M2003 UART 模拟 LIN 总线从机代码

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本代码用 M2003 的 UART 来实现 LIN 总线从机功能数据收发
BSP 版本	M2003_Series_BSP_CMSIS_V3.00.001
开发平台	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

本代码用 M2003 MCU 实现 LIN 总线从机功能，代码中定义了两条报文，一条用于发送数据，另一条用于接收数据。

LIN 总线上的报文，以“至少 13 位低电平”开始，后续跟随 0x55、报文 ID、数据以及校验码。数据域不能出现 9 位以上的低电平，从机的数据发送必须在收到与自己匹配的 ID 后才能把数据发送出去。时序如图 1-1。

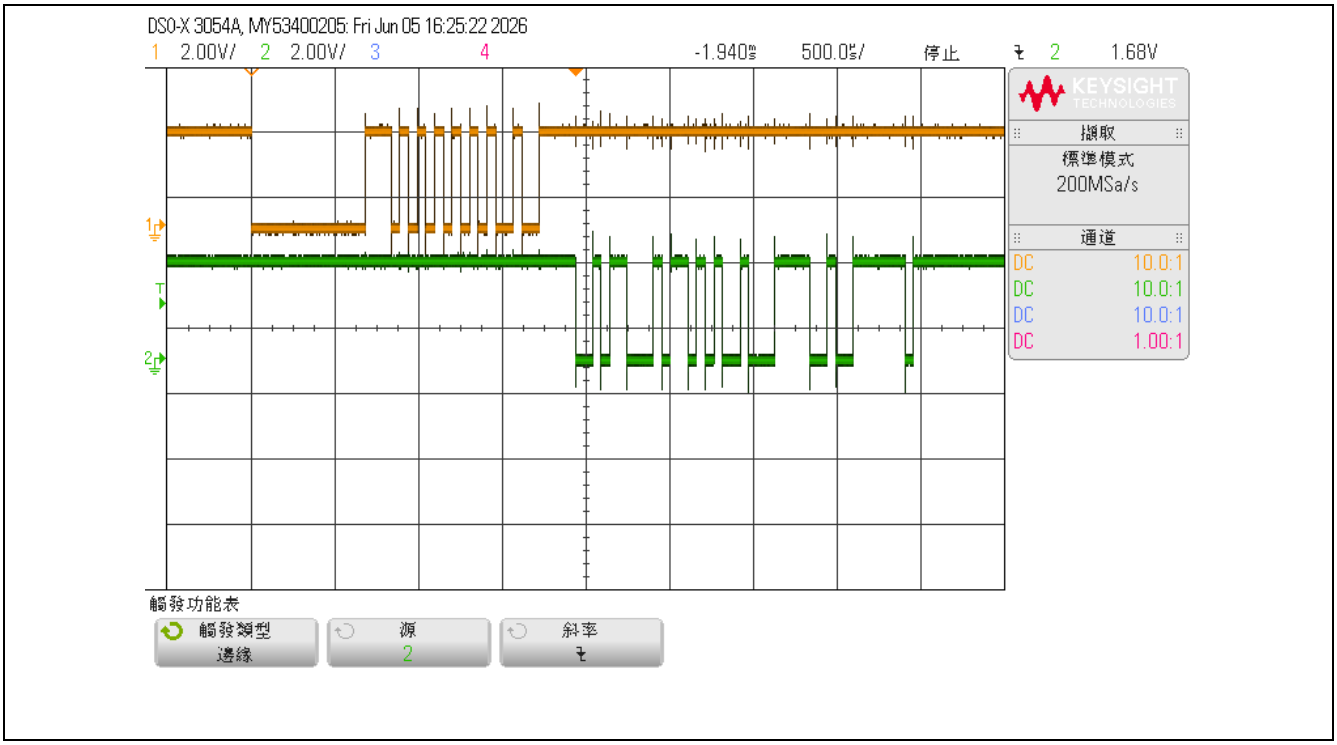


图 1-1 LIN 总线数据

1.1 原理

为简化 LIN 总线从机收发代码，我们用 UART 的 break 中断来检测总线上的帧头，break 中断后开始接收数据，第一个字节应是 0x55，第二个字节是 ID，后续是接收或发送数据。

M2003 做为从机，发送数据时待发送数据全部写入 UART 发送缓存，发送数据最多 8 个字节数据加一个检验码，M2003 的 UART 有 16 级缓存。

LIN 从机接收数据最多 11 个字节：0x55+ID+Data+Checksum。收到的数据先暂存在数组 StringRx[ ]中，收到当前报文所有数据后再转存到报文结构体数组中。

1.2 执行结果

LIN 从机必须在收到总线上与自己匹配的 ID 后才能收发，所以测试此代码必须挂到 LIN 总线上，M2003 的 UART0 会打印出收发的报文信息，如图 1-2。

