在 M460 uCOSii 中做 4 路 CANFD 收发

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码用 M460 系列移植 uCOSii 和 4 路 CANFD 收发报文
BSP 版本	M460_Series_BSP_CMSIS_V3.00.002
开发平台	NuMaker-M467HJ V1.0

1. 概述

用 M460 系列单片机移植 uCOSii，并且为 4 路 CANFD 分别建立了任务，负责报文收发。

1.1 原理

M460 系列的 CANFD 功能强大，发送时最多可配置多达 32 个报文缓存等待发送，如图 1-1，报文缓存个数和首地址由寄存器 TXBC 配置，这个配置在函数 CANFD_Tx_Init()中。在多任务系统中，需发送 CANFD 报文的任务，可独占其中几个缓存用于发送报文，如此就不必再用信号量管理 CANFD 外设了。

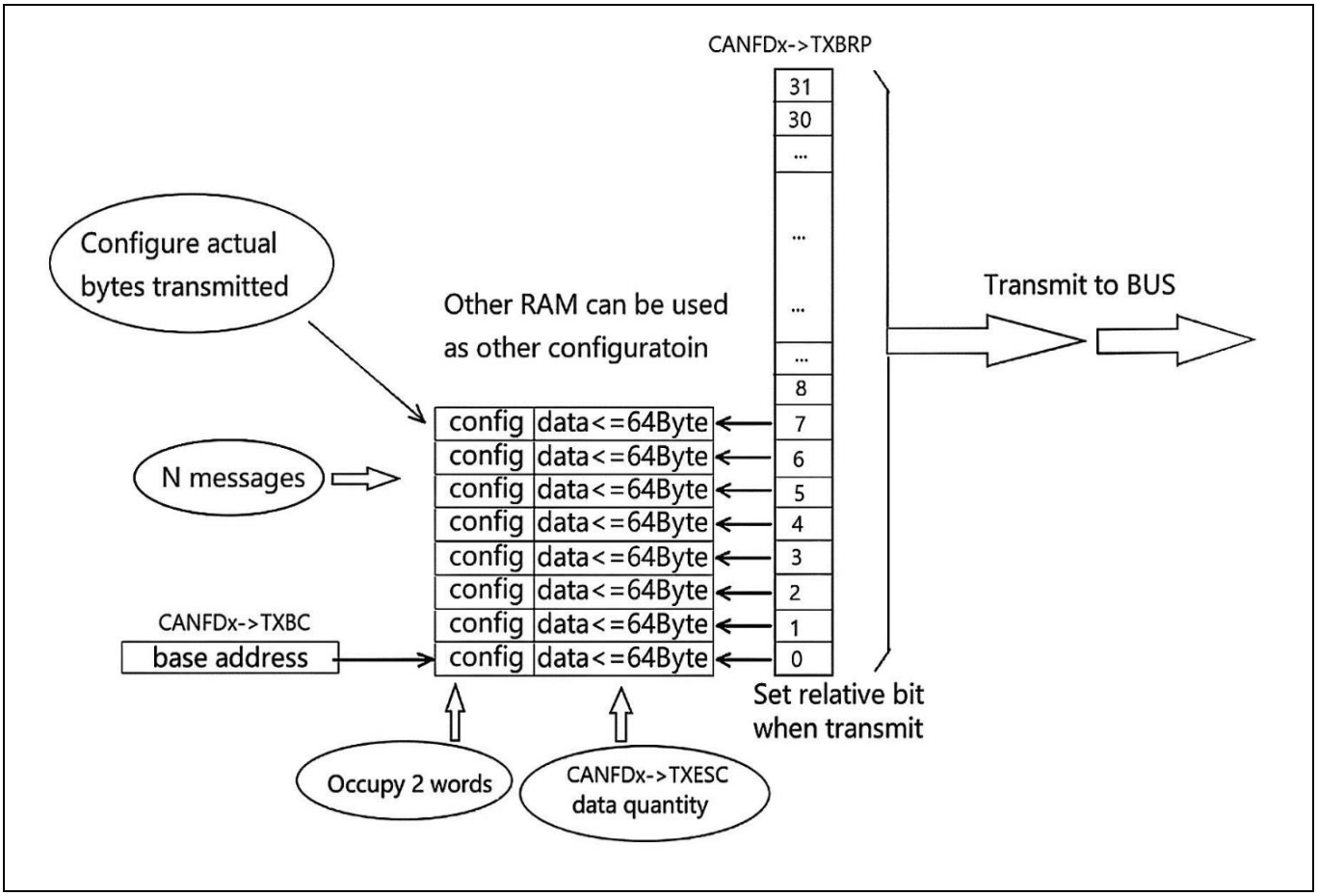

图 1-1 M460 CANFD Tx Register

每条等待发送的报文，数据结构是 “配置 + 数据(最多 64 字节)” ，报文数据结构如图 1-2。

Bit	31			24	23					16	15		8	7		0
T0	ESI	XTD	RTR	ID[28:0]												
