7. CAN Bus Communication

7.1. Experimental purpose

Using STM32 CAN communication, using loop mode, key control to send CAN data, interrupt receiving CAN data and print out through serial assistant.

7.2. Configure pin information

Since each new project needs configuration information, it is troublesome. Fortunately, STM32CubeIDE provides the function of importing.IOC file, which can help us save time.
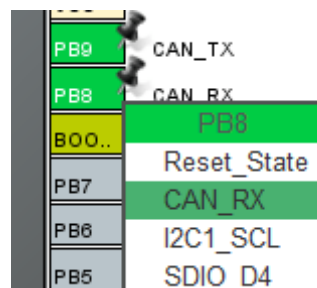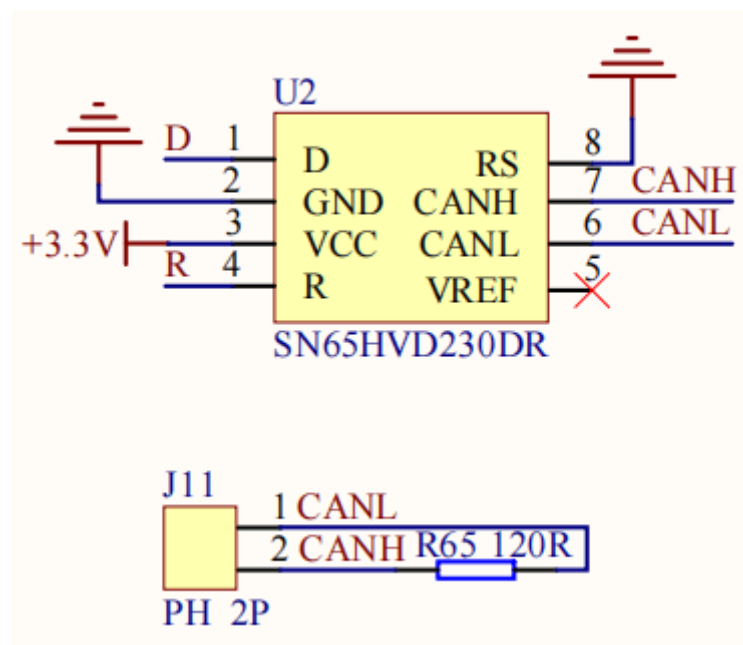
1. Import ioc file from Serial project and name it CAN.

