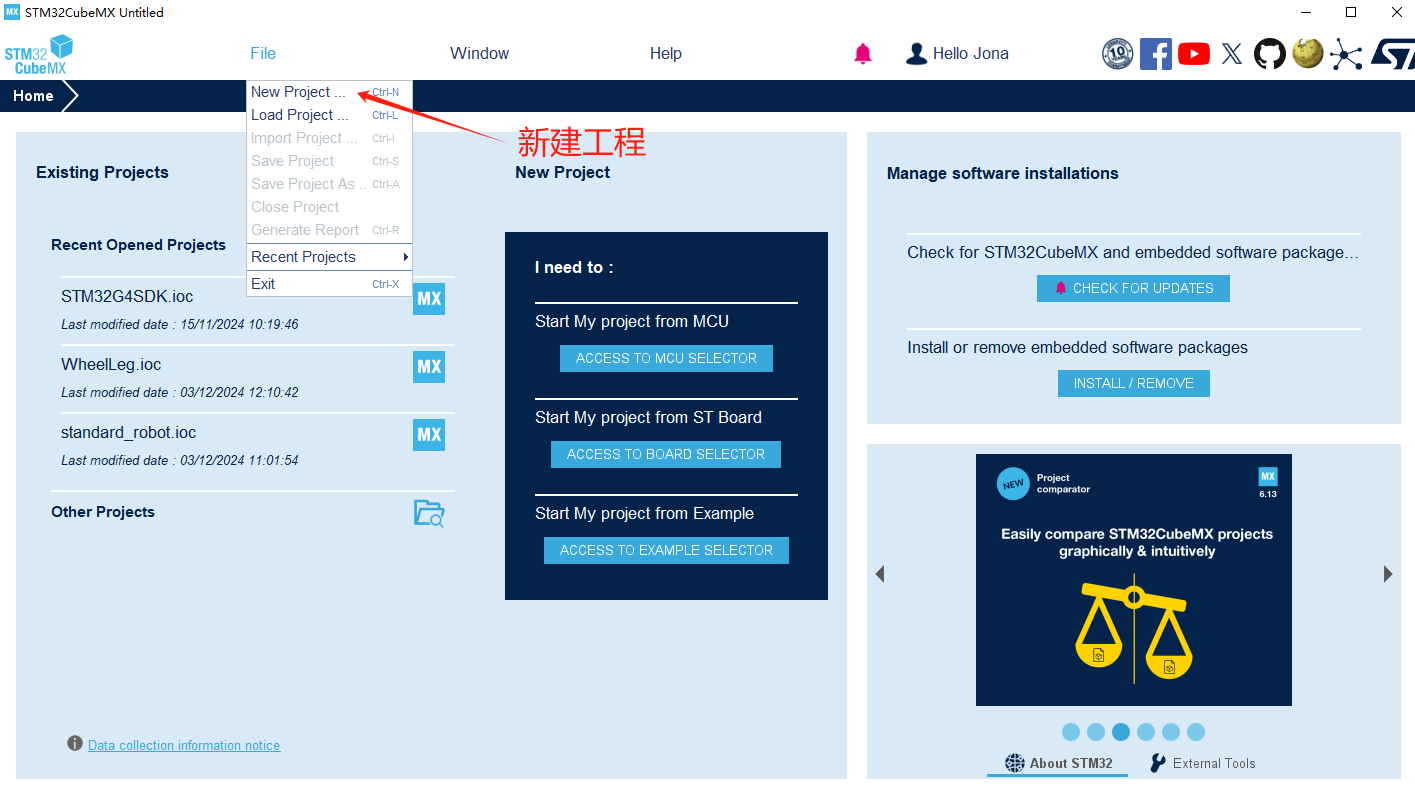
**Guardian Beast DriverSDKUser Manual**

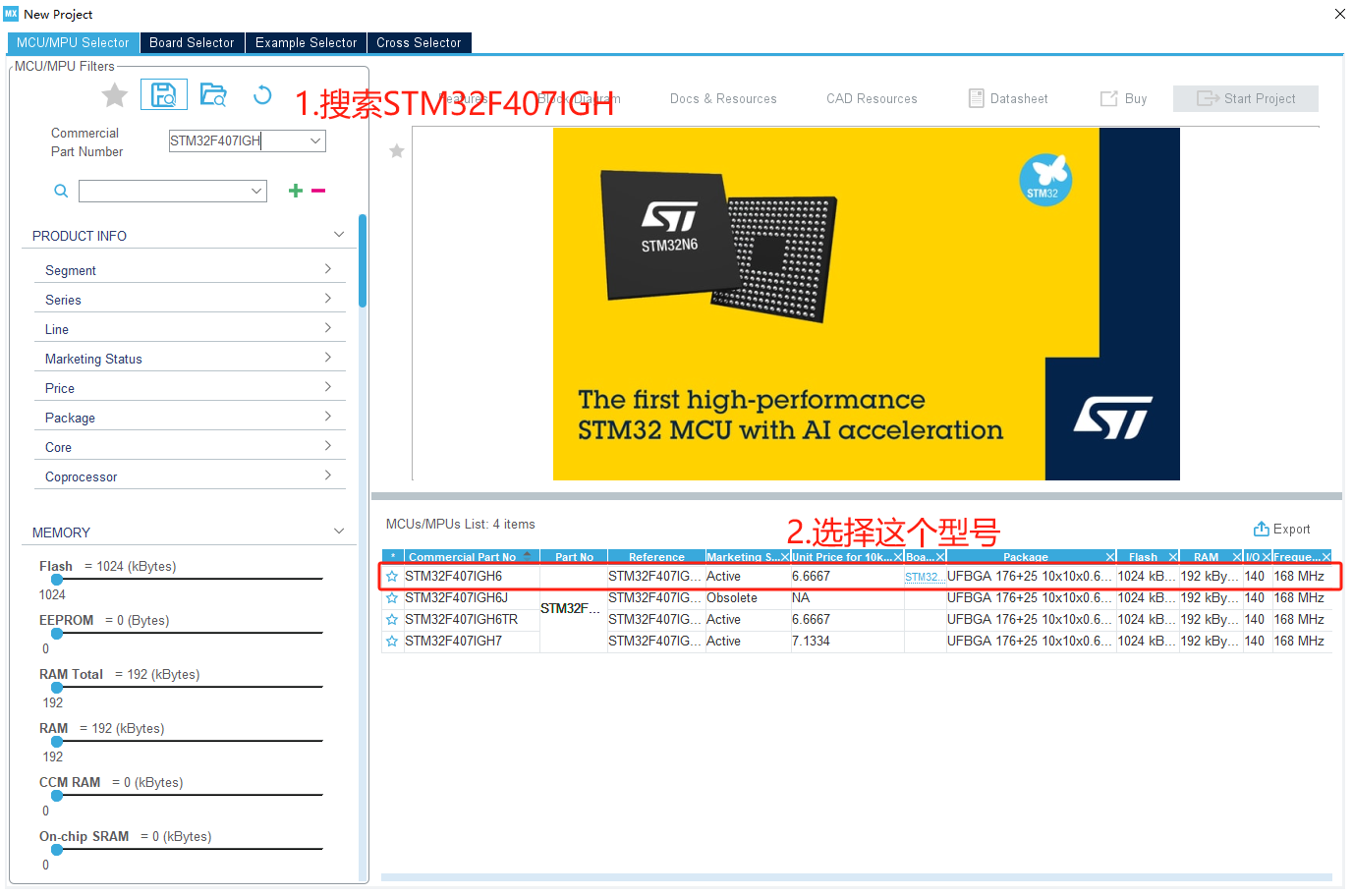
|  |
| --- |
| This instruction uses the DJI C Board STM32F407IG as an example to illustrate some functions of the SDK. The MWMotor.c and.h files used can be used in any system that supports a C language compilation environment, not limited to STM32. Code repository:  [Beijing Shouhoushou Technology Co., Ltd./MWMotorSDK: MW Motor Software Development Kit (SDK)](https://gitee.com/cyberbeast/mwmotorsdk) |

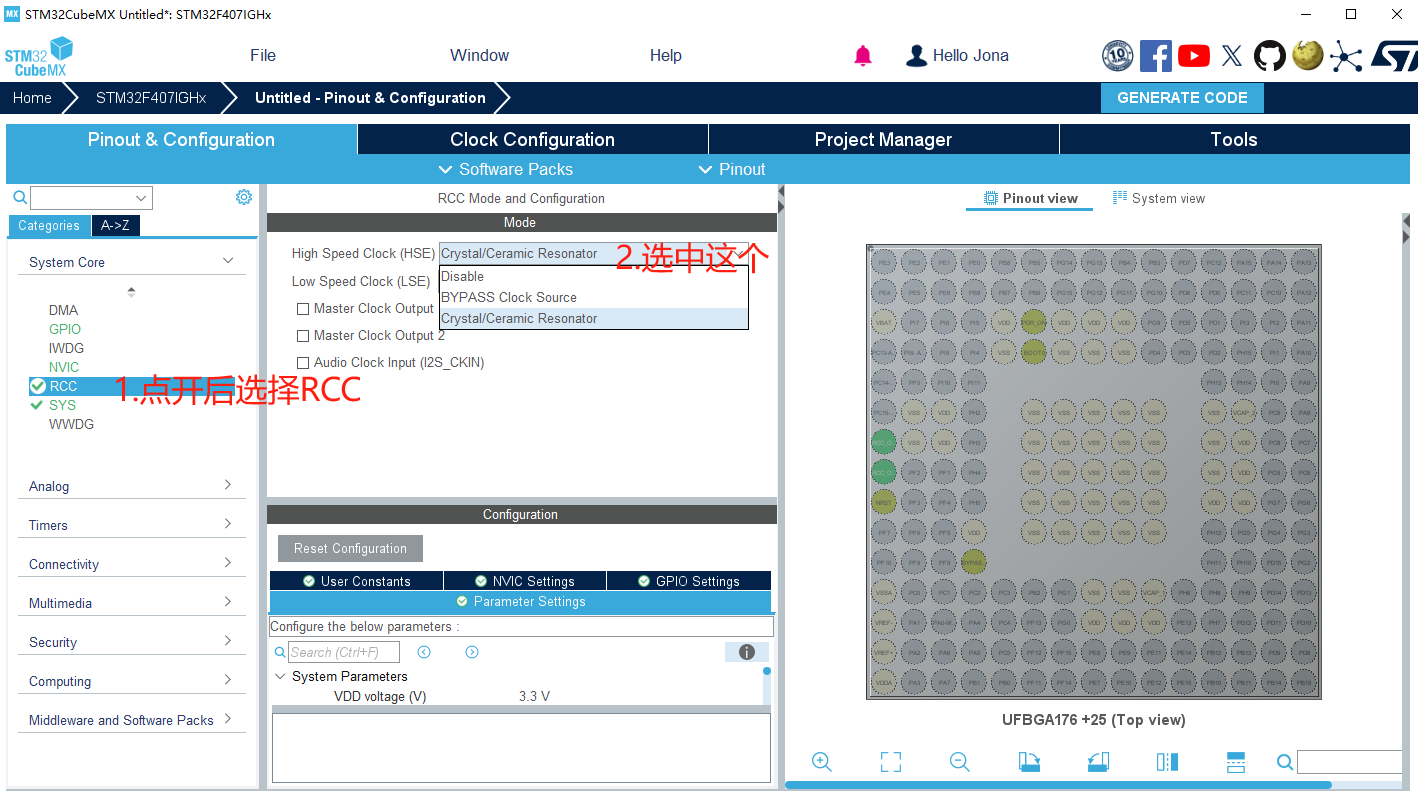
1. **STM32CUBEMX and Keil5 Project Creation**

