M2003 UART 模擬 LIN 匯流排從機代碼

NuMicro® 32 位元系列微控制器範例代碼介紹

檔案資訊

應用簡述	本代碼用 M2003 的 UART 來實現 LIN 匯流排從機功能資料收發
BSP 版本	M2003_Series_BSP_CMSIS_V3.00.001
開發平臺	NuMaker-M2003FC V1.1

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

本代碼用 M2003 MCU 實現 LIN 匯流排從機功能，代碼中定義了兩條報文，一條用於發送資料，另一條用於接收資料。

LIN 匯流排上的報文，以 “至少 13 位低電平” 開始，後續跟隨 0x55、報文 ID、資料以及校驗碼。資料欄不能出現 9 位以上的低電平，從機的資料發送必須在收到與自己匹配的 ID 後才能把資料發送出去。時序如圖 1-1。

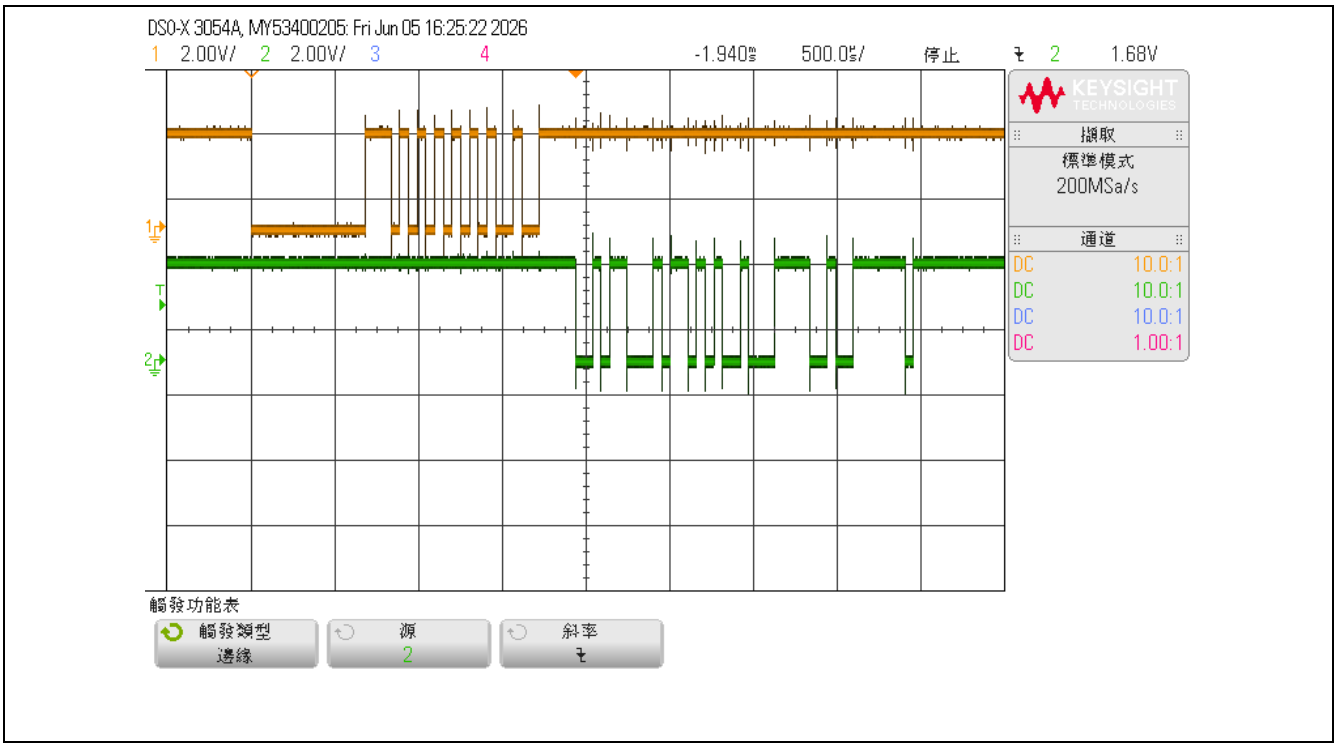


圖 1-1 LIN 匯流排資料

1.1 原理

為簡化 LIN 匯流排從機收發代碼，我們用 UART 的 break 中斷來檢測匯流排上的幀頭，break 中斷後開始接收資料，第一個位元組應是 0x55，第二個位元組是 ID，後續是接收或發送資料。

