Simple Code for M460 CANFD Transmission

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses M460 series MCU to transmit CANFD messages cyclically .
BSP Version	M460_Series_BSP_CMSIS_V3.00.001
Hardware	NuMaker-M467HJ V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. Overview

This example code uses to test CANFD transmitting for M460 series MCU.

1.1 Principle

A message is composed of “2words ID + data”. The format is shown in Figure 1-1.

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]							

Figure 1-1 Message Format Waiting for Transmission

A message may be waiting for bus idle after it is enabled to transmit. There can be configured 32 messages waiting for transmitting in M467 special CANFD RAM which has 6144 bytes to buffer message ID, received messages, etc.

Figure 1-2 shows registers relative to transmission. The messages quantity of waiting for transmitting is configured by CANFDx->TXBC. The data quantity in one message is configured by CANFDx->TXESC. A message will not be transmitted until bus idle after the relative bit of CANFDx->TXBRP is set. This relative bit will be cleared by hardware after a message is transmitted.

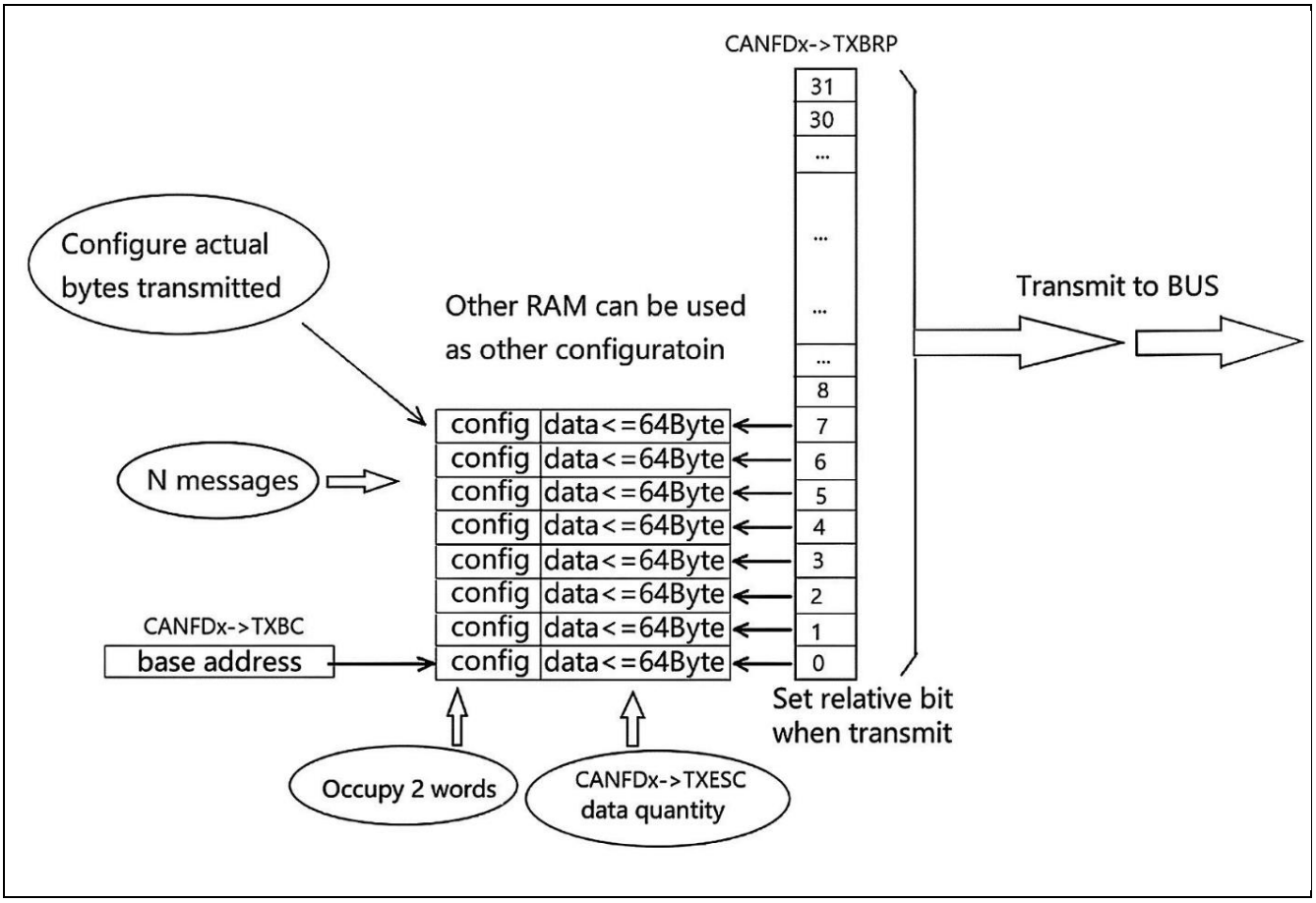

Figure 1-2 Register Relative to Transmission

In this example code, bit time of CANFD is configured at first. Then other register of CANFD is configured. At last, CANFDx->TXBRP is set periodically for transmission.

