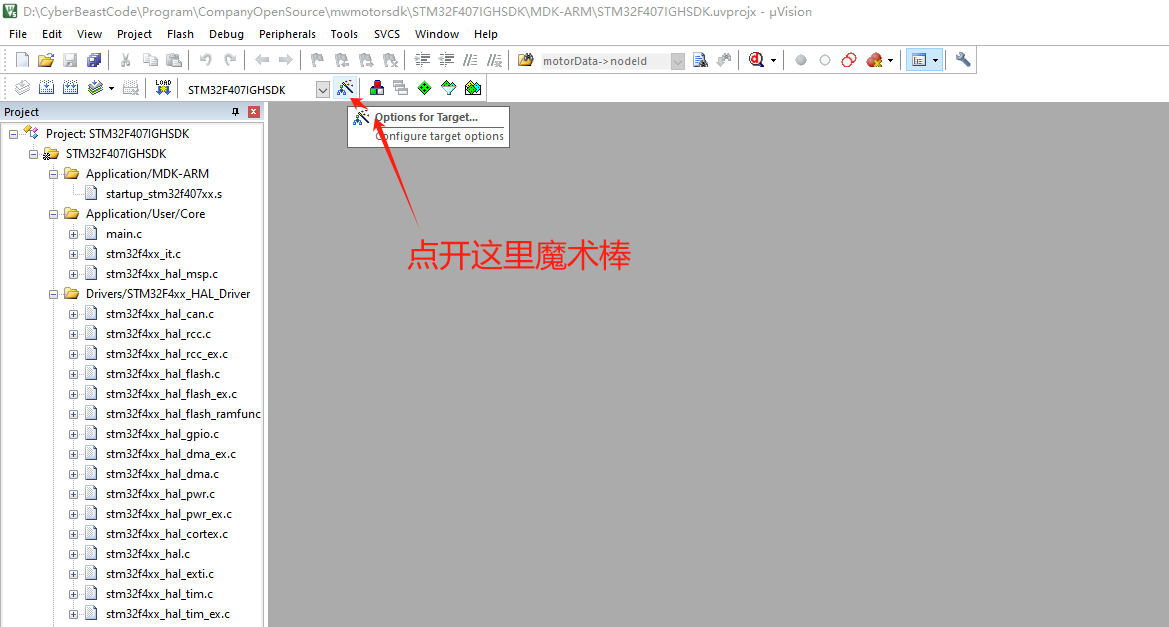
|  |
| --- |
| 注意，STM32F407IGH用的是CAN，而另一个工程STM32G431里的使用的是FDCAN，两个工程下的CANDrive.c .h虽然同名，但是内容不同！ |

