UART 轉 CAN 的橋接傳輸

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 M0A23 演示 UART 與 CAN 的橋接傳輸
BSP 版本	M0A21_Series_BSP_CMSIS_V3.03.000
開發平台	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

本範例程式碼示範如何在 M0A23 系列平台完成 UART 與 CAN 的橋接傳輸。

相關應用場景，如在汽車電子領域中，當系統需要嵌入 UART 感測器，但又必須透過 CAN 總線進行數據傳輸時，可採用 UART 與 CAN 之間的橋接轉換技術來實現數據互通，並確保穩定傳輸。

1.1 原理

因 UART 與 CAN 的傳輸原理與格式皆不同，此章節將著重於介紹 UART 與 CAN 之間的傳輸格式轉換與傳輸流程。

CAN 的傳輸方式是以 Frame 為單位，而 UART 是以位元組 (bytes) 為單位，由於需要將 UART 收到的數據格式轉換成 CAN 可接收的數據格式，因此需對 UART 接收資料設定為符合 CAN frame 的格式，對應關係設定如表 1-1，而由於 UART 不像 CAN 有明確的 frame 範圍與定義，因此多設定 data header，否則有可能產生數據解析錯誤。

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

當 UART 接收到數據亦確認解析正確後，便會將數據封裝成 CAN frame 並以 CAN_Transmit 送出，流程與判斷方式如圖 1-1。

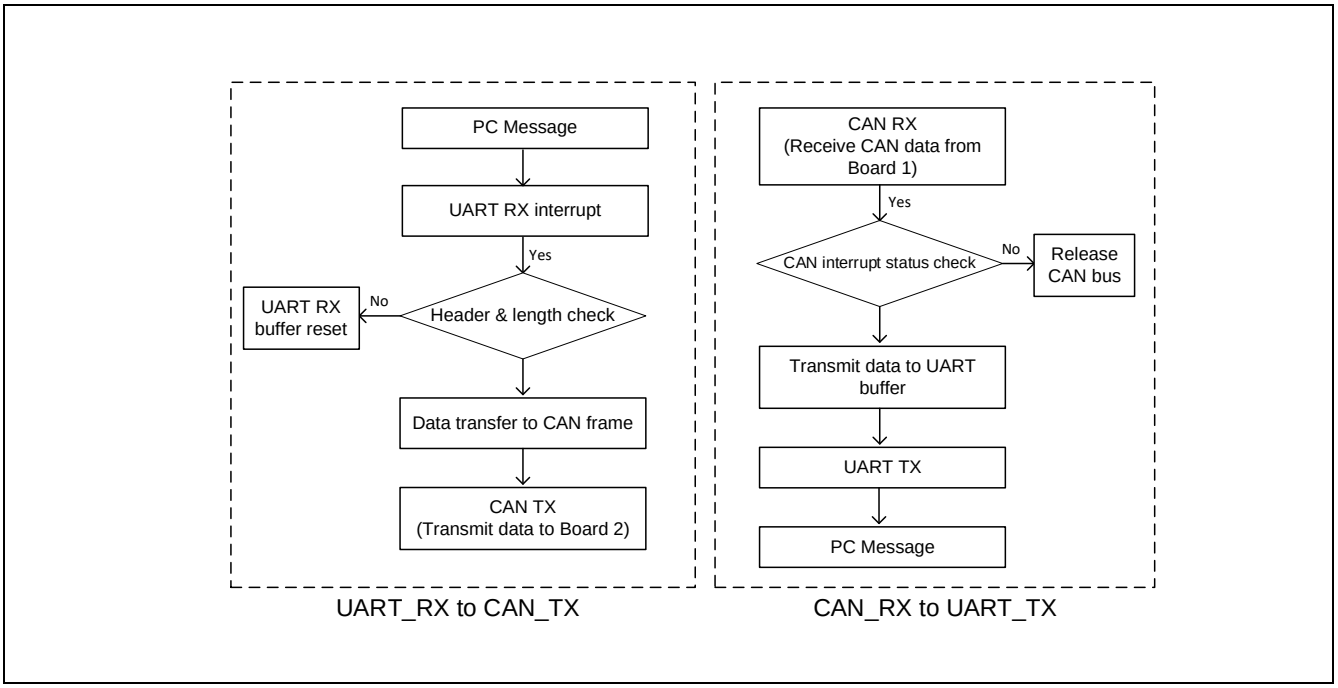

圖 1-1 數據接收與轉換流程

舉例來說，若 UART 接收到的數據，如表 1-1 中的 Example 欄位，轉換為 CAN 封包後，經由 CAN 傳送的數據將如表 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 執行結果