1.2 Demo Result

The CANFD interface will transmit continuously and cyclically 8 messages as Figure 1-3 if this code executive on NuMaker-M467HJ V1.0 board. Another NuMaker-M467HJ board can be used to receive and print messages.

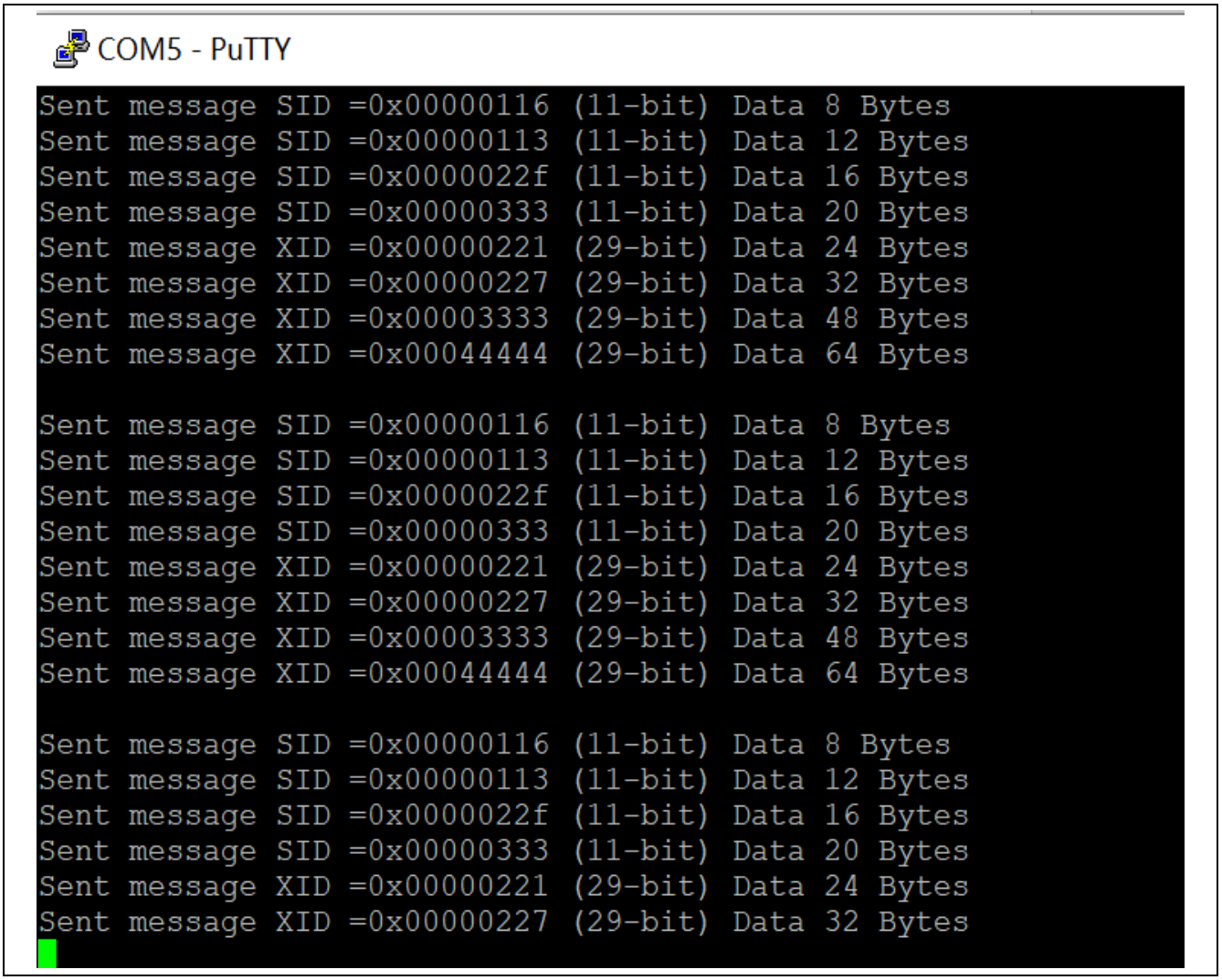

Figure 1-3 Result Execution and Printed Messages

Messages can be received and printed by the other NuMaker-M467HJ board which executes code:M460_Series_BSP_CMSIS_V3.00.001\SampleCode\StdDriver\CANFD_CANFD_TxRxINT. On that receiving board, please don't forget to configure CANFD interface on PJ10, PJ11 pin. Received Messages from CANFD will be printed from UART0 which is shown in Figure 1-4

