在 M460 uCOSii 中做 4 路 CANFD 收發

NuMicro® 32 位元系列微控制器範例代碼介紹

檔資訊

應用簡述	本範例代碼用 M460 系列移植 uCOSii 和 4 路 CANFD 收發報文
BSP 版本	M460_Series_BSP_CMSIS_V3.00.002
開發平臺	NuMaker-M467HJ V1.0

1. 概述

用 M460 系列單片機移植 uCOSii，並且為 4 路 CANFD 分別建立了任務，負責報文收發。

1.1 原理

M460 系列的 CANFD 功能強大，發送時最多可配置多達 32 個報文緩存等待發送，如圖 1-1，報文緩存個數和首位址由寄存器 TXBC 配置，這個配置在函數 CANFD_Tx_Init()中。在多工系統中，需發送 CANFD 報文的任務，可獨佔其中幾個緩存用於發送報文，如此就不必再用信號量管理 CANFD 外設了。

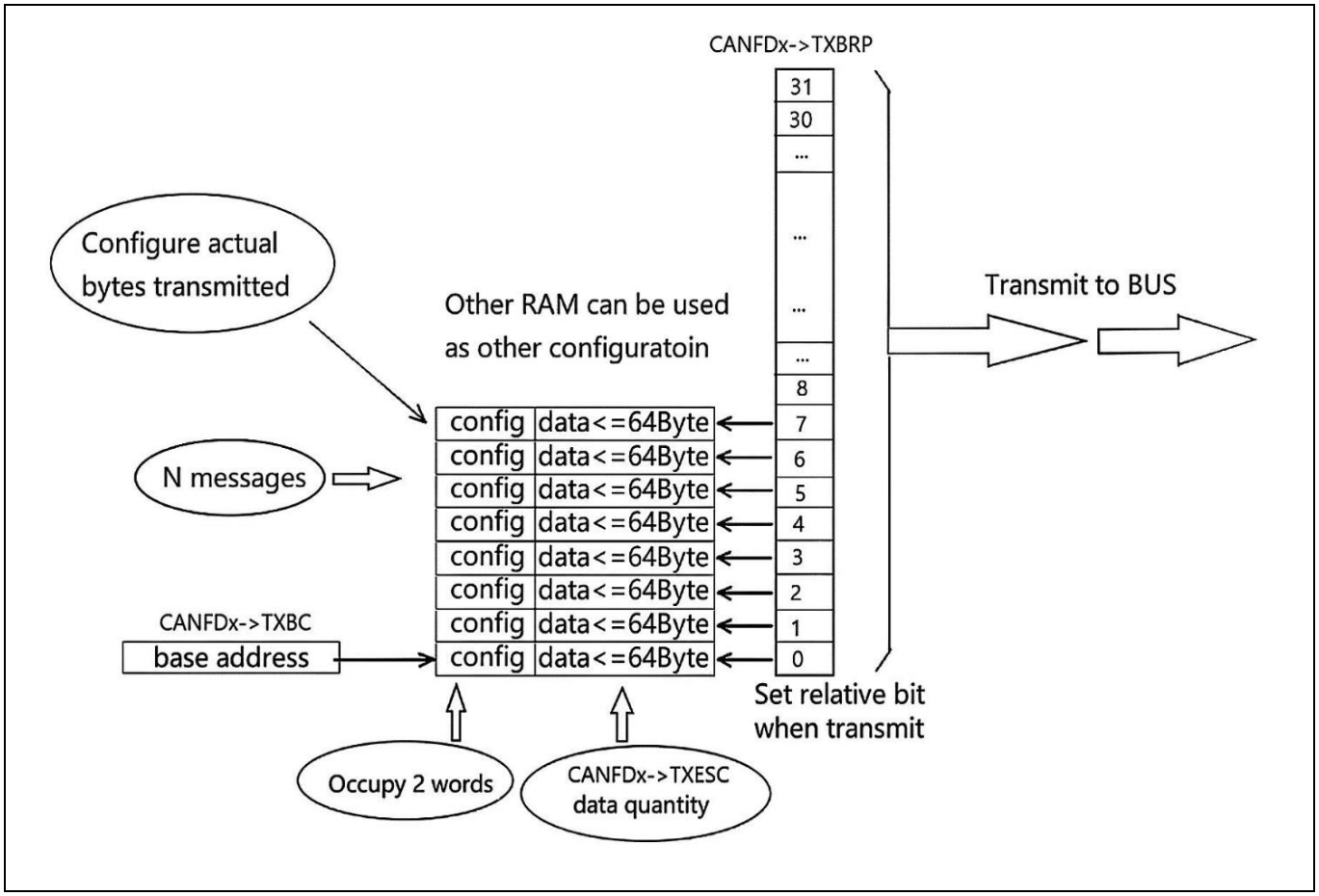

圖 1-1 M460 CANFD Tx Register

每條等待發送的報文，資料結構是“配置 + 資料(最多 64 位元組)”，報文資料結構如圖 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]			

圖 1-2 發送報文格式

接收報文時，對於 CANFD 匯流排上的報文，經過最多 128 組標準 SID 和最多 64 組擴展 XID 過濾後，接收下來的報文可放入 64 個緩存中的某個指定空間，多工系統中的某個任務可獨佔某個接收緩存，如此可不用信號量管理 CANFD 報文的接收。接收報文也可以放入有 64 級深度的 FIFO0 或 FIFO1，如圖 1-3。在匯流排報文較多時，使用 FIFO 可以減輕 CPU 的負荷。本代碼接收報文就是放入 CANFD0~CANFD3 各自的 FIFO1 的。