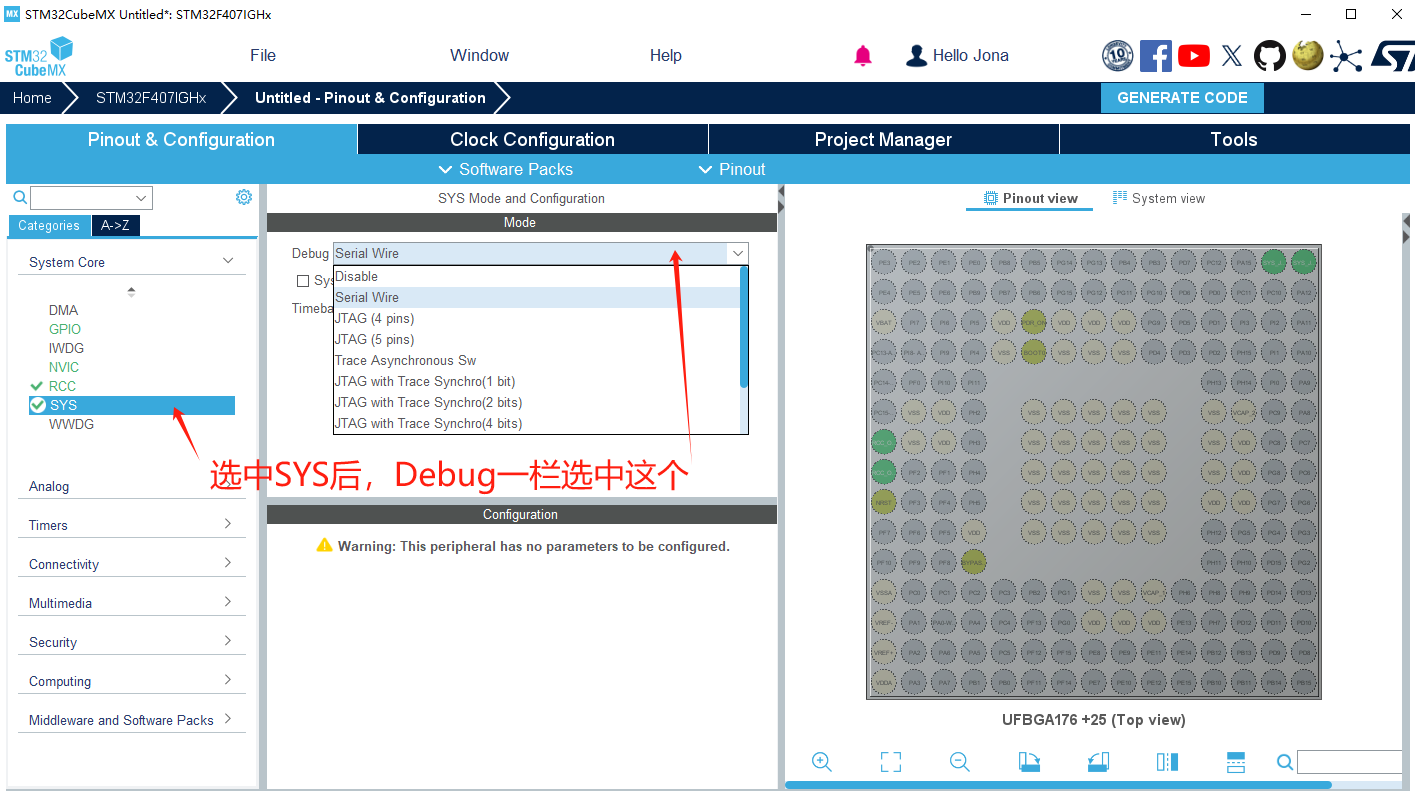
If only transplantation is required, simply copy MWMotor.c.h to your own project directory for compilation. After successful compilation, directly refer to  [Section 2](https://rcneo0sp7vgy.feishu.cn/docx/ZHCmdl10FoFsFQxQBmmcwTLOnwf#doxcnaf926uv6hv5K3XkU959U3d) .