M2003 做為從機，發送資料時待發送資料全部寫入 UART 發送緩存，發送資料最多 8 個位元組資料加一個檢驗碼，M2003 的 UART 有 16 級緩存。

LIN 從機接收資料最多 11 個位元組：0x55+ID+Data+Checksum。收到的資料先暫存在陣列 StringRx[ ]中，收到當前報文所有資料後再轉存到報文結構體陣列中。

1.2 執行結果

LIN 從機必須在收到匯流排上與自己匹配的 ID 後才能收發，所以測試此代碼必須掛到 LIN 匯流排上，M2003 的 UART0 會列印出收發的報文資訊，如圖 1-2。

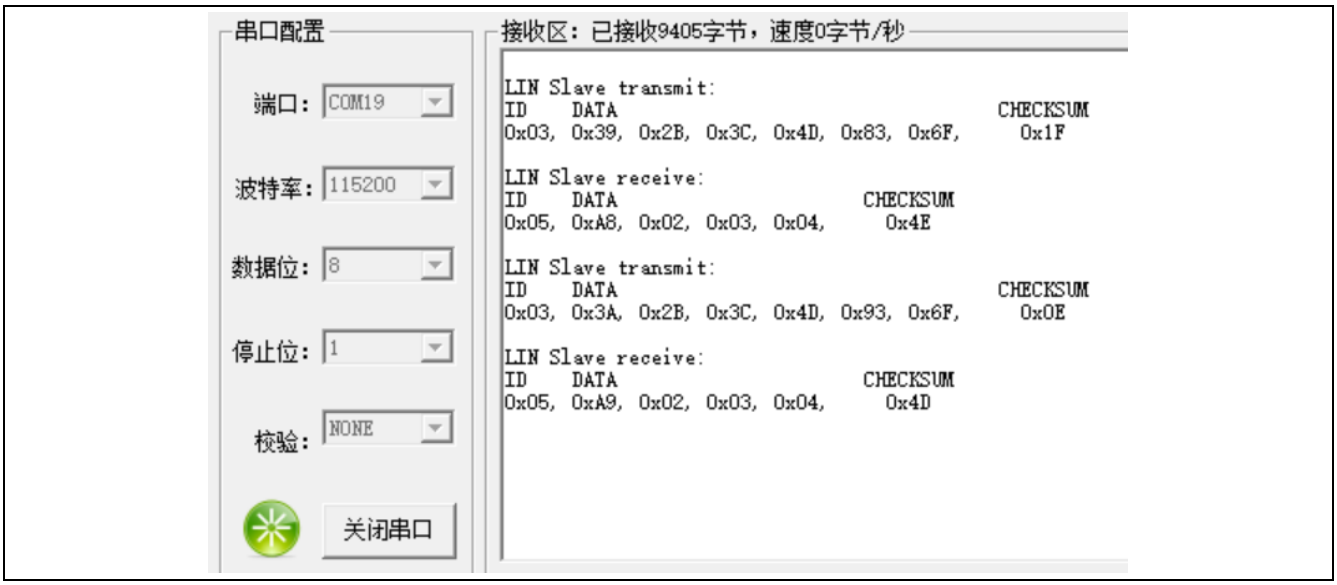


圖 1-2 列印資訊

2. 代碼介紹

LIN 匯流排資料結構體

結構體成員 TX_Frame_Cnt 非 0 時，收到匹配 ID 幀頭才會啟動對應資料幀的發送。在 TX_Frame_Cnt=0 時更新資料，可保證更新資料時未在發送，防止資料混疊。

```
typedef struct
{
    uint8_t  PID;
    /* If frame is for reception, this will be written : 0x55 + PID + Data + CheckSum */
    uint8_t  Data[12];
    /* Data number. Only data number exclude PID 0x55 or others. */
    uint8_t  DataNum;
    /* It will decrease after transmission and stop transmission when it equal 0 */
    uint8_t  TX_Frame_Cnt;
    /* When it equals 0, Data[ ] won't be updated */
    uint8_t  Rx_Frame_Num;
    /* 0 = Classic checksum, 1 = Enhanced checksum */
    uint8_t  CheckSum_Type;
} Message_Slave_Type ;

/*---------------------------------------------*/
/* Define global variables and constants       */
/*---------------------------------------------*/
/* The checksum type must match the Master configuration for the same PID. */
Message_Slave_Type  Message_Slave_TxData =
{
    0x03, {0x1A, 0x2B, 0x3C, 0x4D, 0x5E, 0x6F}, 6, 0, 1, 0
};

/* Here random number is only for example. */
Message_Slave_Type  Message_Slave_RxData =
{
    0x05, { }, 4, 0, 1, 0
};

/* 0x55 + PID + Data + CheckSum = [0]~[10], StringRx[11] is no used. */
uint8_t StringRx[12];
```