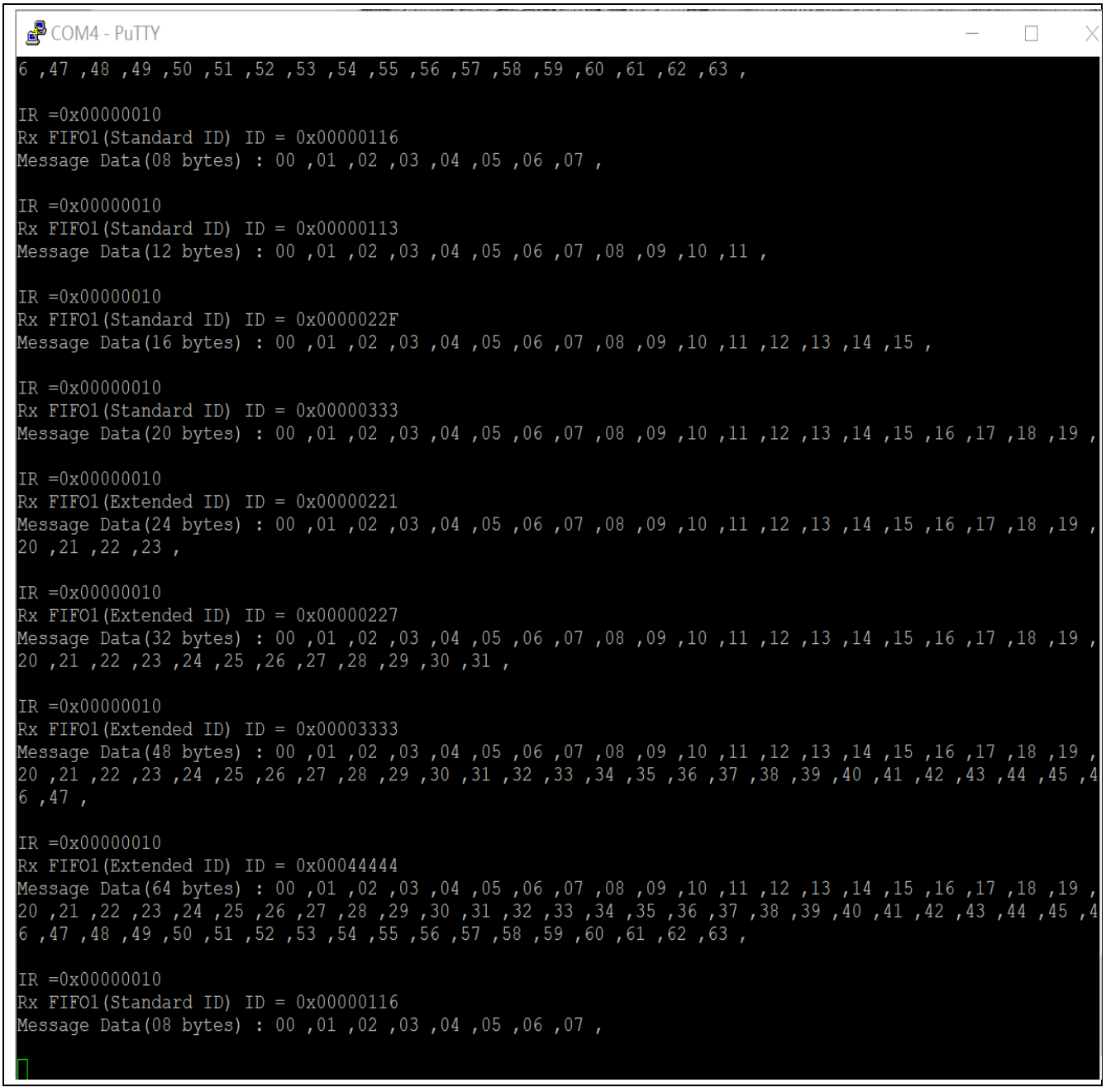

Figure 1-4 Messages on CAN Bus

2. Code Description

Configure CANFD bit time. The code in main.c is as follows.

```
/// input: psCanfd, The value must be CANFD0,CANFD1,CANFD2 or CANFD3
void CANFD_BitRate_Init(CANFD_T *psCanfd)
{                                       // CANFD_CLk = 200M/5 = 40MHz
    psCanfd->NBTP = (3 << 25) +         // NSJW  = 3+1 =4  CLK
                    (1 << 16) +         // NBRP  = 1+1 =2 ,     // prescaler = 2
                    (13 << 8) +         // NTSG1 =13+1=14 CLK  // One bit =1+14+5 =20 CLK
                    (5 - 1);            // NTSG2 = 5  CLK

    psCanfd->DBTP = ((1 - 1) << 16) +   // DBRP  = 1 prescaler
                    ((6 - 1) << 8) +    // DTSG1 = 6 CLK       // One bit = 1+6+3 =10 CLK
                    ((3 - 1) << 4) +    // DTSG2 = 3 CLK
                    (2 - 1);            // DSJW  = 2CLK
}
```

CANFD takes sample between DTSG1 and DTSG2 when data field is received, and between NTSG1 and NTSG2 when other field received. The total clock quantity of one bit must be added with one clock synchronous field, as shown in Figure 2-1.