T1	MM[7:0]				EFC	-	FDF	BRS	DLC[3:0]		MM[15:8]			-		
T2	DB3[7:0]				DB2[7:0]						DB1[7:0]			DB0[7:0]		
T3	DB7[7:0]				DB6[7:0]						DB5[7:0]			DB4[7:0]		

图 1-2 发送报文格式

接收报文时，对于 CANFD 总线上的报文，经过最多 128 组标准 SID 和最多 64 组扩展 XID 过滤后，接收下来的报文可放入 64 个缓存中的某个指定空间，多任务系统中的某个任务可独占某个接收缓存，如此可不用信号量管理 CANFD 报文的接收。接收报文也可以放入有 64 级深度的 FIFO0 或 FIFO1，如图 1-3。在总线报文较多时，使用 FIFO 可以减轻 CPU 的负荷。本代码接收报文就是放入 CANFD0~CANFD3 各自的 FIFO1 的。

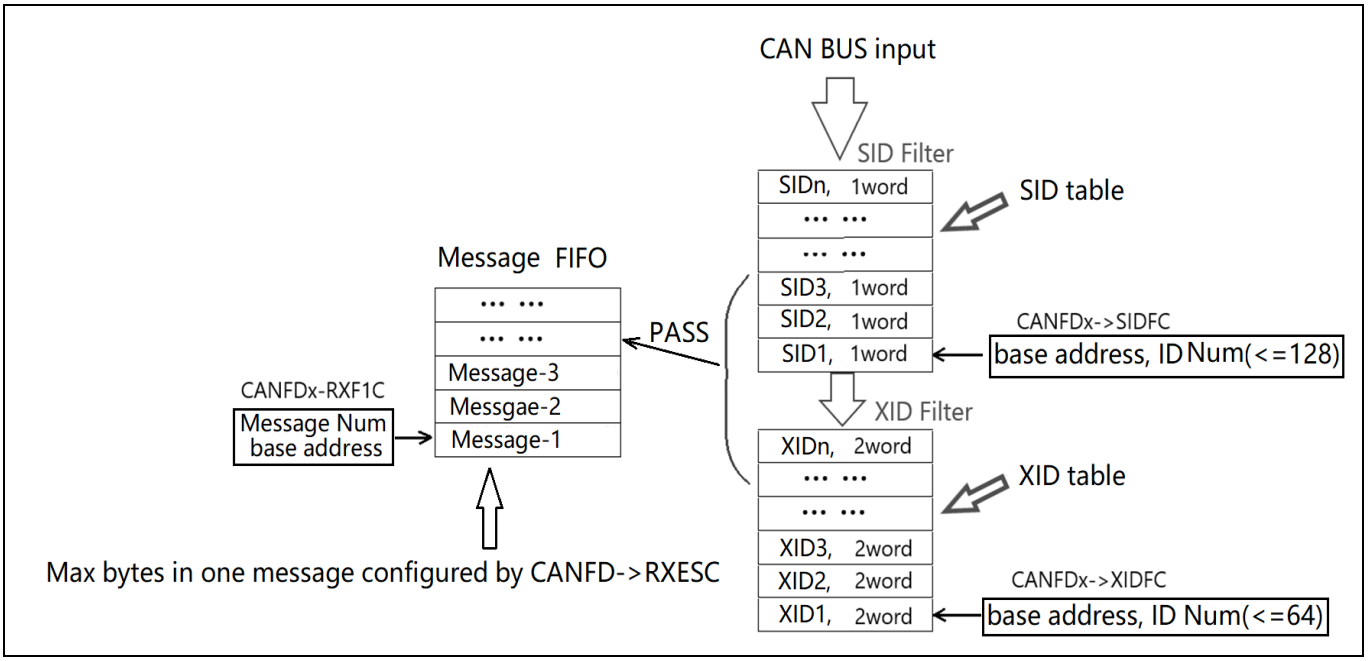

图 1-3 CANFD FIFO 接收结构

图1-4展示了用于接收时过滤报文的标准ID的数据结构

Bit	31		24	23	16	15	8	7	0
S0	SFT [1:0]	SFEC [2:0]	SID1[10:0]			-	SID2(or Mask)[10:0]		

