使用 M2A23 ACMP 作為 CAN FD 收發器

NuMicro® 32 位元系列微控制器範例程式碼介紹

文件資訊

應用簡述	本範例程式碼使用 M2A23 ACMP 作為 CAN FD 收發器
BSP 版本	M2A23_Series_BSP_CMSIS_V3.00.001
開發平台	NuMaker-M2A23SG5AC_V1.0 CAN_ ACMP_TRANSCEIVER_EVB_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

本範例程式碼使用 M2A23 比較器配合外部 BJT 以替代傳統收發器。此方案透過拆分 CAN TX 路徑及利用比較器作為 CAN RX 輸入機制以實現 CAN FD 通訊。在滿足較為低速的應用同時還可以降低建置通訊平台的成本。

1.1 原理

傳統 CAN FD 系統常使用價格較高的收發器元件進行訊號收發。本方案改為使用晶片內建比較器搭配外部 BJT 電路來處理 CAN TX 與 CAN RX 路徑。具體做法為：

- 將 CAN TX 分為 TXH 與 TXL 兩組訊號，分別控制外部 BJT 以輸出高電位與低電位。
- 依據腳位設定，CAN RX 切換為內建比較器的輸出模式，並以 CAN_H 與 CAN_L 作為比較器之輸入，進行訊號判讀。

