M3334 UART 模拟 LIN 总线主机代码

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 M3334 的 UART1 实现 LIN 总线主机收发代码
BSP 版本	M3334_Series_BSP_CMSIS_V3.00.002
开发平台	NuMaker-M3334KI V2.0

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

LIN 总线是基于 UART/SCI 的、单主多从结构的低成本接口，最高速率 20Kbps，常用速率 19200bps，9600bps。其报文帧结构以 13 位 0 开始，后续跟着标准 UART 通信的多个字节：0x55+ID+数据+校验和。数据个数双方约定好，最多 8 字节。13bit0+0x55+ID 是帧头，只能主机发送。从机只能在收到与自己 ID 匹配的帧头后，把数据发出。故主机必须周期性的发送帧头，以便从机及时把数据发出。

本代码用 UART 实现了 LIN 主机功能。代码中定义了一个含六条报文的报文列表，列表内的报文循环发送，发送间隔是 20 ms，由 Timer 完成定时。每次帧头发完后，后续若是主机发送就继续发送数据，后续若是主机接收，就启动数据接收，总线时序如图 1-1。

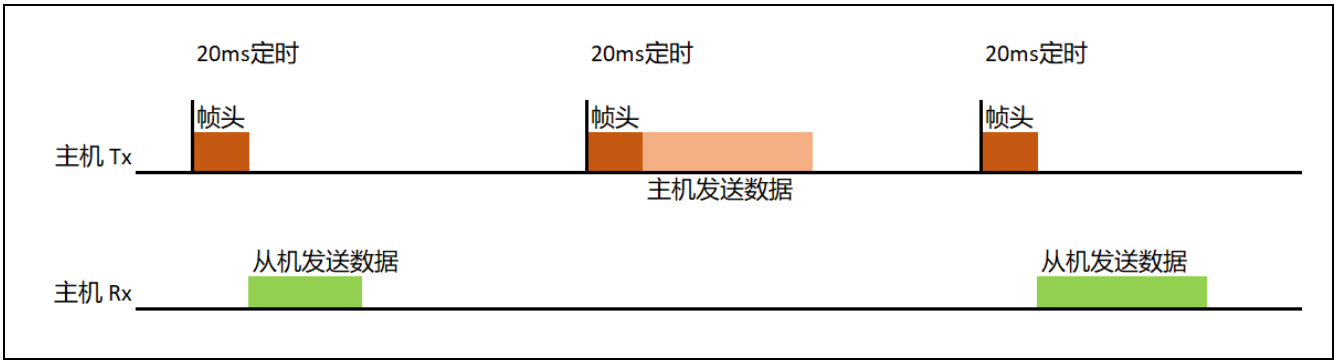


图 1-1 LIN 总线时序

1.1 原理

帧头结构如 图 1-2 所示。

- 帧 ID 取值 0~0x3F 共 64 个值，放在 ID 段的 bit0~bit5。
- ID 段 bit6 = bit0，bit1，bit2，bit4 的异或值。Bit7=bit1，bit3，bit4，bit5 异或值再取反。
- 波特率最高 20 kbps，本代码中取 19200 bps。

在 Timer 的 20 ms 周期中断里，把波特率降为 9/13 发送一个字节 0，在总线上产生正常波特率的 13 位时长的 0。13 位 0 发送结束后，进 UART 中断发送 0x55 和 ID，到此帧头发送结束。