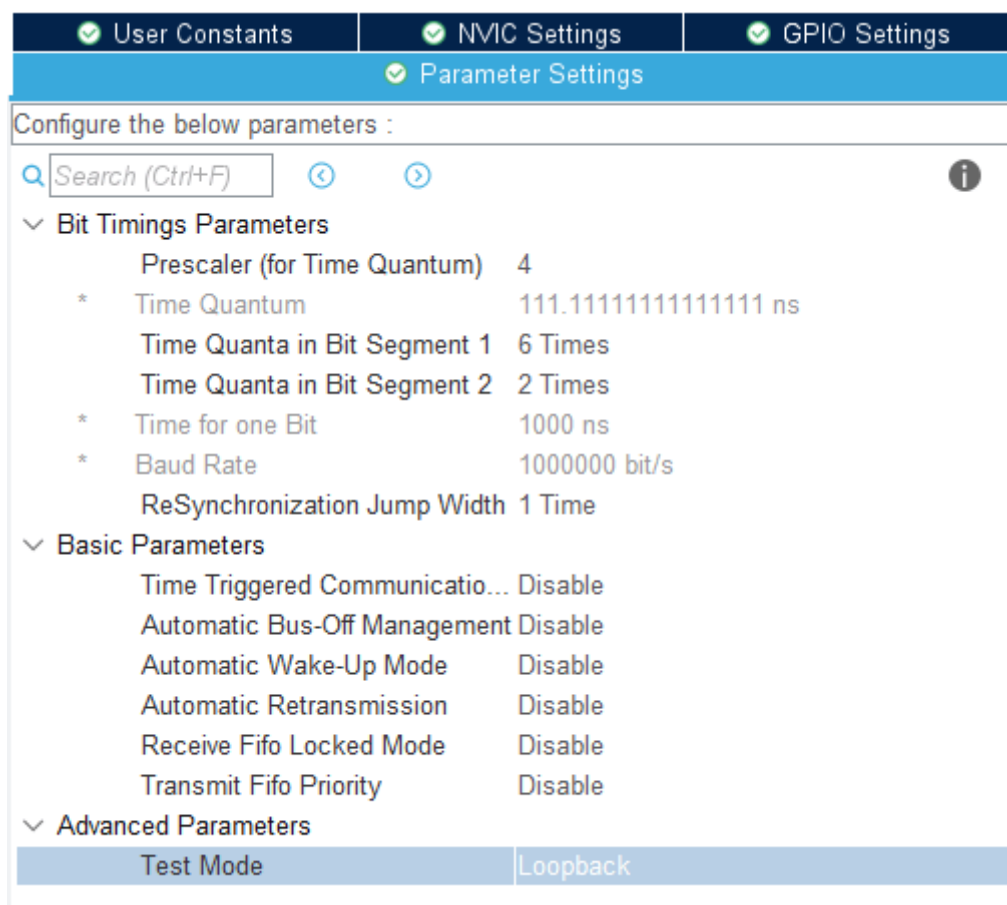Find CAN in Connectivity and check Activated to enable the CAN peripheral.
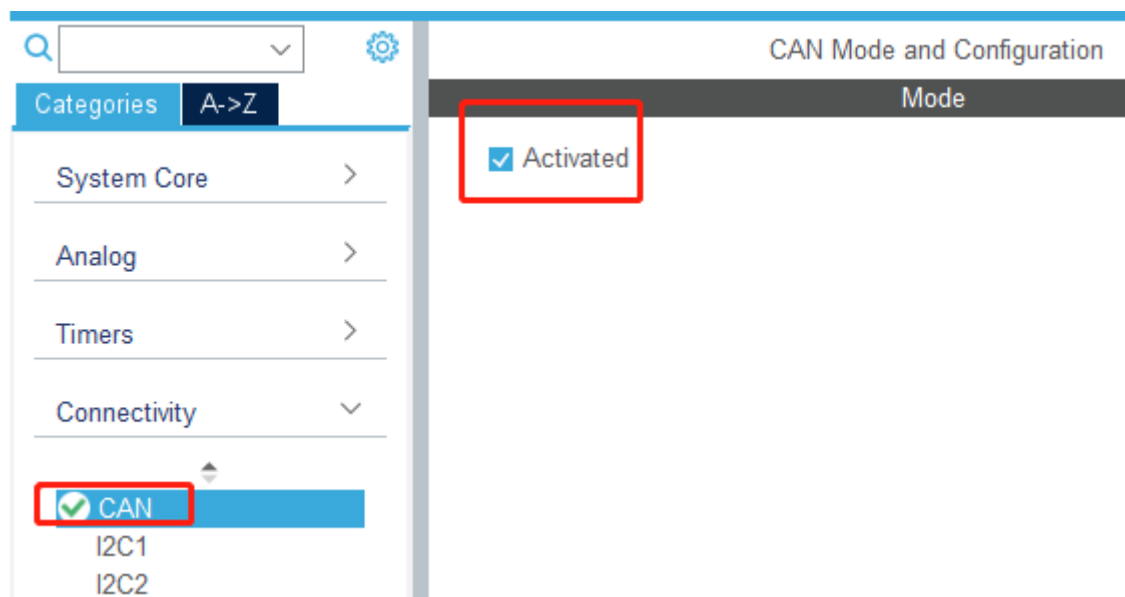
2. According to the schematic diagram, the pins of the CAN bus connection are PB8 and PB9, while the default CAN bus pins are PA11 and PA12, so it is necessary to manually modify the pins of the CAN bus to PB8 and PB9.

U2
SN65HVD230DR

J11
1 CANL
2 CANH R65_120R
PH 2P

PB9 CAN_TX
PB8 CAN_RX
BOO..
PB7
PB6
PB5

PB8
Reset_State
CAN_RX
I2C1_SCL
SDIO_D4

3. Set the parameters of the CAN peripheral. Here, we set the baud rate to 1000kbps and the mode to Loopback.

✔ User Constants	✔ NVIC Settings	✔ GPIO Settings

✔ Parameter Settings

Configure the below parameters :

Q Search (Ctrl+F)   ⊙   ⊙   ⓘ

∨ Bit Timings Parameters

Prescaler (for Time Quantum)	4
* Time Quantum	111.11111111111111 ns
Time Quanta in Bit Segment 1	6 Times
Time Quanta in Bit Segment 2	2 Times
* Time for one Bit	1000 ns
* Baud Rate	1000000 bit/s
ReSynchronization Jump Width	1 Time