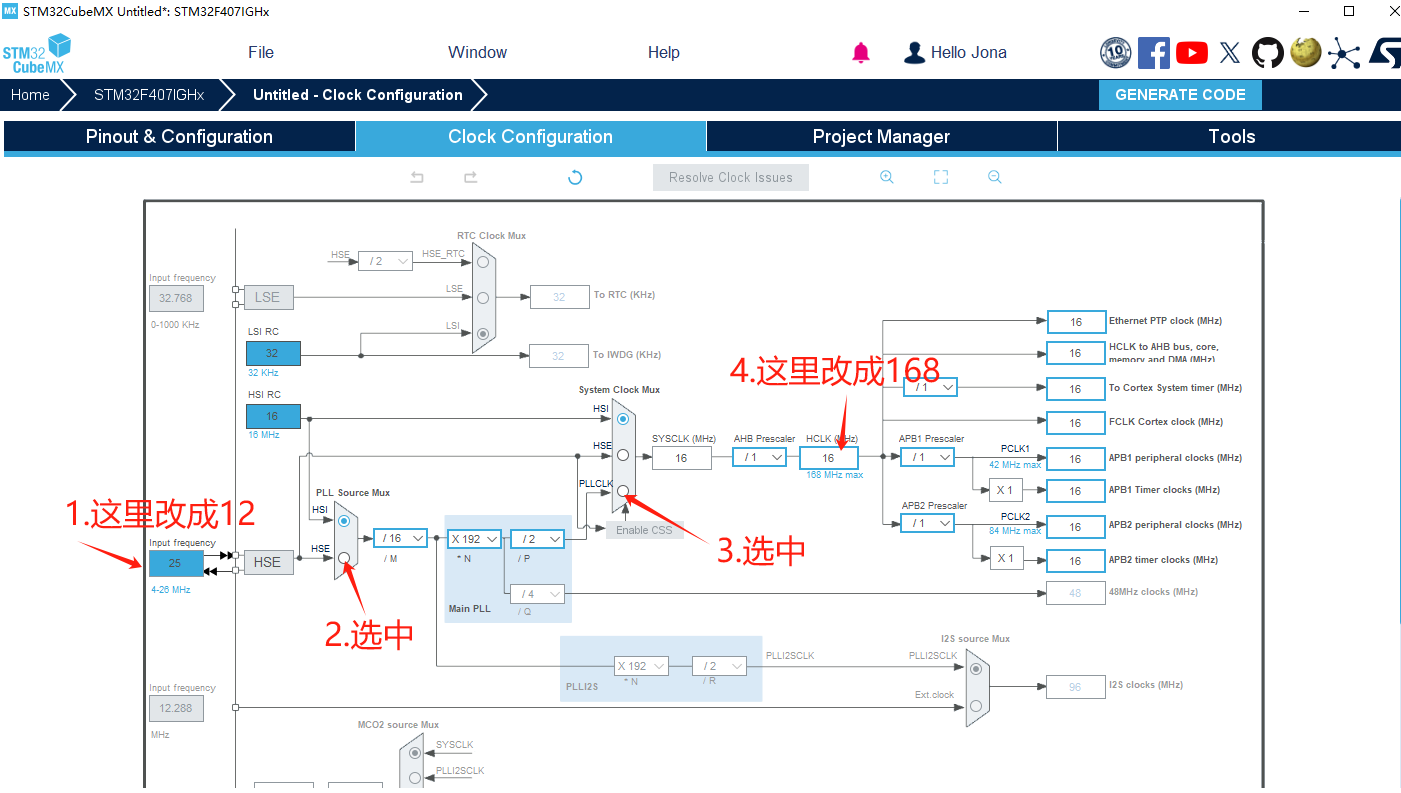
1.1 **Project Creation**

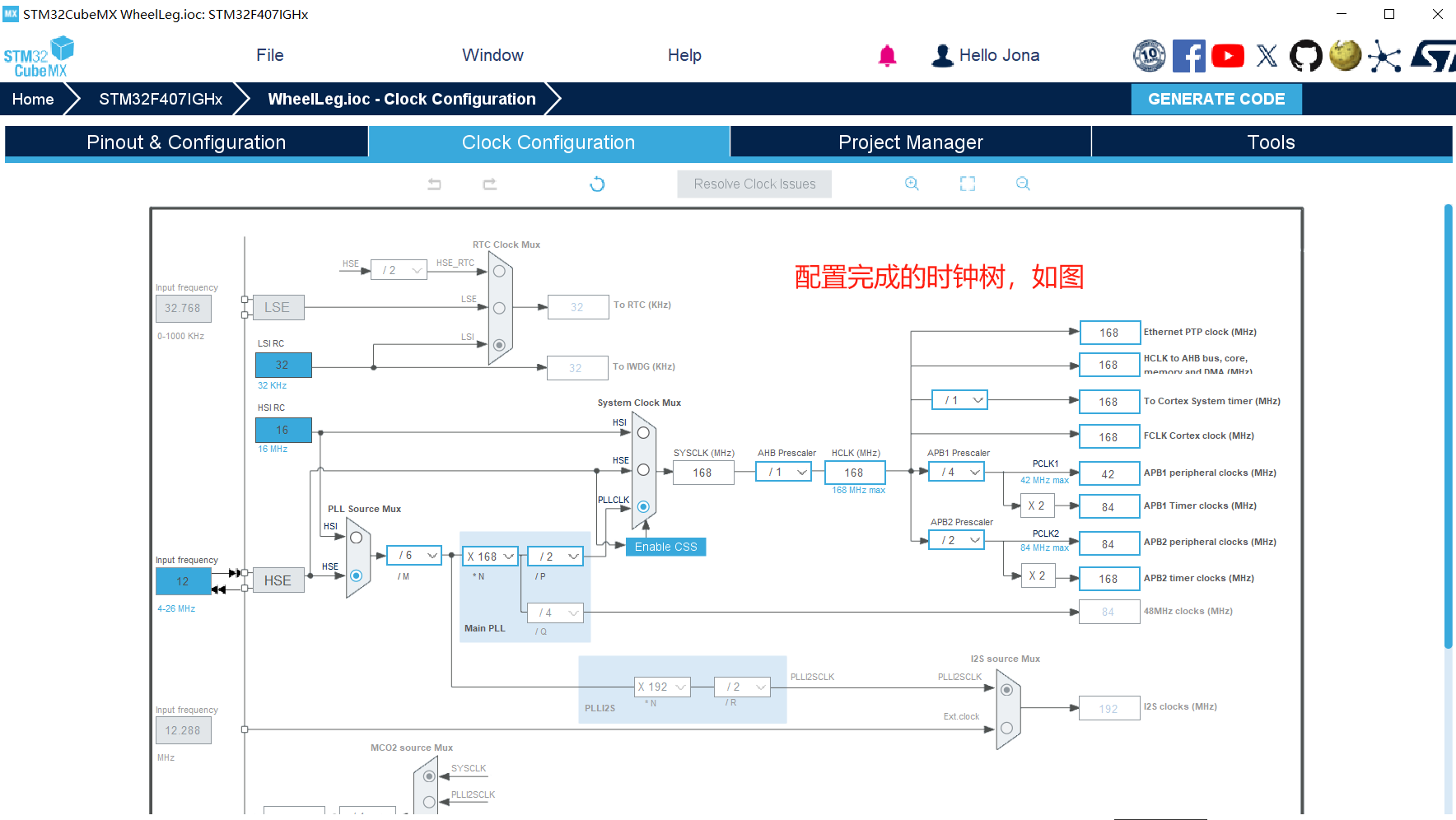


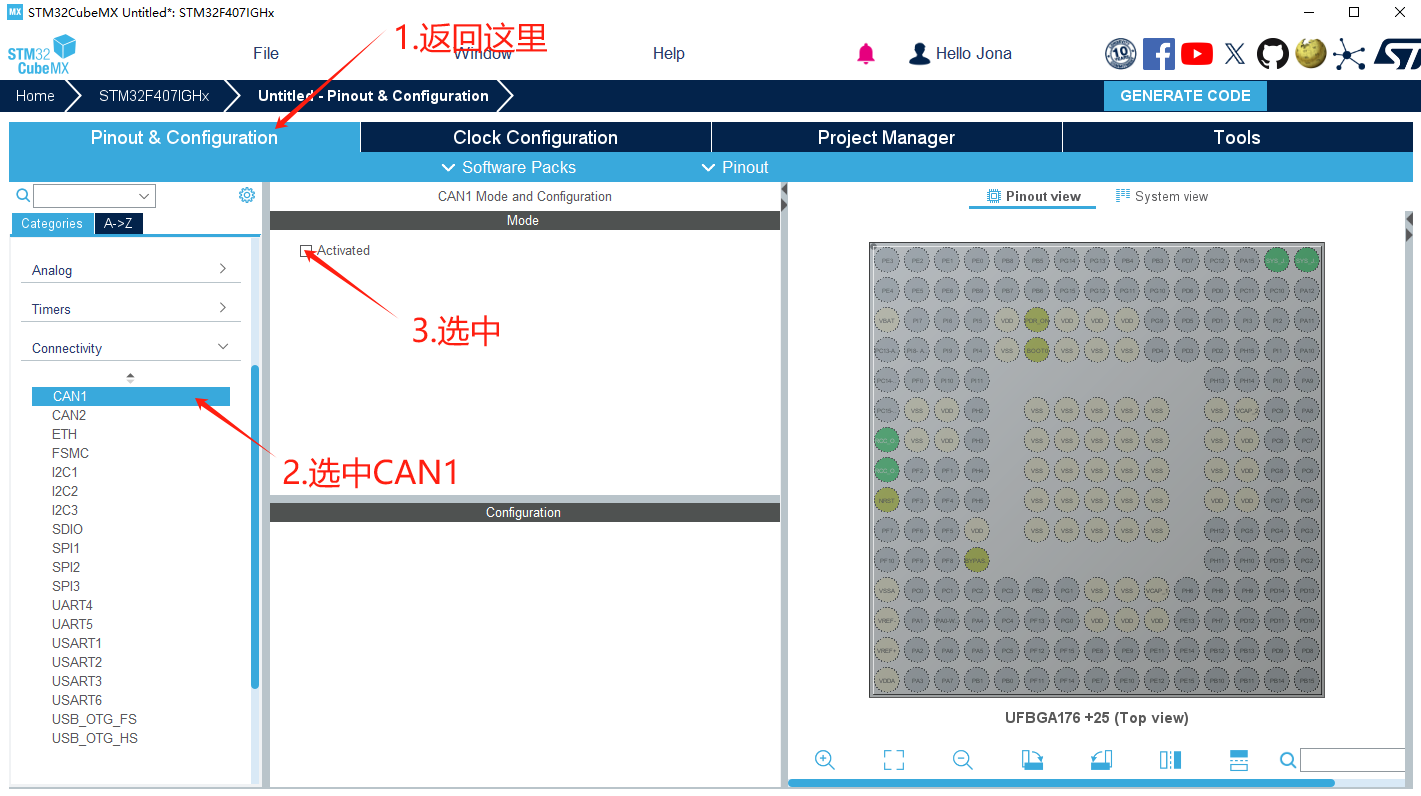


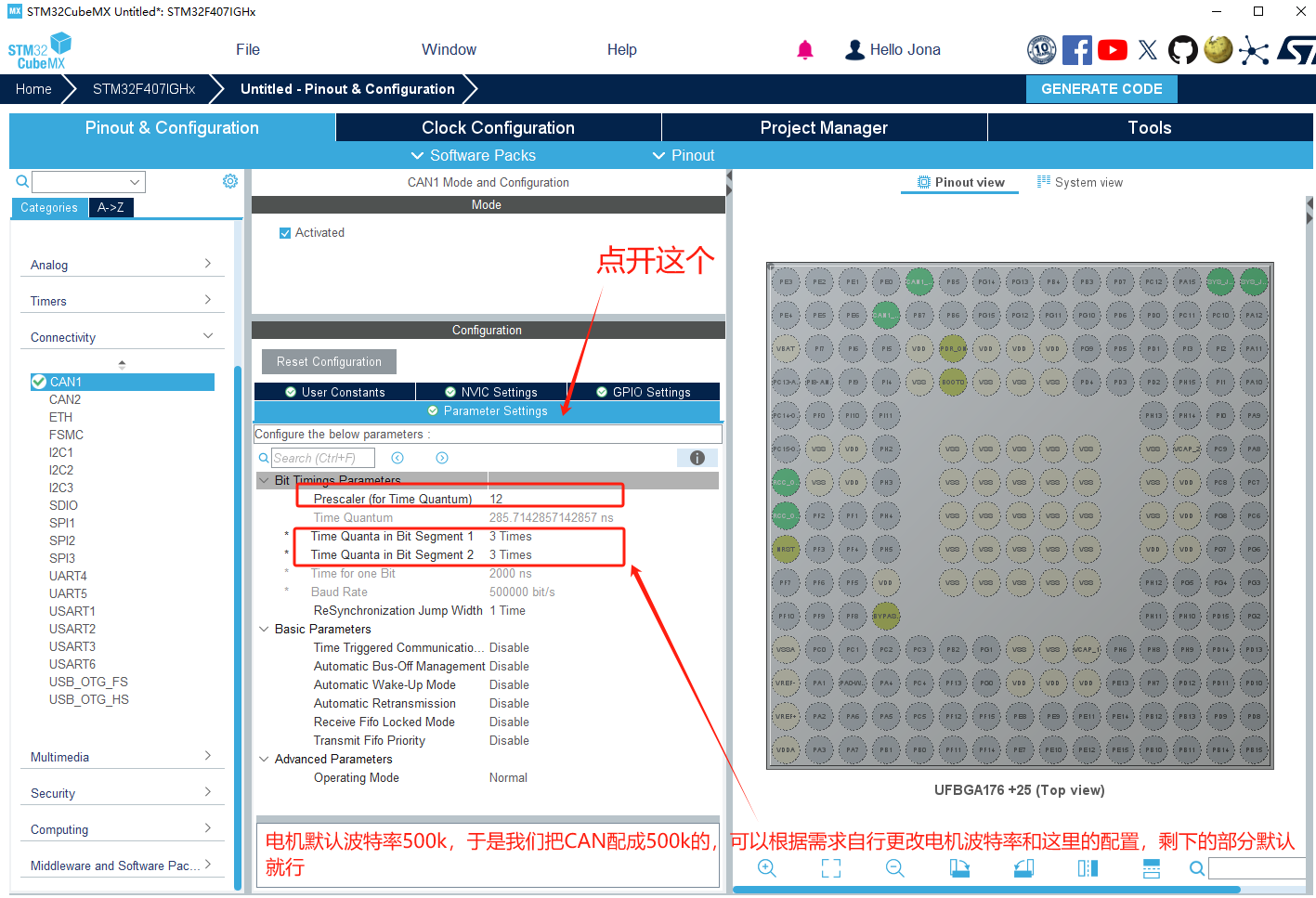