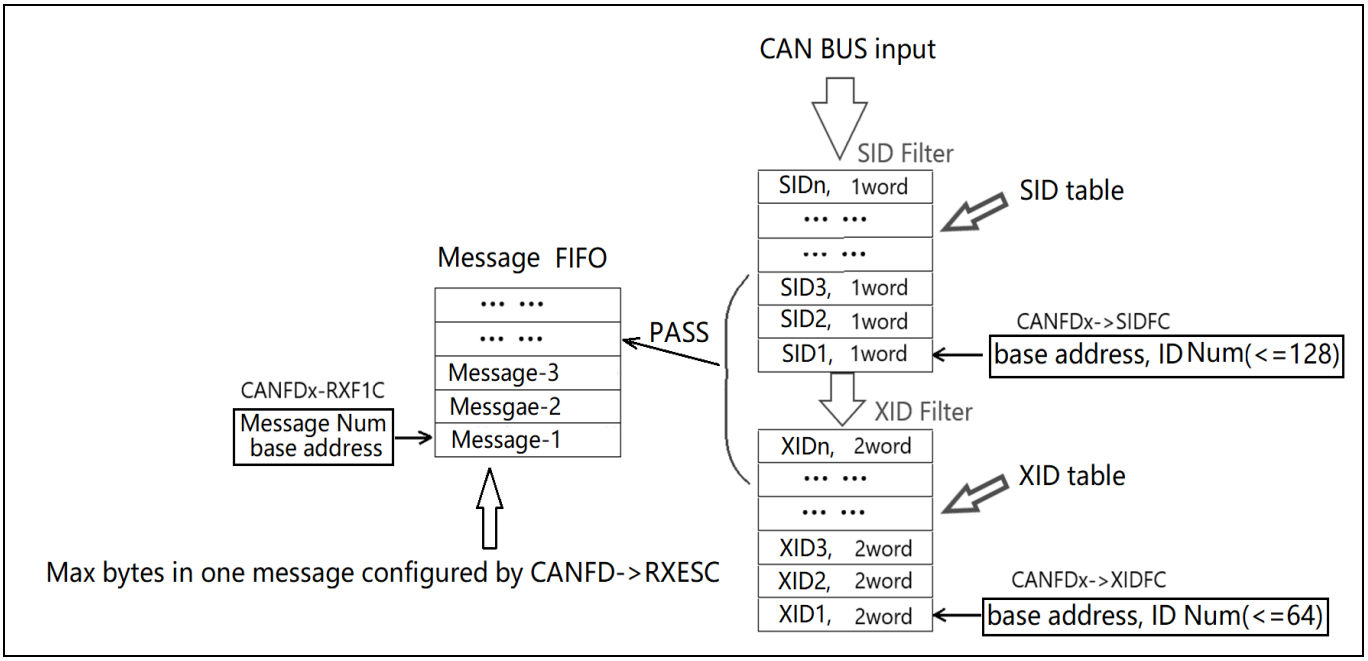

圖 1-3 CANFD FIFO 接收結構

圖1-4展示了用於接收時過濾報文的標準ID的資料結構

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

圖1-4 SID資料結構

圖1-5 展示了用於過濾接收報文的擴展ID的資料結構。

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]						

圖 1-5 XID 資料結構

1.2 執行結果

本代碼在只有一個 CANFD 收發器的 NuMaker-M467HJ V1.0 板上執行結果如圖 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,
```

圖 1-6 執行結果

如果把 CANFD0~CANFD3 介面連到一條 CANFD 匯流排上，執行本代碼可得到如圖 1-7 所示列印資訊。Task0~Task3 是對應的任務在執行，對應 CANFD0~CANFD3 在發送。CAN0~CAN3 是對應的 CANFD 介面收到的報文。每個 CANFD 發送後，其它 3 個 CANFD 都會收到報文。

```
  Task A
Task 0
CAN1 SID=0x0116(04 Bytes): 16,00,00,00,
CAN2 SID=0x0116(04 Bytes): 16,00,00,00,
CAN3 SID=0x0116(04 Bytes): 16,00,00,00,
Task 1
CAN0 SID=0x0128(01 Bytes): 20,
CAN2 SID=0x0128(01 Bytes): 20,
CAN3 SID=0x0128(01 Bytes): 20,
Task 2
CAN0 SID=0x0188(02 Bytes): 20,00,
CAN1 SID=0x0188(02 Bytes): 20,00,
CAN3 SID=0x0188(02 Bytes): 20,00,
Task 3
