M460 CANFD 收發器延遲補償功能

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式介紹如何使用 M460 CAN FD 收發器延遲補償功能
BSP 版本	M460_Series_BSP_CMSIS_V3.00.002
開發平臺	NuMaker-M476HJ_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. 概述

本範例程式介紹了 CAN FD 的收發器延時補償功能。以確保 CANFD 在高速通訊發訊息的時候，避免發送器進入 BUSOFF。

Transceiver Delay Compensation 收發器延遲補償(簡稱 TDC)，主要解決收發器迴路物理延遲，對信號回讀可靠性校驗的問題。

為了方便理解 TDC 的原理本範例先從以下幾點來介紹 TDC 功能，然後再介紹程式配置方式：

- 什麼是收發延遲
- 為什麼要用發送器延遲補償
- TDC 的設置方法
- 實例分析

本範例程式需要配一個 CAN FD 的調試工具，用於觀測收發的資料。本例中已經有通訊信號的捕捉圖示，如果需要進行驗證，建議搭配示波器進行觀測。

1.1 什麼是收發延遲

當我們進行數位通訊的時候信號通過 MCU 的 TTL 電平進行傳輸，而單一的資料信號不適合進行長距離、高可靠性的傳輸，這時我們需要對信號做一些變換，讓變換後的信號適合長距離的可靠通訊。

本文中的使用 CAN 差分信號進行對稱傳輸，圖 1-1 是文中 CAN 轉換晶片的內部結構和傳輸延遲示意：

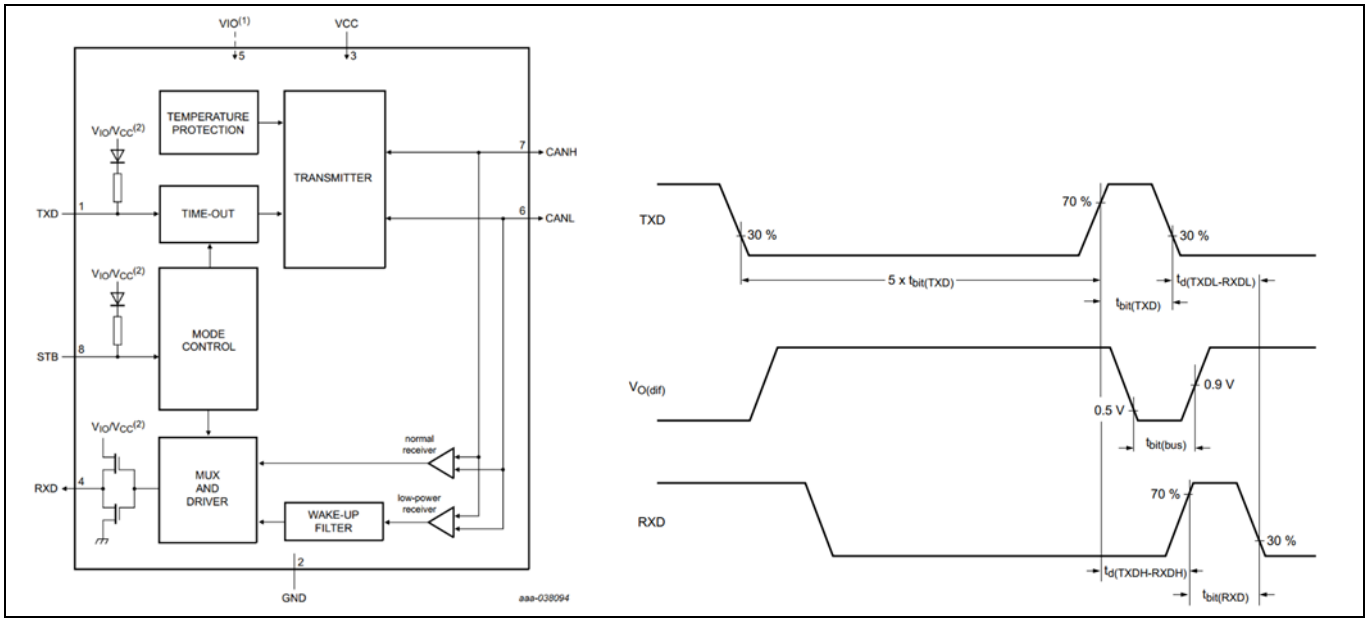


圖 1-1 內部結構和傳輸延遲

TXD 為控制器信號發送，Vo(dif)為外部匯流排的實體信號，RXD 為控制器信號接收。

圖 1-1 僅僅是轉換器的延遲，同樣在環境惡劣的通訊環境中，我們為了保護控制器還會加入信號隔離器，以免外界雜訊干擾控制器的運行，但是同樣也會帶來更長時間的信號延遲。

圖 1-2 是本例中使用的隔離器的傳輸延遲時間：

Parameter	Symbol	Min	Typ	Max	Unit	Test Conditions
ADuM120xBR						
Minimum Pulse Width[2]	PW			100	ns	C_L = 15 pF, CMOS signal levels
Maximum Data Rate[3]		10			Mbps	C_L = 15 pF, CMOS signal levels
Propagation Delay[4]	t_{PHL}, t_{PLH}	20		60	ns	C_L = 15 pF, CMOS signal levels
Pulse-Width Distortion, $\|t_{PLH} - t_{PHL}\|$[4]	PWD			3	ns	C_L = 15 pF, CMOS signal levels
Change Versus Temperature			5		ps/°C	C_L = 15 pF, CMOS signal levels
Propagation Delay Skew[5]	t_{PSK}			22	ns	C_L = 15 pF, CMOS signal levels
Channel-to-Channel Matching, Codirectional Channels[6]	t_{PSKCD}			3	ns	C_L = 15 pF, CMOS signal levels
Channel-to-Channel Matching, Opposing Directional Channels[6]	t_{PSKOD}			22	ns	C_L = 15 pF, CMOS signal levels
Output Rise/Fall Time (10% to 90%)	t_R/t_F		3.0		ns	C_L = 15 pF, CMOS signal levels
ADuM120xAR						
Minimum Pulse Width[2]	PW			1000	ns	C_L = 15 pF, CMOS signal levels
Maximum Data Rate[3]		1			Mbps	C_L = 15 pF, CMOS signal levels
Propagation Delay[4]	t_{PHL}, t_{PLH}	50		150	ns	C_L = 15 pF, CMOS signal levels
Pulse-Width Distortion, $\|t_{PLH} - t_{PHL}\|$[4]	PWD			40	ns	C_L = 15 pF, CMOS signal levels
Propagation Delay Skew[5]	t_{PSK}			100	ns	C_L = 15 pF, CMOS signal levels
Channel-to-Channel Matching[6]	$t_{PSKCD/OD}$			50	ns	C_L = 15 pF, CMOS signal levels
Output Rise/Fall Time (10% to 90%)	t_R/t_F		10		ns	C_L = 15 pF, CMOS signal levels