1.2 訊號路徑設計

下圖 1-1 示意本方案的電路概述，說明 TXH/TXL 與外部 BJT 的連接，以及內建比較器與 CAN_H/CAN_L 之間的訊號路徑。

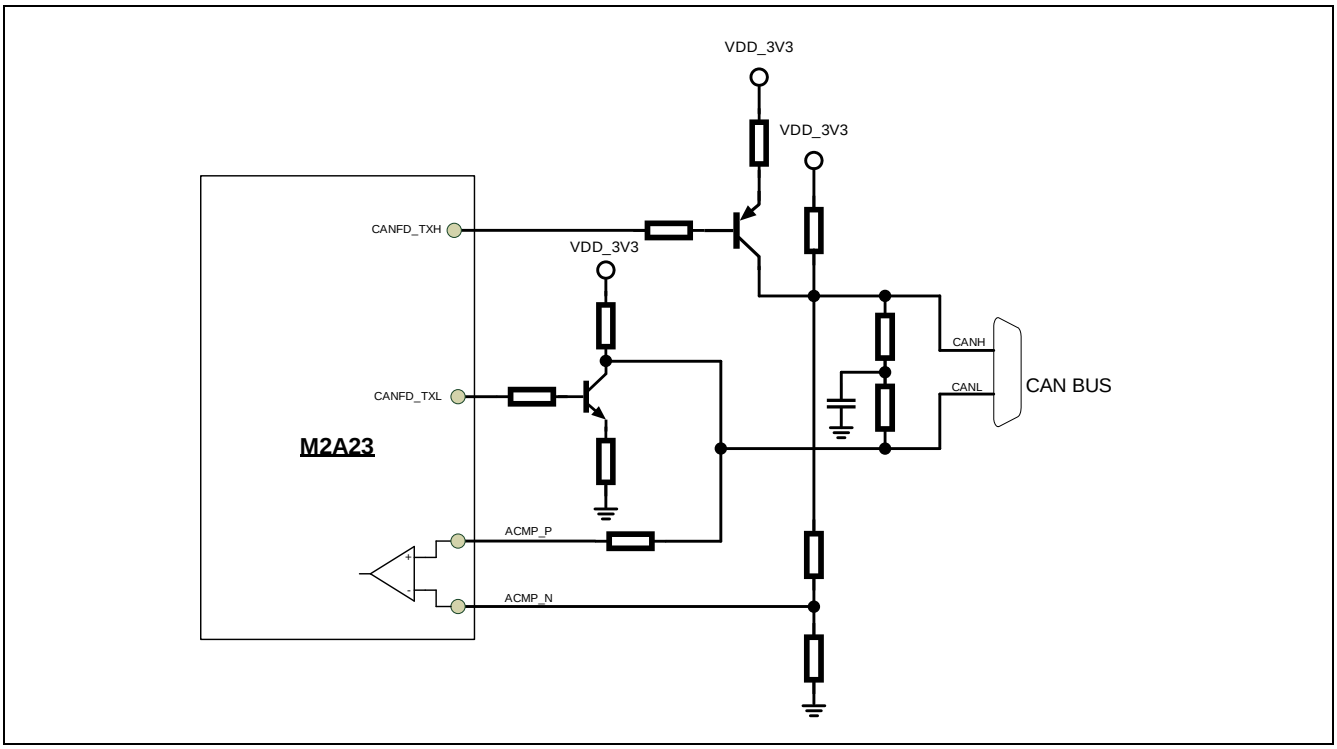


圖 1-1 電路概述圖

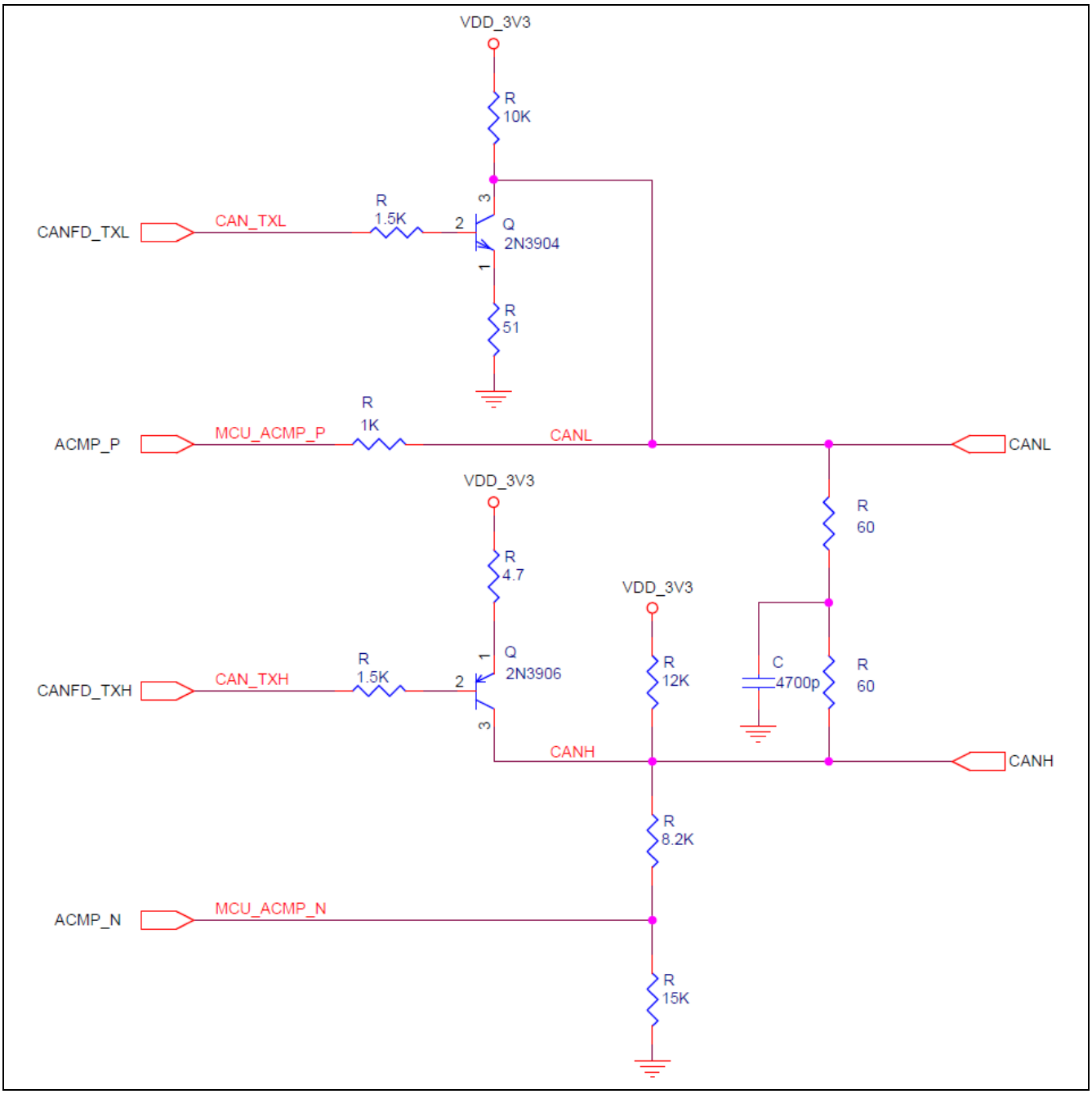

VDD_3V3
R
10K
CANFD_TXL
CAN_TXL
R
1.5K
2
3
1
Q
2N3904
R
51
ACMP_P
MCU_ACMP_P
R
1K
CANL
CANL
VDD_3V3
R
4.7
R
60
VDD_3V3
CANFD_TXH
CAN_TXH
R
1.5K
2
1
3
Q
2N3906
R
12K
C
4700p
R
60
CANH
CANH
R
8.2K
ACMP_N
MCU_ACMP_N
R
15K

圖 1-2 參考電路圖

1.2.1 CAN TX 路徑

拆分訊號：

- CAN TX 將訊號拆分成 TXH 與 TXL 兩組，用以分別控制 BJT 的開關狀態，實現高、低電位的輸出。

訊號控制：

- 採用外部 BJT 配合設計，能夠確保發送訊號的正確性。

1.2.2 CAN RX 路徑