代碼中僅定義了一收一發兩個結構體，若需要，還可再多定義幾個。

UART 中斷函數

本代碼用 UART 的 break 中斷檢測幀頭，檢測到幀頭後，匯流排上的資料接收下來保存到陣列 StringRx[ ]。收到的第一個位元組是 0x55，在收到第二個位元組、ID 後，判斷若與某個報文中的 ID 匹配，就開始報文資料的收發。

報文接收完，只有在 Rx_Frame_Num 非 0 時才把資料從暫存陣列 StringRx[ ]拷貝到報文結構體陣列中。中斷外讀數據時先讓 Rx_Frame_Num=0 可避免資料更新混疊。

```
void UART1_IRQHandler(void)
{
    uint8_t temp8;
    uint32_t INT_Status, FIFO_Status;
    Message_Slave_Type static *ptr;

    uint8_t static RxCnt = 11, RxCnt_Max = 11;

    /* Read out interrupt status */
    INT_Status  = UART1->INTSTS;
    /* Read out FIFO status */
    FIFO_Status = UART1->FIFOSTS;

    /* Received a byte */
    if (INT_Status & UART_INTSTS_RDAIF_Msk)
    {
        /* Read out the received byte */
        temp8 = UART1->DAT;

        /* Receive Line interrupt */
        if (INT_Status & UART_INTSTS_RLSIF_Msk)
        {
            /* Break Interrupt */
            if (FIFO_Status & UART_FIFOSTS_BIF_Msk)
            {
                /* Frame head is received */
                if (temp8 == 0) RxCnt = 0;
            }
        }
        /* StringRx[0]=0x55, StringRx[1]=PID, */
        else StringRx[RxCnt++] = temp8;

        if (RxCnt == 2)
        {
            /*
             * PID received = the PID of the Slave response frame.
             * The Master must send this header before the Slave transmits.
             */
            if (temp8 == Message_Slave_TxData.PID)
            {
                /* This value will avoid processing received data */
                RxCnt_Max = 11;

                /* Permit transmission && (Datanum != 0) */
                if ((Message_Slave_TxData.TX_Frame_Cnt) && (Message_Slave_TxData.DataNum))
                {
                    for (temp8 = 0; temp8 <= Message_Slave_TxData.DataNum; ++temp8)
                    {
                        /* Transmit data */
                        UART1->DAT = Message_Slave_TxData.Data[temp8];
                    }
```

```
                    Message_Slave_TxData.TX_Frame_Cnt --;
                }
            }
            /* PID received = the PID of Message 4 */
            else if (temp8 == Message_Slave_RxData.PID)
            {
                ptr = &Message_Slave_RxData;
                /* Prepare to receive data */
                RxCnt_Max = Message_Slave_RxData.DataNum;