图1-4 SID数据结构

图1-5 展示了用于过滤接收报文的扩展ID的数据结构。

Bit	31		24	23	16	15	8	7	0
F0	EFECV [2:0]		XID1[28:0]						
F0	EFT [1:0]	-	XID2(or MASK)[28:0]						

图 1-5 XID 数据结构

1.2 执行结果

本代码在只有一个 CANFD 收发器的 NuMaker-M467HJ V1.0 板上执行结果如图 1-6 所示。

```
  Task A

Task 0
CAN0 SID=0x0116(04 Bytes): 139,05,00,00,
Task 1
CAN0 SID=0x0116(04 Bytes): 140,05,00,00,
Task 2
Task 3
Task 0
CAN0 XID=0x0118(08 Bytes): 00,00,00,00,225,255,255,255,
CAN0 SID=0x0116(04 Bytes): 141,05,00,00,
Task 1

  Task A

Task 0
Task 2
Task 3
CAN0 SID=0x0116(04 Bytes): 142,05,00,00,
Task 0
Task 1
Task 0
CAN0 XID=0x0118(08 Bytes): 00,00,00,00,226,255,255,255,
Task 2
Task 3
CAN0 SID=0x0116(04 Bytes): 143,05,00,00,
Task 0
CAN0 SID=0x0116(04 Bytes): 144,05,00,00,
```

图 1-6 执行结果

如果把 CANFD0~CANFD3 接口连到一条 CANFD 总线上，执行本代码可得到如图 1-7 所示打印信息。Task0~Task3 是对应的任务在执行，对应 CANFD0~CANFD3 在发送。CAN0~CAN3 是对应的 CANFD 接口收到的报文。每个 CANFD 发送后，其它 3 个 CANFD 都会收到报文。