圖 1-2 隔離器的傳輸延遲

從圖 1-2 我們可以看到 10Mbps（20~60ns）的比 1Mbps（50~150ns）的隔離器延時長了近一倍。

綜上在實際的應用中一共存在3種延遲：

- 收發器迴路的延遲（收發器接收發送資料後再回傳給控制器）
- 發送延遲（發送隔離）
- 接收延遲（接收隔離）

1.2 為什麼要用收發器延遲補償

CAN 通訊綫的設計之初就對其通訊提出了高性能和可靠性的要求，所以在 CAN 協議中規定：發送方發送資料時，需檢測回傳的資料與發送是否一致，若不一致則產生錯誤幀，就是用回讀的方式判斷信號是否發送成功。

CAN 檢測到發送資訊回讀不正確並達到一定數量後，控制器會認為匯流排異常進入 BUSOFF 狀態，並將自己暫停不影響匯流排其他控制器的通訊。

在經典 CAN ISO11898-5 的白皮書中規定了總迴路的延時為 255ns。這個規定在低速模式下能夠確保迴路的延時不會對資料回讀造成影響。但是當 CAN FD 模式在 5Mbps 資料段通訊，實際位元發送時間僅有 200ns，如果不考慮收發延遲，那麼匯流排將完全不能正常收發資料了。另外在章節 1.1 的分析中幾類延遲在實際應用時疊加起來就很有可能會超過 255ns。

針對上述的問題，BOSCH 設計了一個叫 Transceiver Delay Compensation 的功能（簡稱 TDC），這個功能可以自動去測量電路中的延時時間，並且通過延時時間重新設置資料段的信號取樣點位置。這個取樣點的位置定義為二次取樣點 Secondary Sample Point (簡稱 SSP)。

圖 1-3 展示了自動延時和 SSP 取樣點的計算方法。

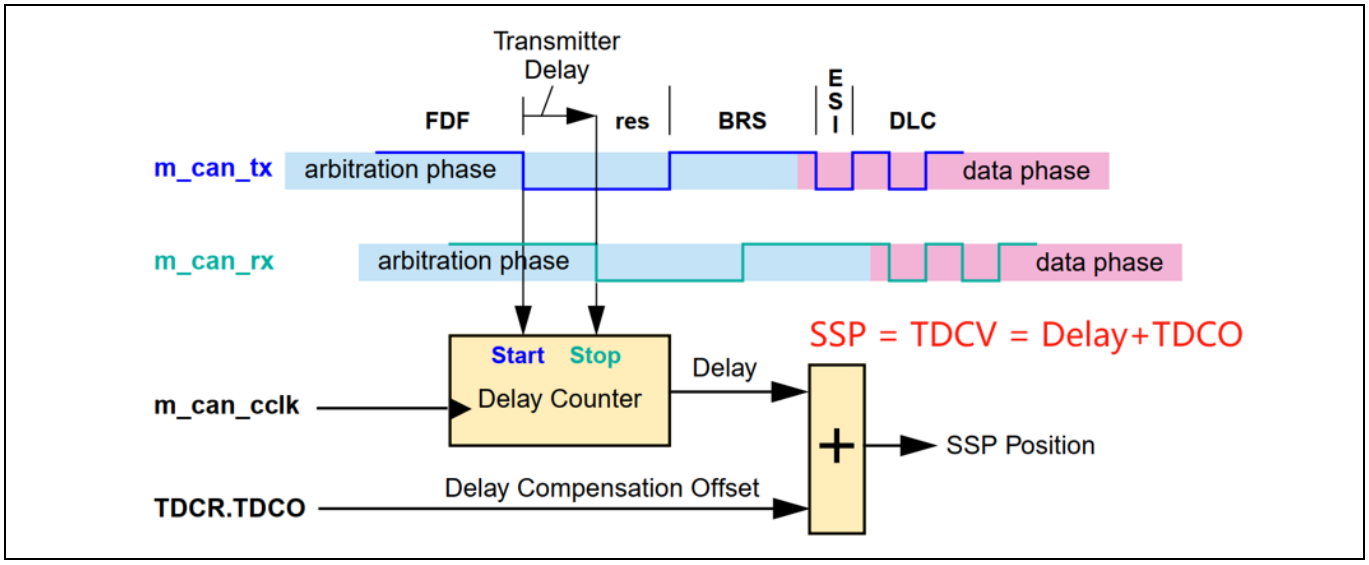


圖 1-3 自動計算 SSP 的示意圖

1.3 TDC 的設置方法：

TDC 主要是用來設置二次取樣點 SSP，此功能涉及 3 個暫存器，4 個參數值。

TDCF，TDCO 和 TDCV 均以最小時間 tq 為計數單位，計算方法同串列傳輸速率暫存器。

- DBTP 暫存器中的 TDC 位：此位表示延時取樣功能的使能，大於 1Mbps 通訊時建議打開。
- TDCR 暫存器中的 TDCO 和 TDCF：二次取樣點 SSP = Delay+ TDCO 。

  TDCO ：二次取樣延時偏移量，一般參考 DTSEG1 值進行設置。

  TDCF ：最小二次取樣點。同時這個值是必須大於 TDCO 的 。

  在不知道具體延時參數的時候可以默認 TDCF =DTSEG1+1 ，確保 TDCF > TDCO。

  TDCF 可根據實際硬體的測試情況做些調整，硬體確認後物理延時也就定下來了。
- PSR 暫存器中的 TDCV：自動計算的延時取樣點 TDCV = TDCO + 回讀物理延時（Delay）。

  當 TDCO = 0 時實際測量到的就是回讀的物理延時（Delay）。

  當 TDCF ≥ TDCV 時，二次取樣點 SSP = TDCF 。

  當 TDCF < TDCV 時，二次取樣點 SSP = TDCV 。

1.4 實例分析

1.4.1 波形分析

實驗環境：

- 10Mbps 的隔離晶片 ADUM1201BRZ（延遲 20~60ns）
- 1Mbps 的隔離晶片 ADUM1201ARZ（延遲 50~150ns）
- CAN 收發器：TJA1462A
- MCU 控制：M467HJHAN

隔離器的延時數據見圖 1-2.

