M460 的 4 路 CAN FD 消息通过 VCOM 转 SLCAN 协议

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码将 M460 的 4 路 CAN FD 消息通过 VCOM 接口转成 SLCAN 协议发送到电脑上。
BSP 版本	M460_Series_BSP_CMSIS_V3.00.002
开发平台	3-In-1 Analyzer Tool

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

CAN FD 的消息以 2 进制的方式存 MCU 内，不方便观测操作 CAN FD 总线上的数据。本范例通过虚拟串口 VCOM 将 CAN FD 消息转换成 ASCII 码形式的 SLCAN 协议用以与 PC 上的调试助手进行通讯。

1.1 概述介绍以下 4 个基本内容。

- CAN FD 简介
- VCOM 简介
- SLCAN 协议介绍
- CAN 调试助手

1.2 CAN FD 简介

2012 年，博世推出 CAN FD（具有灵活数据速率的 CAN），以克服传统 CAN 的比特率限制为 1 Mbps，并将每个 CAN 帧的数据字节数从最多 8 个扩展到 64 个，从而缩小经典 CAN 与其他协议之间的差距。这是通过修改 CAN 帧格式来实现的，其中比特率可以在 CAN 帧内切换到更快的值，并通过新的数据长度编码来实现。CAN FD 标准化为 ISO11898-1:2015。CAN FD 协议控制器兼容经典的 CAN 通信。

CAN FD 对比经典的 CAN 的主要特点：

- 更高的传输速率：数据域最最高速率可达到 8 Mbps。
- 更多的数据长度：最高可达到 64 Bytes。

1.3 VCOM 简介

虚拟串口 Virtual Communication Port 简称 VCOM，使用 USB 接口替换之前 PC 的硬件串行通讯口。本例中使用高性能单片机 M467 的 1 个高速 USB 接口虚拟最多 4 路 VCOM，可同时收发 4 路不同速率的 CAN FD 消息。

1.4 SLCAN 协议介绍

Serial Line CAN(简称 SLCAN)是一种基于文本 ASCII 码的串行通讯口协议，也是 LAWICEL 硬件设备的协议，本文中使用的 SLCAN 通讯协议兼容部分 CAN232 设备的协议，CAN232 最新版本为 2023 年的 Version 3.0B。

该协议可以通过如下网站进行下载 LAWICEL CAN232 | canusb.com。

SLCAN 协议格式：命令+参数+[CR]。

本例程已支持的命令如下图：

命令	命令码	参数1	参数2	参数3	参数4
打开SLCAN口	O	[CR]			
关闭SLCAN口	C	[CR]			
设置仲裁域波特率	S	1~8	[CR]		
设置数据域波特率	Y	2，5	[CR]		
CAN标准远程帧	r	11位帧ID	数据长度	[CR]	
CAN标准帧	t	11位帧ID	数据长度	数据	[CR]
CAN FD标准帧	d	11位帧ID	数据长度	数据	[CR]
CAN FD变速帧	b	11位帧ID	数据长度	数据	[CR]
CAN 扩展远程帧	R	29位帧ID	数据长度	[CR]	[CR]
CAN扩展帧	T	29位帧ID	数据长度	数据	[CR]
CAN FD扩展帧	D	29位帧ID	数据长度	数据	[CR]
CAN FD变速帧	B	29位帧ID	数据长度	数据	[CR]
软件版本	V	[CR]			
硬件版本	v	[CR]			
序列号	N	[CR]			

图 1–1 已支持的 SLCAN 协议命令

仲裁域支持的波特率参数：

命令：S	参数和仲裁域波特率对照								
参数1	0	1	2	3	4	5	6	7	8
仲裁域波特率	10kbps	20kbps	50kbps	100kbps	125kbps	250kbps	500kbps	800kbps	1Mbps

图 1–2 仲裁域参数波特率对照

数据域支持的波特率参数：

命令：Y	参数和数据域波特率对照	
参数1	2	5
数据域波特率	2Mbps	5Mbps

图 1–3 数据域参数波特率对照

数据长度的索引码：

帧长度	参数2和数据长度对照						
参数2	9	A	B	C	D	E	F
数据长度	12Bytes	16Bytes	20Bytes	24Bytes	32Bytes	48Bytes	64Bytes

图 1–4 数据长度参数和长度对照

1.5 CAN 调试助手

CAN 调试助手可以使用 2 种：串口调试助手和 Cangaroo 调试助手。

- 串口调试助手：

  通过标准串口收发数据，接收 SLCAN 协议数据需要自行进行解析，发送 CAN FD 消息需要自行进行 SLCAN 协议编码。本例程使用串口调试助手监测固件的运行日志。

- Cangaroo 调试助手：

  Cangaroo，一个开源的 CAN 总线数据分析工具，由 Hubert Denkmair 开发并维护。这个小巧且功能强大的软件旨在帮助电子工程师和爱好者解析和理解车载网络通信数据。Cangaroo 基于 Qt 框架构建，提供一种直观的图形用户界面，支持多种操作系统。