比較器輸出設定：

- CAN RX 腳位可依需求配置為內建比較器的輸出模式，利用內部比較器對輸入訊號進行即時電平判讀。

訊號比較：

- 比較器以 CAN_H 與 CAN_L 兩條訊號線作為輸入，根據兩者電位差的判斷，產生對應的邏輯輸出，供後端控制單元進行處理。

1.3 電路測試結果

本章節說明在主機端與從機端皆採用本方案時，不同接線長度下的最大可達資料傳輸速率，以及在特定資料位元率下的CAN_H 與CAN_L 示波器觀測波形。

1.3.1 線長與最大傳輸速率測試

下表 1-1 彙整了在跳線、1 m、5 m 及 10 m 接線長度下，測試系統最高可達之資料位元率（Data bit rate）：

Core clock (MHz)	Data bit rate (Mbps)	Wire length (m)
72	4.5	Jump wire
72	4.5	1
72	4.5	5
72	4	10

表 1-1 線長與可達最大傳輸速率測試結果

1.3.2 示波器波形觀測

在資料位元率 500 kHz 時，於示波器上可觀測到差分訊號 CAN_H 及 CAN_L 之上下切換波形，如下圖 1-3 所示，其中訊號 2 為 CAN_H，訊號 3 為 CAN_L。

