M460 的 4 路 CAN FD 訊息通過 VCOM 轉 SLCAN 協議

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式將 M460 的 4 路 CAN FD 訊息通過 VCOM 介面轉成 SLCAN 協議發送到 PC 上。
BSP 版本	M460_Series_BSP_CMSIS_V3.00.002
開發平臺	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 的訊息以 2 進制的方式存 MCU 內，不方便觀測操作 CAN FD 匯流排上的資料。本範例通過虛擬串口 VCOM 將 CAN FD 訊息轉換成 ASCII 碼形式的 SLCAN 協議用以與電腦上的調試助手進行通訊。

1.1 概述介紹以下 4 個基本內容。

- CAN FD 簡介
- VCOM 簡介
- SLCAN 協議介紹
- CAN 調試助手

1.2 CAN FD 簡介

2012 年，博世推出 CAN FD（CAN with Flexible Data-rate，具有靈活資料速率的 CAN），以克服傳統 CAN 的位元速率限制為 1 Mbps，並將每個 CAN 幀的資料位元組數從最多 8 個擴展到 64 個，從而縮小經典 CAN 與其他協議之間的差距。這是通過修改 CAN 框架格式來實現的，其中位元速率可以在 CAN 幀內切換到更快的值，並通過新的資料長度編碼來實現。CAN FD 標準化為 ISO11898-1:2015。CAN FD 協議控制器相容經典的 CAN 通信。

CAN FD 對比經典的 CAN 的主要特點：

- 更高的傳輸速率：資料最高速率可達到 8 Mbps。
- 更多的資料數量：最高可達到 64 Bytes。

1.3 VCOM 簡介

虛擬串口 Virtual Communication Port 簡稱 VCOM，使用 USB 介面替換之前電腦的硬體串列通訊口。本例中使用高性能微控制器 M467 的 1 個高速 USB 介面虛擬最多 4 路 VCOM，可同時收發 4 路不同速率的 CAN FD 訊息。

1.4 SLCAN 協議介紹

Serial Line CAN(簡稱 SLCAN)是一種基於文本 ASCII 碼的串列通訊口協議，也是 LAWICEL 硬體設備的協議。本文中使用的 SLCAN 通訊協議相容部分 CAN232 設備的協議，CAN232 最新版本為 2023 年的 Version 3.0B。

該協議可以通過如下網站進行下載 LAWICEL CAN232 | canusb.com。

SLCAN 協議格式：命令+參數+[CR]。

本範例已支援的 SLCAN 命令如下圖：

Command	Code	Param1	Param2	Param3	Param4
Open Port	O	[CR]			
Close Port	C	[CR]			
Set the Baud Rate of the Arbitration Domain	S	1~8	[CR]		
Set the Baud Rate of the Data Domain	Y	2,5	[CR]		
CAN Standard Remote Frame	r	11bit Frame ID	Data Length	[CR]	
CAN Standard Frame	t	11bit Frame ID	Data Length	Data	[CR]
CAN FD Standard Frame	d	11bit Frame ID	Data Length	Data	[CR]
CAN FD BRS Standard Frame	b	11bit Frame ID	Data Length	Data	[CR]
CAN Extended Remote Frame	R	29bit Frame ID	Data Length	[CR]	[CR]
CAN Extended Format	T	29bit Frame ID	Data Length	Data	[CR]
CAN FD Extended Frame	D	29bit Frame ID	Data Length	Data	[CR]
CAN FD BRS Extended Frame	B	29bit Frame ID	Data Length	Data	[CR]
Software Version	V	[CR]			
Hardware Version	v	[CR]			
Serial Number	N	[CR]			

圖 1–1 已支援的 SLCAN 協議命令

仲裁域支援的串列傳輸速率參數：

Code：S	Param1 & Baud Rate of the Arbitration								
Param1	0	1	2	3	4	5	6	7	8
Baud Rate of the Arbitration	10kbps	20kbps	50kbps	100kbps	125kbps	250kbps	500kbps	800kbps	1Mbps

圖 1–2 仲裁域參數串列傳輸速率對照

資料域支援的串列傳輸速率參數：

Code：Y	Param1 & Baud Rate of the Data	
Param1	2	5
Baud Rate of the Data	2Mbps	5Mbps