∨ Basic Parameters

Time Triggered Communicatio...	Disable
Automatic Bus-Off Management	Disable
Automatic Wake-Up Mode	Disable
Automatic Retransmission	Disable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable

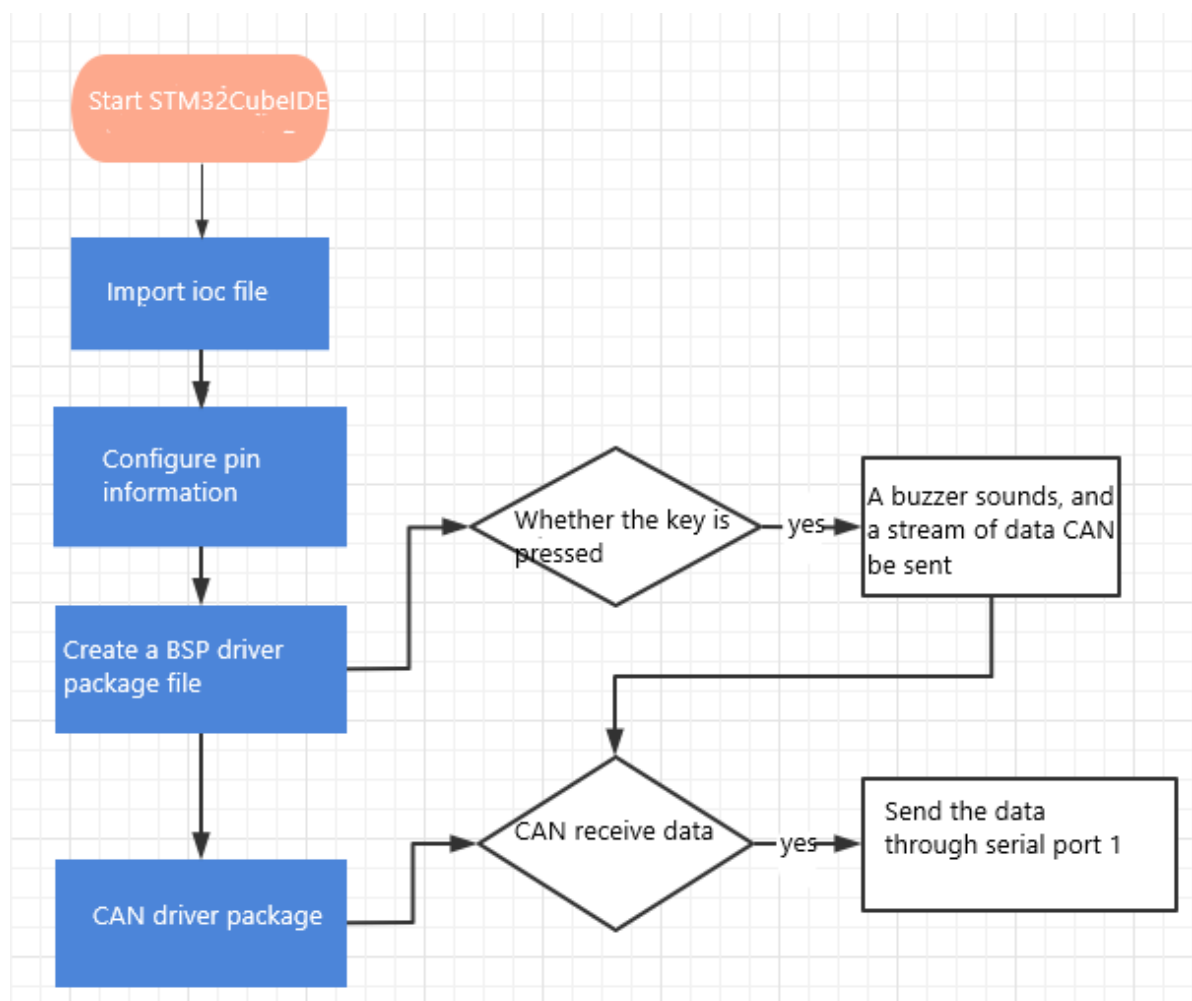∨ Advanced Parameters

Test Mode	Loopback

Since this is only used to test communication, select Loopback mode (data collection). If you need to connect a third-party CAN device, select the Normal mode (data receiving/sending is independent).
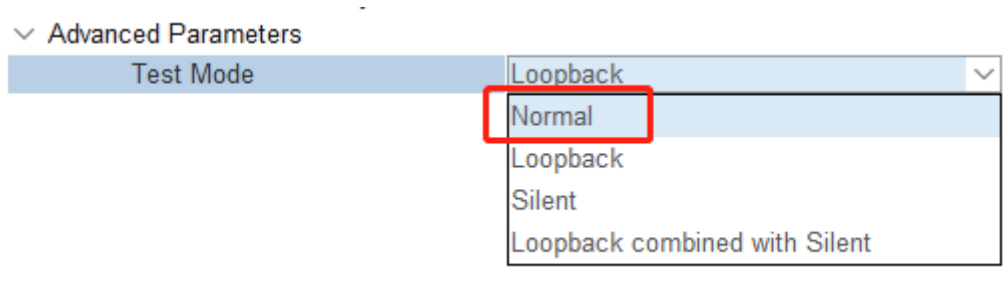
4. Turn on the CAN RX0 interrupt in the interrupt setting. If you do not turn on the interrupt, you cannot receive data.

7.3. Analysis of experimental flow chart

7.4. Core code interpretation

1. Create the buzzer driver library bsp_can.h and bsp_can.c files in the BSP. Add the following to bsp_can.h:

```
void Can_Init(void);
void Can_Test_Send(void);
```

2. Add the following content to bsp_can.c:

Can_Init(): initializes CAN peripherals, sets CAN receiving filters, and enables CAN bus communication.

```c
// Initialize the CAN   初始化CAN
void Can_Init(void)
{
    sFilterConfig.FilterBank = 0;
    sFilterConfig.FilterMode = CAN_FILTERMODE_IDMASK;
    sFilterConfig.FilterScale = CAN_FILTERSCALE_32BIT;
    sFilterConfig.FilterIdHigh = 0x0000;
    sFilterConfig.FilterIdLow = 0x0000;
    sFilterConfig.FilterMaskIdHigh = 0x0000;
    sFilterConfig.FilterMaskIdLow = 0x0000;
    sFilterConfig.FilterFIFOAssignment = CAN_FILTER_FIFO0;
    sFilterConfig.SlaveStartFilterBank = 27;
    sFilterConfig.FilterActivation = CAN_FILTER_ENABLE;

    // Setting the CAN Filter   设置CAN过滤器
    if (HAL_CAN_ConfigFilter(&hcan, &sFilterConfig) != HAL_OK)
    {
        Error_Handler();
    }

    // Start the CAN peripheral   启动CAN
    if (HAL_CAN_Start(&hcan) != HAL_OK)
    {
        Error_Handler();
    }