                if (RxCnt_Max > 8) RxCnt_Max = 8;
            }

        }
        /* CheckSum has been received into StringRx[RxCnt_Max+2] */
        else if (RxCnt == RxCnt_Max + 3)
        {
            /* Received data will be saved only when this value is non-zero */
            if (ptr->Rx_Frame_Num)
            {
                /* Avoid the value becoming 0 and discard the received data */
                if (LIN_Checksum_Verify(ptr->PID,
                                        &StringRx[2],
                                        ptr->DataNum,
                                        0))
                {
                    for (temp8 = 0; temp8 < RxCnt_Max + 3; ++temp8)
                    {
                        /* Copy data: 0x55 + PID + Data + CheckSum */
                        ptr->Data[temp8] = StringRx[temp8];
                    }

                    /* When read the data, Rx_Frame_Num=0 can avoid data overwriting. */
                    if (++ptr->Rx_Frame_Num == 0) ptr->Rx_Frame_Num = 1;
                }
            }
        }

        /* StringRx[11] will be discarded */
        if (RxCnt > 11) RxCnt = 11;
    }

    UART1->FIFOSTS |= UART_FIFOSTS_PEF_Msk
                    | UART_FIFOSTS_FEF_Msk
                    | UART_FIFOSTS_BIF_Msk;
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M2003_Series_BSP_CMSIS_V3.00.001
- IDE 版本
  - Keil uVision 5.43.1.0
  - VS Code 1.115.3

3.2 硬體需求

- 電路元件
  - NuMaker-M2003FC V1.1
  - 測試本代碼必須掛到 LIN 匯流排上,主機可以用 EC_M3334_UART_LIN_Master_V1.00。
  - 將兩個板子 UART1 相連後，即可互相收發資料。

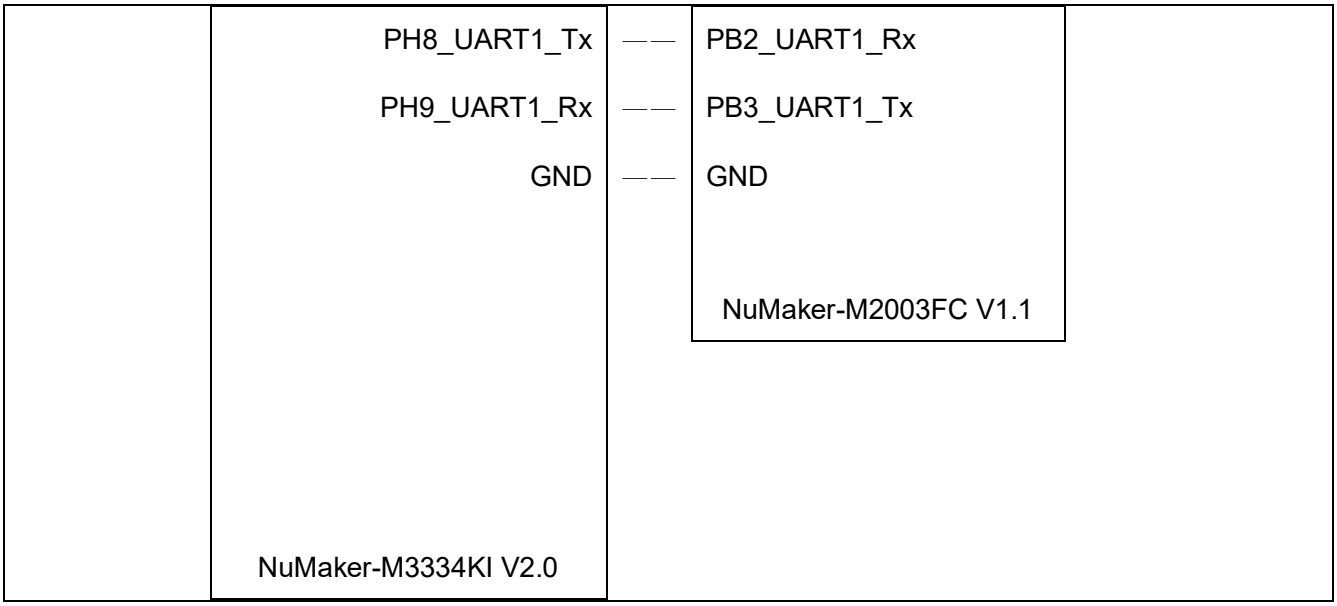


圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

EC_M2003_UART_LIN_Slave_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	
M2003_UART_LIN_Slave	Source file of example code

圖 4-1 目錄資訊

5. 範例程式執行

5.1 Keil 平臺

1. 在 Windows 資源管理器中進入 Keil 工程資料夾，再開啟 *M2003_UART_LIN_Slave.uvprojx* 工程檔。
2. 編譯代碼
3. 下載代碼
4. 進入模擬模式
   - 執行代碼

5.2 VS Code 平臺

1. 在 VS Code IDE 中開啟 **File** 菜單
   - 按一下 **Open Folder**
   - 選取工程目錄下的 **VSCode** 資料夾
2. 選擇 **CMSIS**
   - 按一下 **Build solution** 編譯工程
   - 按一下 **Load & Run application** 下載映射檔到目標晶片
   - 按一下 **Load & Debug application** 進入模擬模式

6. 修訂記錄

Date	Revision	Description
2026.08.19	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.