使用 M2A23 ACMP 作为 CAN FD 收发器

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例程序使用 M2A23 ACMP 作为 CAN FD 收发器
BSP 版本	M2A23_Series_BSP_CMSIS_V3.00.001
开发平台	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

本示例代码使用 M2A23 比较器配合外部 BJT 以替代传统收发器。该方案通过拆分 CAN TX 路径并利用比较器作为 CAN RX 输入机制以实现 CAN FD 通讯。在满足较为低速的应用同时还可以降低建立通信平台的成本。

1.1 原理

传统 CAN FD 系统多使用价格较高的收发器组件进行信号收发，本方案改为使用芯片内建的比较器搭配外部 BJT 电路来处理 CAN TX 与 CAN RX 路径。具体做法为：

- 将 CAN TX 分为 TXH 与 TXL 两组信号，分别控制外部 BJT 以输出高电位与低电位。
- 依据脚位设定，CAN RX 切换为内建比较器的输出模式，并以 CAN_H 与 CAN_L 作为比较器之输入，进行信号识别。

1.2 信号路径设计

下图 1-1 示意本方案的电路概述，说明 TXH/TXL 与外部 BJT 的连接，以及内建比较器与 CAN_H/CAN_L 之间的信号路径。

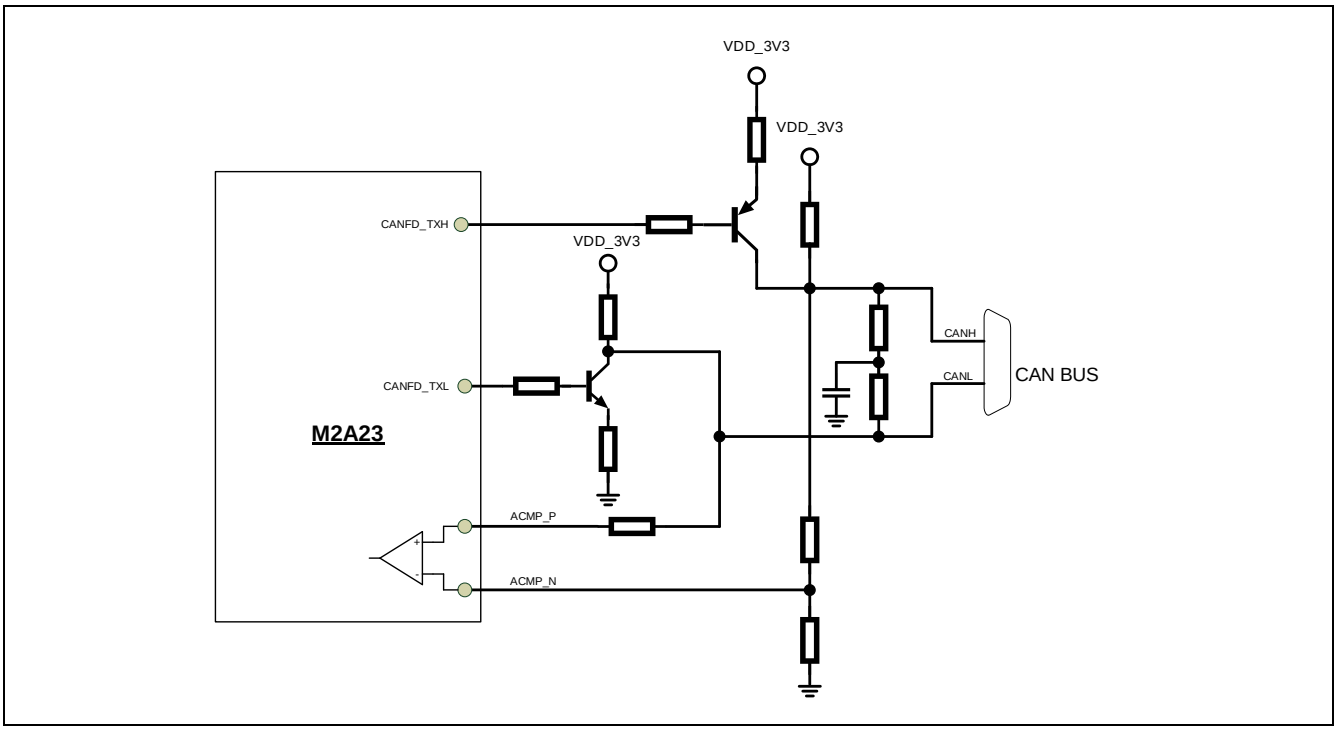


图 1-1 电路概述图

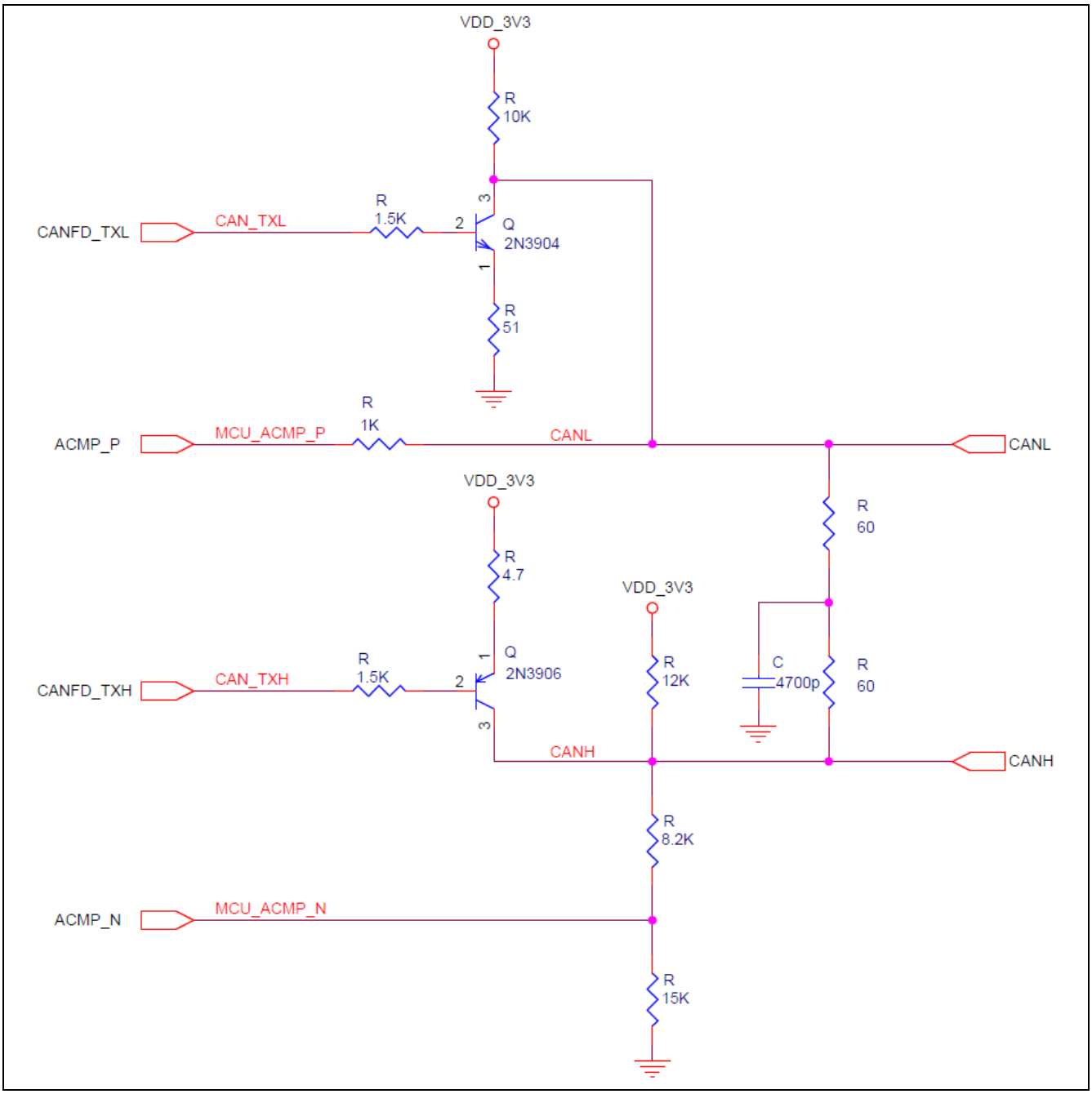

VDD_3V3
R
10K
CANFD_TXL
CAN_TXL
R
1.5K
2
3
1
Q
2N3904
R
51
ACMP_P
MCU_ACMP_P
R
1K
CANL
CANL
VDD_3V3
R
4.7
R
60
VDD_3V3
CANFD_TXH
CAN_TXH
R
1.5K
2
1
3
Q
2N3906
R
12K
C
4700p
R
60
CANH
CANH
R
8.2K
ACMP_N
MCU_ACMP_N
R
15K

图 1-2 参考电路图

1.2.1 CAN TX 路径

拆分信号：

- CAN TX 將信号拆分为 TXH 与 TXL 两组，分别用于控制 BJT 的导通状态，实现高低电位输出。

信号控制：

- 采用外部 BJT 配合设计，能够确保发送信号的正确性。

1.2.2 CAN RX 路径

比较器输出设置：

- CAN RX 引脚可根据需求配置为内部比较器的输出模式，通过内部比较器对输入信号进行实时电平判断。

信号比较：

- 比较器以 CAN_H 与 CAN_L 两条信号线作为输入，根据两者电位差的判断，产生对应的逻辑输出，供后端控制单元进行处理。

1.3 电路测试结果

本章节说明当主机端与从机端均采用本方案时，不同连线长度下的最大可实现数据传输速率，以及在特定数据比特率下 CAN_H 和 CAN_L 的示波器观测波形。

1.3.1 线长与最大传输速率测试

下表 1-1 汇总了在跳线、1 m、5 m、及 10 m 连线长度下，测试系统最高可达之数据比特率（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 线长与最大可实现传输速率测试结果

1.3.2 示波器波形观察

在数据比特率 500 kHz 时，于示波器上可观测到差分信号 CAN_H 及 CAN_L 之上下切换波形，如下图 1-3 所示，其中信号 2 为 CAN_H，信号 3 为 CAN_L。