  本例程的使用的 Cangaroo 针对新唐硬件平台进行优化，软件内含解析功能，可以方便直接发送需要的 CAN FD 消息，并且可以直观的观测到 CAN 总线上接收到的数据。1 次可以打开 4 个调试口，发送的时候可以快速选择需要发送的数据端口，并可同时监测 4 个数据端口接收到的消息。

2. 代码介绍

本节内容先介绍代码工作流程，以方便理解运行机制，进而更好的对代码进行介绍。最后再分析一个代码的运行实例。

- 流程图：数据收发流程，CAN 调试操作流程。
- 代码简介：USB 虚拟 VCOM，数据收发，协议转换。
- 运行实例：1 发 3 收，监测运行日志，分析对比收数据。

2.2 流程图

数据收发流程：

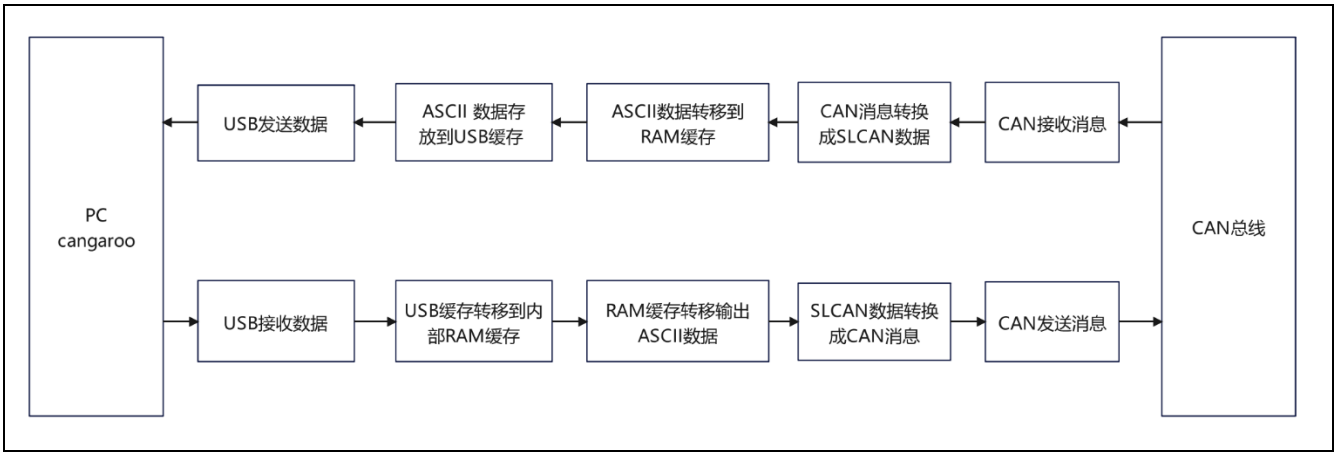

图 2–1 数据收发流程图

数据收发主要涉及到 4 个缓存区域：

- USB 缓存：存放 USB 收发实时的包数据。
- RAM 缓存：用于双向数据的传递，提高数据响应的实时性。
- SCLAN 缓存：用于转换 CAN 消息的存储区。
- CAN 缓存：存放 CAN 收发实时的消息数据。