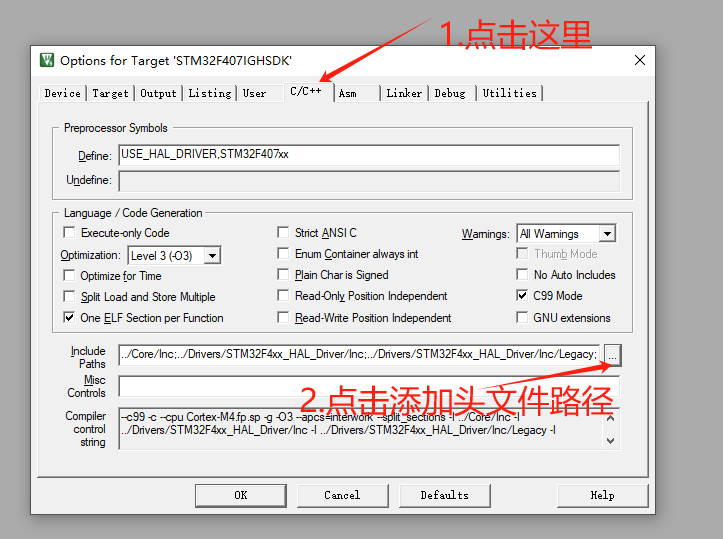


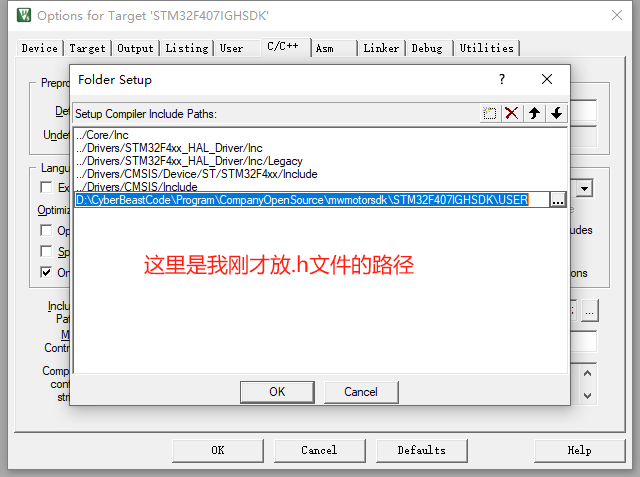


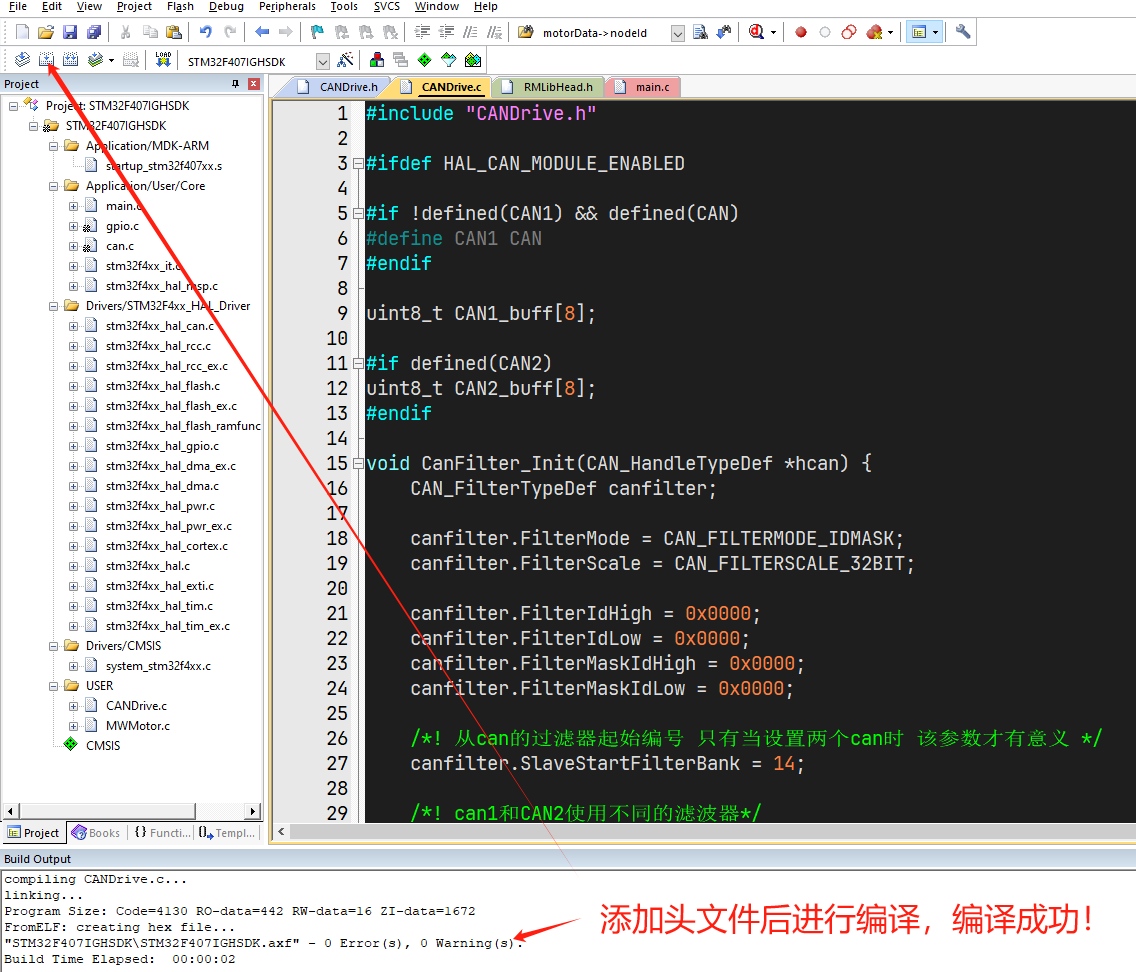












2. **开始使用SDK**

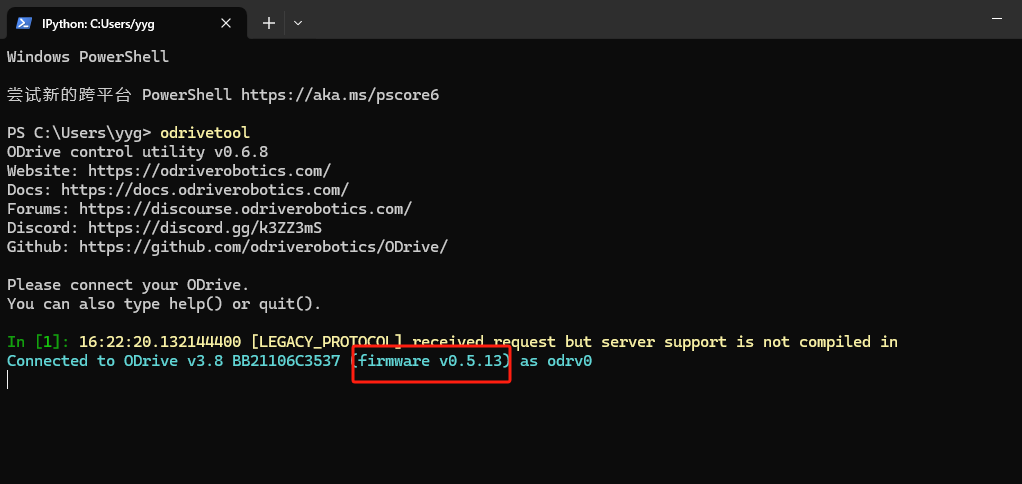
|  |
| --- |
| 使用SDK需要创建什么变量，什么函数，根据固件版本定义固件号，怎么查看 |

2.1 **修改固件版本号**

由于固件版本不同，导致心跳版本不同，为了得到正确的心跳反馈，我们需要先得到固件版本号在MWMotor.h中更改固件号。

2.1.1 **odrivetool查看固件版本**

按照说明书正确安装odrivetool，电机电源灯亮蓝色，并用type-c线连接电机，打开终端，执行命令odrivetool

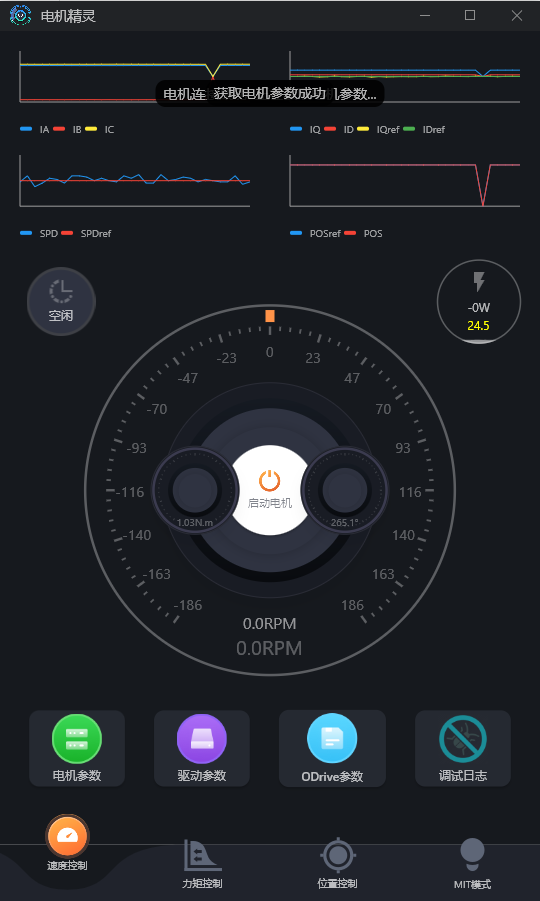


如图，红框中的即为电机固件版本为0.5.13，前一个V3.8是硬件版本号

|  |
| --- |
| 打开SDK中的头文件MWMotor.h，上文中我们的固件是0.5.13，那我们对应的固件版本号为513，如下图所示    如果固件版本是0.5.12，则固件版本号为512，以此类推。 |

2.1.2 **电机精灵查看固件版本**

按照说明书正确安装电机精灵后，点开电机精灵，等待电脑成功连接电机后，会显示参数获取成功



打开绿色图标电机参数，会看到如下图红框识别到的固件版本为0.5.13，硬件版本为3.8



|  |
| --- |
| 打开SDK中的头文件MWMotor.h，上文中我们的固件是0.5.13，那我们对应的固件版本号为513，如下图所示    如果固件版本是0.5.12，则固件版本号为512，以此类推。  同步自文档: <https://mcnzslglxogb.feishu.cn/docx/AcGwdCBIxo4ok4xacAqckqRlnif#IuFAdQV2rsETjdbOSiQcLQr2n9b> |