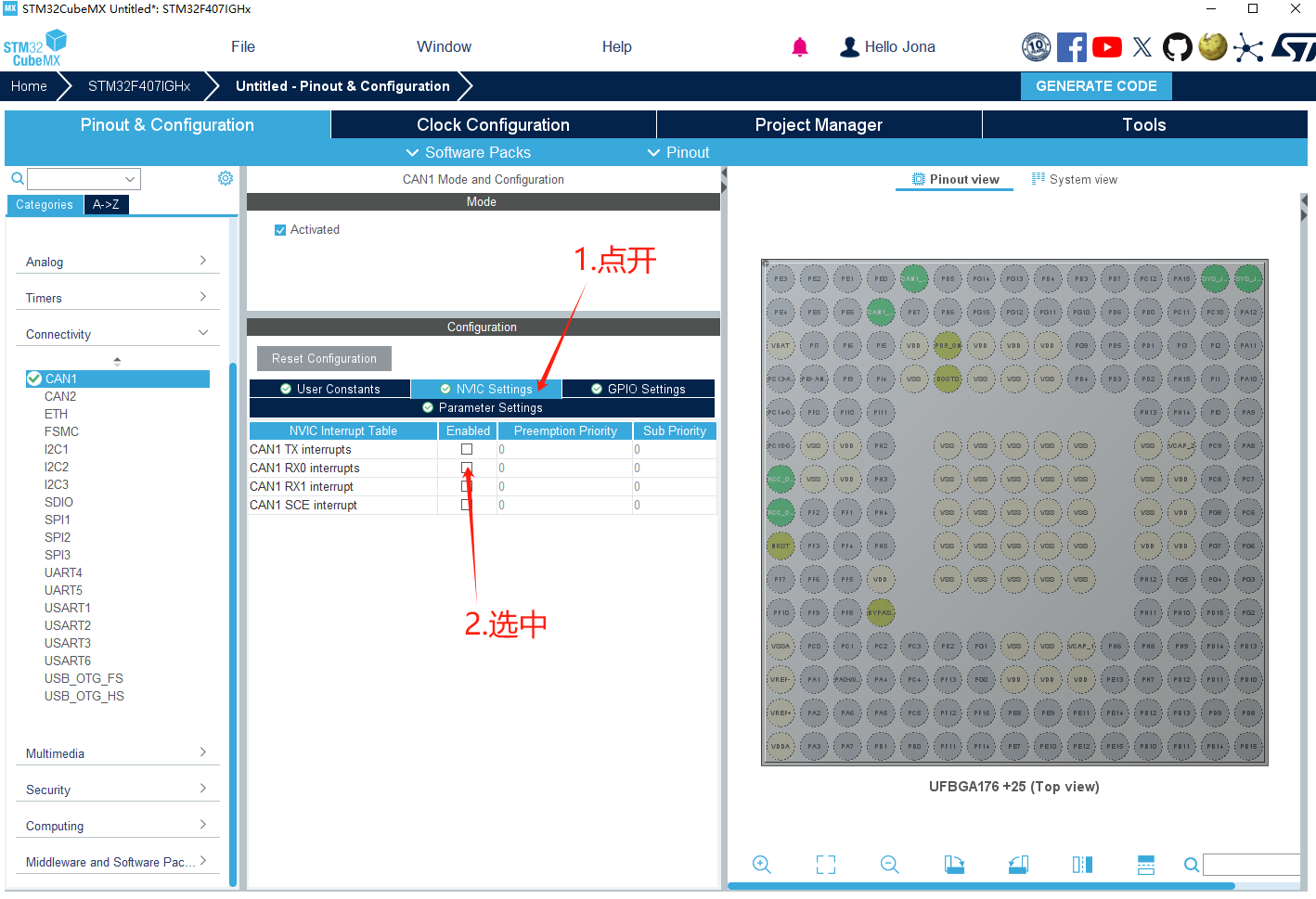


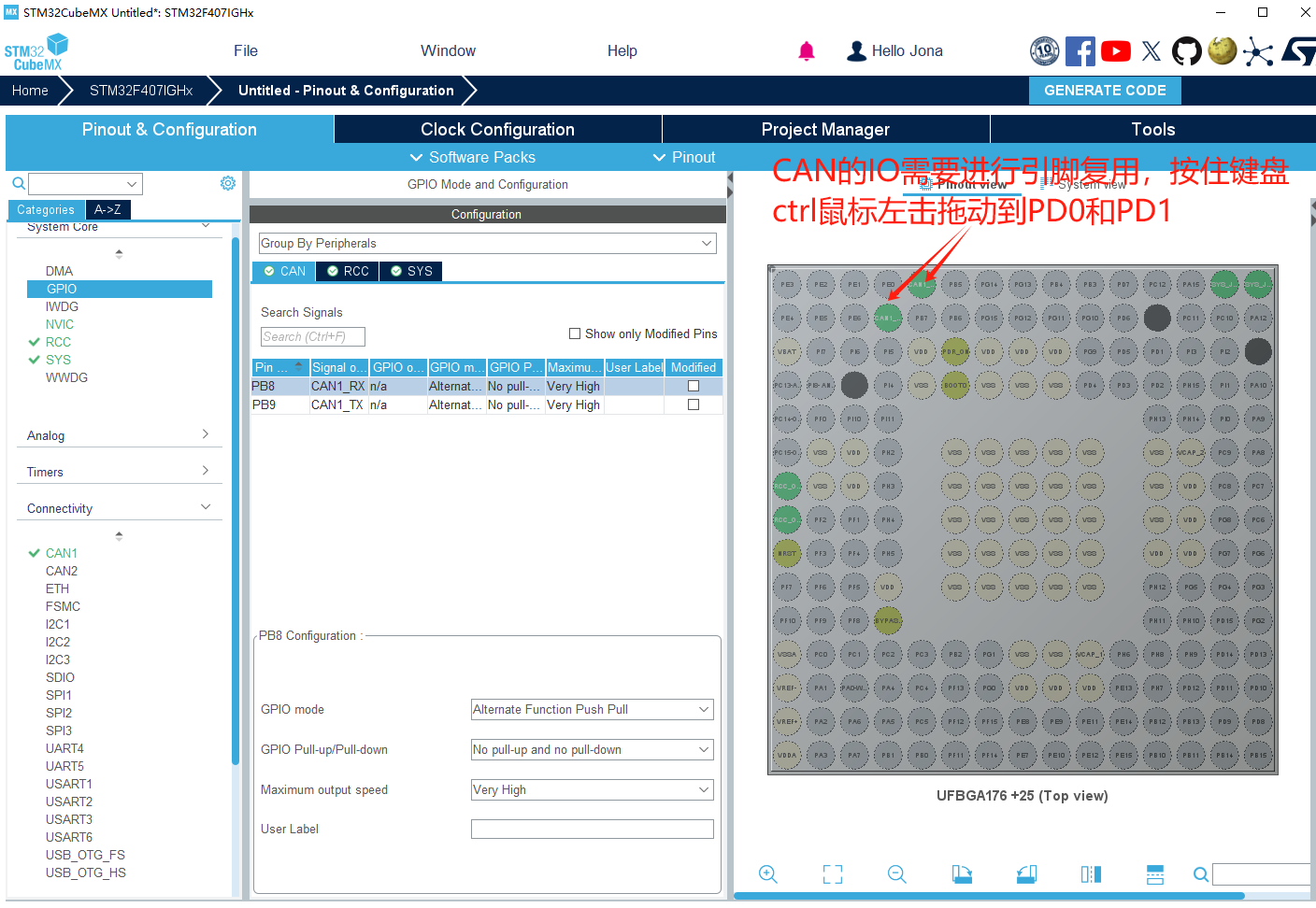


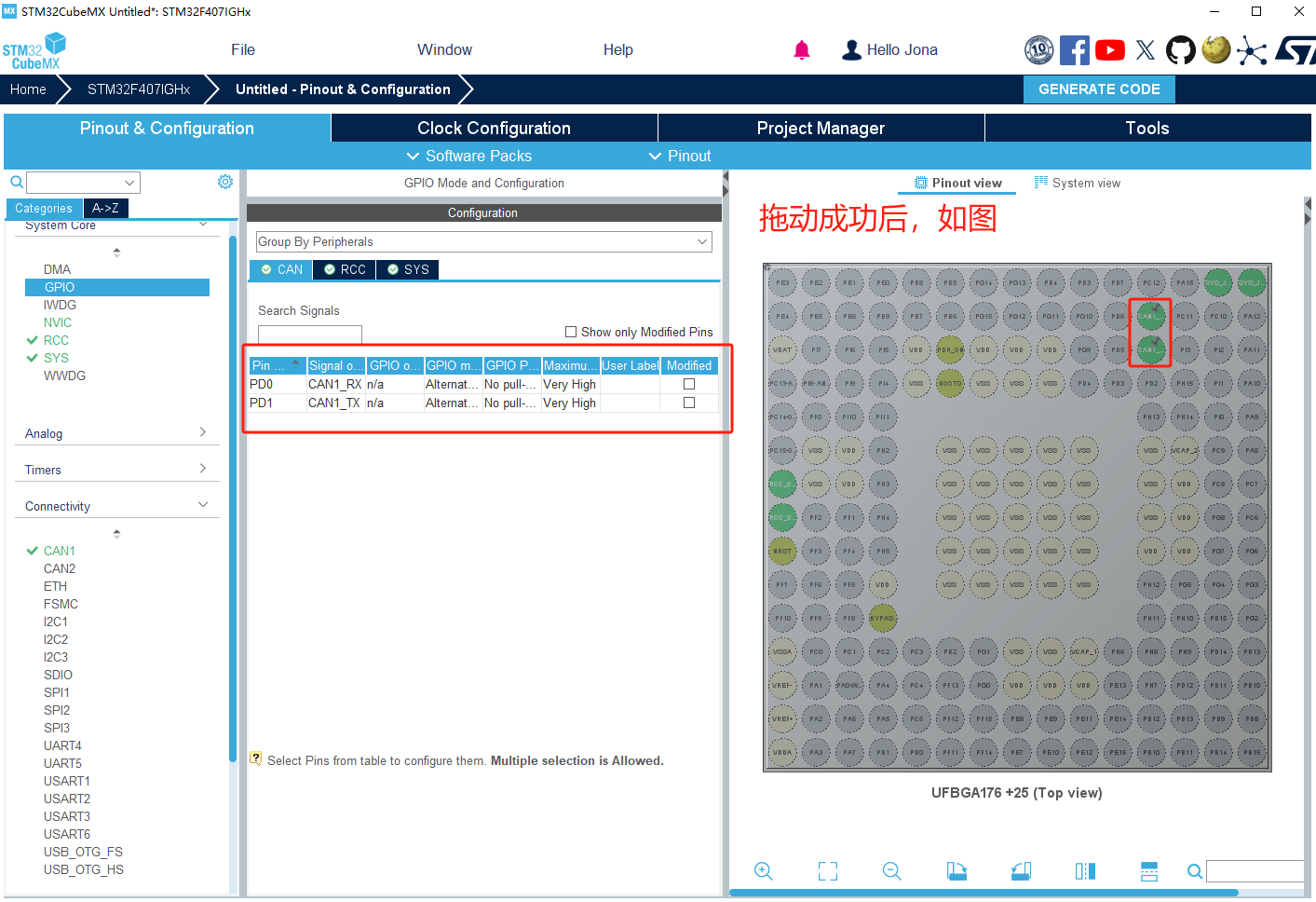


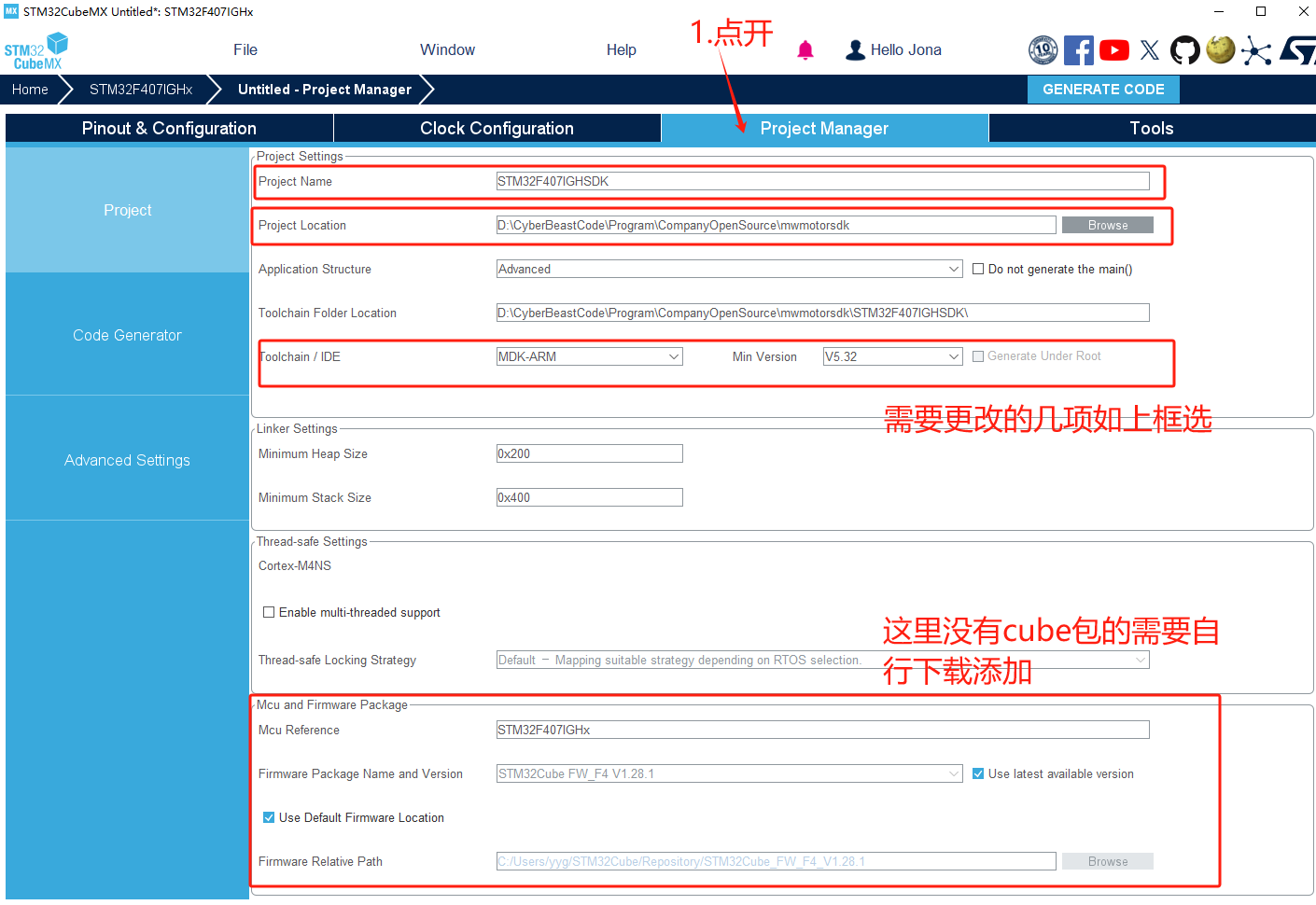


