UART 转 CAN 的桥接传输

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码使用 M0A23 演示 UART 与 CAN 的桥接传输
BSP 版本	M0A21_Series_BSP_CMSIS_V3.03.000
开发平台	NuMaker-M0A23EC 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

本范例代码示范如何在 M0A23 系列平台完成 UART 与 CAN 的桥接传输。

相关应用场景，如在汽车电子领域中，当系统需要嵌入 UART 传感器，但又必须透过 CAN 总线进行数据传输时，可采用 UART 与 CAN 之间的桥接转换技术来实现数据互通，并确保稳定传输。

1.1 原理

因 UART 与 CAN 的传输原理与格式皆不同，此章节将着重于介绍 UART 与 CAN 之间的传输格式转换与传输流程。

CAN 的传输方式是以 Frame 为单位，而 UART 是以字节 (bytes) 为单位，由于需要将 UART 收到的数据格式转换成 CAN 可接收的数据格式，因此需对 UART 接收数据设定为符合 CAN frame 的格式，对应关系设定如表 1-1，而由于 UART 不像 CAN 有明确的 frame 范围与定义，因此多设定 data header，否则有可能产生数据解析错误。

UART Bytes	UART Data	Contents	Example (in Hex)
0-1	0x55, 0xAA	Start header	55, AA
2-5	ID1, ID2, ID3, ID4	CAN ID	00, 00, 01, 33
6	Length	Data length Note: The length cannot be 0.	04
7-N	DataN	CAN Data	11, 22, 33, 44

表 1-1 Data Format

当 UART 接收到数据亦确认解析正确后，便会将数据封装成 CAN frame 并以 CAN_Transmit 送出，流程与判断方式如图 1-1。

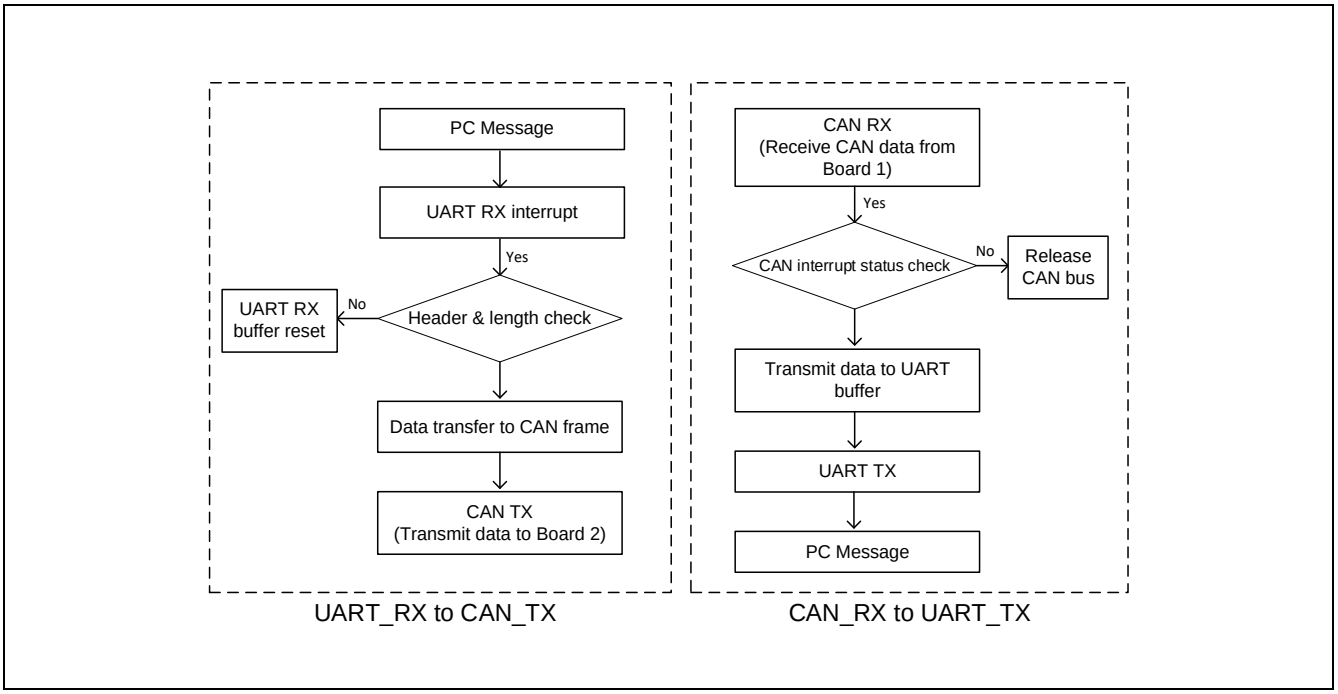

图 1-1 数据接收与转换流程

举例来说，若 UART 接收到的数据，如表 1-1 中的 Example 字段，转换为 CAN 封包后，经由 CAN 传送的数据将如表 1-2 所示。

Frame Type	ID Type	ID	DLC	Data
CAN_Data_FRAME	CAN_STD_ID CAN_EXT_ID	0x00000133	4	11,22,33,44