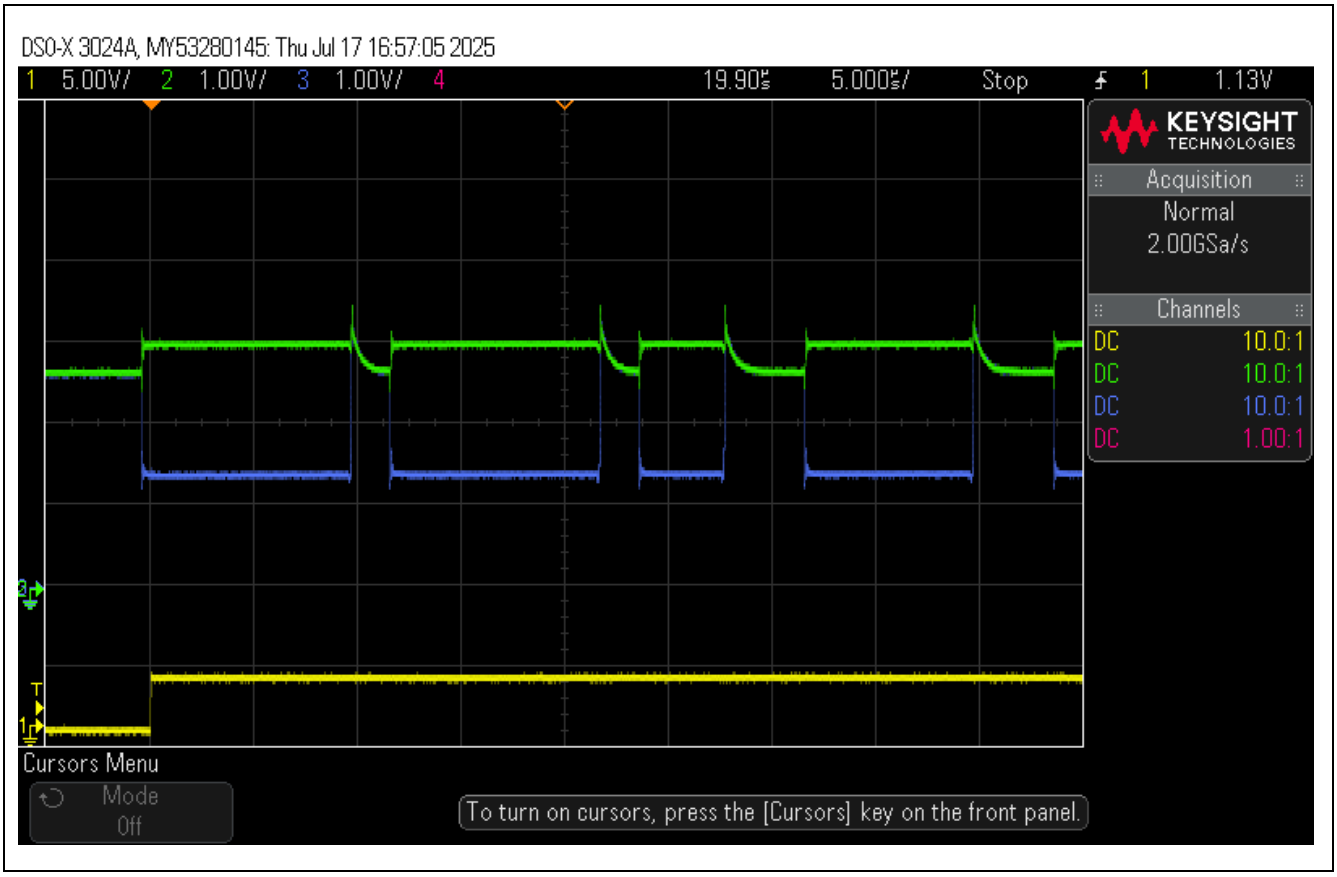


图 1-3 示波器波形

2. 代码介绍

设置多功能引脚（Multi-Function Pin）、CAN FD TX、RX 信号形式（CANFD_STSEL）与 ACMP 初始化的函数，程序代码位于 *main.c* 中。

在 *SYS_Init()* 函数中分别设置 CAN FD TXH、TXL 以及 ACMP 多功能引脚，并启用 ACMP01 外设时钟。

```
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;
}
```

初始化系统后，接着在 *ACMP_Init()* 开启 ACMP0，并设定 ACMP_CTL_POSSEL 为 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() 除了设置传输速率与 ID 外，同时需要设置 CANFD_STSEL。将 RXSS 设置为 ACMP0 的输出脚位；将 TXLS 设为逻辑 0 时输出高电平，逻辑 1 时输出低电平；将 TXHS 设为逻辑 0 时输出低电平，逻辑 1 时输出高电平。

```
/*---------------------------------------------------------------------------------------------------------*/
/* 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_STSEL_RXSS_Pos) |
         (CANFD_TX_L0_HIGH_L1_LOW << CANFD_STSEL_TXLS_Pos) |
         (CANFD_TX_L0_LOW_L1_HIGH << CANFD_STSEL_TXHS_Pos));
}
```

连接 UART 并执行程序后，可在终端机（terminal）中看到如下菜单画面：

```
+-------------------------------------------------------------------------+
|                           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)           |
+-------------------------------------------------------------------------+
```