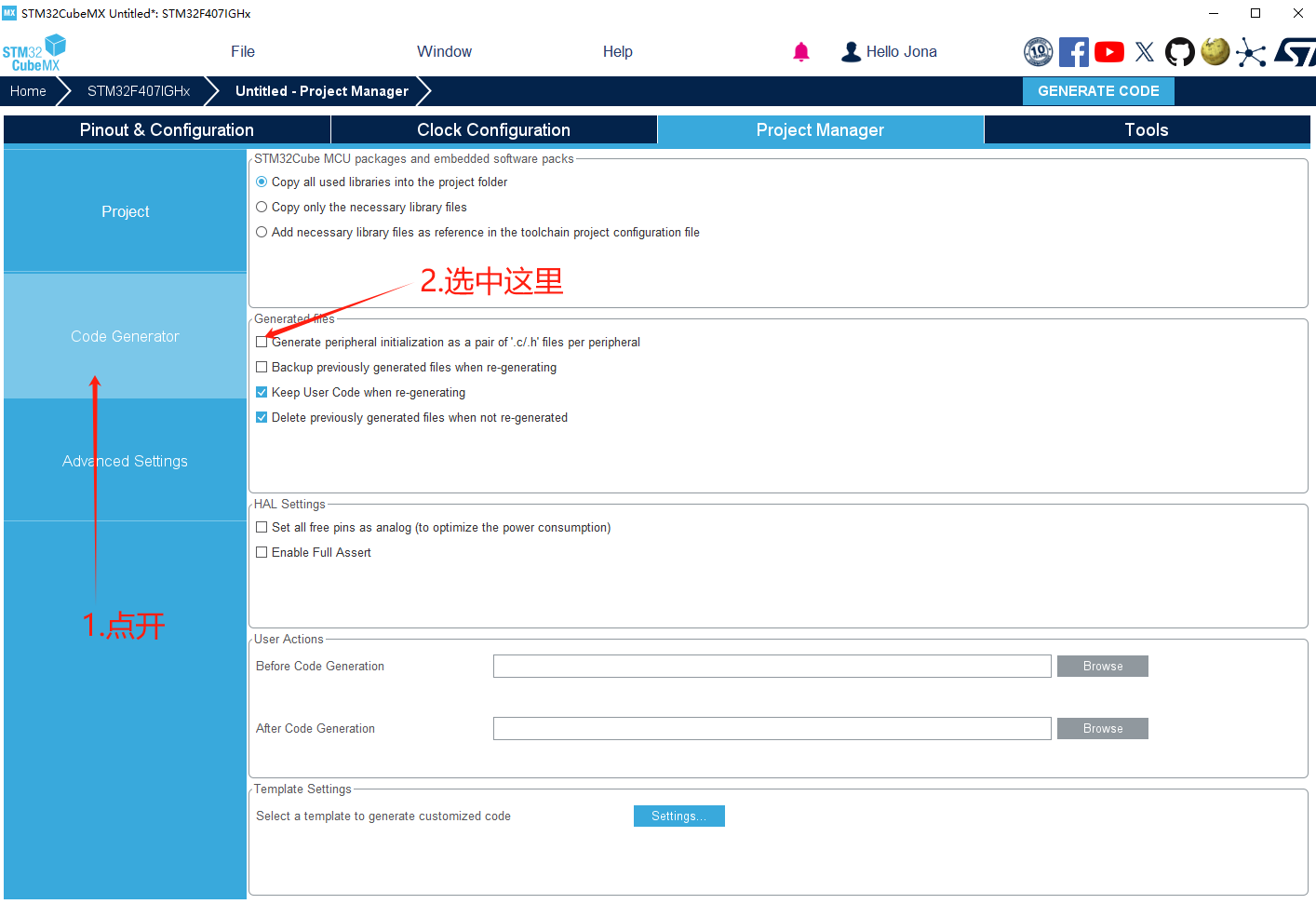


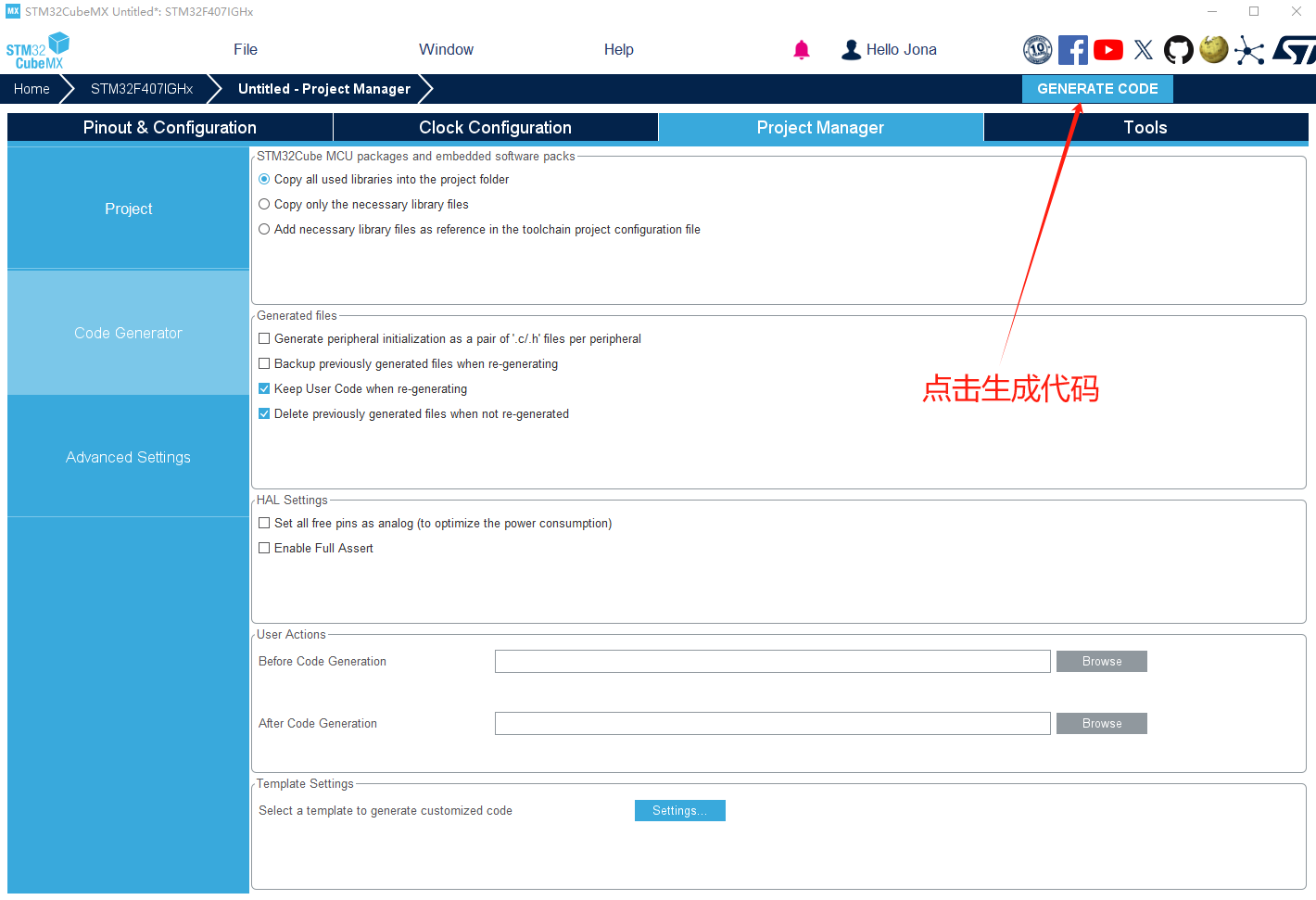


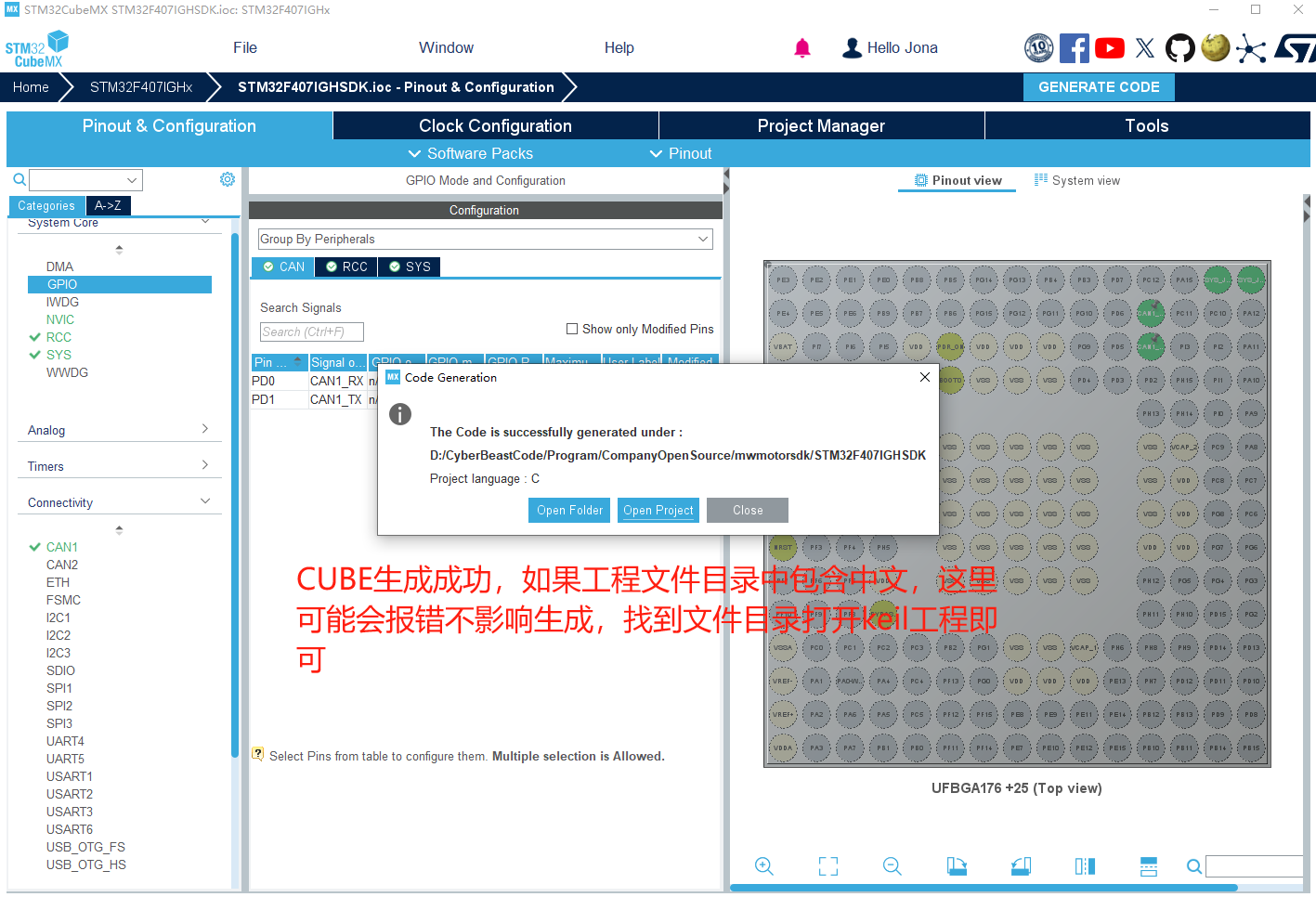




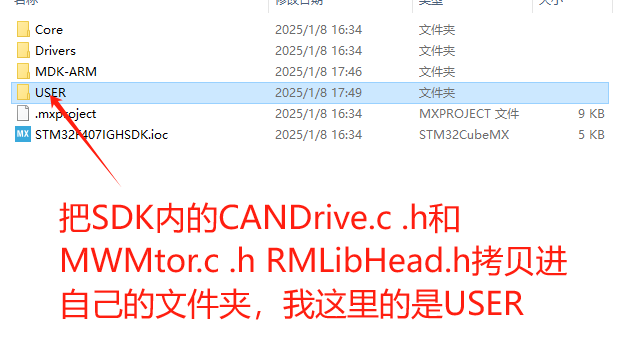