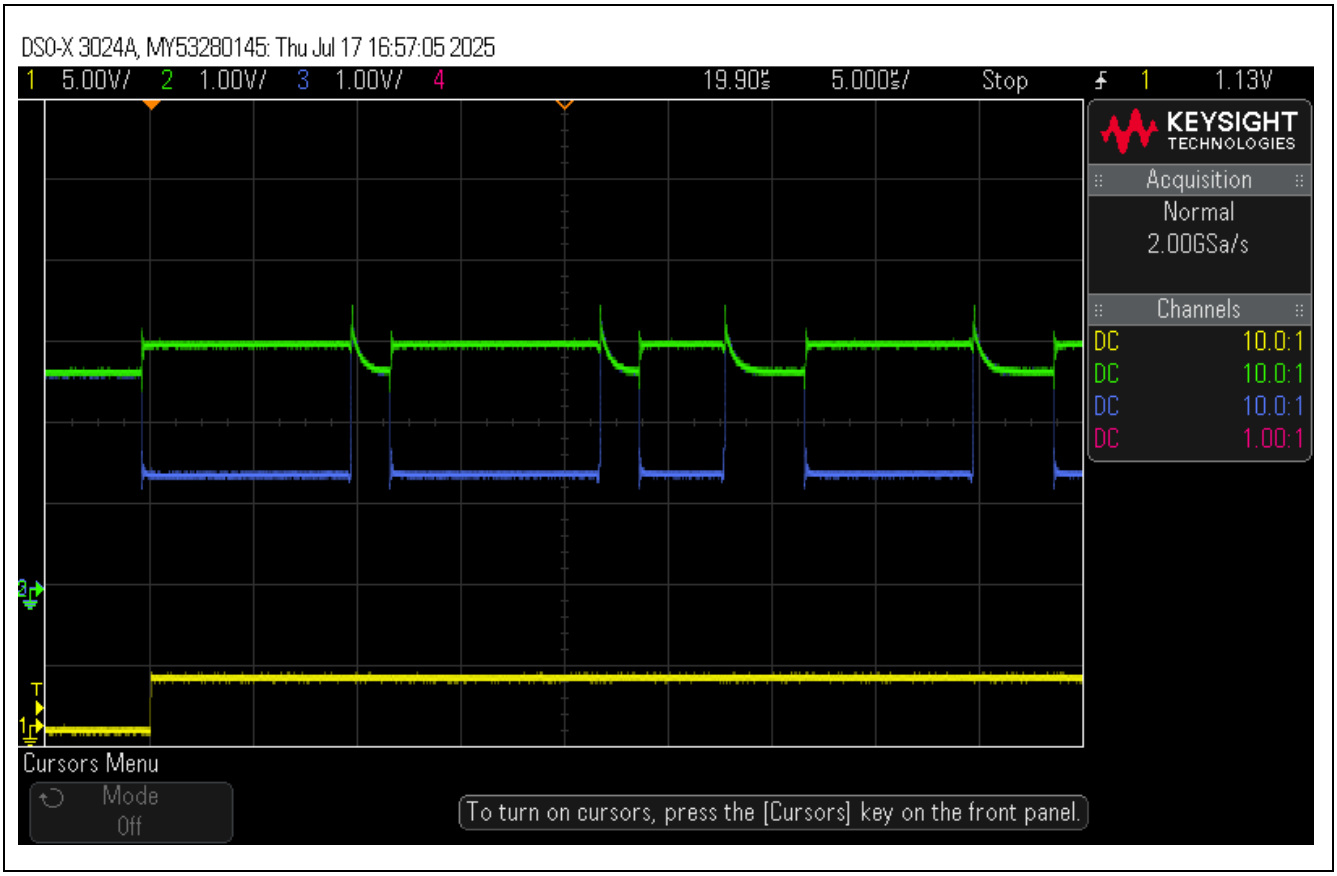


圖 1-3 示波器波形圖

2. 程式碼介紹

設定多功能腳位（Multi-Function Pin）、CAN FD TX、RX 訊號形式（CANFD_STSEL）與 ACMP 初始化的函式，程式碼位於 *main.c*。

在 *SYS_Init()* 函式內分別設定 CAN FD TXH、TXL 與 ACMP 多功能腳位，並啟用 ACMP01 周邊時鐘。

```
void SYS_Init(void)
{
    /*---------------------------------------------------------------------------------------------------------*/
    /* Init System Clock                                                                                       */
    /*---------------------------------------------------------------------------------------------------------*/
    /* Enable HIRC clock */
    CLK_EnableXtalRC(CLK_PWRCTL_HIRCEN_Msk);

    /* Wait for HIRC clock ready */
    CLK_WaitClockReady(CLK_STATUS_HIRCSTB_Msk);

    /* Enable HXT clock */
    CLK_EnableXtalRC(CLK_PWRCTL_HXTEN_Msk);

    /* Wait for HXT clock ready */
    CLK_WaitClockReady(CLK_STATUS_HXTSTB_Msk);


    /* Set core clock to 72MHz */
    CLK_SetCoreClock(72000000);

    /* Enable all GPIO clock */
    CLK->AHBCLK |= CLK_AHBCLK_GPIOACKEN_Msk | CLK_AHBCLK_GPIOBCKEN_Msk | CLK_AHBCLK_GPIOCCKEN_Msk |
                   CLK_AHBCLK_GPIODCKEN_Msk | CLK_AHBCLK_GPIOFCKEN_Msk;

    SYS->GPF_MFPL = (SYS->GPF_MFPL & (~SYS_GPF_MFPL_PF2MFP_Msk)) | SYS_GPF_MFPL_PF2MFP_XT1_OUT;
    SYS->GPF_MFPL = (SYS->GPF_MFPL & (~SYS_GPF_MFPL_PF3MFP_Msk)) | SYS_GPF_MFPL_PF3MFP_XT1_IN;

    /* Enable UART0 module clock */
    CLK_EnableModuleClock(UART0_MODULE);