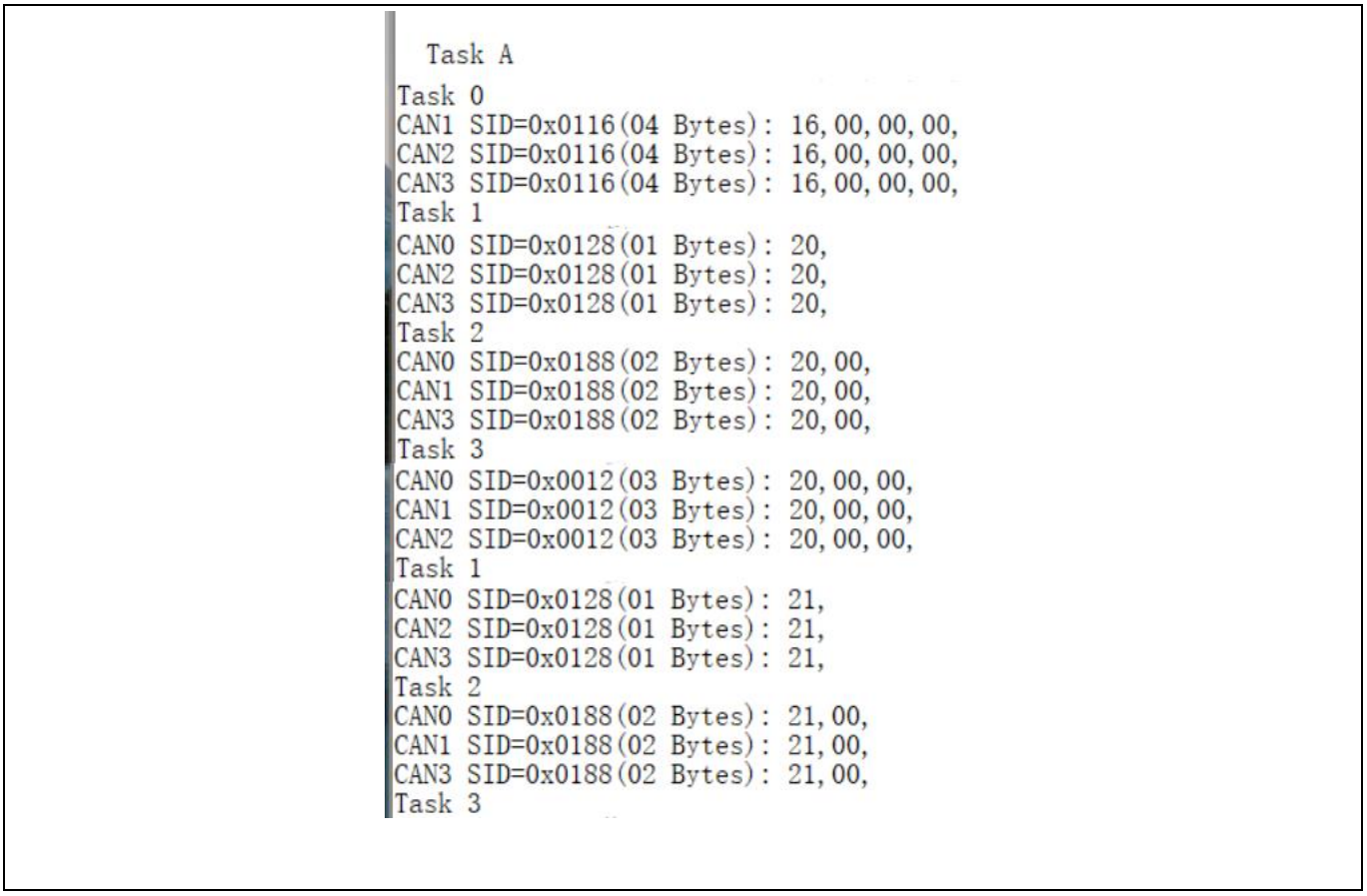

图 1-7 4 个 CANFD 收发器连在一起的执行打印结果

2. 代码介绍

本代码移植了 uCOSii，如何移植 uCOSii 请参考 EC_M480_uCOS_II_Porting_Readme，可在 https://www.nuvoton.com.cn/resource-download.jsp?tp_GUID=EC012022101106314002 下载。本代码建立了 6 个任务，task_Highest()是优先级最高的任务，系统节拍定时器必须在此任务开头配置并开始工作，UART0 打印也在此任务中，其它任务若需 UART0 打印信息，必须通过队列 Q 把待打印数据发过来，避免发生访问 UART0 冲突。

task0~task3 四个任务分别控制 CANFD0~CANFD3 向 CANFD 总线发送报文。

CANFD0~CANFD3 每收到一个报文，对应的中断里会发出一个信号量（信号量加 1），中断外检查信号量从对应的 FIFO1 中读取报文。 本例程力求简化，统一在 task_A 中读取并打印报文，为了及时打印，task_A 的优先级一般高于其它需通过 UART0 打印信息的任务。

CANFD 的引脚功能配置、通信速率以及收发初始化代码，在 main()函数开头配置。发送 CAN 报文还是 CANFD 报文也在此配置。

```
//== Configure CANFD0 ======================================
    CANFD0->CCCR = CANFD_CCCR_CCE_Msk | CANFD_CCCR_INIT_Msk | CANFD_CCCR_BRSE_Msk |
CANFD_CCCR_FDOE_Msk;

#ifndef  __M467SJHAE            // If test this project on NuMaker-M467HJ board
    SET_CAN0_RXD_PJ11();        // Set PJ multi-function pins for CAN FD0 RXD and TXD
    SET_CAN0_TXD_PJ10();
#else                           // If test this project on M467SJHAE board
    SET_CAN0_RXD_PB10();
    SET_CAN0_TXD_PB11();
#endif
    CANFD_BitRate_Init(CANFD0);
                          // 32-Txbuf,StartAddress=0xE0 Please reference CANFD_TxBuff[32]
    CANFD_Tx_Init(CANFD0, 32, 0xE0);
    CANFD0_RX_Initial(CANFD0);
//    CANFD0->CCCR = CANFD_CCCR_BRSE_Msk | CANFD_CCCR_FDOE_Msk ;            // CANFD frame
    CANFD0->CCCR = 0 ;                                  // CAN frame,Write is OK after 2 CLKs
```