从机端选择 1，主机端选择 0。接着主机端将会发送 6 笔不同消息，同时从机端的终端机画面会显示接收到的消息及其比对结果。

主机端发送消息画面：

```
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
```

从机端接收消息画面：

```
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. 软件与硬件需求

3.1 软件需求

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

3.2 硬件需求

- 电路组件
  - NuMaker-M2A23SG5AC_V1.0
  - CAN_ ACMP_TRANSCEIVER_EVB_V1.0
- 线路示意图
  - 将 M2A23SG5AC CAN FD TXH、TXL 和 ACMP_N、ACMP_P 连接至 CAN_ ACMP_TRANSCEIVER 开发板。
  - 将主机端与从机端的 CAN_ ACMP_TRANSCEIVER 开发板的 CAN_H 与 CAN_L 对接以确保能够正常通信。

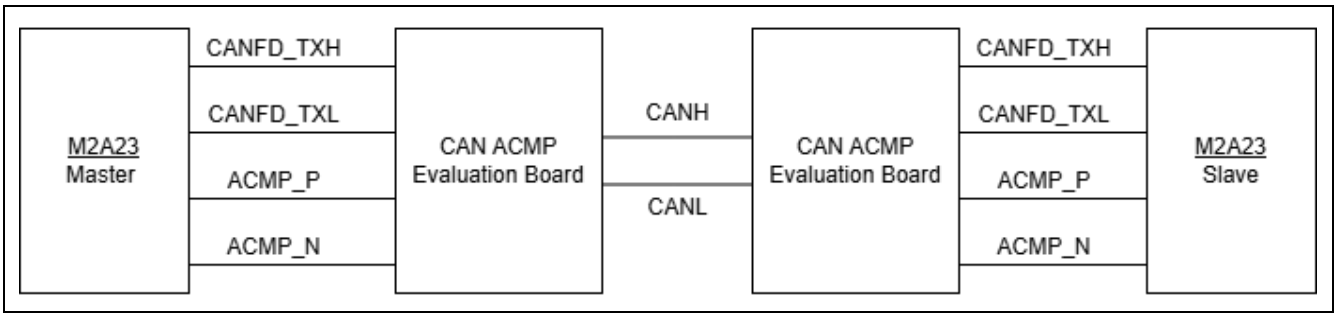


图 3-1 线路示意图

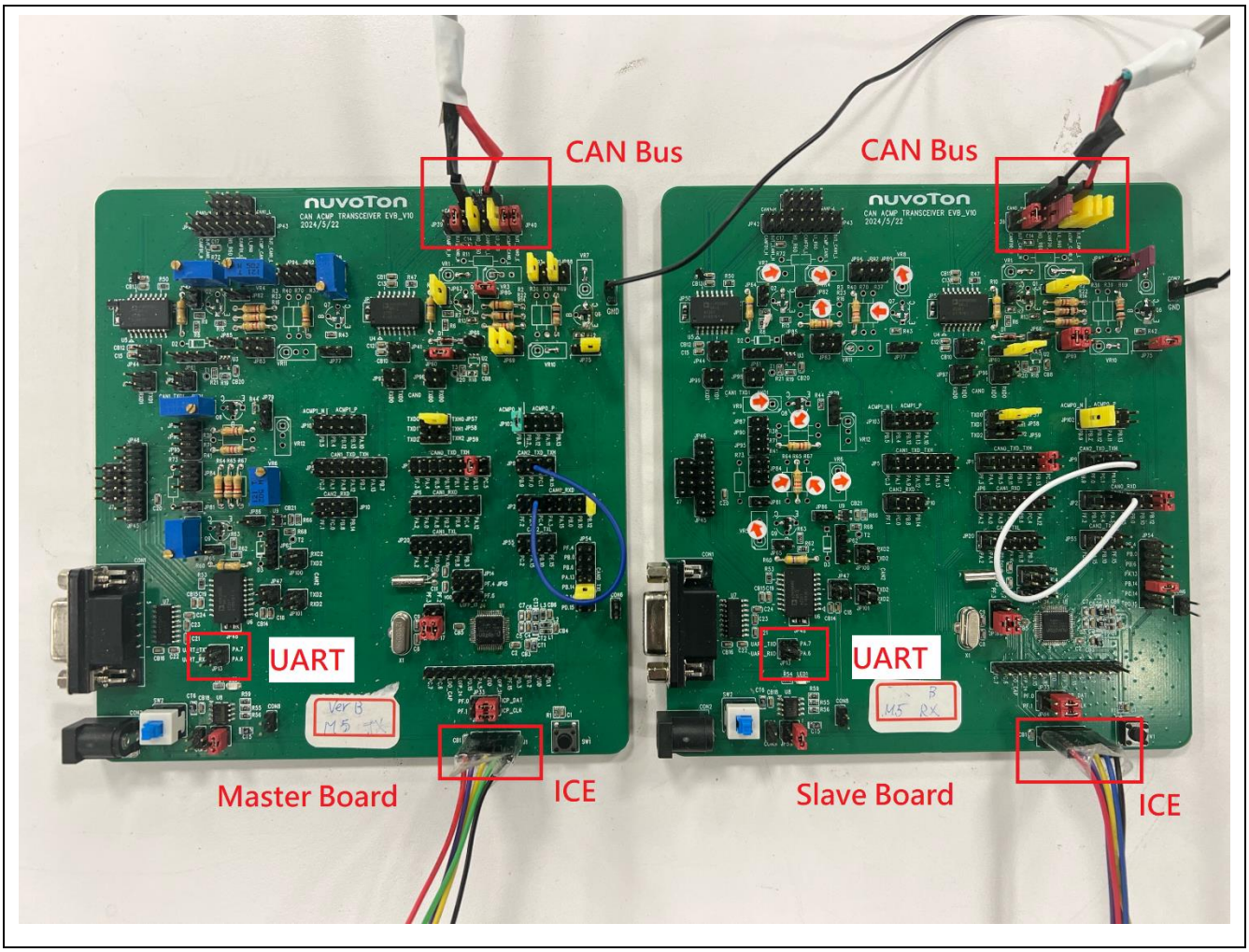


图 3-2 实物接线

4. 目录信息

目录结构如下图所示。

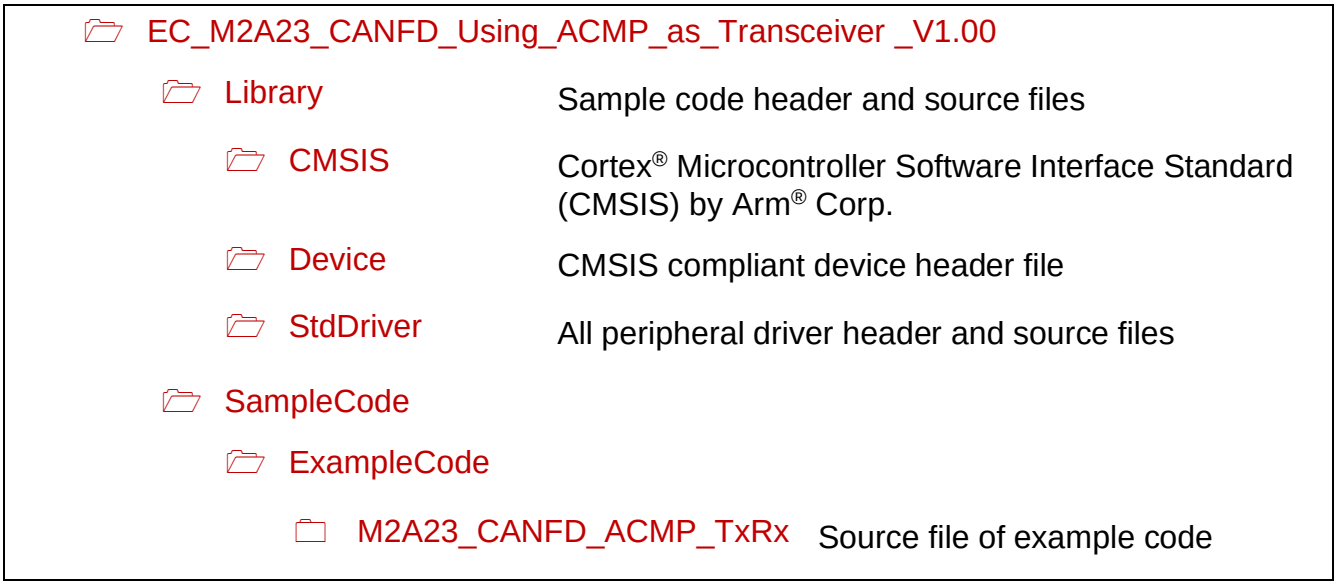


图 4-1 目录信息

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 *M2A23_CANFD_ACMP_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
2025.04.25	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.

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.