    /* Select UART0 module clock source as HIRC and UART0 module clock divider as 1 */
    CLK_SetModuleClock(UART0_MODULE, CLK_CLKSEL2_UART0SEL_HIRC, CLK_CLKDIV0_UART0(1));

    /* Enable ACMP01 peripheral clock */
    CLK_EnableModuleClock(ACMP01_MODULE);

    /* Select CAN FD0 clock source is HCLK */
    CLK_SetModuleClock(CANFD0_MODULE, CLK_CLKSEL0_CANFD0SEL_HCLK, CLK_CLKDIV1_CANFD0(1));

    /* Enable CAN FD0 peripheral clock */
    CLK_EnableModuleClock(CANFD0_MODULE);

    /*---------------------------------------------------------------------------------------------------------*/
    /* Init I/O Multi-function                                                                                 */
    /*---------------------------------------------------------------------------------------------------------*/
```

```
    /*---------------------------------------------------------------------------------------------------------*/

    /* Set PA multi-function pins for UART0 RXD and TXD */
    SET_UART0_RXD_PA6();
    SET_UART0_TXD_PA7();

    /* Set PB multi-function pins for CAN FD0 TXH and TXL */
    SET_CANFD0_TXH_PB13();
    SET_CANFD0_TXL_PB14();

    /* Set PB2 multi-function pin for ACMP0 positive input pin and PB2 multi-function pin for ACMP0 negative input pin */
    SET_ACMP0_P1_PB2();
    SET_ACMP0_N_PB3();

    /* Disable digital input path of analog pin ACMP to prevent leakage */
    GPIO_DISABLE_DIGITAL_PATH(PB, BIT2);

    /* Disable pull-up resistor */
    PB->PUSEL = 0;
}
```