CAN 调试操作流程：

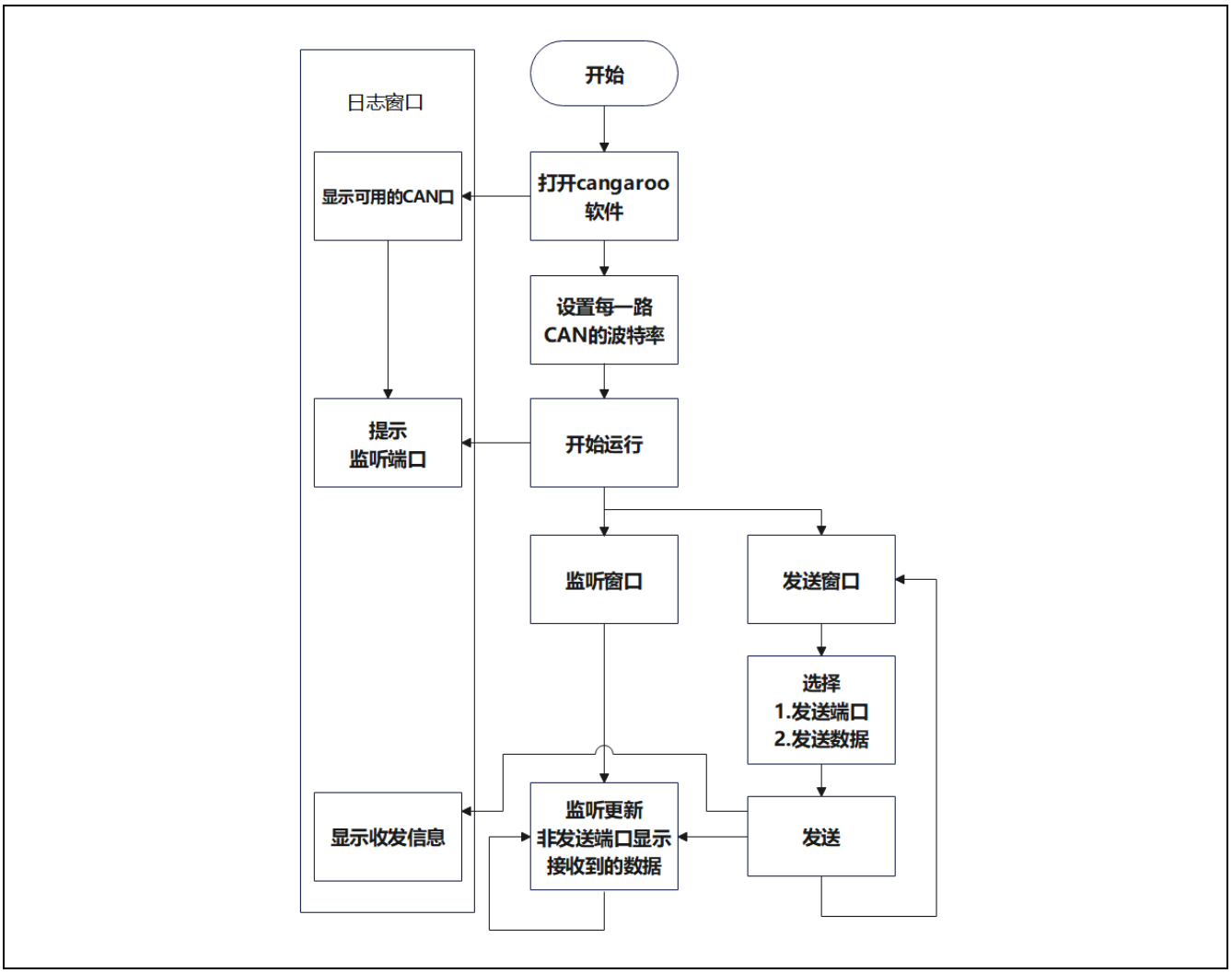

图 2–2 上位机调试操作流程图

CAN 调试助手有 3 个窗口（图 2–3）：

- 日志窗口：显示当先的设备状态，已发送的 CAN 消息。
- 监听窗口：显示当前 CAN 接收到的消息，根据时间戳可调整显示方式。
- 发送窗口：选择需要发送的端口，发送消息的格式和类型。