圖 1–3 資料參數串列傳輸速率對照

資料長度的索引碼：

Data Length	Param2 & Data Length						
Param2	9	A	B	C	D	E	F
Data Length	12Bytes	16Bytes	20Bytes	24Bytes	32Bytes	48Bytes	64Bytes

圖 1–4 資料長度參數和長度對照

1.5 CAN 調試助手

CAN 調試助手可以使用 2 種：串口調試助手和 Cangaroo 調試助手。

- 串口調試助手：

  通過標準串口收發資料，接收 SLCAN 協議資料需要自行進行解析；發送 CAN FD 訊息則需要自行進行 SLCAN 協議編碼。本範例使用串口調試助手以監測韌體的運行日誌。

- Cangaroo 調試助手：

  Cangaroo，一個開源的 CAN 匯流排資料分析工具，由 Hubert Denkmair 開發並維護。這個小巧且功能強大的軟體旨在協助電子工程師愛好者解析和理解車載網路通信資料。Cangaroo 基於 Qt 框架構建，提供一直觀的圖形化使用者介面，並支援多種作業系統。

  本範例的使用的 Cangaroo 針對新唐硬體平臺進行最佳化，軟體內含解析功能，可以方便直接發送需要的 CAN FD 訊息，並且可以直觀地觀測到 CAN 匯流排上接收到的資料。1 次可以打開 4 個調試口，發送的時候可以快速選擇需要發送的資料埠，並可同時監測 4 個資料埠接收到的訊息。

2. 程式介紹

本節內容先介紹程式工作流程，以方便理解運行機制，進而更好地對程式進行介紹。最後再分析一個程式的運行實例。

- 流程圖：資料收發流程，CAN 調試操作流程。
- 程式簡介：USB 虛擬 VCOM，資料收發，協議轉換。
- 運行實例：1 發 3 收，監測運行日誌，分析對比收資料。

2.2 流程圖

資料收發流程：

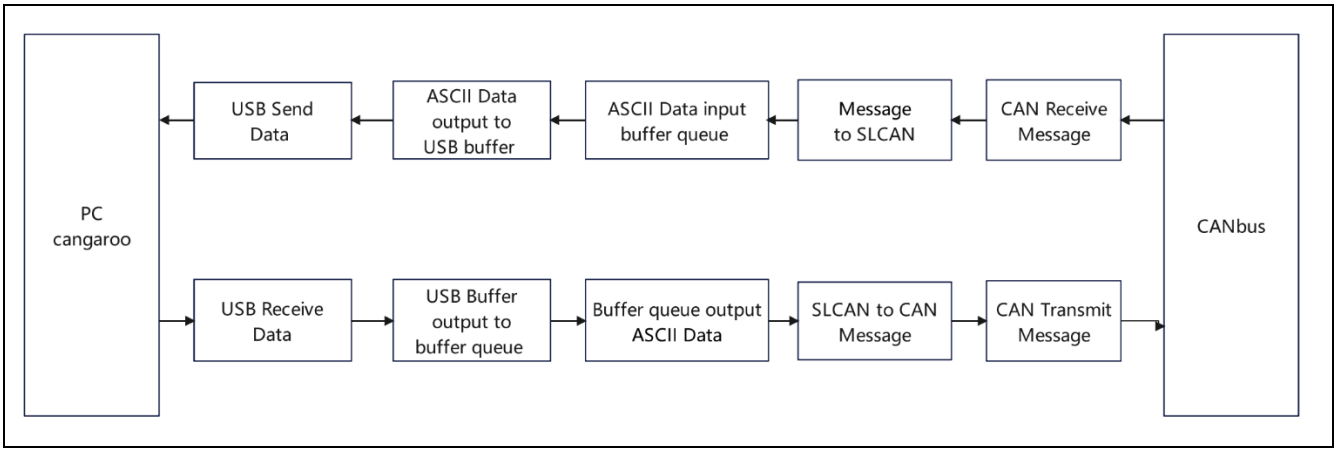

圖 2–1 資料收發流程圖

資料收發主要涉及到 4 個快取區域：

- USB 快取：存放 USB 即時收發的資料。
- RAM 快取：用於雙向資料的傳遞，提高資料回應的即時性。
- SCLAN 快取：用於轉換 CAN 訊息的存儲區。
- CAN 快取：存放 CAN 收發的即時訊息資料。