CAN0 SID=0x0012(03 Bytes): 20,00,00,
CAN1 SID=0x0012(03 Bytes): 20,00,00,
CAN2 SID=0x0012(03 Bytes): 20,00,00,
Task 1
CAN0 SID=0x0128(01 Bytes): 21,
CAN2 SID=0x0128(01 Bytes): 21,
CAN3 SID=0x0128(01 Bytes): 21,
Task 2
CAN0 SID=0x0188(02 Bytes): 21,00,
CAN1 SID=0x0188(02 Bytes): 21,00,
CAN3 SID=0x0188(02 Bytes): 21,00,
Task 3
```

圖 1-7 4 個 CANFD 收發器連在一起的執行列印結果

2. 代碼介紹

本代碼移植了 uCOSii，如何移植 uCOSii 請參考 EC_M480_uCOS_II_Porting_Readme，可在 https://www.nuvoton.com.cn/resource-download.jsp?tp_GUID=EC012022101106314002 下載。本代碼建立了 6 個任務，task_Highest()是優先順序最高的任務，系統節拍計時器必須在此任務開頭配置並開始工作，UART0 列印也在此任務中，其它任務若需 UART0 列印資訊，必須通過佇列 Q 把待列印資料發過來，避免發生訪問 UART0 衝突。

task0~task3 四個任務分別控制 CANFD0~CANFD3 向 CANFD 匯流排發送報文。

CANFD0~CANFD3 每收到一個報文，對應的中斷裡會發出一個信號量（信號量加 1），中斷外檢查信號量從對應的 FIFO1 中讀取報文。 本常式力求簡化，統一在 task_A 中讀取並列印報文，為了及時列印，task_A 的優先順序一般高於其它需通過 UART0 列印資訊的任務。

CANFD 的引腳功能配置、通信速率以及收發初始化代碼，在 main()函數開頭配置。發送 CAN 報文還是 CANFD 報文也在此配置。