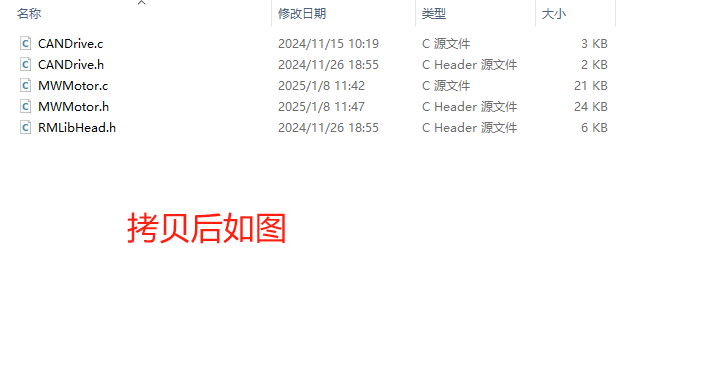




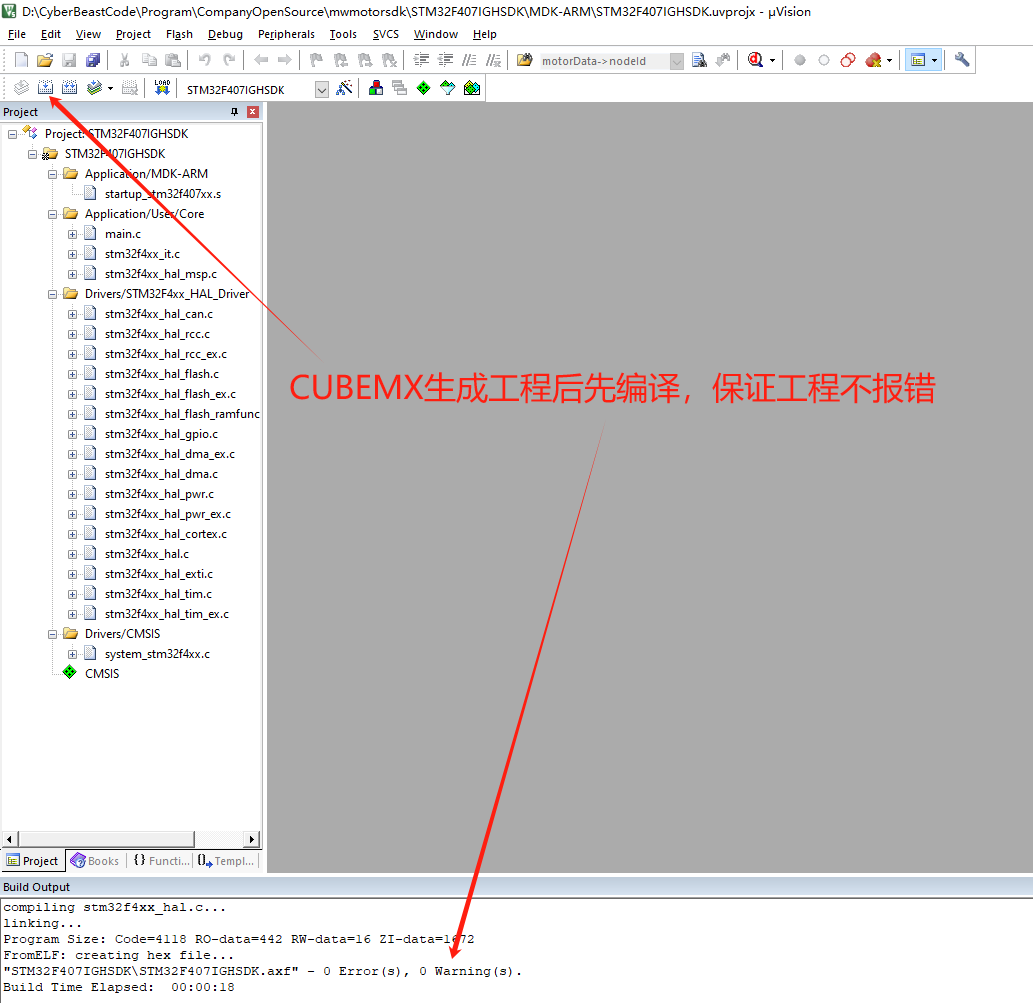


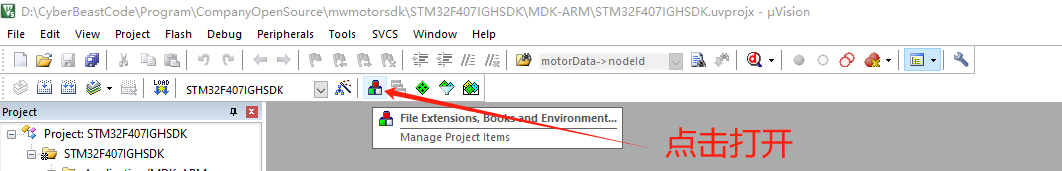
1.2 **Import files into Keil5 and compile them**