正确配置报文接收采样点，是提高 CAN 或 CANFD 通信可靠性的关键，如图 2-1。考虑到数据传输时间，采样前段 DTSG1 要比采样后段 DTSG2 时间要长一点。这部分由函数 CANFD_BitRate_Init()配置，前段时间占位时间的 75%。

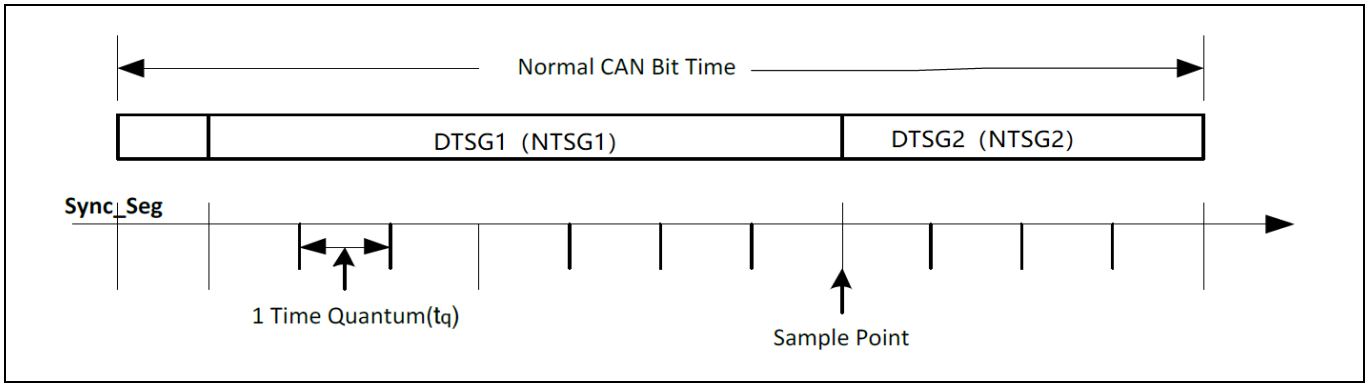

图 2-1 报文接收采样点

```
void CANFD_BitRate_Init(CANFD_T *psCanfd)
{
    // CANFD_CLk = 200M/10 = 20MHz       Baud Rate = 20M/20 = 1Mbps

    psCanfd->NBTP = (3 << 25) +        // NSJW  = 3+1 =4 CLK
                    (0 << 16) +        // NBRP  = 0+1 =1       // prescaler =1
                    (13 << 8) +        // NTSG1 = 13+1 =14 CLK
                    (5 - 1) ;          // NTSG2 = 5 CLK       // One bit =1+14+5 =20 CLK

// If CANFD frame and permit data-rate change, define date rate as following.

//    psCanfd->DBTP = ((1 - 1) << 16) +      // DBRP  = 1 prescaler
//                    ((15 - 1) << 8) +      // DTSG1 = 15 CLK
//                    ((5 - 1) << 4) +       // DTSG2 = 5 CLK   // One bit = 15+5 = 20 CLK
//                    (4 - 1) ;              // DSJW  = 4 CLK
}
```

函数 CANFD_Tx_Init()配置发送缓存首地址，每个报文缓存配置为最多可存放 64 字节，但发送报文实际字节数，发送时再具体定义。

```
void CANFD_Tx_Init(CANFD_T *psCanfd, int16_t TxBuff_Quantity, uint16_t TxBuff_StartAddr)
{
    psCanfd->TXBC = (TxBuff_Quantity << CANFD_TXBC_NDTB_Pos)
                    + TxBuff_StartAddr ;            // Start address of messages RAM

    psCanfd->TXESC = 7 << CANFD_TXESC_TBDS_Pos;     // 7 mean 64 Bytes
}
```