表 1-2 CAN Package

1.2 执行结果

本范例代码执行编译和烧录的结果如图 1-2 所示。硬件连接方式请参考 3.2 章节的线路示意图，软件工具则需使用支持 Hex 发送与接收的串口调适工具，使用两块 NuMaker-M0A23EC，择一于串口调适器输入符合的 hex 字符串，如表 1-1 的范例字符串，即可看到另一个串口成功接收指定的字符串。

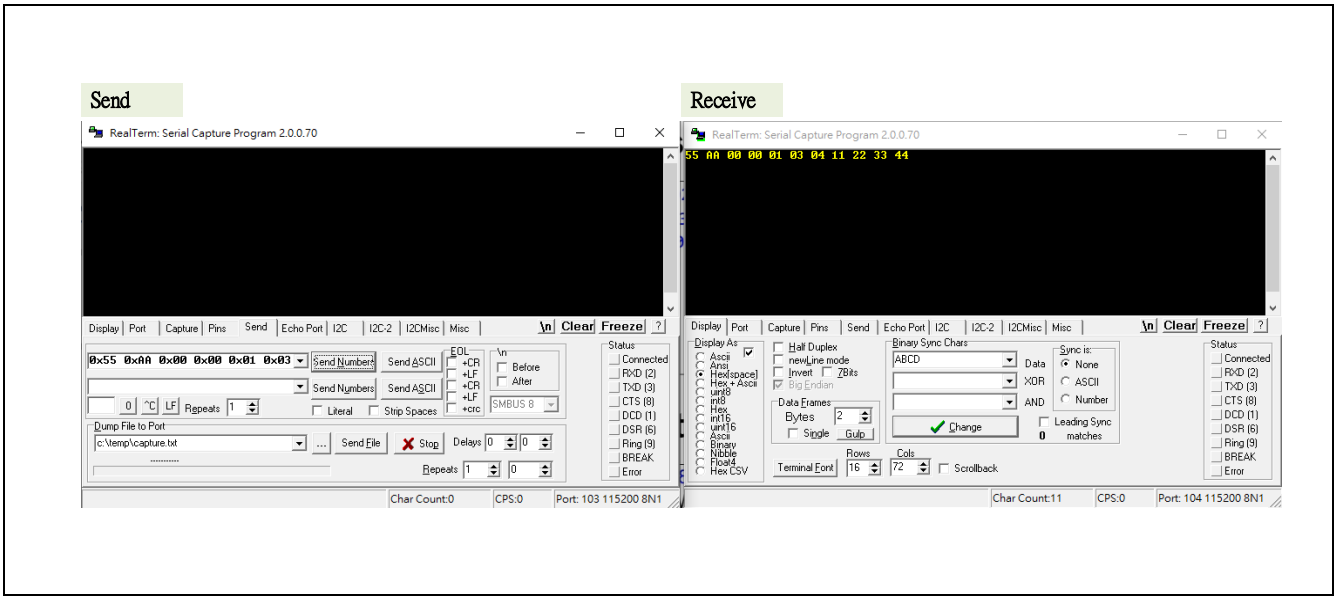

图 1-2 执行编译和烧录结果

2. 代码介绍

UART 以 RDA interrupt 接收数据并储存在 UART ring buffer 中。

```
void UART0_IRQHandler(void)
{
    uint32_t u32IntStatus;
    int32_t size;
    uint8_t loc_data[1];
    u32IntStatus = UART0->INTSTS;

    if((u32IntStatus & UART_INTSTS_RDAINT_Msk))
    {
        /* Get all the input characters */
        while(UART_GET_RX_EMPTY(UART0) == 0)
        {
            loc_data[0] = UART_READ(UART0);
            lwrb_write(&lw_uart_rx_buff, loc_data, 1);
        }
    }

    /* Check if Tx is working */
    if((UART0->INTEN & UART_INTEN_THREIEN_Msk) != 0)
    {
        if(u32IntStatus & UART_INTSTS_THREIF_Msk)
        {
            if((lw_uart_tx_buff.w - lw_uart_tx_buff.r) != 0)
            {
                /* Fill the Tx FIFO */
                size = (lw_uart_tx_buff.w - lw_uart_tx_buff.r);
                if(size >= TX_FIFO_SIZE)
                {
                    size = TX_FIFO_SIZE;
                }
                while(size)
                {
                    lwrb_read(&lw_uart_tx_buff, loc_data, 1);
                    while(UART0->FIFOSTS & UART_FIFOSTS_TXFULL_Msk) {};
                    UART0->DAT = loc_data[0];
                    size--;
                }
            }
            else
            {
                if((lw_uart_tx_buff.w - lw_uart_tx_buff.r) == 0)
                {
                    /* No more data, just stop Tx (Stop work) */
                    UART_DISABLE_INT(UART0, UART_INTEN_THREIEN_Msk);
                    lwrb_reset(&lw_uart_tx_buff);
                }
            }
        }
    }
}
```