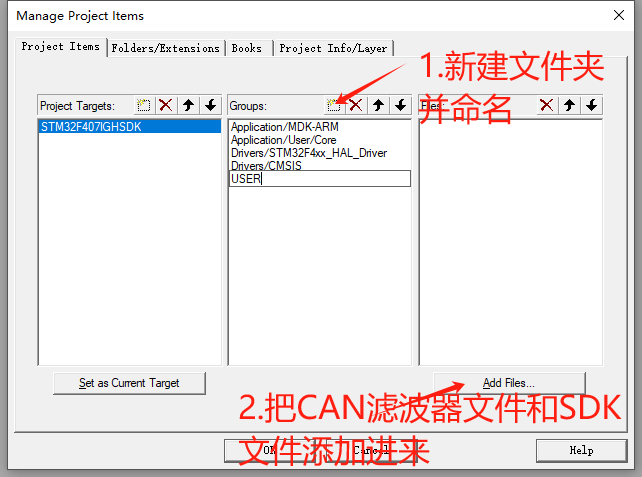


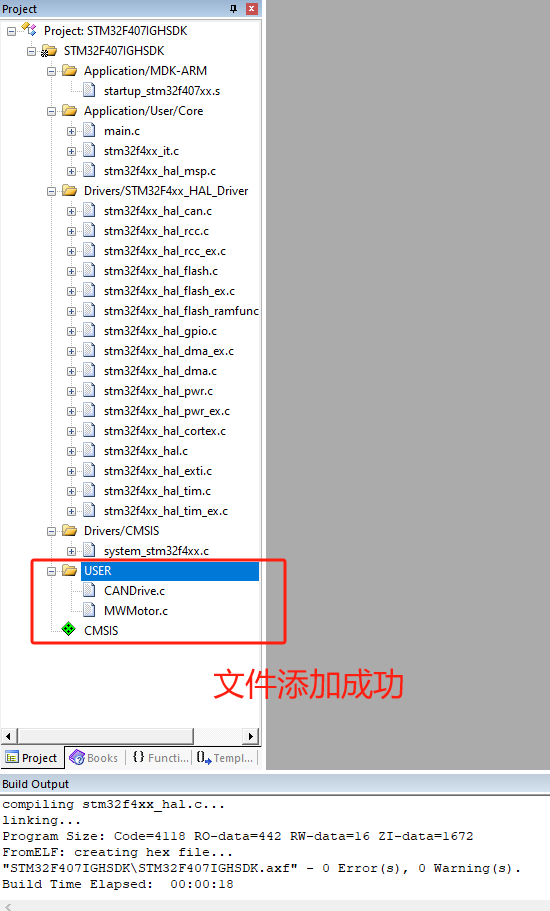


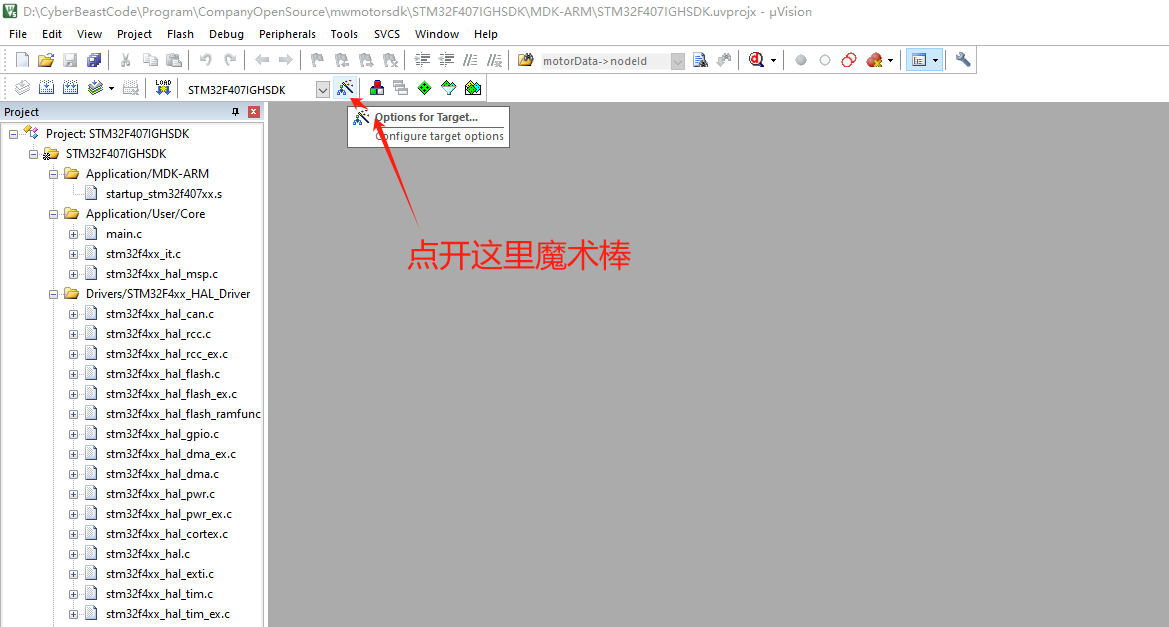
|  |
| --- |
| Note that STM32F407IGH uses CAN, while another project, STM32G431, uses FDCAN. Although the CANDrive.c and CANDrive.h files in the two projects have the same name, their contents are different! |