```
//== 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
```

正確配置報文接收採樣點，是提高 CAN 或 CANFD 通信可靠性的關鍵，如圖 2-1。考慮到資料傳輸時間，採樣前段 DTSG1 要比採樣後段 DTSG2 時間要長一點。這部分由函數 CANFD_BitRate_Init()配置，前段時間占位時間的 75%。

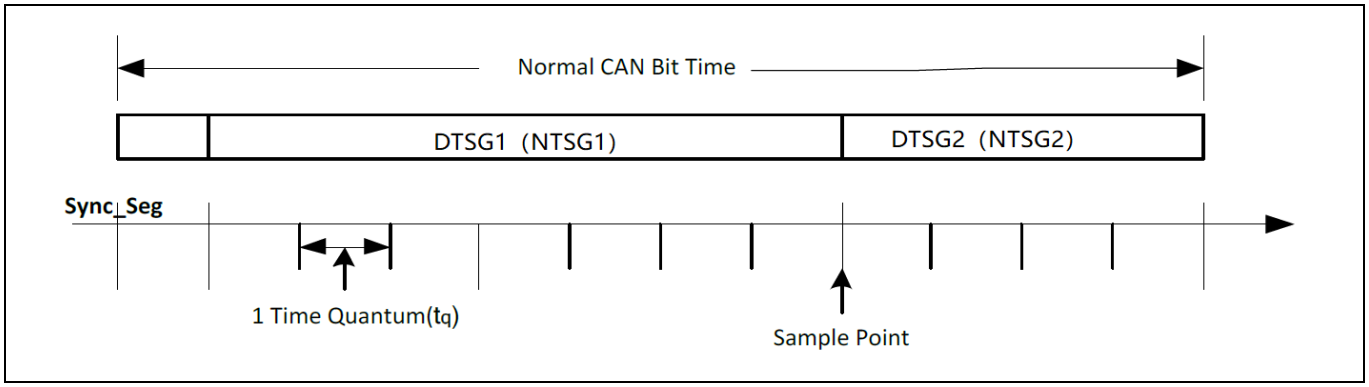

圖 2-1 報文接收採樣點

```
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
}
```

函數 CANFD_Tx_Init()配置發送緩存首位址，每個報文緩存配置為最多可存放 64 位元組，但發送報文實際位元組數，發送時再具體定義。

```
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
}
```

函數 CANFD0_RX_Initial()是 CANFD0 接收配置。前面配置了報文緩存最多可存放 64 個位元組，然後配置了 SID 和 XID 的存放位址在專用 RAM 區的偏移值。#if 1 代碼是接收所有報文放入 FIFO1 並且拒絕接收遠端幀。#else 代碼是配置 SID 值或 XID 值過濾報文的幾個示例。