于 *main.c* 的 while loop 中检查 UART buffer 中是否有数据，以及分析 UART 接收到的数据格式是否符合定义。

```
        /* analyze UART data */
        if((lw_uart_rx_buff.w - lw_uart_rx_buff.r) > 0)
        {
            __disable_irq();
            lwrb_read(&lw_uart_rx_buff, loc_data_m, 1);
            gUartReciveGroup[u8UART0_RX_Count] = loc_data_m[0];

            /* compare start byte, and the start byte should be 0x55 */
            if(u8UART0_RX_Count == 0 && gUartReciveGroup[u8UART0_RX_Count] != 0x55)
            {
                dbg_printf("The first byte is not 0x55.\n");
            }
            else
            {
                u8UART0_RX_Count++;
                /*check the format
                1. The first 2 bytes are 0x55 0xAA
                2. The data length should not be 0
                3. Data received finish, when the length is equal to gUartReciveGroup[6]+7
                */
                if( (gUartReciveGroup[0] == 0x55) && (gUartReciveGroup[1] == 0xaa) &&
                    (gUartReciveGroup[6] != 0) &&
                    ((u8UART0_RX_Count >= (gUartReciveGroup[6] + 7))) )
                {
                    u8UART0_RX_Flag = 1;
                }
            }
            if((lw_uart_rx_buff.w - lw_uart_rx_buff.r) == 0)
            {
                lwrb_reset(&lw_uart_rx_buff);
            }
            __enable_irq();
        }
```