2.2 **创建电机结构体**

完成以上操作步骤后，打开keil任意一个位置定义两个全局结构体

|  |
| --- |
| C /\* 电机数据接收 \*/ MW\_MOTOR\_DATA MWtestData; /\* 电机登记数据 \*/ MW\_MOTOR\_ACCESS\_INFO MWtest = {.busId = 1,  .nodeId = 0,  .motorData = &MWtestData,  .sender = MotorBusSend,  .notifier = MotorNotice}; |

**MW\_MOTOR\_DATA**

此类型的结构体是用来接收电机的反馈数据无需初始化

**MW\_MOTOR\_ACCESS\_INFO**

此类型的结构体用来保存电机的使用信息

**busId**

是电机所属的总线号（用户自行进行总线区分，如果不知道如何使用，使所有电机都在一个总线号上即可）

**nodeId**

是电机的ID号，用户需要在上位机上配置好电机的nodeId后，将配置的ID写到这里

* odrivetool中设置nodeId

在odrivetool中执行这个指令后保存

|  |
| --- |
| C odrv0.axis0.config.can.node\_id=0 odrv0.save\_configuration() |

* 电机精灵中设置电机nodeId

打开电机精灵，点开电机参数，**请在电机处于空闲状态再同步参数**



**motorData**

把刚刚创建的MW\_MOTOR\_DATA结构体的指针传到这里。

**sender**

SDK发送函数，由于每个系统CAN发送的函数不一样，且在SDK中需要使用CAN发送函数，于是SDK中定义了一个函数指针类型来做不同控制系统的兼容。用户需要自己定义一个函数，这个函数我们已经固定了函数格式：

|  |
| --- |
| C /\* 定义一个函数指针类型，用于发送电机控制指令 \*/ typedef void (\*MotorSender)(uint8\_t busId, uint8\_t canId, uint8\_t \*data, uint8\_t dataSize); |

函数名称任意，需要传入的参数需要包含uint8\_t类型的busId总线ID，uint8\_t类型的canId，uint8\_t\*的CAN发送数组的指针，uint8\_t类型的dataSize数据大小。

**在这个定义的函数内，调用自己系统的CAN发送函数即可**。

|  |
| --- |
| C void MotorBusSend(uint8\_t busId, uint8\_t can\_id, uint8\_t \*data, uint8\_t dataSize) {  /\* 用户自己的CAN发送函数 \*/ } |

例如，在SMT32的HAL库上：

|  |
| --- |
| C void MotorBusSend(uint8\_t busId, uint8\_t can\_id, uint8\_t \*data, uint8\_t dataSize) {  if(busId == 0x001) {  /\*等待发送总线空闲\*/   /\*发送函数\*/  CAN\_Send\_StdDataFrame(&hcan1, can\_id, data);  } } |

然后**把创建的这个函数的指针传给结构体内用来初始化。**

**notifier**

SDK通知函数，和上面的发送函数一样，SDK定义了一个函数指针类型来做不同控制系统的兼容。如下：

|  |
| --- |
| C /\* 定义一个函数指针类型，用于接收电机状态通知 \*/ typedef void (\*MotorNotifier)(uint8\_t busId, uint8\_t nodeId, MW\_CMD\_ID cmdId); |

需要定义一个同类型的函数：

|  |
| --- |
| C /\* 用户自创建总线消息函数 \*/ void MotorNotice(uint8\_t busId, uint8\_t nodeId, MW\_CMD\_ID cmdId) {  if(busId == 0x001) return; } |