CAN 收發器 TJA1462A 的延時資料圖 1-4：

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{d(TXDL-RXDL)}$	delay time from TXD LOW to RXD LOW	Normal mode	-	-	190	ns
		Normal mode; V_{CC} = 4.5 V to 5.5 V	-	-	255	ns
$t_{d(TXDH-RXDH)}$	delay time from TXD HIGH to RXD HIGH	Normal mode	-	-	190	ns
		Normal mode; V_{CC} = 4.5 V to 5.5 V	-	-	255	ns

圖 1-4 CAN 收發器延時資料

實驗方法：

M467 開發板 CAN 發送一幀資料，用示波器 2 個通道分別接 M467CAN 的 RXD 和 TXD，發送資料的時候同時測量 TXD 和 RXD 的信號，記錄測量 1Mbps 隔離和 10Mbps 隔離的延遲差異。

圖 1-5 為 10Mbps 隔離晶片 CANFD 收發的一幀資料：

標準幀，位址 0x000，資料長度 12byte，可變串列傳輸速率。

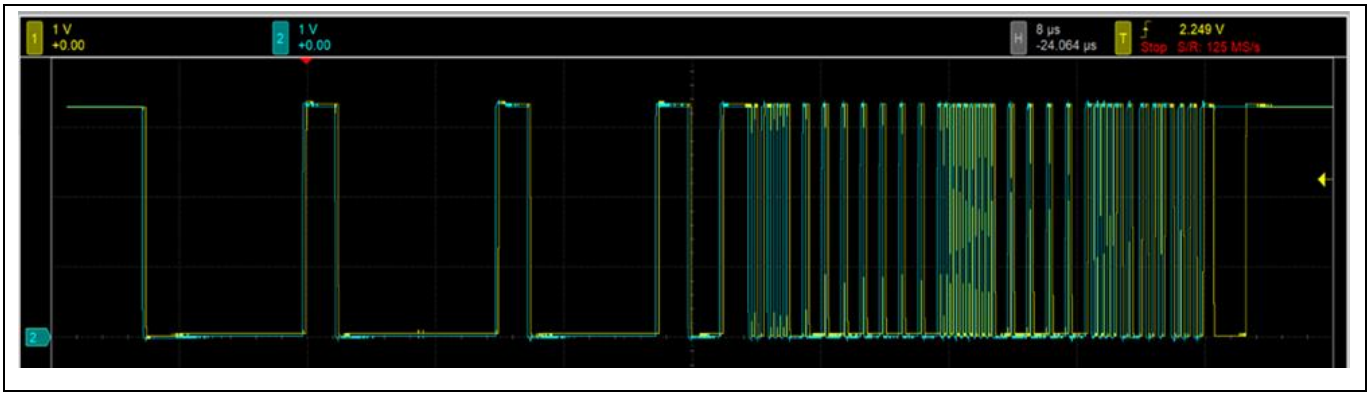


圖 1-5 10Mbps 隔離的 CAN FD 一幀收發資料波形

測量 MCU 端 TXD 和 RXD 接腳，可以很明顯的看到 TXD 藍線和 RXD 黃線信號有疊影。

放大資料段 1 個位的波形圖 1-6：

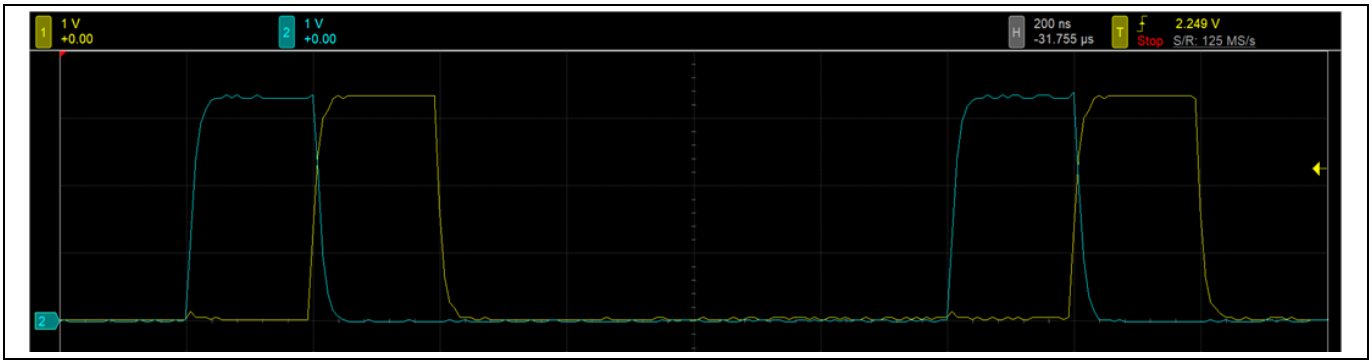


圖 1-6 10Mbps 隔離資料段 1 個位的收發波形

從圖 1-6 可以很明顯的看到 RXD 黃線延遲於 TXD 藍線約 200ns 才到 MCU 端，這個時候如果依然還用發射時序的取樣點，那麼會導致信號無法取樣正確。這個時候 TDC 功能非常就重要了。