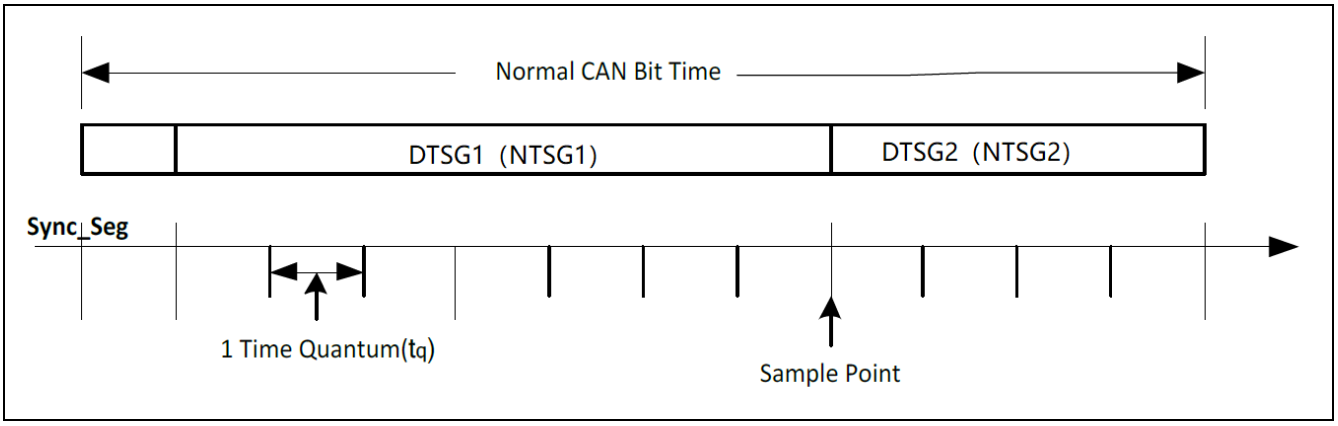

Figure 2-1 Bit Time Format

The first 2words of message is used to configure message ID, data length, etc. That is performed by function CANFD_Tx_Init(). Data is added into the message just before transmission.

```
    // There configured 8 Txbuff elements
void CANFD_Tx_Init(CANFD_T *psCanfd, int16_t TxBuff_Quantity, int16_t DLC_Max, uint16_t
TxBuff_StartAddr)
{
    psCanfd->TXBC = (TxBuff_Quantity << CANFD_TXBC_NDTB_Pos)
                    + TxBuff_StartAddr ;                    // Start address of messages RAM

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

//  u32Config = 0x00300000 mean EFC=0 · FDF=1, BRS=1 // If DLC <8,the Byte quantiry =DLC

    CANFD_TxBuff[0].u32Id       = 0x111 << 18 ;                     // SID = 0x111
```

```
    CANFD_TxBuff[0].u32Config   = 0x00300000 + 0x00080000 ;      // DLC=8 = 8 Bytes
//  CANFD_TxBuff[0].u32Config = 0x00300000 + 0x00050000 ;  // It has DLC data if(DLC <=8)

    CANFD_TxBuff[1].u32Id       = 0x113 << 18 ;                    // SID = 0x113
    CANFD_TxBuff[1].u32Config   = 0x00300000 + 0x00090000 ;      // DLC=9 = 12 Bytes
    CANFD_TxBuff[2].u32Id       = 0x22F << 18 ;                    // SID = 0x22F
    CANFD_TxBuff[2].u32Config   = 0x00300000 + 0x000A0000 ;      // DLC=A = 16 Bytes
    CANFD_TxBuff[3].u32Id       = 0x333 << 18 ;                    // SID = 0x333
    CANFD_TxBuff[3].u32Config   = 0x00300000 + 0x000B0000 ;      // DLC=B = 20 Bytes

    CANFD_TxBuff[4].u32Id       = 0x40000000 + 0x221 ;             // XID = 0x221
    CANFD_TxBuff[4].u32Config   = 0x00300000 + 0x000C0000 ;      // DLC=C = 24 Bytes