若符合定义即可封装成 CAN frame 格式，并开始传送。

```
        /* Start CAN transmit */
        if(u8UART0_RX_Flag == 1)
        {
            /* The format of UartReceiveGroup is
                55 AA ID1 ID2 ID3 ID4 Length Data1 Data2 ...
            */
            uint32_t id = 0;
            uint32_t count = 0;

            /* Get device address */
            id = gUartReciveGroup[2] << 24;
            id += gUartReciveGroup[3] << 16;
            id += gUartReciveGroup[4] << 8;
            id += gUartReciveGroup[5];
            STR_CANMSG_T tMsg;

            /* Send a 11-bits message */
```

```
            tMsg.FrameType = CAN_DATA_FRAME;

#ifdef CAN_STD_FRAME
            tMsg.IdType = CAN_STD_ID;
#else
            tMsg.IdType = CAN_EXT_ID;
#endif
            tMsg.Id = id;
            tMsg.DLC = gUartReciveGroup[6];

            for(count = 0; count < gUartReciveGroup[6]; count++)
                tMsg.Data[count] = gUartReciveGroup[7 + count];

            /* Configure Msg RAM and send the Msg in the RAM */
            if(CAN_Transmit(tCAN, MSG(1), &tMsg) == FALSE)
            {
                dbg_printf("Set Tx Msg Object failed\n");
            }

            /* clear buffer */
            for(count = 0; count < 7; count++)
                gUartReciveGroup[count] = 0;

            u8UART0_RX_Count = 0;

            /* clear flag */
            u8UART0_RX_Flag = 0;
    }
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M0A21_Series_BSP_CMSIS_V3.03.000
- IDE 版本
  - Keil uVersion 5.38

3.2 硬件需求

- 电路组件
  - NuMaker-M0A23EC V1.0
- 线路示意图
  - 将两块 NuMaker-M0A23EC 开发版的 CAN_H 与 CAN_L 分别相连，并透过 PC 使用 UART 接收与传输数据，以显示范例代码的执行结果。在 NuMaker-M0A23EC 上，可以直接使用板上的 Nu-Link2-Me USB port(ICEJ3)与 PC 相连，并将 VCOM switch 的 RX 与 TX 拨至 ON。

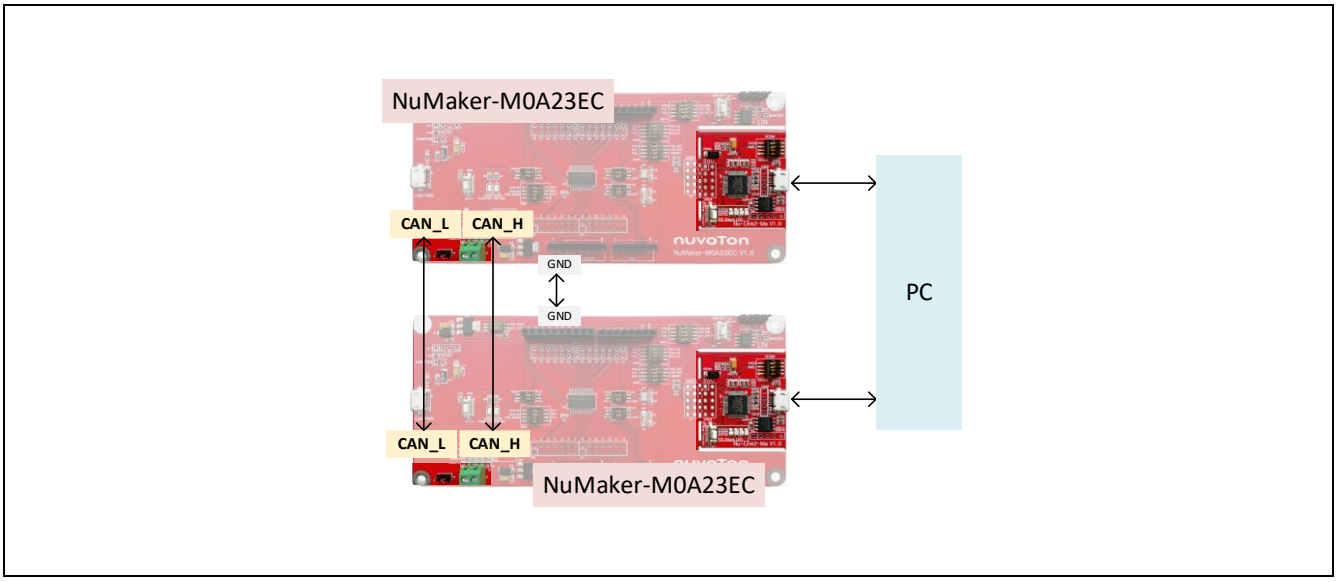

图 3-1 线路示意图

4. 目录信息

目录结构如下图所示。

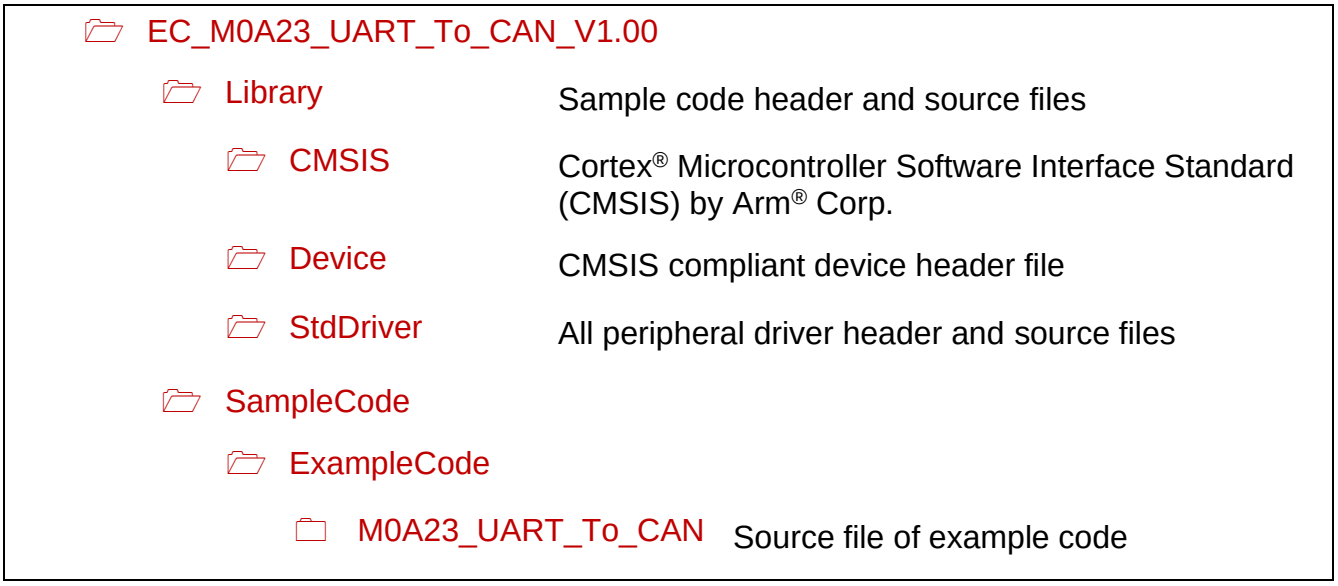

图 4-1 目录信息

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 *M0A23_UART_To_CAN.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
2025.02.28	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.