    // Activate CAN RX notification   启动CAN RX通知
    if (HAL_CAN_ActivateNotification(&hcan, CAN_IT_RX_FIFO0_MSG_PENDING) != HAL_OK)
    {
        Error_Handler();
    }
}
```

3. In order to test sending data, a new Can_Test_Send() function is created to send data through CAN and print it to the serial assistant. If you need to modify the sent data, just modify the TxData array before sending.

```
// The test sends data through CAN   测试通过CAN发送数据
void Can_Test_Send(void)
{
    uint8_t TxData[8];
    uint32_t TxMailbox = 0;
    TxHeader.StdId = 0x000F;
    TxHeader.ExtId = 0x00;
    TxHeader.RTR = CAN_RTR_DATA;
    TxHeader.IDE = CAN_ID_STD;
    TxHeader.DLC = 8;
    TxHeader.TransmitGlobalTime = DISABLE;

    for (int i = 0; i < 8; i++)
    {
        TxData[i] = 1 << i;
    }
    printf("CAN Send:%02X %02X %02X %02X %02X %02X %02X %02X \n",
            TxData[0], TxData[1], TxData[2], TxData[3],
            TxData[4], TxData[5], TxData[6], TxData[7]);
    // Send Data   发送数据
    if (HAL_CAN_AddTxMessage(&hcan, &TxHeader, TxData, &TxMailbox) != HAL_OK)
    {
        Error_Handler();
    }
}
```

4.CAN receive interrupt callback function, and print out the received CAN data through serial port. This function name cannot be changed, otherwise this function cannot be called.

```
// CAN receives interrupt callbacks   CAN接收中断回调
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    if (hcan->Instance == CAN1)
    {
        uint8_t RxData[8];
        if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader, RxData) != HAL_OK)
        {
            Error_Handler();
        }
        else
        {
            printf("CAN Receive:%02X %02X %02X %02X %02X %02X %02X %02X \n",
                    RxData[0], RxData[1], RxData[2], RxData[3],
                    RxData[4], RxData[5], RxData[6], RxData[7]);
        }
    }
}
```