函数 CANFD0_RX_Initial()是 CANFD0 接收配置。前面配置了报文缓存最多可存放 64 个字节，然后配置了 SID 和 XID 的存放地址在专用 RAM 区的偏移值。#if 1 代码是接收所有报文放入 FIFO1 并且拒绝接收远程帧。#else 代码是配置 SID 值或 XID 值过滤报文的几个示例。

```
/*---------------------------------------------------------------------------*/
/*                        Init CANFD0 Rx                                     */
/*---------------------------------------------------------------------------*/
void CANFD0_RX_Initial(CANFD_T *psCanfd)
```

```
{
    // 0=>8Byte, 1=>12Byte, 2=>16Byte, 3=>20Byte,4=>24Byte,5=>32Byte,6=>48Byte, 7=>64Byte
    psCanfd->RXESC = (psCanfd->RXESC & (~CANFD_RXESC_F1DS_Msk)) | (7 <<
CANFD_RXESC_F1DS_Pos);                                          // 7 mean MAX 64 Bytes

    psCanfd->RXF1C = (30 << CANFD_RXF1C_F1WM_Pos)           // watermark = 60 messages
                   | (32 << CANFD_RXF1C_F1S_Pos)          // FIFO1 can hold 64 messages
                   | 0x9E0 ;                  // FIFO1 start address =0x40020000 +0x9E0

    psCanfd->SIDFC = (24 << 16) + 0 ;                       // 24-SID at 0  address
    psCanfd->XIDFC = (16 << 16) + 0x60 ;                    // 16-XID at 0x60 address

#if 1
    CANFD0_SID_Buff[0].VALUE = CANFD_RX_FIFO1_STD_MASK(0, 0) ; // receive all SID messages

    CANFD0_XID_Buff[0].LOWVALUE  =CANFD_RX_FIFO1_EXT_MASK_LOW(0) ;
    CANFD0_XID_Buff[0].HIGHVALUE=CANFD_RX_FIFO1_EXT_MASK_HIGH(0);  //receive all XID

    CANFD0->GFC = 3 ;                                           // reject remote frames

#else
    CANFD0_SID_Buff[0].VALUE = CANFD_RX_FIFO1_STD_MASK(0x110, 0x7F0) ;   // SID, Mask
    CANFD0_SID_Buff[1].VALUE = CANFD_RX_FIFO1_STD_MASK(0x22F, 0x7FF) ;   // SID, Mask
    CANFD0_SID_Buff[2].VALUE = CANFD_RX_FIFO1_STD_MASK(0x333, 0x7FF) ;   // SID, Mask

    CANFD0_XID_Buff[0].LOWVALUE  = CANFD_RX_FIFO1_EXT_MASK_LOW(0x220) ;       // XID
    CANFD0_XID_Buff[0].HIGHVALUE = CANFD_RX_FIFO1_EXT_MASK_HIGH(0x1FFFFFF0) ; // MASK

    CANFD0_XID_Buff[1].LOWVALUE  = CANFD_RX_FIFO1_EXT_MASK_LOW(0x3333) ;      // XID
    CANFD0_XID_Buff[1].HIGHVALUE = CANFD_RX_FIFO1_EXT_MASK_HIGH(0x1FFFFFFF) ; // MASK

    CANFD0_XID_Buff[2].LOWVALUE  = CANFD_RX_FIFO1_EXT_MASK_LOW(0x44444) ;     // XID
    CANFD0_XID_Buff[2].HIGHVALUE = CANFD_RX_FIFO1_EXT_MASK_HIGH(0x1FFFFFFF) ; // MASK

    CANFD0->GFC = 0x20 + 0x08 + 3 ;  // reject all no-match SID &XID, reject remote frames

    //  CANFD0->GFC = 0x10 + 0x04 + 3 ;     //no-match ID into FIFO1. reject remote frames

#endif

    CANFD_EnableInt(CANFD0, (CANFD_IE_TOOE_Msk | CANFD_IE_RF1NE_Msk), 0, 0, 0);
    NVIC_EnableIRQ(CANFD00_IRQn);
}
```

本代码是在只能“字写入”不能字节写入的、CANFD 的专用 RAM 区直接组织报文数据的，所以要在编译器的 Option 选项里，配置专用 RAM 区的地址，如图 2-2 中 IRAM2。