本範例程式碼執行編譯和燒錄的結果如圖 1-2 所示。硬體連接方式請參考 3.2 章節的線路示意圖，軟體工具則需使用支援 Hex 發送與接收的串列埠監測軟體，使用兩塊 NuMaker-M0A23EC，擇一於串列埠監測軟體中輸入符合的 hex 字串，如表 1-1 的範例字串，即可看到另一個串列埠成功接收指定的字串。

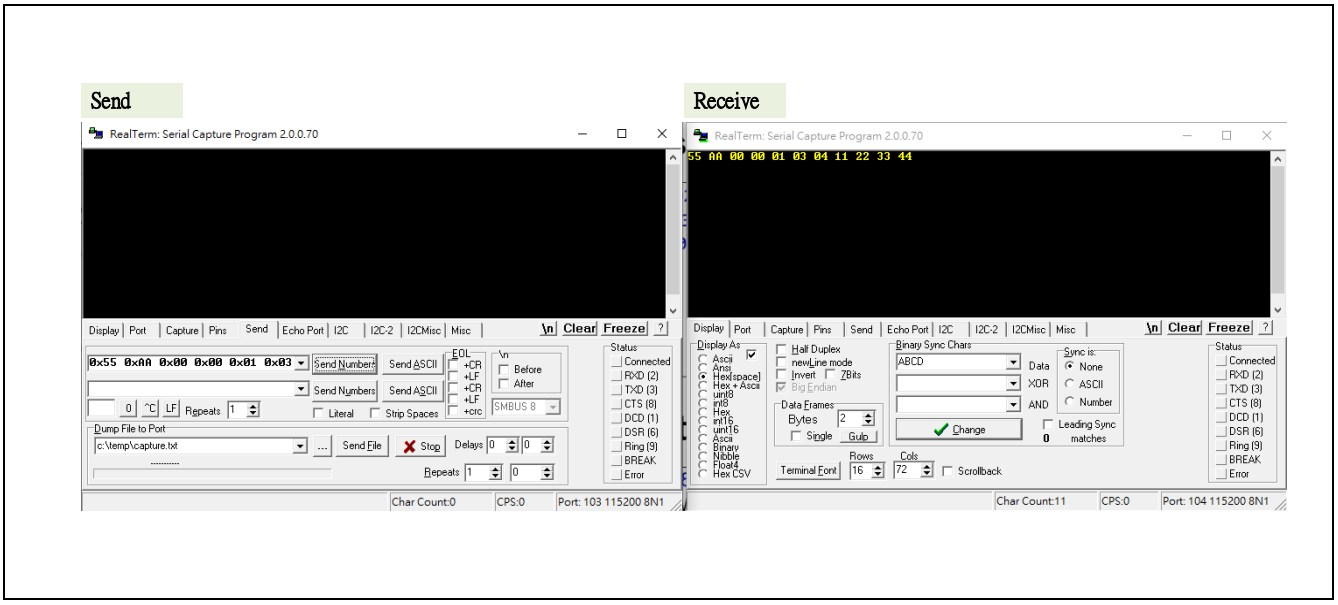

圖 1-2 執行編譯和燒錄結果

2. 程式碼介紹

UART 以 RDA interrupt 接收資料並儲存在 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);
                }
            }
        }
    }
}
```

於 *main.c* 的 while loop 中檢查 UART buffer 中是否有數據，以及分析 UART 接收到的數據格式是否符合定義。

```
        /* 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();
        }
```

若符合定義即可封裝成 CAN frame 格式，並開始傳送。

```
        /* 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. 軟體與硬體需求

3.1 軟體需求

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

3.2 硬體需求

- 電路元件
  - NuMaker-M0A23EC V1.0
- 線路示意圖
  - 將兩塊 NuMaker-M0A23EC 開發版的 CAN_H 與 CAN_L 分別相連，並透過 PC 使用 UART 接收與傳輸數據，以顯示範例程式碼的執行結果。在 NuMaker-M0A23EC 上，可以直接使用板上的 Nu-Link2-Me USB port(ICEJ3)與 PC 相連，並將 VCOM switch 的 RX 與 TX 撥至 ON。