    CANFD_TxBuff[5].u32Id       = 0x40000000 + 0x227 ;             // XID = 0x227
    CANFD_TxBuff[5].u32Config   = 0x00300000 + 0x000D0000 ;      // DLC=D = 32 Bytes
    CANFD_TxBuff[6].u32Id       = 0x40000000 + 0x3333 ;            // XID = 0x3333
    CANFD_TxBuff[6].u32Config   = 0x00300000 + 0x000E0000 ;      // DLC=E = 48 Bytes
    CANFD_TxBuff[7].u32Id       = 0x40000000 + 0x44444 ;           // XID = 0x44444
    CANFD_TxBuff[7].u32Config   = 0x00300000 + 0x000F0000 ;      // DLC=F = 64 Bytes
}
```

Up to 32 messages can be configured. When one message is expected to transmit, please set the relative bit of register TXBAR.

```
while (CANFD_GET_COMMUNICATION_STATE(psCanfd) != eCANFD_IDLE)
{
   if (--u32TimeOutCnt == 0) break;            // waiting for CANFD idle
}

   psCanfd->TXBAR = 1 << 3 ;       // Tx_Element_Buff 3 Request transmitting
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M460_Series_BSP_CMSIS_V3.00.001
- IDE version
  - Keil uVersion 5.38

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M467HJ V1.0
- Pin Connect
  - Two NuMaker-M467HJ boards are connected to each other with CANFD interface as to show the execution result. Messages will be printed from UART0.

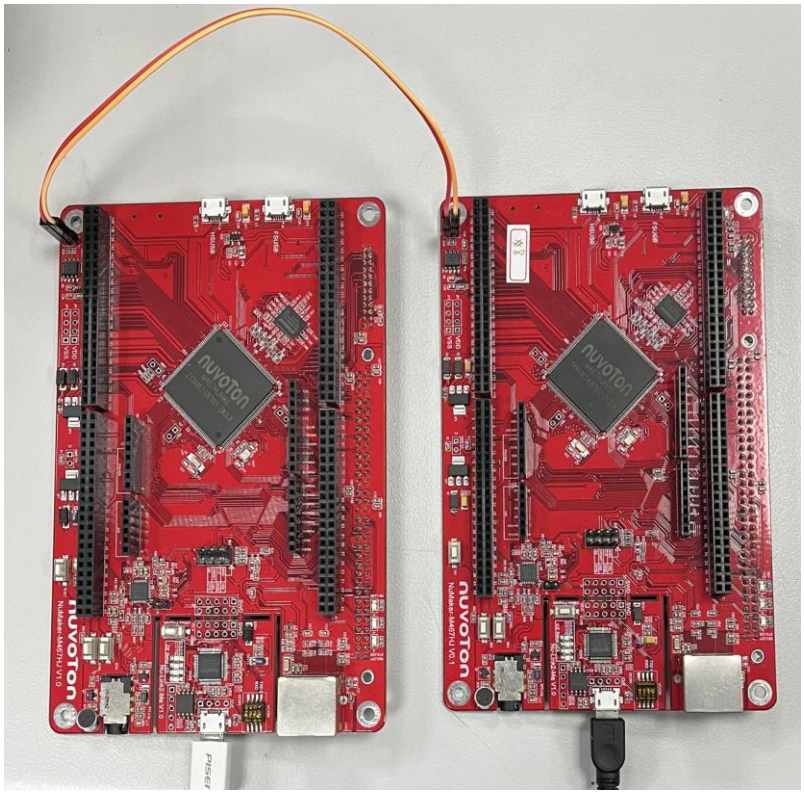

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_ M460_CANFD_Tx_Simply _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

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *EC_M460_CANFD_Tx_Simply.uvprojx*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2023.02.20	1.00	Initial version.

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.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.