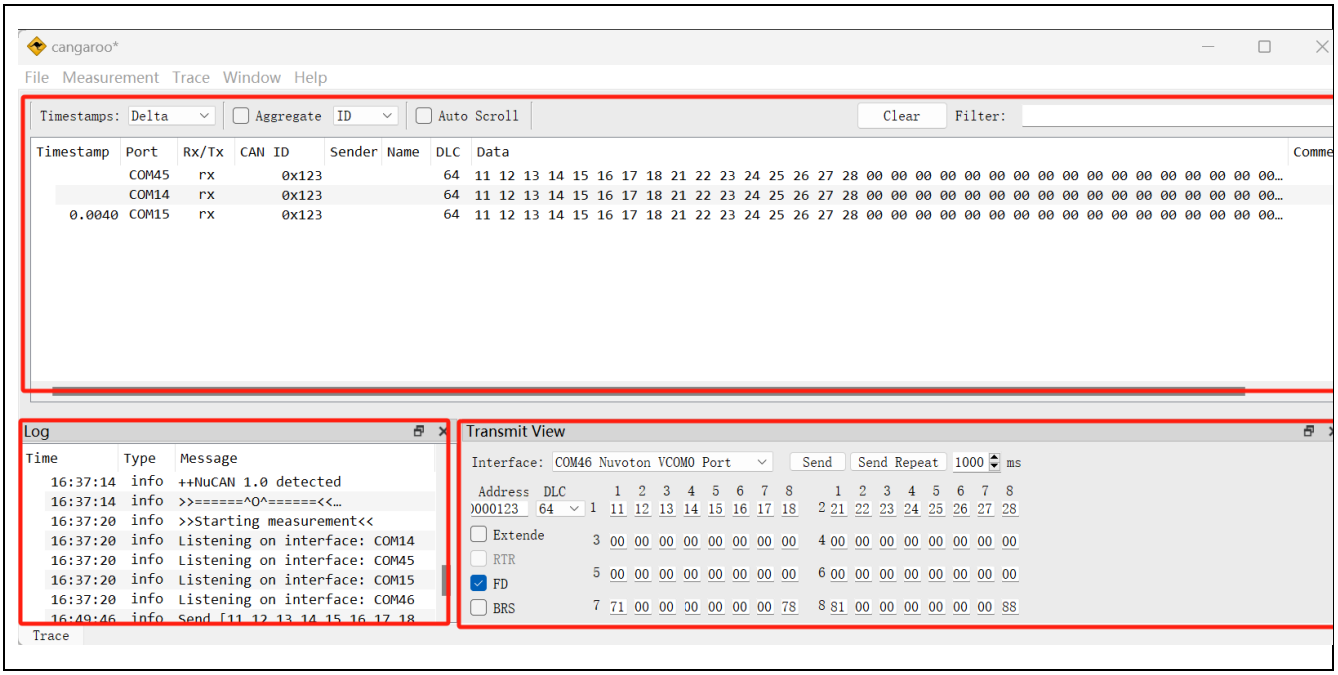

图 2–3 CAN 调试助手窗口

图 2–3 最上方的为监听窗口，左下角的小窗口为日志窗口，右下角的为消息发送窗口。

2.3 代码简介

USB 虚拟 VCOM，虚拟 4 路 VCOM 由 *descriptors.c* 描述符文件描述 USB 的设备的类型和节点信息。每个 VCOM 都有各自如下文的描述。

```
//-----------------VCOM0------------------//
#if (VCOM_CNT >= 1)
    // IAD
    LEN_IAD,        // bLength        : Interface Descriptor size
    DESC_IAD,       // bDescriptorType: IAD
    INTF_NUM_VCOM0, // bFirstInterface
    0x02,           // bInterfaceCount
    0x02,           // bFunctionClass     : CDC
    0x02,           // bFunctionSubClass
    0x01,           // bFunctionProtocol
    0x02,           // iFunction

    /* INTERFACE descriptor */
    LEN_INTERFACE,                          /* bLength             */
    DESC_INTERFACE,                         /* bDescriptorType     */
    INTF_NUM_VCOM0,                         /* bInterfaceNumber    */
    0x00,                                   /* bAlternateSetting   */
    0x01,                                   /* bNumEndpoints       */
    DESC_CDC_COMMUNICATION_INTERFACE_CLASS, /* bInterfaceClass     */
    DESC_CDC_ABSTRACT_CONTROL_MODEL,        /* bInterfaceSubClass  */
    0x01,                                   /* bInterfaceProtocol  */
    0x00,                                   /* iInterface          */
```

```
//LEN_VCOM_COMMAND_INTR
    /* Communication Class Specified INTERFACE descriptor */
    0x05,                           /* Size of the descriptor, in bytes */
    DESC_CDC_CS_INTERFACE,          /* CS_INTERFACE descriptor type */
    DESC_CDC_HEADER,                /* Header functional descriptor subtype */
    0x10, 0x01,  /* Communication device compliant to the communication spec. ver. 1.10 */

    /* Communication Class Specified INTERFACE descriptor */
    0x05,                           /* Size of the descriptor, in bytes */
    DESC_CDC_CS_INTERFACE,          /* CS_INTERFACE descriptor type */
    DESC_CDC_CALL_MANAGEMENT,       /* Call management functional descriptor */
    0x00,                      /* BIT0: Whether device handle call management itself. */
    /* BIT1: Whether device can send/receive call management information over a Data Class
Interface 0 */
    INTF_NUM_VCOM0 + 1,            /* Interface number of data class interface optionally
used for call management */

    /* Communication Class Specified INTERFACE descriptor */
    0x04,                             /* Size of the descriptor, in bytes */
    DESC_CDC_CS_INTERFACE,            /* CS_INTERFACE descriptor type */
    DESC_CDC_ABSTRACT_CONTROL_MODEL,  /* Abstract control management functional
descriptor subtype */
    0x00,                             /* bmCapabilities       */

    /* Union Functional descriptor */
    0x05,                             /* bLength              */
    DESC_CDC_CS_INTERFACE,            /* bDescriptorType: CS_INTERFACE descriptor type */
    DESC_CDC_UNION,                   /* bDescriptorSubType:Union Functional Descriptor*/
    INTF_NUM_VCOM0,                   /* bMasterInterface     */
    INTF_NUM_VCOM0 + 1,               /* bSlaveInterface0     */
//END  ILEN_VCOM_COMMAND_INTR

    /* ENDPOINT descriptor */
    LEN_ENDPOINT,                     /* bLength          */
    DESC_ENDPOINT,                    /* bDescriptorType  */
    (EP_INPUT | INT_IN_EP_NUM_0),     /* bEndpointAddress */
    EP_INT,                           /* bmAttributes     */
    GET_BYTE0(EPC_MAX_PKT_SIZE),
    GET_BYTE1(EPC_MAX_PKT_SIZE),      /* wMaxPacketSize   */
    0x01,                             /* bInterval:1 ms   */

    /* INTERFACE descriptor */
    LEN_INTERFACE,      /* bLength              */
    DESC_INTERFACE,     /* bDescriptorType      */
    INTF_NUM_VCOM0 + 1, /* bInterfaceNumber     */
    0x00,               /* bAlternateSetting    */
    0x02,               /* bNumEndpoints        */
    0x0A,               /* bInterfaceClass:CDC-Data */
    0x00,               /* bInterfaceSubClass   */
    0x00,               /* bInterfaceProtocol   */
    0x00,               /* iInterface           */

    /* ENDPOINT descriptor */
    LEN_ENDPOINT,                     /* bLength          */
    DESC_ENDPOINT,                    /* bDescriptorType  */
    (EP_INPUT | BULK_IN_EP_NUM_0),    /* bEndpointAddress */
```

```
    EP_BULK,                            /* bmAttributes     */
    GET_BYTE0(EPA_MAX_PKT_SIZE),
    GET_BYTE1(EPA_MAX_PKT_SIZE),        /* wMaxPacketSize   */
    0x00,                               /* bInterval        */

    /* ENDPOINT descriptor */
    LEN_ENDPOINT,                       /* bLength          */
    DESC_ENDPOINT,                      /* bDescriptorType  */
    (EP_OUTPUT | BULK_OUT_EP_NUM_0),    /* bEndpointAddress */
    EP_BULK,                            /* bmAttributes     */
    GET_BYTE0(EPB_MAX_PKT_SIZE),
    GET_BYTE1(EPB_MAX_PKT_SIZE),        /* wMaxPacketSize   */
    0x00,                               /* bInterval        */
#endif
}
```