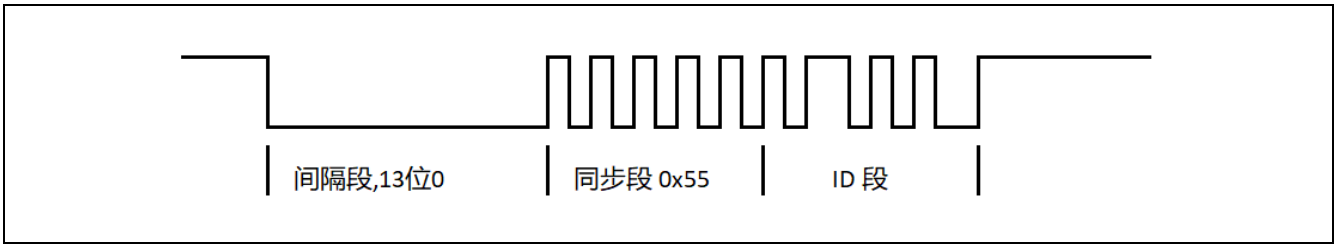


图 1-2　LIN 总线帧头

UART 中断里发送完 0x55 和 ID 后，再判断当前帧若是主机发送，就继续发送数据；当前帧若是主机接收，则配置接收长度、做接收前的准备。

1.2 执行结果

代码中定义的六条报文 Message_Buffer[6]，四条发送，两条接收。为方便查看本代码执行效果，main 函数中周期性的启动了 ID 等于 0x05 的报文的发送，打印信息如图 1-3 所示。