初始化系統後，接著在 *ACMP_Init()* 開啟 ACMP0，並設定 ACMP_CTL_POSSEL 為 ACMP0_P1。

```
/*---------------------------------------------------------------------------------------------------------*/
/* Init ACMP                                                                                               */
/*---------------------------------------------------------------------------------------------------------*/
void ACMP_Init(void)
{
    /* Configure ACMP0. Enable ACMP0 and select ACMP0 N as the source of ACMP negative input. */
    ACMP_Open(ACMP01, 0, ACMP_CTL_NEGSEL_PIN, ACMP_CTL_HYSTERESIS_DISABLE);
    /* Select P1 as ACMP positive input channel */
    ACMP_SELECT_P(ACMP01, 0, ACMP_CTL_POSSEL_P1);
}
```

CANFD_Init() 除了設定傳輸速度及 ID 外，同時需要設定 CANFD_STSEL。將 RXSS 設定為 ACMP0 的輸出腳位；將 TXLS 設為邏輯 0 時輸出高電位，邏輯 1 時輸出低電位；將 TXHS 設為邏輯 0 時輸出低電位，邏輯 1 時輸出高電位。

```
/*---------------------------------------------------------------------------------------------------------*/
/* Init CAN FD                                                                                             */
/*---------------------------------------------------------------------------------------------------------*/
void CANFD_Init(void)
{
    CANFD_FD_T sCANFD_Config;

    printf("+-------------------------------------------------------------+\n");
    printf("|                        Pin Configure                        |\n");
    printf("+-------------------------------------------------------------+\n");
    printf("|        CAN_TXD                               CAN_TXD        |\n");
    printf("|        CAN_RXD                               CAN_RXD        |\n");
    printf("|           +-----------+  CANBUS  +-----------+              |\n");
    printf("|    ------>|           |          |           |<------       |\n");
```

```
    printf("|     CAN_TXL|            |            |            |CAN_TXL     |\n");
    printf("|     ------>|            |<-------->  |            |<------     |\n");
    printf("|     CAN_TXH|   CANFD    |   CAN_H    |   CANFD    |CAN_TXH     |\n");
    printf("|            | ACMP+BJTs  |            | ACMP+BJTs  |            |\n");
    printf("|     <------|            |<-------->  |            |------>     |\n");
    printf("|      ACMP_P|            |   CAN_L    |            |ACMP_P      |\n");
    printf("|     <------|            |            |            |------>     |\n");
    printf("|      ACMP_N|            |            |            |ACMP_N      |\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;
    sCANFD_Config.sBtConfig.sDataBitRate.u32BitRate = 500000;

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

    NVIC_EnableIRQ(CANFD00_IRQn);

    /* Receive 0x111 (11-bit id) in CAN FD rx message buffer 0 by setting mask 0 */
    CANFD_SetSIDFltr(g_pCanfd, 0, CANFD_RX_BUFFER_STD(0x111, 0));
    /* Receive 0x22F (11-bit id) in CAN FD rx message buffer 0 by setting mask 1 */
    CANFD_SetSIDFltr(g_pCanfd, 1, CANFD_RX_BUFFER_STD(0x22F, 0));
    /* Receive 0x333 (11-bit id) in CAN FD rx message buffer 0 by setting mask 2 */
    CANFD_SetSIDFltr(g_pCanfd, 2, CANFD_RX_BUFFER_STD(0x333, 0));

    /* Receive 0x222 (29-bit id) in CAN FD rx message buffer 1 by setting mask 3 */
    CANFD_SetXIDFltr(g_pCanfd, 0, CANFD_RX_BUFFER_EXT_LOW(0x222, 1),
CANFD_RX_BUFFER_EXT_HIGH(0x222, 1));
    /* Receive 0x3333 (29-bit id) in CAN FD rx message buffer 1 by setting mask 3 */
    CANFD_SetXIDFltr(g_pCanfd, 1, CANFD_RX_BUFFER_EXT_LOW(0x3333, 1),
CANFD_RX_BUFFER_EXT_HIGH(0x3333, 1));
    /* Receive 0x44444 (29-bit id) in CAN FD rx message buffer 1 by setting mask 3 */
    CANFD_SetXIDFltr(g_pCanfd, 2, CANFD_RX_BUFFER_EXT_LOW(0x44444, 1),
CANFD_RX_BUFFER_EXT_HIGH(0x44444, 1));
    /* CAN FD Run to Normal mode */
    CANFD_RunToNormal(g_pCanfd, TRUE);

    /* Set RXSS from ACMP0 output,
       TXLS as output high while logical 0 and output low while logical 1,
       TXHS as output low while logical 0 and output high while logical 1 */
    g_pCanfd->STSEL =
        ((CANFD_RXSS_ACMP0) |
         (CANFD_TX_L0_HIGH_L1_LOW << CANFD_STSEL_TXLS_Pos) |
         (CANFD_TX_L0_LOW_L1_HIGH << CANFD_STSEL_TXHS_Pos));
}
```

接上 UART 並執行程式後，可在終端機 (terminal) 上看到如下選擇頁面：

```
+-------------------------------------------------------------------------+
|                           CAN FD Function Test                          |
+-------------------------------------------------------------------------+
|  Description :                                                          |
|    The sample code needs two boards. One is master(CAN FD transmitter)  |
|    and the other is slave(CAN FD receiver). Master will send 6 messages |
|    with different sizes of data and ID to the slave. Slave will check if|
|    received data is correct after getting 6 messages data.              |
|  Please select Master or Slave test                                     |
|  [0] Master(CAN FD transmitter)    [1] Slave(CAN FD receiver)           |
+-------------------------------------------------------------------------+
```

從機端選擇 1，主機端選擇 0。接著主機端將會發送 6 筆不同訊息，同時從機端的終端機畫面會顯示接收到的訊息及其比對結果。

主機端發送訊息：

```
Start to CAN FD Bus Transmitter :
Send to transmit message 0x00000111 (11-bit)
Data Message : 00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,

Start to CAN FD Bus Transmitter :
Send to transmit message 0x0000022f (11-bit)
Data Message :
01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,
31,32,

Start to CAN FD Bus Transmitter :
Send to transmit message 0x00000333 (11-bit)
Data Message :
02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,
62,63,64,65,

Start to CAN FD Bus Transmitter :
Send to transmit message 0x00000222 (29-bit)
Data Message : 03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,

Start to CAN FD Bus Transmitter :
Send to transmit message 0x00003333 (29-bit)
Data Message :
04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
34,35,

Start to CAN FD Bus Transmitter :
Send to transmit message 0x00044444 (29-bit)
Data Message :
05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,
35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,
65,66,67,68,


 Transmit Done
```