當使用 1Mbps 隔離晶片時資料段 1 個位波形的延時測試如圖 1-7：

ADUM1201ARZ（延遲 50~150ns），CAN 收發器同樣為 TJA1462A 。

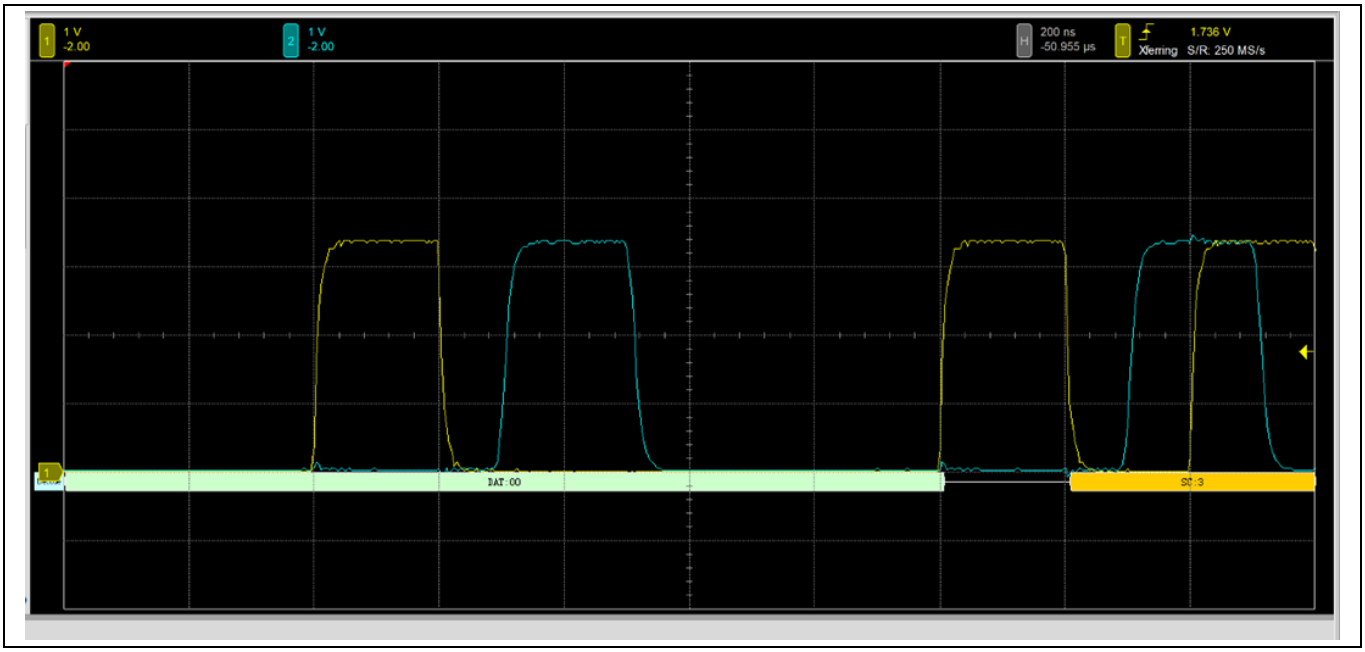


圖 1-7 1Mbps 隔離資料段 1 個位的收發波形

從圖 1-7 中可以看到 1Mbps 隔離 RXD 藍線延遲於 TXD 黃線近 300ns，不同的隔離器延時性能差異較大。

當硬體設計完成之後，這個物理延時時間也就基本定下來了，可以透過示波器來確認準確的延時時間，即確認 TDCF 的值。

1.4.2 TDC 的設置實例

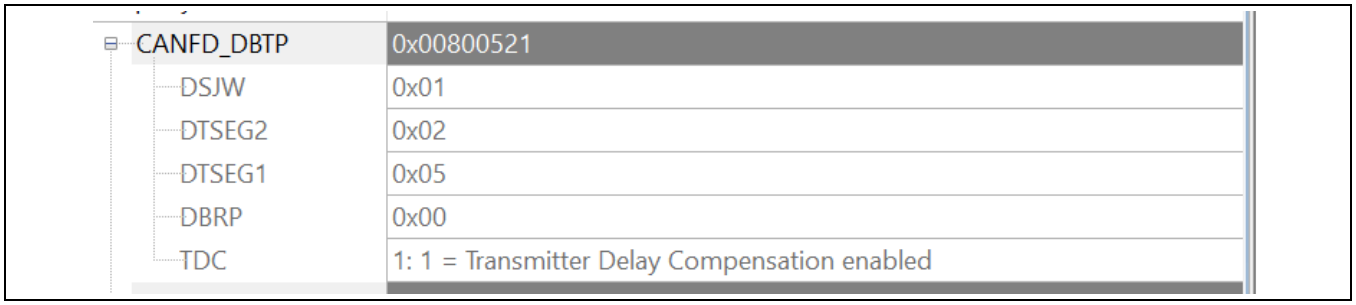
實例通過 Nu-Link 截圖觀測 Keil 內實際 TDC 暫存器的設置，並且校驗演算過程是否正確。

仲裁段串列傳輸速率為 500bkps，資料段為 5Mbps，資料段串列傳輸速率暫存器：

總 Tq =（DTSEG1+1 ）+（DTSEG2+1）+ DSJW =（5 + 1）+（2 + 1 ）+ 1 = 10tq

1 個位的時間為 1 個總 Tq，總 Tq 拆分成 10 個元素，用這個 10 個元素來進行取樣點的設置。

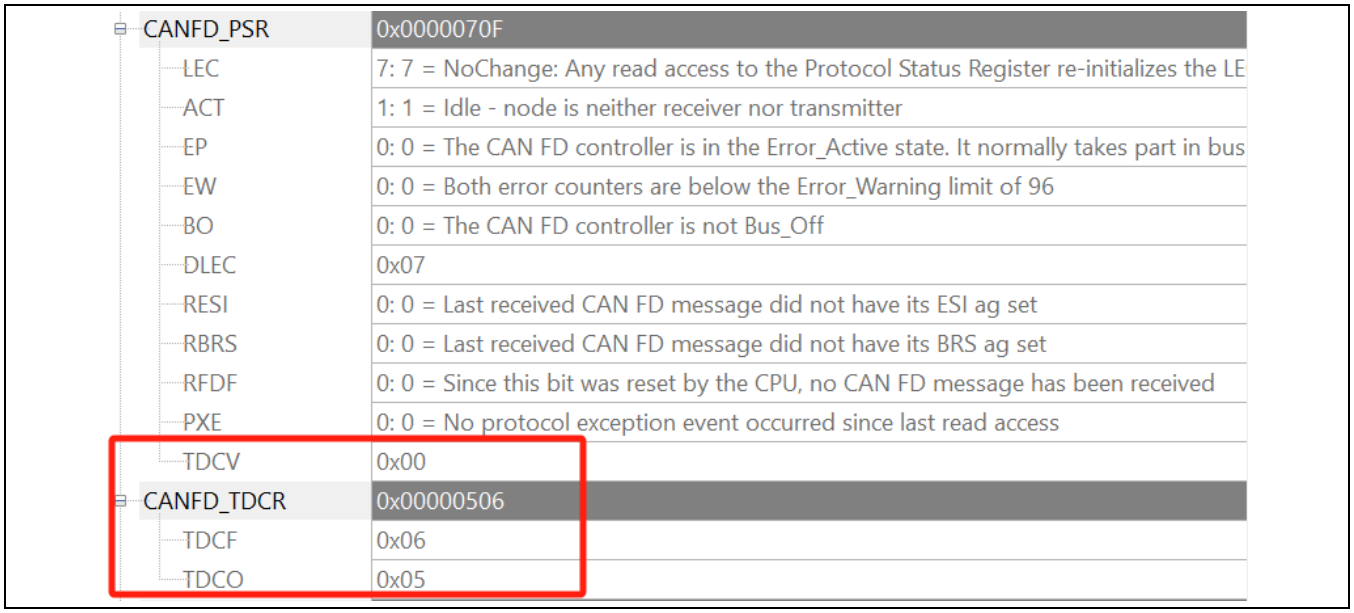
實際的暫存器設置如圖 1-8：



CANFD_DBTP	0x00800521
DSJW	0x01
DTSEG2	0x02
DTSEG1	0x05
DBRP	0x00
TDC	1: 1 = Transmitter Delay Compensation enabled