图 1-3 打印信息 1

若挂个 LIN 从机，打印接口还可会打印出接收报文的信息，如图 1-4 所示。

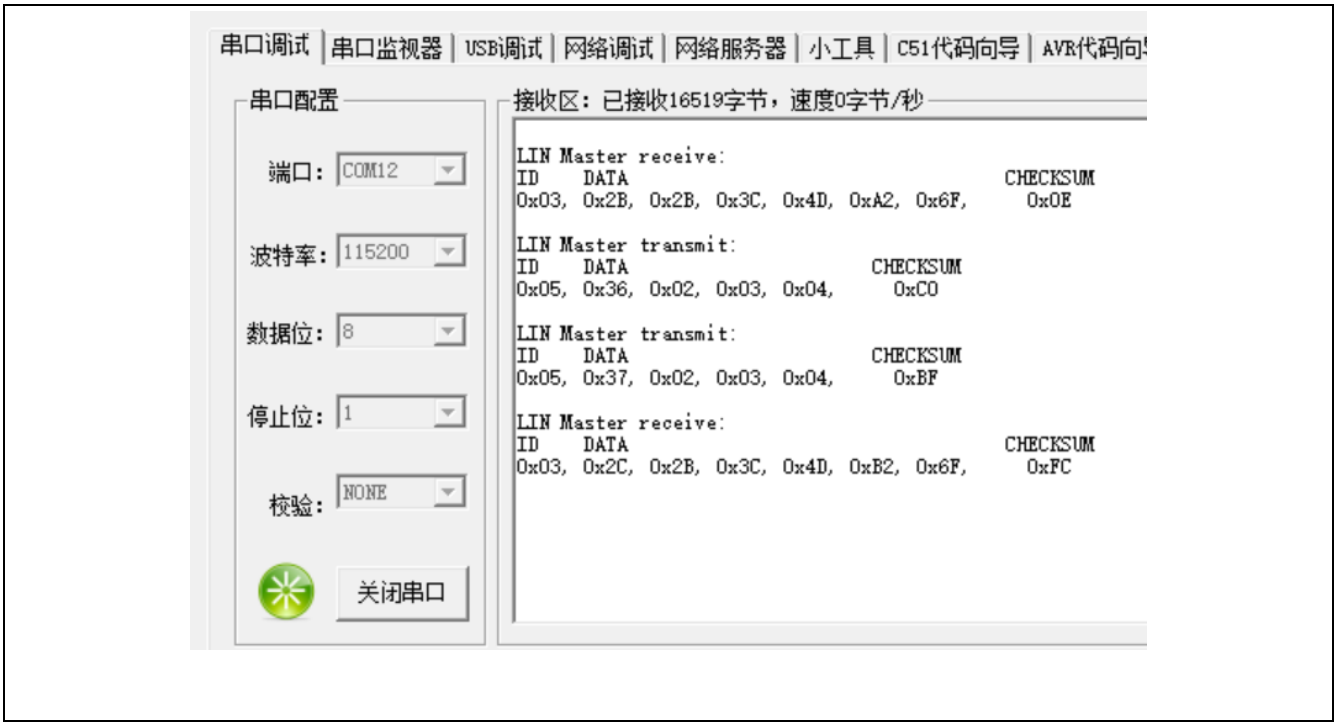

串口调试
串口监视器
USB调试
网络调试
网络服务器
小工具
C51代码向导
AVR代码向
串口配置
端口: COM12
波特率: 115200
数据位: 8
停止位: 1
校验: NONE
关闭串口
接收区：已接收16519字节，速度0字节/秒
LIN Master receive:
ID DATA CHECKSUM
0x03, 0x2B, 0x2B, 0x3C, 0x4D, 0xA2, 0x6F, 0x0E
LIN Master transmit:
ID DATA CHECKSUM
0x05, 0x36, 0x02, 0x03, 0x04, 0xC0
LIN Master transmit:
ID DATA CHECKSUM
0x05, 0x37, 0x02, 0x03, 0x04, 0xBF
LIN Master receive:
ID DATA CHECKSUM
0x03, 0x2C, 0x2B, 0x3C, 0x4D, 0xB2, 0x6F, 0xFC

图 1-4 打印信息 2

2. 代码介绍

Timer1 20 ms 中断代码

进中断先把波特率降为 9/13 发一个字节 0，如此总线上产生正常波特率的 13 位 0。发送 ID 前重新计算一下 ID 的 bit6 bit7，如此，代码中修改报文 ID 不必手算 ID 校验码。13 位 0 发送结束后产生能 UART 发送结束中断，由 UART 中断收发后续数据。

```
void TMR1_IRQHandler(void)
{
    int32_t temp32, temp32_1;

    Timer1_Period_Dec --;

    /*
     * Generate LIN Break by transmitting UART 0x00 at a reduced
     * baud rate. UART 0x00 has 9 LOW bits (start + 8 data bits).
     * Scaling the baud rate makes these approximately 13 LIN bits.
     *
     * At 19.2 kbps:
     *     13 LIN bits = 677.08 us
     *
     * Verify the actual waveform because the UART baud-rate divider
     * may introduce a small timing error.
     */
    UART_Open(UART1, LIN_BREAK_BAUD_RATE);
    UART1->DAT = 0x00;
    Frame_Head = TRUE;

    if (++ptr_Msg > Message_End) ptr_Msg = Message_Buffer;

    /* Clear TIMER1 time-out interrupt flag */
    TIMER_ClearIntFlag(TIMER1);
    /* bit6 = bit1 ^ bit4 */
    temp32   = (ptr_Msg->PID << 5) ^ (ptr_Msg->PID << 2);
    /* bit2 = bit0 ^ bit2 */
    temp32_1 = (ptr_Msg->PID << 6) ^ (ptr_Msg->PID << 4);
    /* bit6 = bit0 ^ bit1 ^ bit2 ^ bit4 */
    temp32_1 = (temp32 ^ temp32_1) & 0x40;

    temp32 <<= 1;
    temp32  ^= ptr_Msg->PID << 4;
    temp32  ^= ptr_Msg->PID << 2;
    temp32  ^= 0x80;

    /* bit7 = ~(bit1 ^ bit3 ^ bit4 ^ bit5) */
    temp32 &= 0x80;
    /* Calculate bit6,7 of PID */
    ptr_Msg->PID = (ptr_Msg->PID & 0x3F) | temp32 | temp32_1;

    if (ptr_Msg->DataNum > 8) ptr_Msg->DataNum = 8;
```

```
    /* Last frame stop receiving */
    UART_DisableInt(UART1, UART_INTEN_RXTOIEN_Msk);

    /* 13bit zero finishing will occur interrupt */
    UART_EnableInt(UART1, UART_INTEN_TXENDIEN_Msk);
}
```

帧数据结构

成员 Tx_Frame_Cnt 只用于发送数据帧，报文数据每发一次减 1，减到 0 对应 ID 的数据帧不再发送数据，但仍会发送帧头。故此值=0 时可更新待发送的数据，不会引起发送数据混叠。