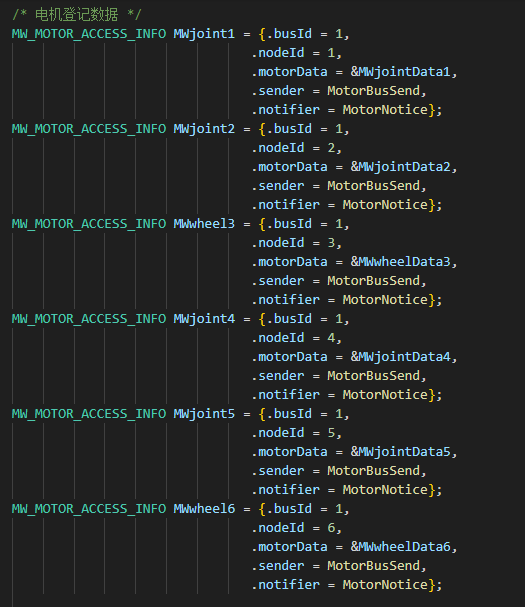
如果这个功能用不上可以定义一个空函数传给结构体做初始化。

**2.3 调用函数登记电机信息**

找到我们系统运行的主函数，在系统CAN初始化完成后调用电机信息登记函数MWRegisterMotor()将我们刚才创建的

|  |
| --- |
| C  /\* 电机总线创建 \*/  MWRegisterMotor(MWtest); |

如果有多个电机，就需要定义多个结构体并调用登记函数，例如：





**2.5 数据接收**

调用MWReceiver()函数即可实现电机数据的接收，其中需要传入电机的busId、canId、CAN接收到数组的指针。

例如，在STM32的例程中，我是在HAL库的回调函数中调用实现的数据接收：

|  |
| --- |
| C void HAL\_CAN\_RxFifo0MsgPendingCallback(CAN\_HandleTypeDef \*hcan){  /\* 判断是否是CAN1收到的数据 \*/  if(hcan->Instance == CAN1) {  uint32\_t canId = CAN\_Receive\_DataFrame(&hcan1, CAN1\_buff);  /\* 传入CAN接收的数据和ID \*/  MWReceiver(1, canId, CAN1\_buff);  } } |

|  |
| --- |
| 至此，电机已全部配置完成！ |

3. **函数详解**

**功能简述**

|  |  |
| --- | --- |
| 名称 | 功能 |
| MWRegisterMotor | 注册电机信息 |
| MWReceiver | 接收电机反馈数据 |
| MWEstop（0x002） | 紧急停止电机 |
| MWGetMotorError（0x003） | 获取电机错误信息 |
| MWRxTxSdo（0x004 0x005） | 用endpoint\_id访问或写入电机其他功能 |
| MWSetAxisNodeID（0x006） | 设置电机的节点ID |
| MWSetAxisState（0x007） | 设置电机状态 |
| MWMitControl（0x008） | MIT运动控制 |
| MWGetEncoderEstimates（0x009） | 获取编码器的速度和位置信息 |
| MWGetEncoderCount（0x00A） | 获取编码器计数函数，从电机控制器获取编码器的计数信息 |
| MWSetControllerMode（0x00B） | 设置电机控制器的控制模式和输入模式 |
| MWPosControl（0x00C） | 位置控制，输入电机位置、速度前馈和力矩前馈 |
| MWVelControl（0x00D） | 速度控制，输入电机速度和力矩前馈 |
| MWTorqueControl（0x00E） | 力矩控制 |
| MWSetLimits（0x00F） | 设置电机限速和限流 |
| MWStartAnticogging（0x010） | 进行力矩纹波校准 |
| MWSetTrajVelLimit（0x011） | 设置梯形曲线位置控制速度限制 |
| MWSetTrajAccelLimits（0x012） | 设置梯形曲线位置控制加速度限制 |
| MWSetTrajInertia（0x013） | 设置梯形曲线位置控制惯量 |
| MWGetIq（0x014） | 获取节点的电流值 |
| MWReboot（0x016） | 电机重启 |
| MWGetBusVoltageCurrent（0x017） | 获取电机电压和电流信息 |
| MWClearErrors（0x018） | 清除所有错误 |
| MWSetLinearCount（0x019） | 设置编码器绝对位置（计数值） |
| MWSetPosGain（0x01A） | 设置电机位置环PID控制的P值 |
| MWSetVelGain（0x01B） | 设置速度环PID控制的P值，I值 |
| MWGetTorques（0x01C） | 获取电机的扭矩信息 |
| MWGetPowers（0x01D） | 获取电机功率数据 |
| MWDisableCAN（0x01E） | 禁用CAN |
| MWSaveConfigeration（0x01F） | 保存电机设置并重启电机 |

4. **参考资料**

例程链接：[北京守护兽科技有限公司/MWMotorSDK](https://gitee.com/cyberbeast/mwmotorsdk)

说明书文档：

[守护兽驱动用户手册](https://cyberbeast.feishu.cn/docx/N3SMd4QyRobzHkx3wP3cT1qXnpf)

[守护兽驱动协议手册](https://cyberbeast.feishu.cn/docx/BPnQd8reEotLWVxqHFNc9qYZnKh)