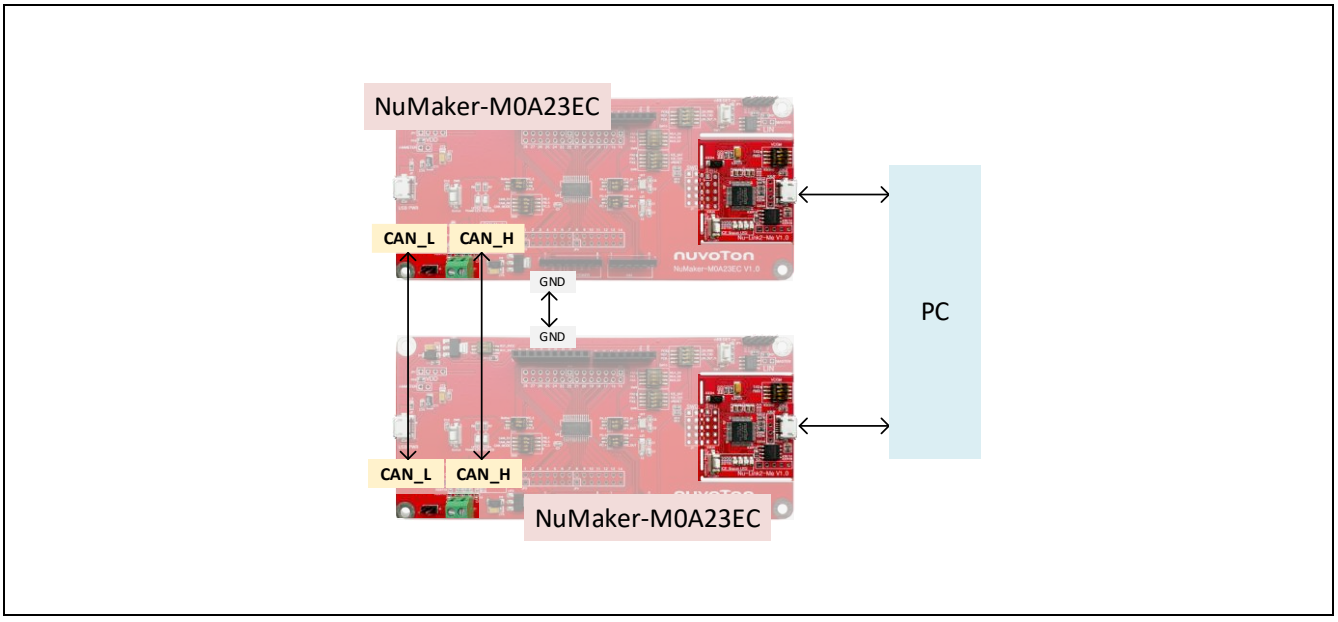

圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

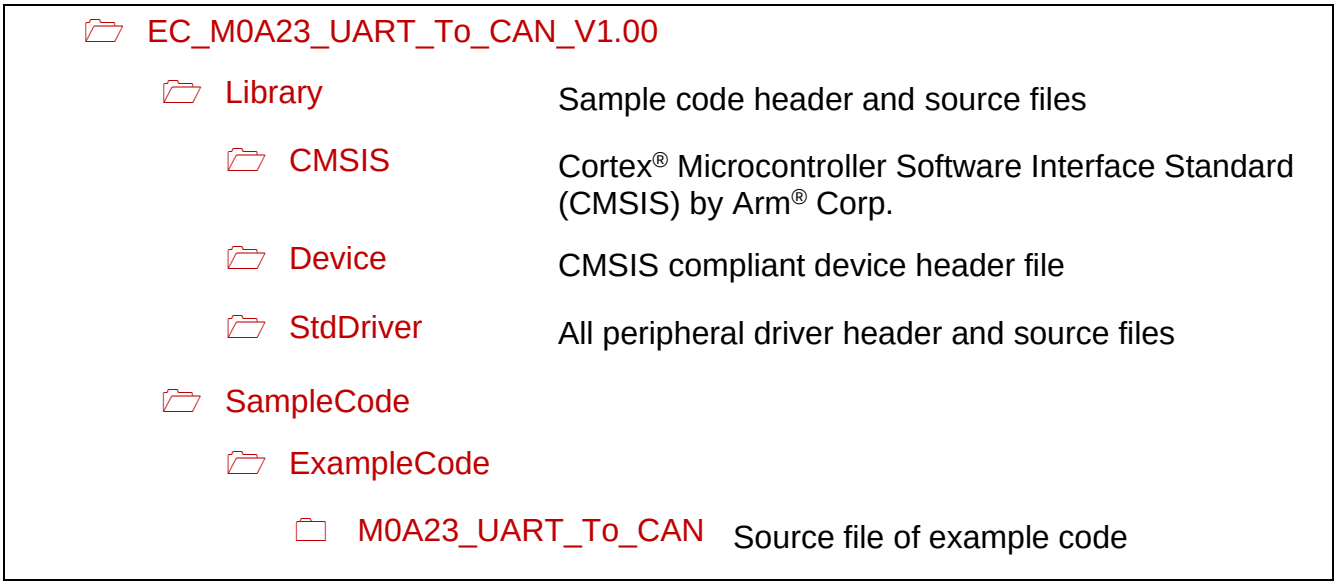

圖 4-1 目錄資訊

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 *M0A23_UART_To_CAN.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
2025.02.28	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.