Rx_Frame_Num 只用于接收数据帧，每收到一帧对应 ID 的完整报文数据，就加 1。此值=0 时收到的数据不更新到帧列表结构体中，因此把此值写 0 再读列表内的数据可避免接收数据混叠。

```
typedef struct
{
    /* 1: This frame will transmit data.  0: This frame is for reception */
    uint8_t  Flag_Master_Tx;
    /* Message ID */
    uint8_t  PID;
    /* Data for transmiting */
    uint8_t  TxData[9];
    /* data number, Max =8 */
    uint8_t  DataNum;
    /* received (data + Checksum) */
    uint8_t  RxData[12];
    /* when Tx_Frame_Cnt != 0, data TxData[] will be transmited */
    uint8_t  Tx_Frame_Cnt;
    /* If Rx_Frame_Num=0, RxData[] won't be updated. */
    uint8_t  Rx_Frame_Num;
    /* If this equals 0 checksum is classic */
    uint8_t  CheckSum_Type;
} Message_Type;
```

UART 中断函数

进中断后若 Frame_Head 非 0 表明前面 Timer 刚发完 13 位 0，恢复 19200 波特率继续发帧头数据 0x55 和 帧 ID。

接下来按帧结构数据中的 Flag_Master_Tx 判断当前帧是主机发送、还是主机接收。因 M3334 UART 有 16 级缓存，而 LIN 总线最多 8 字节数据加校验码。故若是主机发送，则可把数据一次性全部写入发送缓存。若是主机接收，则要等 0x55+ID 发送结束后再启动接收，目的是单线收发时，不接收自己发的 0x55+ID，因此无论单线还是双线应用，统一都只接收报文的数据域。

```
void UART1_IRQHandler(void)
{
    int32_t static TxCnt, RxCnt, CntMax;
```

```
    uint8_t static RxStr_Buf[11];
    uint8_t temp8u;

    /* If frame header has just been transmitted out. */
    if (Frame_Head == TRUE)
    {
        /* Transmit frame head */
        UART_Open(UART1, LIN_BAUD_RATE);
        UART1->DAT = 0x55;
        UART1->DAT = ptr_Msg->PID;

        /* Disable UART Tx interrupt */
        UART_DisableInt(UART1, UART_INTEN_TXENDIEN_Msk);
        Frame_Head = FALSE;

        CntMax = ptr_Msg->DataNum;

        /* Master transmits data subsequently */
        if (ptr_Msg->Flag_Master_Tx)
        {
            /* When CPU updates the data, please set Tx_Frame_Cnt = 0 */
            if (ptr_Msg->Tx_Frame_Cnt)
            {
                /* Decrease every transmision */
                ptr_Msg->Tx_Frame_Cnt --;
                TxCnt = 0;

                while (TxCnt <= CntMax)
                {
                    /* Transmits data and CheckSum */
                    UART1->DAT = ptr_Msg->TxData[TxCnt++];
                }
            }
        }
        /* Master receive data subsequently. */
        else
        {
            /* Wait for 0x55+PID is transmitted completely */
            while ((UART1->FIFOSTS & UART_FIFOSTS_TXEMPTYF_Msk) == 0);

            RxCnt = 0;
            /* Clear Rx FIFO for acceptance */
            UART1->FIFO |= UART_FIFO_RXRST_Msk;

            /* Waitting for Rx FIFO clear completion */
            while ((UART1->FIFOSTS & UART_FIFOSTS_RXIDLE_Msk) == 0) __NOP();

            /* Enable TxD Timer over interrupt */
            UART_EnableInt(UART1, UART_INTEN_RXTOIEN_Msk);
        }

    }
    /* Rx data process */
    else
    {
        /* RxD Timer over interrupt */
```

```
        while (UART1->INTSTS & UART_INTSTS_RXTOINT_Msk)
        {
            while ((UART1->FIFOSTS & UART_FIFOSTS_RXEMPTY_Msk) == 0)
            {
                /* Read out received data */
                temp8u = UART1->DAT;

                /* Copy data and CheckSum */
                if (RxCnt <= CntMax) RxStr_Buf[RxCnt++] = temp8u;
                /* After receiving this frame Data field, disable UART1 TO interrupt */
                else UART_DisableInt(UART1, UART_INTEN_RXTOIEN_Msk);
            }

            /* Last byte, checksum has been received */
            if (RxCnt == CntMax + 1)
            {
                /*
                 * Accept the response only when the LIN checksum valid.
                 */
                if (LIN_Checksum_Verify(ptr_Msg->PID,
                                        RxStr_Buf,
                                        CntMax,
                                        ptr_Msg->CheckSum_Type))
                {
                    /* If Rx_Frame_Num=0, data updates here are prohibited */
                    if (ptr_Msg->Rx_Frame_Num)
                    {
                        for (RxCnt = 0; RxCnt <= CntMax; ++RxCnt)
                        {
                            /* Copy data and CheckSum */
                            ptr_Msg->RxData[RxCnt] = RxStr_Buf[RxCnt];
                        }

                        if (++ptr_Msg->Rx_Frame_Num == 0) ptr_Msg->Rx_Frame_Num = 1;
                    }
                }
                else
                {
                    /* Invalid LIN checksum: discard the response. */
                    LIN_Rx_Checksum_Error++;
                }
            }
        }
    }
}
```