從機端接收訊息：

```
Start to CAN FD Bus Receiver :
Rx buf 0: Received message 0x00000111 (11-bit)
Message Data : 00 ,01 ,02 ,03 ,04 ,05 ,06 ,07 ,08 ,09 ,10 ,11 ,12 ,13 ,14 ,15 ,

Rx buf 0: Received message 0x0000022F (11-bit)
Message Data :
01 ,02 ,03 ,04 ,05 ,06 ,07 ,08 ,09 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,23
,24 ,25 ,26 ,27 ,28 ,29 ,30 ,31 ,32 ,

Rx buf 0: Received message 0x00000333 (11-bit)
```

```
Message Data :
02 ,03 ,04 ,05 ,06 ,07 ,08 ,09 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,23 ,24
,25 ,26 ,27 ,28 ,29 ,30 ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40 ,41 ,42 ,43 ,44 ,45 ,46 ,4
7 ,48 ,49 ,50 ,51 ,52 ,53 ,54 ,55 ,56 ,57 ,58 ,59 ,60 ,61 ,62 ,63 ,64 ,65 ,

Rx buf 1: Received message 0x00000222 (29-bit)
Message Data : 03 ,04 ,05 ,06 ,07 ,08 ,09 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,

Rx buf 1: Received message 0x00003333 (29-bit)
Message Data :
04 ,05 ,06 ,07 ,08 ,09 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,23 ,24 ,25 ,26
,27 ,28 ,29 ,30 ,31 ,32 ,33 ,34 ,35 ,

Rx buf 1: Received message 0x00044444 (29-bit)
Message Data :
05 ,06 ,07 ,08 ,09 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,23 ,24 ,25 ,26 ,27