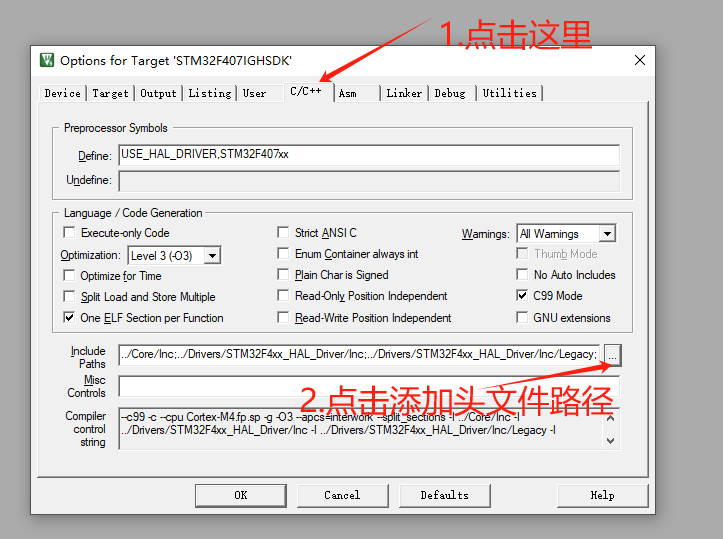


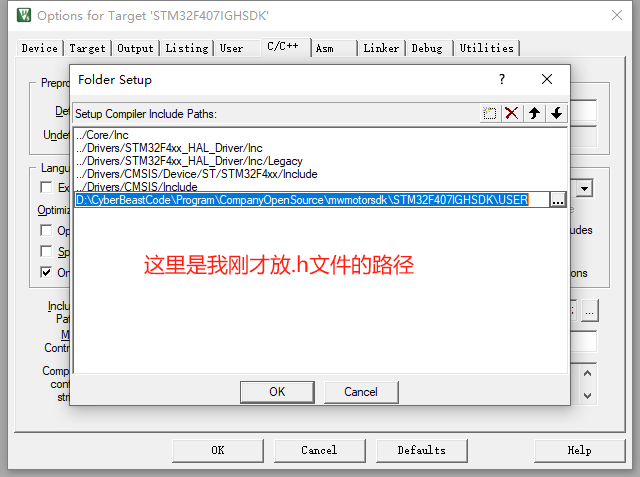


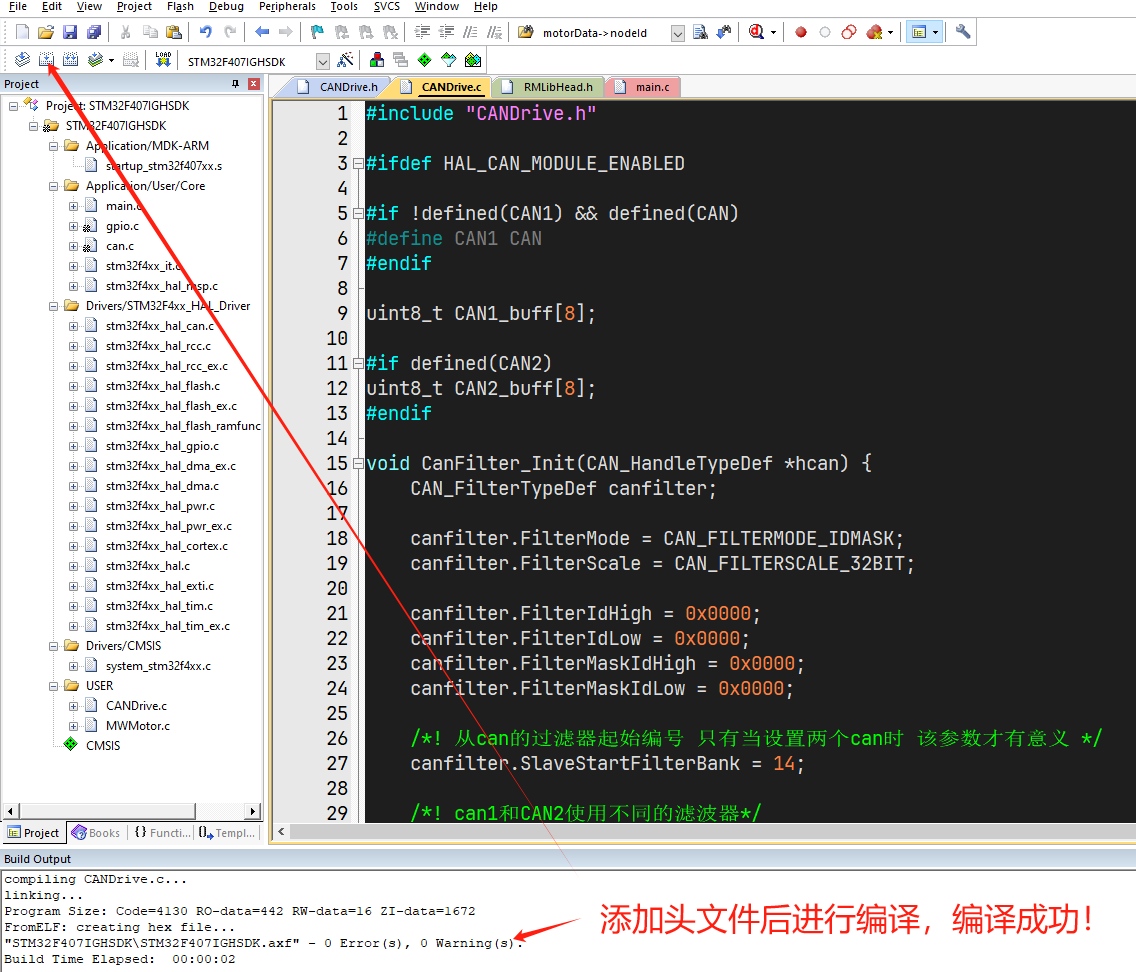












2. **Start Using the Software Development Kit**

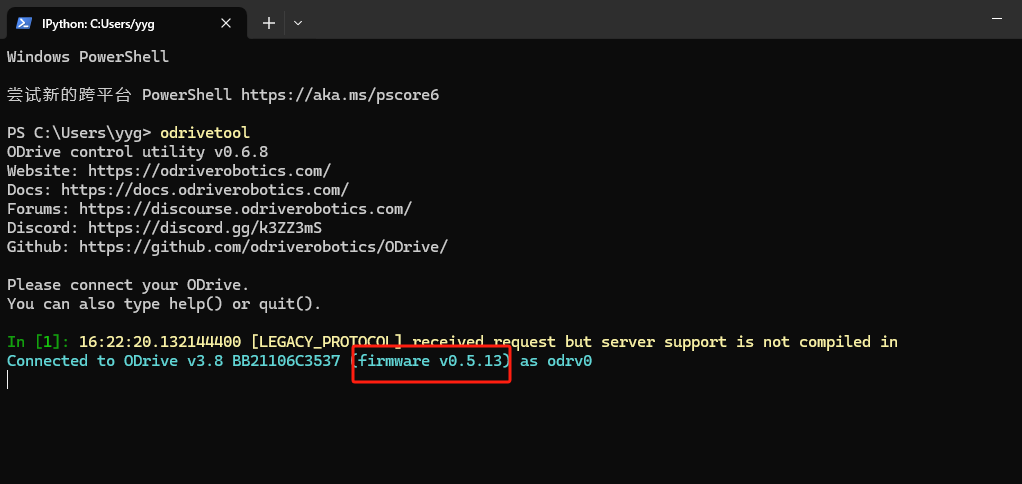
|  |
| --- |
| What variables and functions need to be created when using the Software Development Kit (SDK), how to define the firmware number based on the firmware version, and how to view it? |

2.1 **Modify the firmware version number**

Due to different firmware versions, the heartbeat versions vary. To obtain the correct heartbeat feedback, we need to first obtain the firmware version number and change the firmware number in MWMotor.h.

2.1.1 **odrivetool checks the firmware version**

Install odrivetool correctly according to the instruction manual, ensure the motor power light is blue, connect the motor with a Type-C cable, open the end point, and execute the command odrivetool

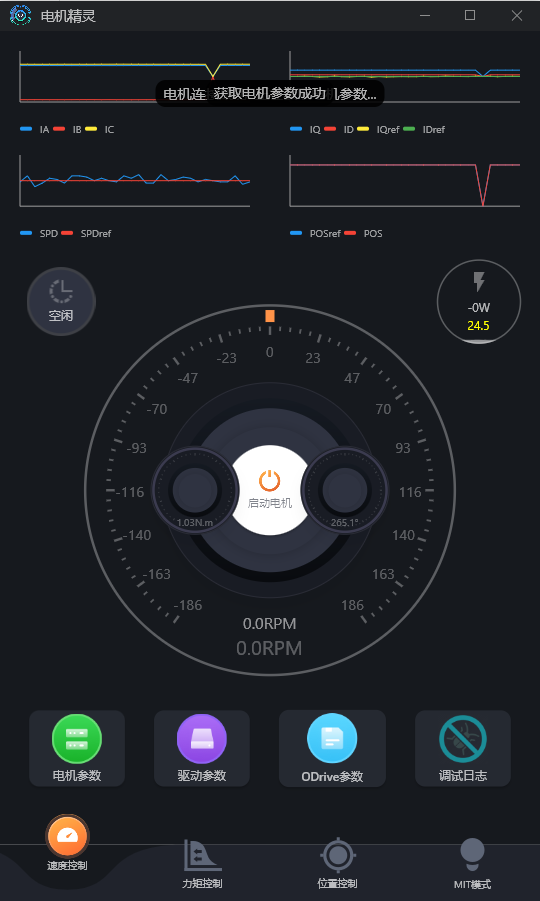


As shown in the figure, the one in the red box is the motor firmware version 0.5.13, and the previous V3.8 is the hardware version number

|  |
| --- |
| Open the Header File MWMotor.h in the SDK. As mentioned above, our firmware version is 0.5.13, so the corresponding firmware version number is 513, as shown in the figure below    If the firmware version is 0.5.12, the firmware version number is 512, and so on. |

2.1.2 **Motor Genius checks firmware version**

After correctly installing Motor Genius according to the instruction manual, open Motor Genius, wait for the computer to successfully connect to the motor, and then it will display that parameter acquisition was successful



When you open the motor parameters of the green icon, you will see that the firmware version identified by the red box in the figure below is 0.5.13, and the hardware version is 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 **Create motor structure**

After completing the above steps, open Keil and define two global structures at any location

|  |
| --- |
| 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**

This type of structure is used to receive the feedback data from the motor and does not need to be initialized

**MW\_MOTOR\_ACCESS\_INFO**

This type of structure is used to store the usage information of the motor

**busId**

It is the bus number to which the motor belongs (users can distinguish the bus by themselves. If they do not know how to use it, they can make all motors on one bus number).

**nodeId**

It is the ID number of the motor. After configuring the nodeId of the motor on the upper computer, the user needs to write the configured ID here

* Set the nodeId in odrivetool

Execute this command in odrivetool and then save

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

* Set the motor nodeId in Motor Wizard

Open the Motor Wizard, click on Motor Parameters,**please synchronize the parameters when the motor is in an idle state**



**motorData**

Pass the pointer of theMW\_MOTOR\_DATAstructure just created here.

**sender**

The SDK send function. Since the CAN send function varies for each system and the CAN send function needs to be used in the SDK, a function pointer type is defined in the SDK to achieve compatibility with different control systems. Users need to define a function themselves, and we have already fixed the function format for this function:

|  |
| --- |
| 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); |

The function name is arbitrary, and the parameters that need to be passed in need to include uint8\_t type of busId bus ID, uint8\_t type of canId , uint8\_t \* CAN send array pointer, uint8\_t type of dataSize data size.

**Within this defined function, simply call the CAN sending function of your own system**.

|  |
| --- |
| 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 \*/ } |

For example, on the HAL library of SMT32:

|  |
| --- |
| 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);  } } |

Then  **pass the pointer of this created function to the structure for initialization.**

**notifier**

The SDK notification function, like the sending function above, defines a function pointer type to achieve compatibility with different control systems, as follows:

|  |
| --- |
| 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); |

A function of the same type needs to be defined:

|  |
| --- |
| 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; } |

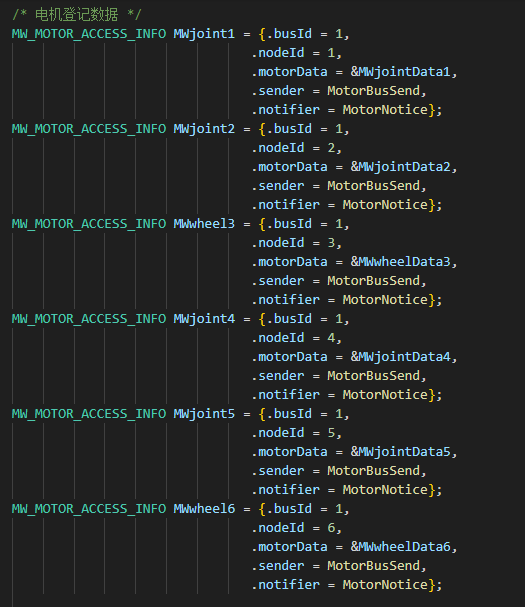
If this function is not needed, you can define an empty function and pass it to the structure for initialization.

**2.3 Call the function to register motor information**

Find the main function of our system operation, and call the motor information registration function after the system CAN initialization is completedMWRegisterMotor()to register the motor we just created

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

If there are multiple motors, multiple structures need to be defined and the registration function called, for example:





**2.5 Data Reception**

CallMWReceiver() function to receive motor data, where the motor's busId, canId, and the pointer to the array received by CAN need to be passed in.

For example, in the STM32 example routine, I call the implemented data reception in the callback function of the HAL library:

|  |
| --- |
| 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);  } } |

|  |
| --- |
| At this point, all motor configurations have been completed! |

3. **Function Details**

**Function Overview**

|  |  |
| --- | --- |
| Name | Function |
| MWRegisterMotor | Register motor information |
| MWReceiver | Receive motor feedback data |
| MWEstop（0x002） | Emergency stop the motor |
| MWGetMotorError（0x003） | Get motor error information |
| MWRxTxSdo（0x004 0x005） | Access or write to other functions of the motor using endpoint\_id |
| MWSetAxisNodeID（0x006） | Set the node ID of the motor |
| MWSetAxisState（0x007） | Set motor status |
| MWMitControl（0x008） | MIT Motion Control |
| MWGetEncoderEstimates（0x009） | Obtain the speed and position information of the encoder |
| MWGetEncoderCount（0x00A） | Get encoder count function, which retrieves the encoder's count information from the motor controller |
| MWSetControllerMode（0x00B） | Set the control mode and input mode of the motor controller |
| MWPosControl（0x00C） | Position control, input motor position, velocity feedforward, and torque feedforward |
| MWVelControl（0x00D） | Speed control, input motor speed and torque feedforward |
| MWTorqueControl (0x00E) | Torque control |
| MWSetLimits（0x00F） | Set motor speed limit and limit viewership of |
| MWStartAnticogging（0x010） | Perform torque ripple calibration |
| MWSetTrajVelLimit（0x011） | Set the speed limit for trapezoidal curve position control |
| MWSetTrajAccelLimits（0x012） | Set the acceleration limit for trapezoidal curve position control |
| MWSetTrajInertia（0x013） | Set the inertia for trapezoidal curve position control |
| MWGetIq（0x014） | Get the current value of the node |
| MWReboot（0x016） | Motor Restart |
| MWGetBusVoltageCurrent（0x017） | Obtain motor voltage and current information |
| MWClearErrors（0x018） | Clear all errors |
| MWSetLinearCount（0x019） | Set encoder absolute position (count value) |
| MWSetPosGain（0x01A） | Set the P-value of the PID control for the motor position loop |
| MWSetVelGain（0x01B） | Set the P-value and I-value of the speed loop PID control |
| MWGetTorques（0x01C） | Obtain the torque information of the motor |
| MWGetPowers（0x01D） | Obtain motor power data |
| MWDisableCAN（0x01E） | Disable CAN |
| MWSaveConfigeration（0x01F） | Save motor settings and restart the motor |

4. **Reference Materials**

Routine Link:  [Beijing Shouhoushou Technology Co., Ltd./MWMotorSDK](https://gitee.com/cyberbeast/mwmotorsdk)

Instruction Manual Document:

[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)