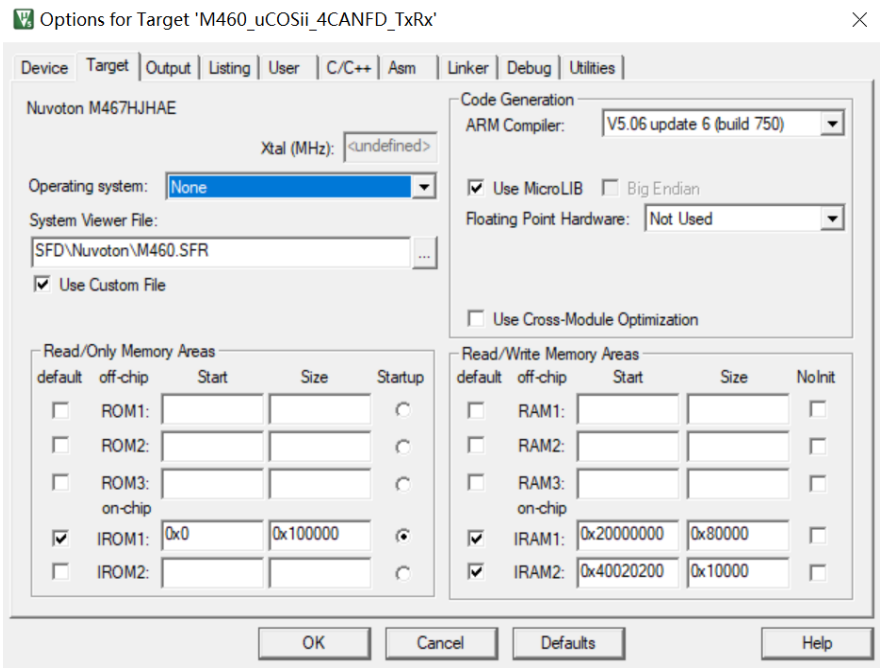

图 2-2 Option 中 RAM 区配置

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M460_Series_BSP_CMSIS_V3.00.002
- IDE 版本
  - Keil uVersion 5.38

3.2 硬件需求

- 电路组件
  - NuMaker-M467HJ V1.0
- 线路示意图
  - 将两个板子的 CANFD 总线连一起，以显示范例代码的执行结果。

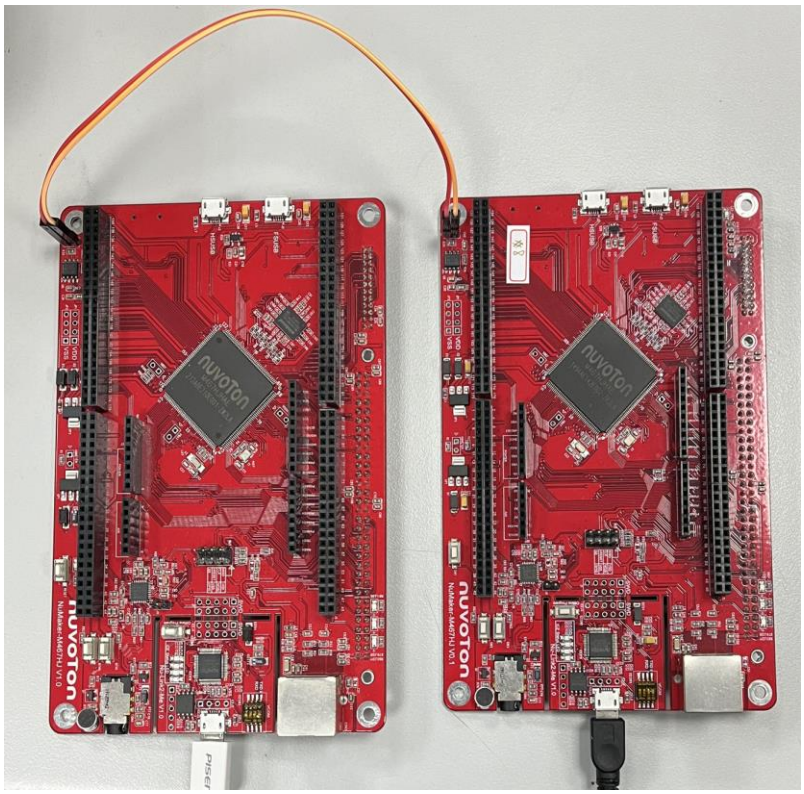

图 3-1 线路示意图

4. 目录信息

EC_M460_uCOSii_4CANFD_TxRx_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code
uCOS_II	
Config	Configuration files of uCOSii for this project
Ports	Porting files of uCOSii for M460 series MCU
Source	Source file of uCOSii code

图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *M460_uCOSii_4CANFD_TxRx.uvprojx*。
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

6. 修订纪录

Date	Revision	Description
2023.11.22	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.