圖 1-8 CAN FD 資料段串列傳輸速率設置暫存器

按照上文 1.3 TDC 的設置方法進行暫存器設置：



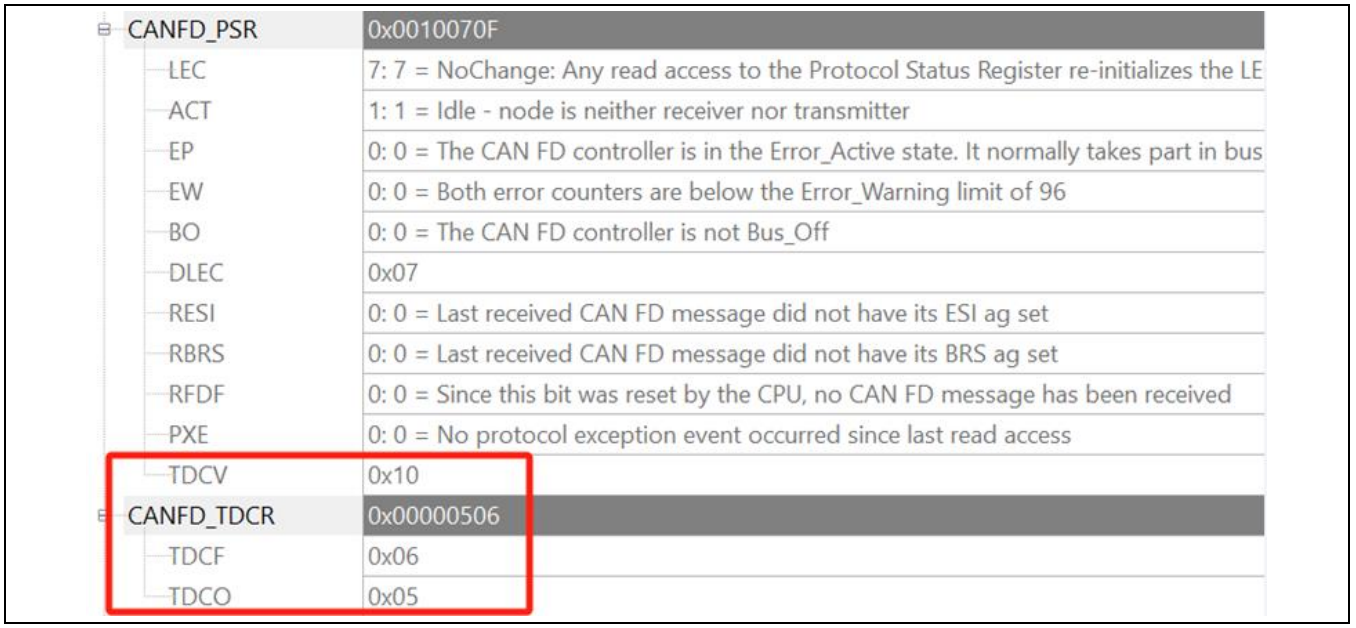
CANFD_PSR	0x0000070F
LEC	7: 7 = NoChange: Any read access to the Protocol Status Register re-initializes the LE
ACT	1: 1 = Idle - node is neither receiver nor transmitter
EP	0: 0 = The CAN FD controller is in the Error_Active state. It normally takes part in bus
EW	0: 0 = Both error counters are below the Error_Warning limit of 96
BO	0: 0 = The CAN FD controller is not Bus_Off
DLEC	0x07
RESI	0: 0 = Last received CAN FD message did not have its ESI ag set
RBRS	0: 0 = Last received CAN FD message did not have its BRS ag set
RFDF	0: 0 = Since this bit was reset by the CPU, no CAN FD message has been received
PXE	0: 0 = No protocol exception event occurred since last read access
TDCV	0x00
CANFD_TDCR	0x00000506
TDCF	0x06
TDCO	0x05

圖 1-9 CAN FD 初始化後 TDC 相關暫存器狀態

CAN 初始化此時還未發送資料：

TDCV = 0； TDCO = DTSEG1 ；TDCF = DTSEG1 + 1 。

CAN 發送一次資料後，匯流排自動測量到延時，並將測量值存入 TDCV 暫存器：



CANFD_PSR	0x0010070F
LEC	7: 7 = NoChange: Any read access to the Protocol Status Register re-initializes the LE
ACT	1: 1 = Idle - node is neither receiver nor transmitter
EP	0: 0 = The CAN FD controller is in the Error_Active state. It normally takes part in bus
EW	0: 0 = Both error counters are below the Error_Warning limit of 96
BO	0: 0 = The CAN FD controller is not Bus_Off
DLEC	0x07
RESI	0: 0 = Last received CAN FD message did not have its ESI ag set
RBRS	0: 0 = Last received CAN FD message did not have its BRS ag set
RFDF	0: 0 = Since this bit was reset by the CPU, no CAN FD message has been received
PXE	0: 0 = No protocol exception event occurred since last read access
TDCV	0x10
CANFD_TDCR	0x00000506
TDCF	0x06
TDCO	0x05