当收到数据个数等于帧数据长度再多一字节时，表明已收到校验和字节，可把收到的数据从 RxStr_Buf [ ] 转存到帧列表对应 ID 的数组中。成员 Rx_Frame_Num=0 时不转存数据，故从帧列表中读数据时，先写 Rx_Frame_Num=0，可防止读报文时发生接收数据混叠。

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M3334_Series_BSP_CMSIS_V3.00.002
- 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-M3334KI V2.0。
  - 若无其它测试从机,可用 EC_M2003_UART_LIN_Slave_V1.00 做测试从机。
- 线路示意图

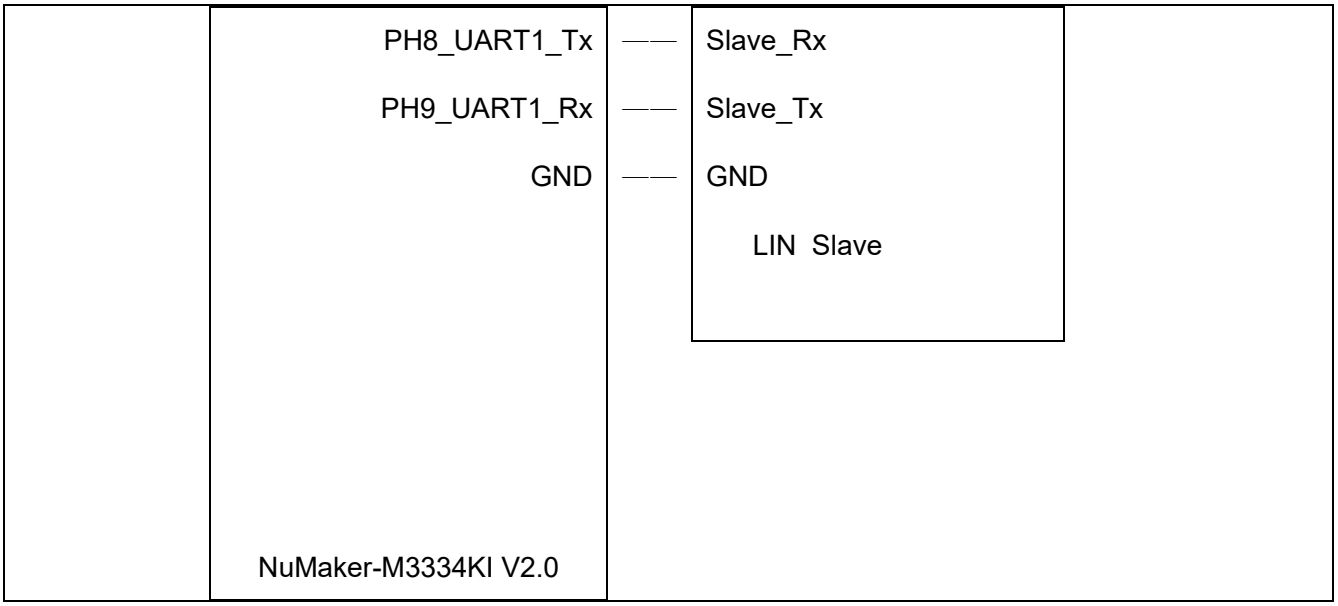


图 3-1 线路示意图

4. 目录信息

目录结构如下图所示。

EC_M3334_UART_LIN_Master_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	
M3334_UART_LIN_Master	Source file of example code

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 *M3334_UART_LIN_Master.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
2026.09.19	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.