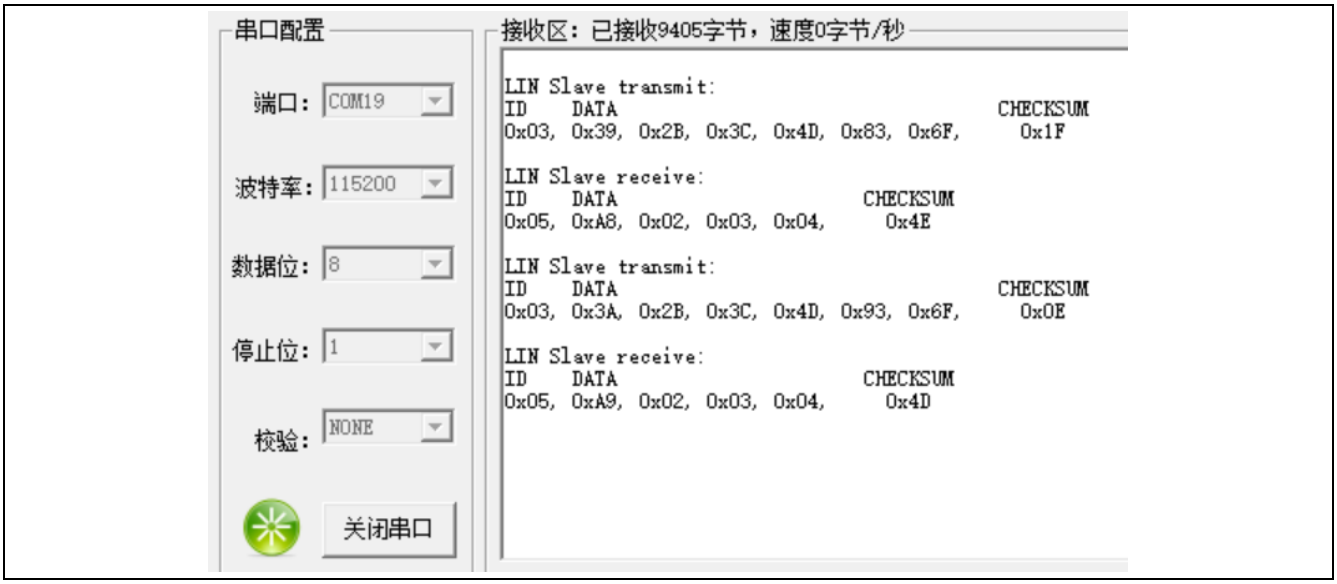


图 1-2 打印信息

2. 代码介绍

LIN 总线数据结构体

结构体成员 TX_Frame_Cnt 非 0 时，收到匹配 ID 帧头才会启动对应数据帧的发送。在 TX_Frame_Cnt=0 时更新数据，可保证更新数据时未在发送，防止数据混叠。

```
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];
```

代码中仅定义了一收一发两个结构体，若需要，还可再多定义几个。

UART 中断函数

本代码用 UART 的 break 中断检测帧头，检测到帧头后，总线上的数据接收下来保存到数组 StringRx[ ]。收到的第一个字节是 0x55，在收到第二个字节、ID 后，判断若与某个报文中的 ID 匹配，就开始报文数据的收发。