圖 1-10 CAN FD 發送資料後 TDC 相關暫存器狀態

發送資料後：TDCV = 16； TDCO = DTSEG1 ；TDCF = DTSEG1 + 1。

資料段 5Mbps 根據波形和參數計算（驗證 1.3 計算方法）：

資料段總 Tq = 10tq = 200ns（5Mbps）；單個元素 1tq = 20ns 。

TDCO= DTSEG1 = 5tq 。

TDCF = DTSEG1 + 1 = 6tq 。

採用 10Mbps 隔離時二次取樣點的位置在（320ns 處）：

TDCV = 16tq

= 11tq（delay） + 5tq（TDCO）

= 11*20ns + 5*20ns

= 320ns

此時 TDCV(16tq) > TDCF(6tq) 二次取樣點按照 TDCV 來計算如下圖：

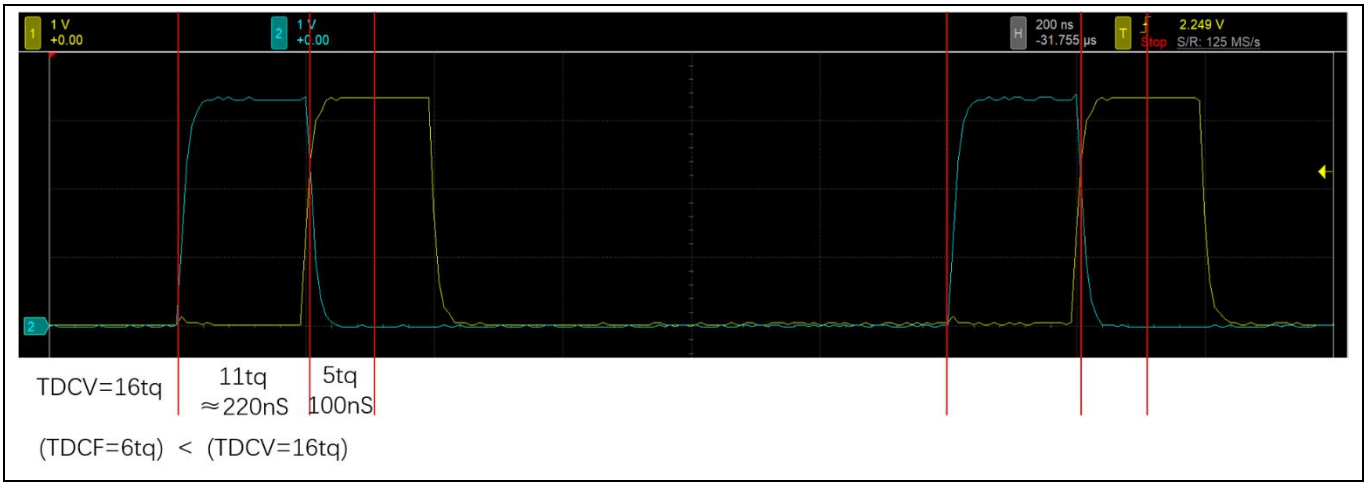


圖 1-11 CANFD 的二次取樣點

由圖 1-11 看二次取樣點落在黃線端紅線處，可見 TDCV= 16tq 能夠正確取樣到回傳的位電平。硬體定型後通過上圖的測量可以選擇一個合適值設置 TDCF。

注：

- TDCV 的測量值和示波器的觀測略有誤差
- 示波器 1 格為 200ns，實際取樣點在 280ns 到 360ns 都能正確取樣(14≤TDCF≤18)
- 藍線為 TXD，黃線為 RXD

2. 程式介紹

首先確認常式串列傳輸速率是否設置成 5Mbps，修改串列傳輸速率位於 *main.c* 檔中的 *CANFD_Init()*函式內。

```
/*---------------------------------------------------------------------------*/
/*                          Init CAN FD0                                     */
/*---------------------------------------------------------------------------*/
void CANFD_Init(void)
{
    printf("+-----------------------------------------------------------------+\n");
    printf("|     Pin Configure                                               |\n");
    printf("+-----------------------------------------------------------------+\n");
    printf("|    CAN_TXD                                 CAN_TXD(Any board)   |\n");
    printf("|    CAN_RXD                                 CAN_RXD(Any board)   |\n");
    printf("|          |-----------| CANBUS  |-----------|                     |\n");
    printf("|  ------> |           |<------->|           |<------              |\n");
    printf("|    CAN_TX|   CANFD   | CANFD_H |    CANFD  |CAN_TX               |\n");
    printf("|          |Transceiver|         |Transceiver|                     |\n");
    printf("|  <------ |           |<------->|           |------>              |\n");
    printf("|    CAN_RX|           | CANFD_L |           |CAN_RX               |\n");
    printf("|          |-----------|         |-----------|                     |\n");
    printf("|                                                                 |\n");
    printf("+-----------------------------------------------------------------+\n\n");

    /*Get the CAN FD configuration value*/
    CANFD_GetDefaultConfig(&sCANFD_Config, CANFD_OP_CAN_FD_MODE);
    sCANFD_Config.sBtConfig.sNormBitRate.u32BitRate =  500000;// NormBitRate = 500Kbps
    sCANFD_Config.sBtConfig.sDataBitRate.u32BitRate = 5000000;// DataBitRate = 5Mbps

    /*Open the CAN FD feature*/
    CANFD_Open(g_pCanfd, &sCANFD_Config);
……
```

CAN FD 的 TDO 功能開啟的函式，程式位於 *canfd.c*，當 DataBitRate 大於 1Mbps 時自動開啟 TDC 功能。

```
/* @brief       Open TDC Function
 *
 * @param[in]   psCanfd     The pointer to CAN FD module base address.
 * @param[in]   psCanfdStr  message ram setting and bit-time setting
 *
 * @return      None.
 *
 * @details     Open Transceiver Delay Compensation
 *               Secondary sampling point  = SSP = TDCV = Delay + TDCO;
 */
void CANFD_Open_TDC(CANFD_T *psCanfd, CANFD_FD_T *psCanfdStr)
{
    uint8_t tdco;//TDCO = DTSEG1 + 1
```

```
    if(psCanfdStr->sBtConfig.sDataBitRate.u32BitRate> 1000000) //bitrate > 1Mbps
      {
     //Get minimum tdco
     tdco = (((psCanfd->DBTP) & CANFD_NBTP_NTSEG1_Msk)>>CANFD_NBTP_NTSEG1_Pos);

     //TDCO need to measure reference DBTP:NTSEG1
     psCanfd->TDCR  = (psCanfd->TDCR & (~CANFD_TDCR_TDCO_Msk)) |
                               (tdco << CANFD_TDCR_TDCO_Pos);

     //TDCF mast greater than TDCO,so set TDCF = TDCO + 1
     psCanfd->TDCR  = (psCanfd->TDCR & (~CANFD_TDCR_TDCF_Msk)) |
                           ((tdco + 1) << CANFD_TDCR_TDCF_Pos);

     //Open compensation function
     psCanfd->DBTP  |= CANFD_DBTP_TDC_Msk;

      }
}
```