vcom_serial.c 文件负责 USB 与 PC 握手和 USB 数据的收发。如下是 VCOM0 的发送中断 *EPA_Handle()* 和接收中断 *EPB_Handle()* 。发送中断提示 USB 缓存数据已发送完成，可进行下一包数据的发送，接收中断负责将 USB 缓存数据拷贝到 RAM 缓存。

```
//-----------------VCOM0------------------//
void EPA_Handle(void)
{   //data out mcu
    volatile uint32_t IrqSt = HSUSBD->EP[EPA].EPINTSTS & HSUSBD->EP[EPA].EPINTEN;
#if (VCOM_CNT>=1)
    volatile struct vcom_var_t *vtp=&vcom_dat[IDX_VCOM0];
    vtp->gu32TxSize = 0;
#endif
    HSUSBD_ENABLE_EP_INT(EPA, 0);
    HSUSBD_CLR_EP_INT_FLAG(EPA, IrqSt);
}

void EPB_Handle(void)
{   //data in mcu
    volatile uint32_t IrqSt = HSUSBD->EP[EPB].EPINTSTS & HSUSBD->EP[EPB].EPINTEN;
#if (VCOM_CNT>=1)
    volatile struct vcom_var_t *vtp=&vcom_dat[IDX_VCOM0];
    int volatile i;
    vtp->gu32RxSize = HSUSBD->EP[EPB].EPDATCNT & 0xffff;

    //pc_vcom -> usb -> mcu
    for(i = 0; i < vtp->gu32RxSize; i++)
        com_putchar(HSUSBD->EP[EPB].EPDAT_BYTE,IDX_VCOM0);

    /* Set a flag to indicate bulk out ready */
    vtp->gi8BulkOutReady = 1;
#endif
    HSUSBD_CLR_EP_INT_FLAG(EPB, IrqSt);
}
```

VCOMapp.c 文件包含 USB 接口的初始化，RAM 缓存数据收发处理函数。数据的收发主要用到类似 *putchar()* 和 *getchar()* 函数的方式，将数据丢入暂存区和从暂存区取数据。文件包含从

USB 缓存拷贝数据到 RAM 缓存，RAM 缓存拷贝到 USB 缓存，SLCAN 缓存拷贝到 RAM 缓存，RAM 缓存拷贝到 SLCAN 缓存，4 个缓存操作函数。如下是本例程 COM 口的 *com_putchar()* 和 *com_getchar()* 函数。

```
/**
 * @brief       pc com send to usb recv in mcu
 * @param[in]   bInChar    : the data will be send
                vcom       : com index
 * @return      1 byte send data,if error return -1
 * @details      pc_vcom -> usb -> mcu
 */
int32_t com_putchar(char bInChar,uint8_t vcom)
{
    volatile struct vcom_var_t *vtp;
    vtp=&vcom_dat[vcom];
    if(vtp->comTbytes < TXBUFSIZE)
    {
        /* Enqueue the character */
        vtp->comTbuf[vtp->comTtail++] = bInChar;
        if(vtp->comTtail >= TXBUFSIZE)
            vtp->comTtail = 0;
        vtp->comTbytes++;
    }
    else
    {
        return -1;
    }
    return bInChar;
}

/**
 * @brief       get 1 byte data from mcu send buffer to vcom
 * @param[in]    vcom :com index
 * @return      1 byte data,if error return -1
 * @details      mcu -> usb -> pc_vcom
 */
int32_t  com_getchar(uint8_t vcom)
{
    uint8_t bInChar;
    volatile struct vcom_var_t *vtp;
    vtp=&vcom_dat[vcom];
    if (vtp->comRbytes)
    {
        bInChar = vtp->comRbuf[vtp->comRhead++];
        if(vtp->comRhead >= RXBUFSIZE)
            vtp->comRhead = 0;
        vtp->comRbytes--;
    }
    else return -1;

    return bInChar;
}
```