5. In BSP initialization, call the Can_Init() function to initialize the CAN peripheral.

```
// The peripheral device is initialized   外设设备初始化
void Bsp_Init(void)
{
    Can_Init();
    USART1_Init();
    Beep_On_Time(50);
    printf("start\n");
}
```

6. Add the function of sending CAN data after the key is pressed.

```
// main.c中循环调用此函数，避免多次修改main.c文件。
// This function is called in a loop in main.c to avoid multiple modifications to the main.c file
void Bsp_Loop(void)
{
    // Detect button down events    检测按键按下事件
    if (Key1_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        Can_Test_Send();
    }

    Bsp_Led_Show_State_Handle();
    // The buzzer automatically shuts down when times out    蜂鸣器超时自动关闭
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}
```

7.5. Hardware connection

Since the loop back mode is used, the CAN interface can not be connected to external devices.
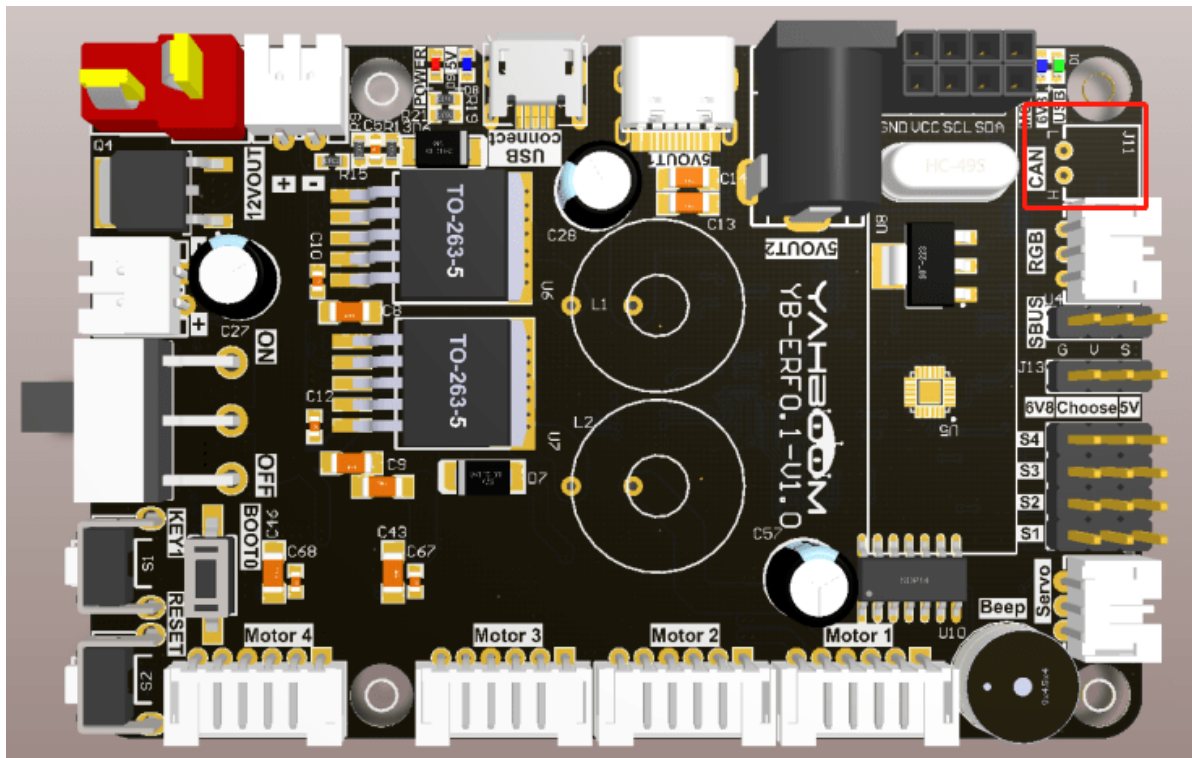
7.6. Experimental effect

After the program is burned, the LED light flashes every 200 milliseconds, the expansion board is connected to the computer through the micro-USB data cable, and the serial port assistant is opened (specific parameters are shown below). Every time you press the key, the buzzer will ring for 50 milliseconds, and you CAN see that the serial port assistant will display the data sent by CAN and the data received by CAN.