报文接收完，只有在 Rx_Frame_Num 非 0 时才把数据从暂存数组 StringRx[ ]拷贝到报文结构体数组中。中断外读数据时先让 Rx_Frame_Num=0 可避免数据更新混叠。

```
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. 软件与硬件需求

3.1 软件需求

- 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 硬件需求

- 电路组件
  - NuMaker-M2003FC V1.1
  - 测试本代码必须挂到 LIN 总线上,主机可以用 EC_M3334_UART_LIN_Master_V1.00。
  - 将两个板子 UART1 相连后，即可互相收发数据。

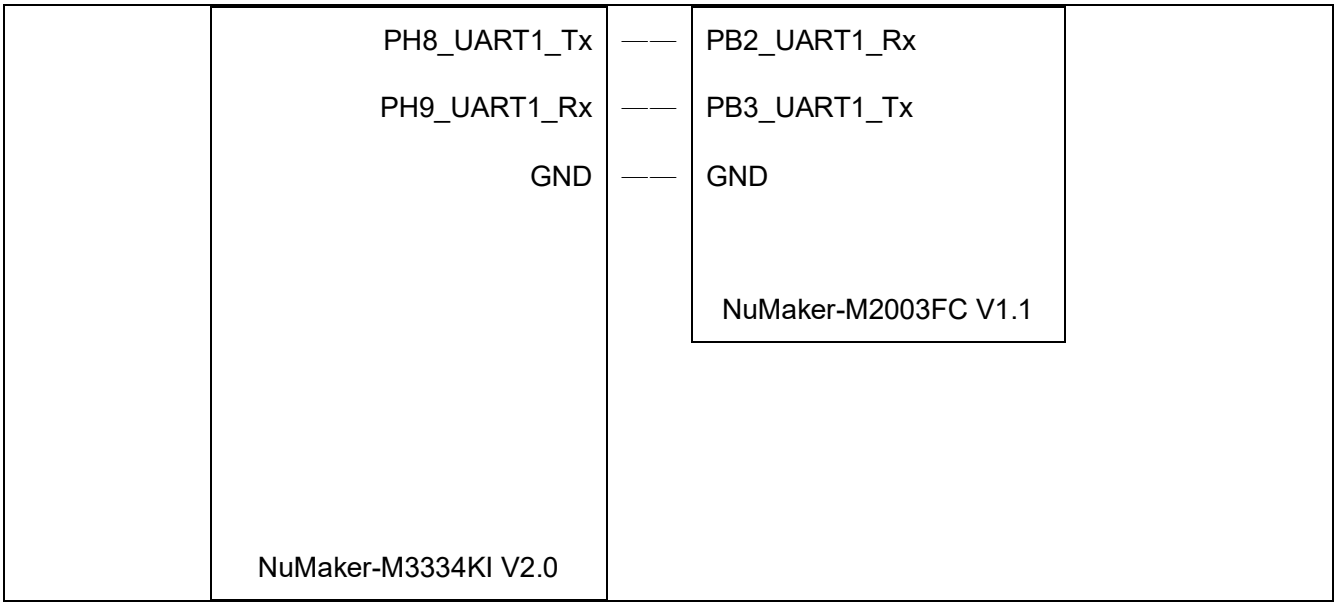


图 3-1 线路示意图

4. 目录信息

目录结构如下图所示。

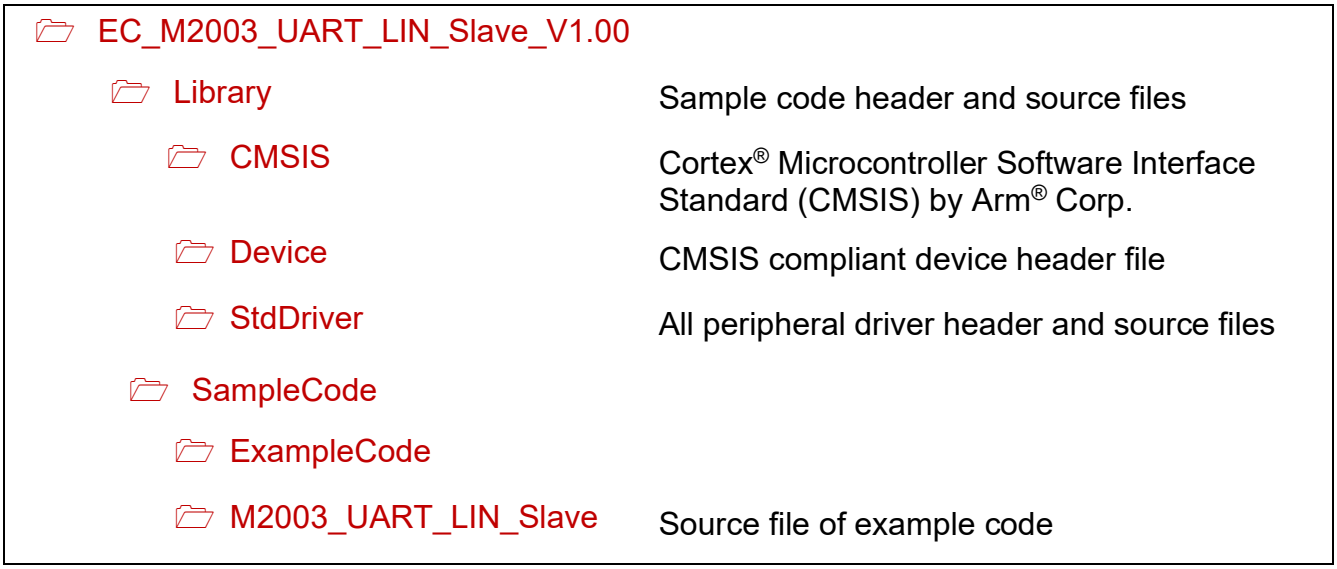


图 4-1 目录信息

5. 范例程序执行

5.1 Keil 平台

1. 在 Windows 资源管理器中进入 Keil 工程文件夹，再开启 *M2003_UART_LIN_Slave.uvprojx* 工程文件。
2. 编译代码
3. 下载代码
4. 进入仿真模式
    - 执行代码

5.2 VS Code 平台

1. 在 VS Code IDE 中开启 **File** 菜单
    - 单击 **Open Folder**
    - 选取工程目录下的 **VSCode** 文件夹
2. 选择 **CMSIS**
    - 单击 **Build solution** 编译工程
    - 单击 **Load & Run application** 下载映像文件到目标芯片
    - 单击 **Load & Debug application** 进入仿真模式

6. 修订记录

Date	Revision	Description
2026.08.19	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.