协议转换主要由 *slcan.c* 文件将 CAN 接收到的数据转换成 SLCAN 协议，发送给 VCOM，并且将 VCOM 发送过来的 SLCAN 数据转换成 CAN 消息发送到总线上。*slcan.c* 文件同时负责解析 SLCAN 的命令。

```
/**
 * @brief       both can and process
 * @param[in]   slcan_port : one can port will to be operated
 * @return
 * @details
 */
void slcan_process_task(struct slcan_t* slcan_port)
{
    slcan_process_can(slcan_port);      //data from CAN FD to VCOM
    slcan_process_uart(slcan_port);     //data from VCOM  to CAN FD
}
```

2.4 运行实例

本例使用相同的波特率设置，测试 1 路 CAN 发送，另外 3 路 CAN 接收。COM46 负责发送数据，COM45，COM14，COM15 三个口负责接收数据。

运行前需要先确认 VCOM 设备的驱动名称已按照新唐 VCOM 名称修改好，可参考例程 *EC_M460_HSUSBD_VCOM_MultiPort_V1.00* 说明文件。

打开上位机设置 CAN 波特率的运行日志：

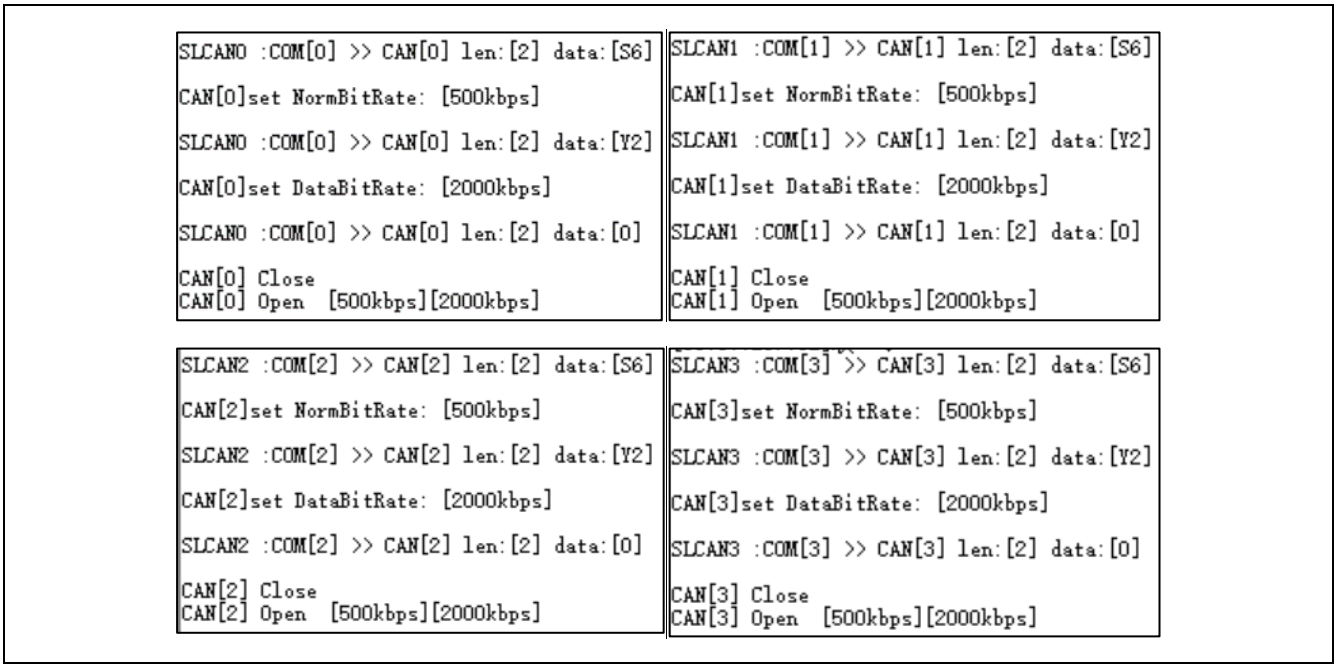

图 2–4 4 个 CAN 口波特率设置日志

4 路 CAN 仲裁域的通讯波特率为 500 kbps，变速帧数据段的波特率为 2 Mbps。

上位机选择 COM46 Nuvoton VCOM0 Port 发送一帧数据：

CAN FD 标准帧，64 字节长度，非变波特率帧，数据收发如图 2–5:

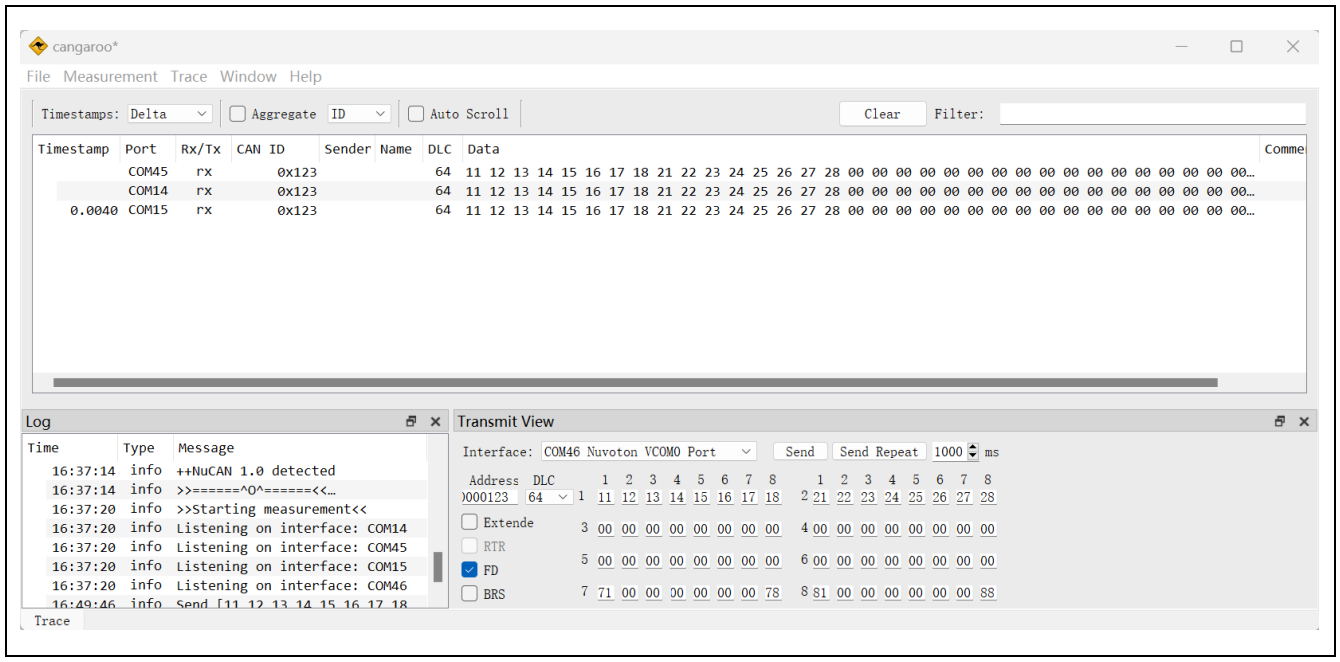

图 2–5 上位机数据收发图

由上图可以看到 COM45，COM14，COM15 3 个 Port 各收到一帧数据，数据内容和 COM46 发送的数据一样。

数据收发的日志如下图：

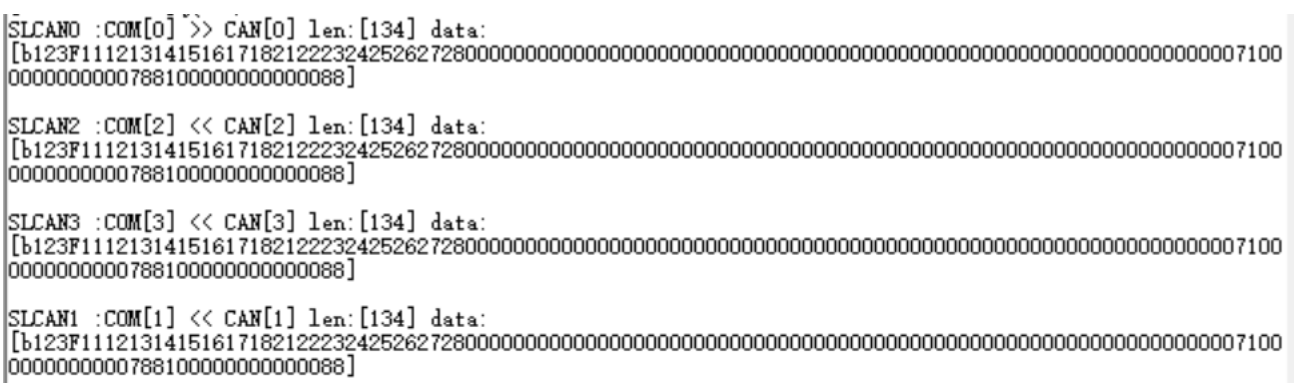

图 2–6 数据收发运行日志

- COM[0] >> CAN[0]表示数据从 VCOM0 发送到 CAN0 总线。
- COM[2] << CAN[2]表示 CAN2 从总线上接收到数据并发送到 VCOM2 端口。
- COM[3] << CAN[3]表示 CAN3 从总线上接收到数据并发送到 VCOM3 端口。
- COM[1] << CAN[1]表示 CAN1 从总线上接收到数据并发送到 VCOM1 端口。