此函式需要放置於 *CANFD_Open(CANFD_T *psCanfd, CANFD_FD_T *psCanfdStr)*函式中，在初始化 CAN FD 時鐘(CANFD_CalculateTimingValues)之後進行串列傳輸速率的判斷。

```
/**
 * @brief       Config message ram and Set bit-time.
 *
 * @param[in]   psCanfd     The pointer to CAN FD module base address.
 * @param[in]   psCanfdStr  message ram setting and bit-time setting
 *
 * @return      None.
 *
 * @details     Converts a Data Length Code into a number of message bytes.
 */
void CANFD_Open(CANFD_T *psCanfd, CANFD_FD_T *psCanfdStr)
{
    uint32_t u32RegLockLevel = SYS_IsRegLocked();

    if (u32RegLockLevel)
        SYS_UnlockReg();

    if (psCanfd == (CANFD_T *)CANFD0)
    {
        CLK_EnableModuleClock(CANFD0_MODULE);
        SYS_ResetModule(CANFD0_RST);
    }
    else if (psCanfd == (CANFD_T *)CANFD1)
    {
        CLK_EnableModuleClock(CANFD1_MODULE);
        SYS_ResetModule(CANFD1_RST);
    }
    else if (psCanfd == (CANFD_T *)CANFD2)
    {
        CLK_EnableModuleClock(CANFD2_MODULE);
        SYS_ResetModule(CANFD2_RST);
    }
    else if (psCanfd == (CANFD_T *)CANFD3)
```

```
    {
        CLK_EnableModuleClock(CANFD3_MODULE);
        SYS_ResetModule(CANFD3_RST);
    }
    else
    {
        if (u32RegLockLevel)
            SYS_LockReg();

        return;
    }

    /* configuration change enable */
    psCanfd->CCCR  = CANFD_CCCR_INIT_Msk;
    psCanfd->CCCR |= CANFD_CCCR_CCE_Msk;

    if(psCanfdStr->sBtConfig.bBitRateSwitch)
    {
        /* enable FD and baud-rate switching */
        psCanfd->CCCR |= CANFD_CCCR_BRSE_Msk;
    }

    if(psCanfdStr->sBtConfig.bFDEn)
    {
        /*FD Operation enabled*/
        psCanfd->CCCR |= CANFD_CCCR_FDOE_Msk;
    }

    /*Clear the Rx Fifo0 element setting */
    psCanfd->RXF0C = 0;
    /*Clear the Rx Fifo1 element setting */
    psCanfd->RXF1C = 0;

    /* calculate and apply timing */
    if(CANFD_CalculateTimingValues(psCanfd,
         psCanfdStr->sBtConfig.sNormBitRate.u32BitRate,
         psCanfdStr->sBtConfig.sDataBitRate.u32BitRate,
         SystemCoreClock, &psCanfdStr->sBtConfig.sConfigBitTing))
    {
        CANFD_SetTimingConfig(psCanfd, &psCanfdStr->sBtConfig.sConfigBitTing);
    }

     /* open TDC function */
     CANFD_Open_TDC(psCanfd,psCanfdStr);
    /* 7=>Default TX maximum data length(64Byte) */
     psCanfd->TXESC = (7 << CANFD_TXESC_TBDS_Pos);
    /* 7=>Default RX maximum data length(64Byte) */
    psCanfd->RXESC = (psCanfd->RXESC & (~CANFD_RXESC_F1DS_Msk)) |
                                    (7 << CANFD_RXESC_F1DS_Pos);


    if (u32RegLockLevel)
        SYS_LockReg();
……
```