```
/*----------------------------------------------------------------------------*/
/*                        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);
}
```

本代碼是在只能“字寫入”不能位元組寫入的、CANFD 的專用 RAM 區直接組織報文資料的，所以要在編譯器的 Option 選項裡，配置專用 RAM 區的位址，如圖 2-2 中 IRAM2。

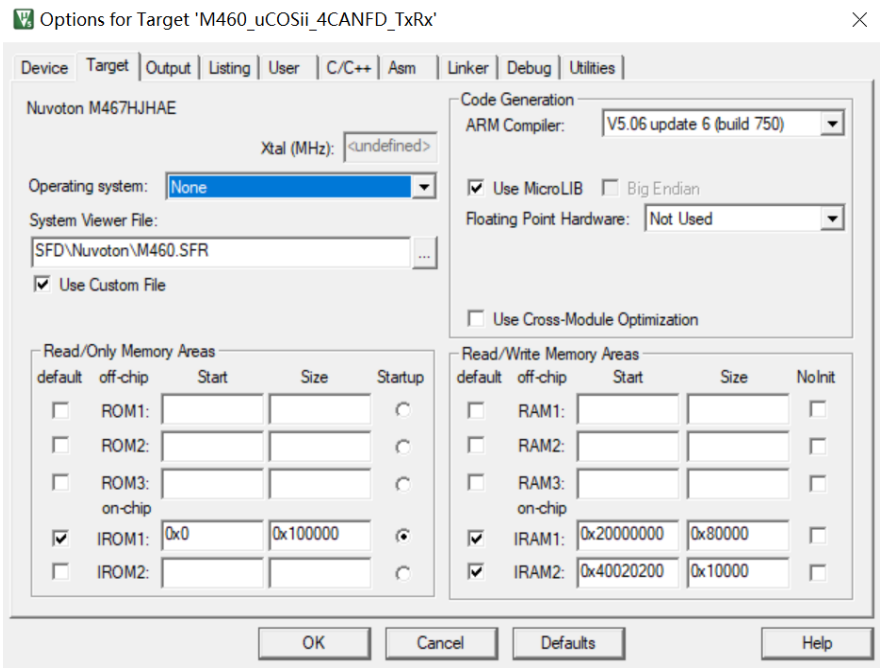

圖 2-2 Option 中 RAM 區配置

3. 軟體與硬體需求

3.1 軟體需求

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

3.2 硬體需求

- 電路元件
  - NuMaker-M467HJ V1.0
- 線路示意圖
  - 將兩個板子的 CANFD 匯流排連一起，以顯示範例代碼的執行結果。

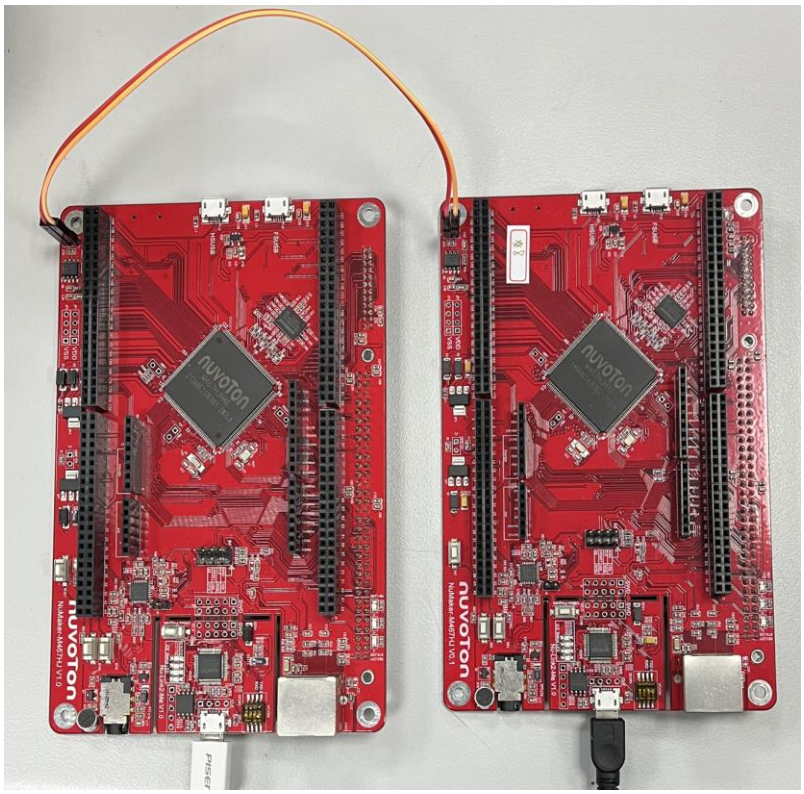

圖 3-1 線路示意圖

4. 目錄資訊

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

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，按兩下 *M460_uCOSii_4CANFD_TxRx.uvprojx*。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開模擬模式
3. 進入模擬模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2023.11.22	1.00	初始發佈。

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.