协议解析：（更多命令参考图 1–1）

len:[134]表示 SLCAN 协议数据长度为 134 Bytes。

134 Bytes SLCAN 协议内容包含：

命令码： b -> CAN FD 标准帧数据（参考图 1–1）

帧 ID： 123 -> 0x123

帧长度： F -> 64 bytes（参考图 1–4）

帧数据(64 Bytes)：

0x 11 12 13 14 15 16 17 18
21 22 23 24 25 26 27 28
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
71 00 00 00 00 00 00 78
81 00 00 00 00 00 00 88

日志解析的数据和收发的数据一致，测试完成。

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M460_Series_BSP_CMSIS_V3.00.002
- IDE 版本
  - Keil uVersion 5.30
- APP 版本
  - Cangaroo V0.3.0

3.2 硬件需求

- 电路组件
  - 3-In-1 Analyzer Tool
- 线路示意图
  - 将 UART0 TX pin 脚连接到 PC UART RX，以显示范例代码的执行结果。
  - 将 HSUSB 连接到 PC，并打开 Cangaroo 上位机，以显示收发的 CAN 消息。

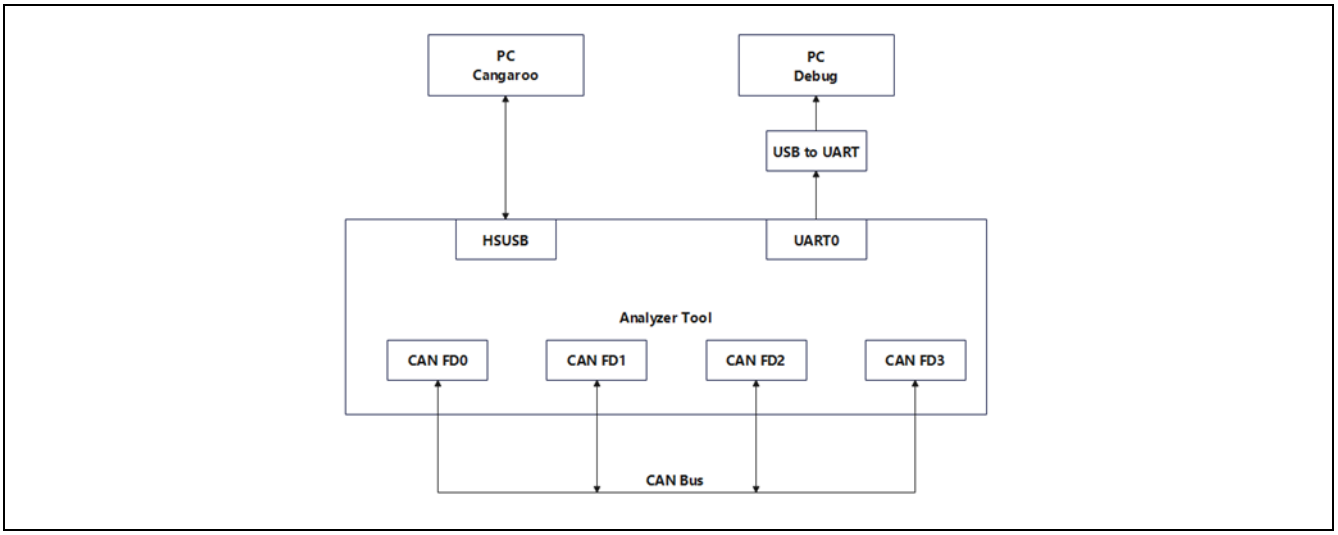

图 3–1 线路示意图

USB-C
UART0
UART2
UART3
UART4
VCOM0 <-> SLCAN0
VCOM1 <-> SLCAN1
VCOM2 <-> SLCAN2
VCOM3 <-> SLCAN3
5V Isolation Power
3-In-1
Analyzer Tool
CAN / UART / RS485
SD Card
K1
K2
Board Status
Reset
RS485 Status
CAN0 Status
CAN1 Status
CAN2 Status
CAN3 Status
CAN3
RS485
CAN0
CAN1
CAN2

图 3–2 实物接线

4. 目录信息

EC_M460_4_CAN_FD_to_SLCAN_by_VCOM_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	
Project	Source file of example code

图 4–1 目录信息

5. 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 *4_CAN_FD_to_SLCAN_by_VCOM.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
2024.06.18	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.