通過序列埠的提示進行 CAN 資料的發送，按數位 8 選擇需要發送的幀，可以看到資料能夠正常發送出去：

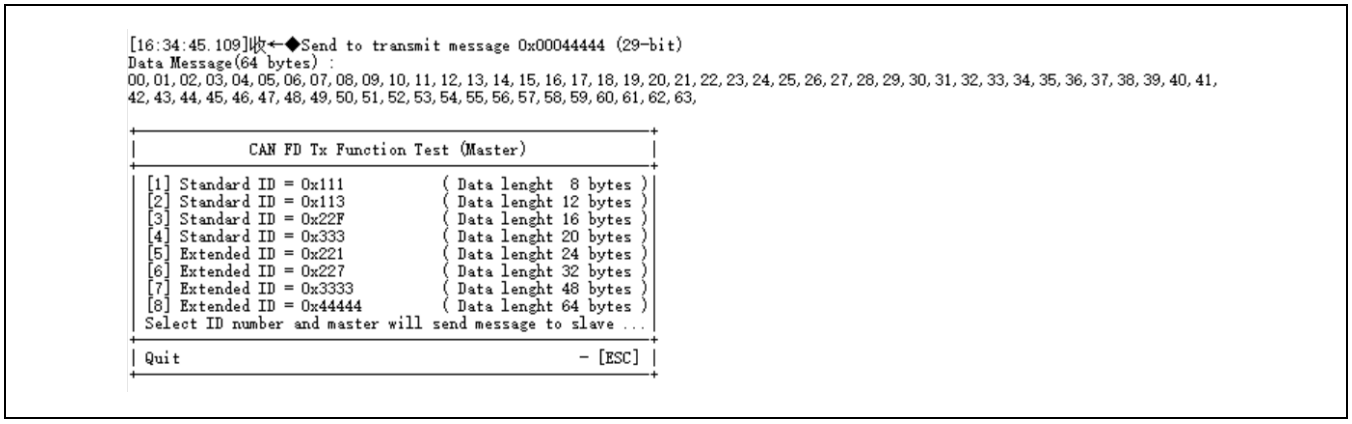


圖 2-1 常式 CAN 發送一幀資料

圖 2-1 CAN FD 序號 8 的數據幀為：

Extended ID=0x44444 變速幀（Data length 64 bytes）

64 個位元組的資料從 0 開始遞增到 63。

CAN 工具接收到的 CAN 資料：

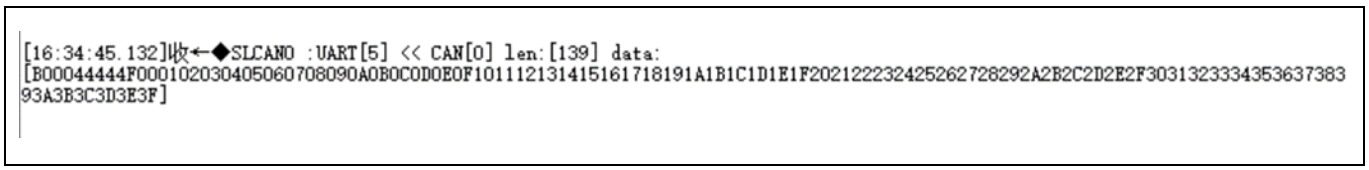


圖 2-2 CAN 工具接收端收到的資料

圖 2-2 中方括號[xxx]內為收到的資料，資料分為 4 個部分：

- 框架類型 ：B -> 變速擴展幀
- 幀 ID ：00044444 -> 0x44444
- 長度 ：F -> 64bytes
- 數據 ：

  00、01、02、03、04、05、06、07、08、09、0A、0F、0C、01、0E、0F、

  10、11、12、13、14、15、16、17、16、19、1A、1B、1C、1D、1E、1F、

  20、21、22、23、24、25、26、27、26、29、2A、2B、2C、2D、2E、2F、

  30、31、32、33、34、35、36、37、38、39、3A、3B、3C、3D、3E、3F

- 5Mbps 可變序列傳輸速率發送腳的資料信號：

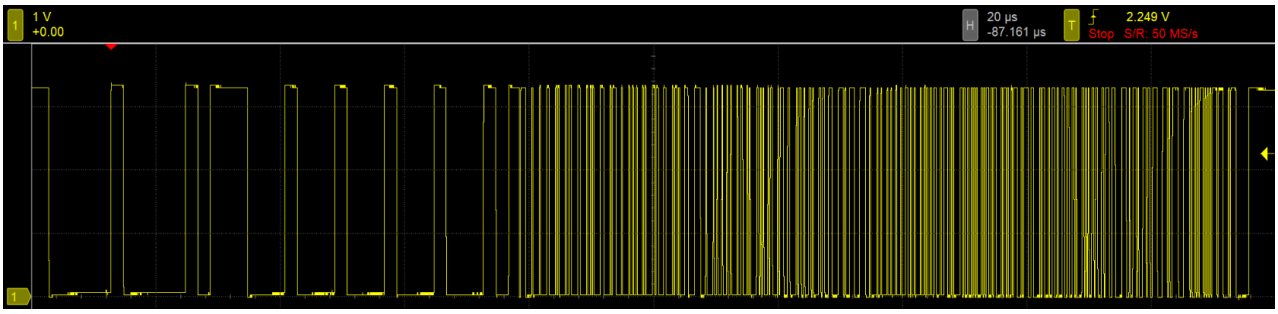


圖 2-3 CAN 發射腳的資料波形

注：如果 TDC 功能沒有打開，500kbps 仲裁段的資料會發送出去，但是由於 5Mbps 的資料段回讀不正確，到資料段就會停止發送並報錯（上圖中信號頻率較高的部分）。圖 2-3 為完整的發送了一幀資料。

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 5.30

3.2 硬體需求

- 電路元件
  - NuMaker-M476HJ_V1.0
  - CANFD 分析儀（圖示中的分析儀可以用其他品牌代替）
- 線路示意圖
  - 將 M476HJ 開發板 UART0 連接到 PC UART，以顯示範例程式的執行結果。
  - 將 CAN_Tool 和 M476 的 CANH/CANL 對接以確保能夠正常通訊

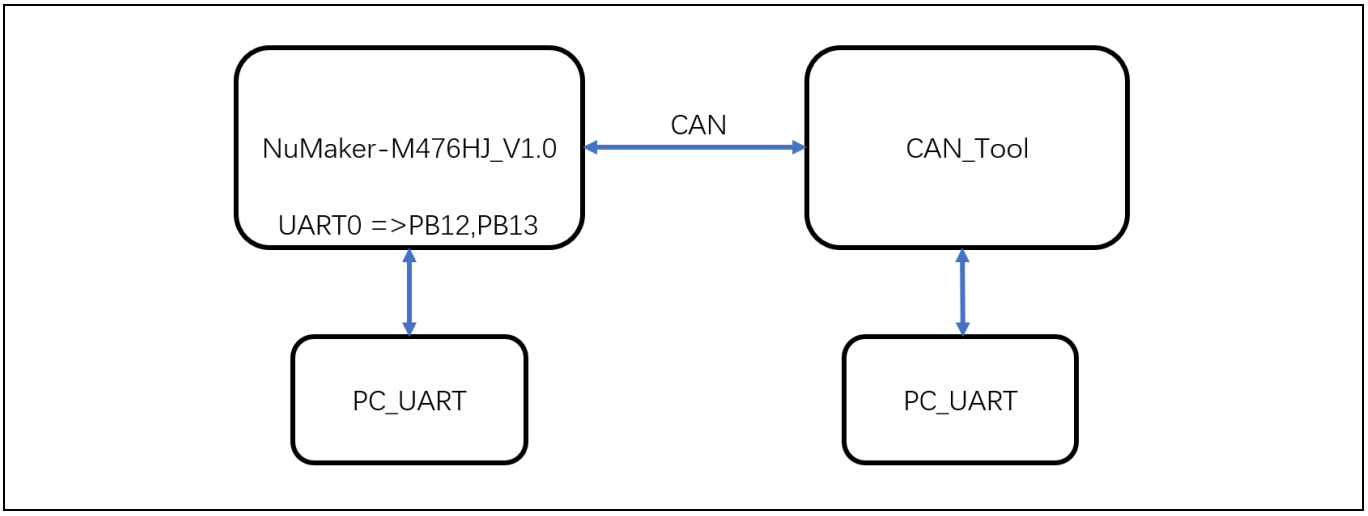


圖 3-1 線路示意圖

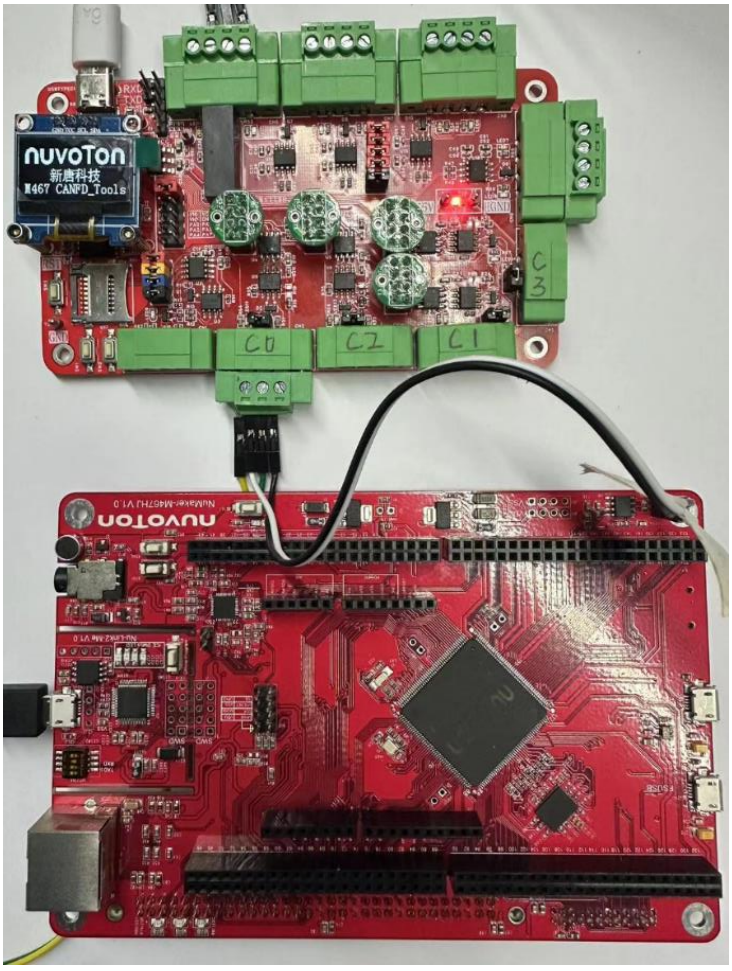


圖 3-1 實物接線

4. 目錄資訊

EC_M460_CANFD_TDC_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	
Project	Source file of example 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 *CANFD_TxRxINT_TDC.uvprojx*
2. 進入編譯模式介面
   - 編譯
   - 下載程式至記憶體
   - 進入 / 離開模擬模式
3. 進入模擬模式介面
   - 執行程式

6. 修訂紀錄

Date	Revision	Description
2024.03.28	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.

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.