,28 ,29 ,30 ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40 ,41 ,42 ,43 ,44 ,45 ,46 ,47 ,48 ,49 ,5
0 ,51 ,52 ,53 ,54 ,55 ,56 ,57 ,58 ,59 ,60 ,61 ,62 ,63 ,64 ,65 ,66 ,67 ,68 ,


 Receive OK & Check OK
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M2A23_Series_BSP_CMSIS_V3.00.001
- IDE 版本
  - Keil uVision 5.41

3.2 硬體需求

- 電路元件
  - NuMaker-M2A23SG5AC_V1.0
  - CAN_ ACMP_TRANSCEIVER_EVB_V1.0
- 線路示意圖
  - 將 M2A23SG5AC CAN FD TXH、TXL 和 ACMP_N、ACMP_P 連接至 CAN_ ACMP_TRANSCEIVER 開發板。
  - 將主機端與從機端的 CAN_ACMP_TRANSCEIVER 開發板的 CAN_H 與 CAN_L 對接以確保能夠正常通訊。

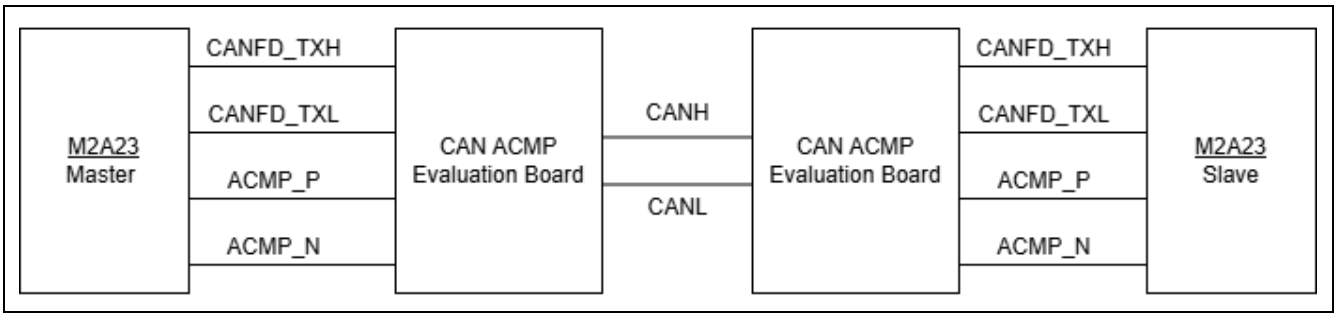


圖 3-1 線路示意圖

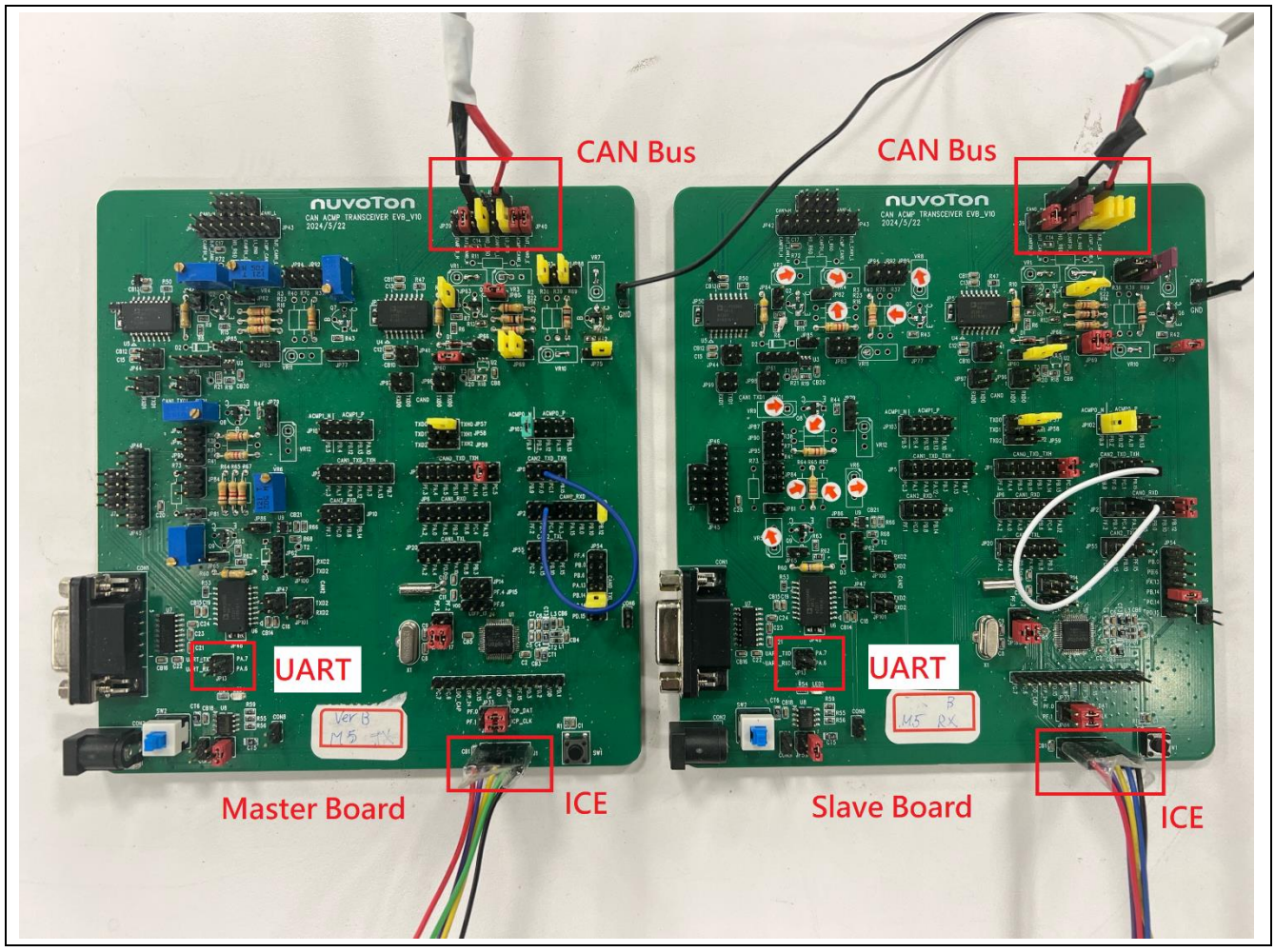


圖 3-2 實體接線

4. 目錄資訊

目錄結構如下圖所示。

EC_M2A23_CANFD_Using_ACMP_as_Transceiver _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	
M2A23_CANFD_ACMP_TxRx	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，雙擊 *M2A23_CANFD_ACMP_TxRx.uvprojx*。
2. 進入編譯模式介面
   - 編譯
   - 下載程式碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行程式碼

6. 修訂紀錄

Date	Revision	Description
2025.04.25	1.00	初始發布。

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.

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.