CAN 調試操作流程：

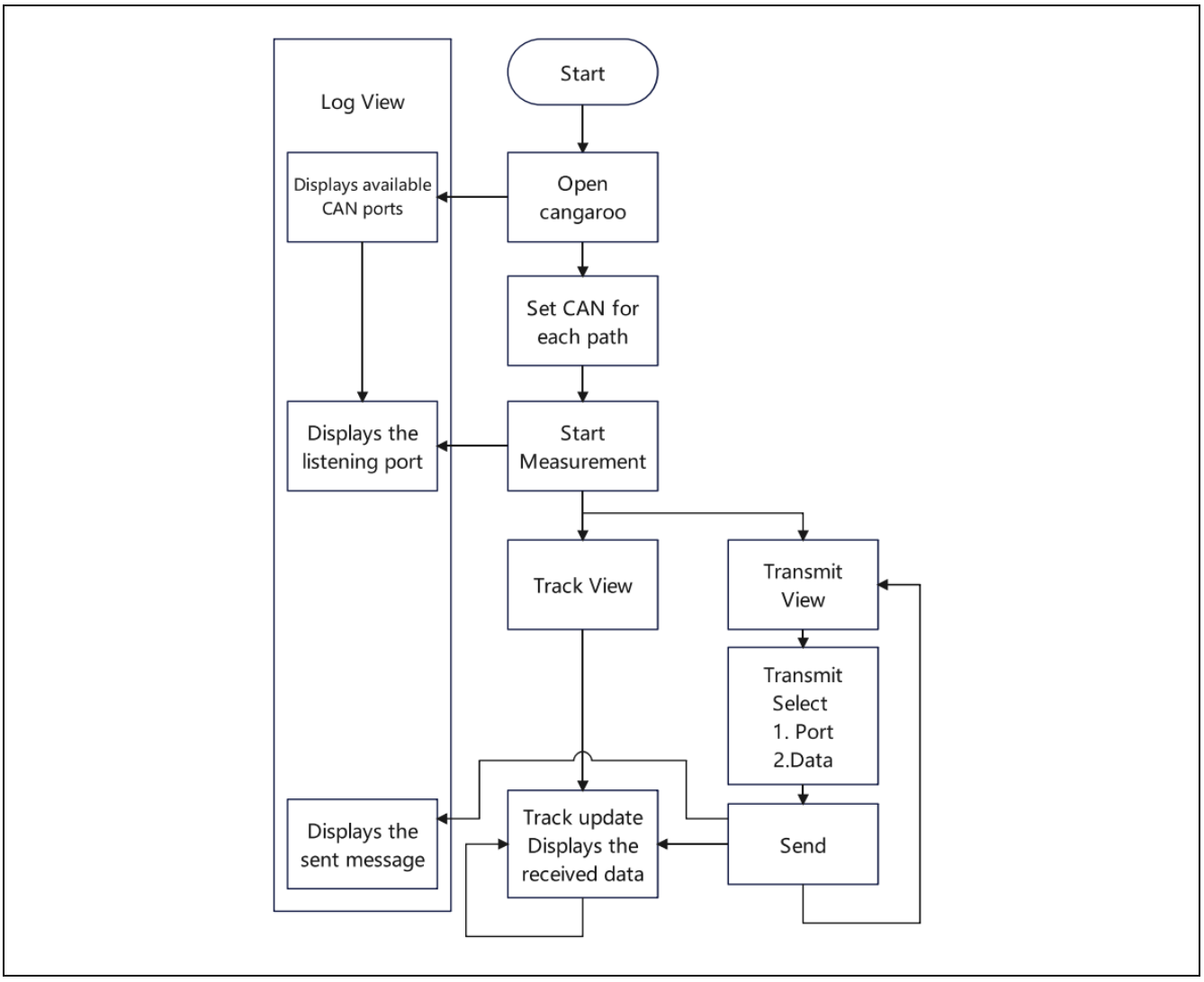

圖 2–2 APP 調試操作流程圖

CAN 調試助手有 3 個窗口（圖 2–3）：

- 日誌視窗：顯示當先的設備狀態，已發送的 CAN 訊息。
- 監聽窗口：顯示當前 CAN 接收到的訊息，可根據時間戳記調整顯示方式。
- 發送窗口：選擇需要發送的埠、發送訊息的格式和類型。

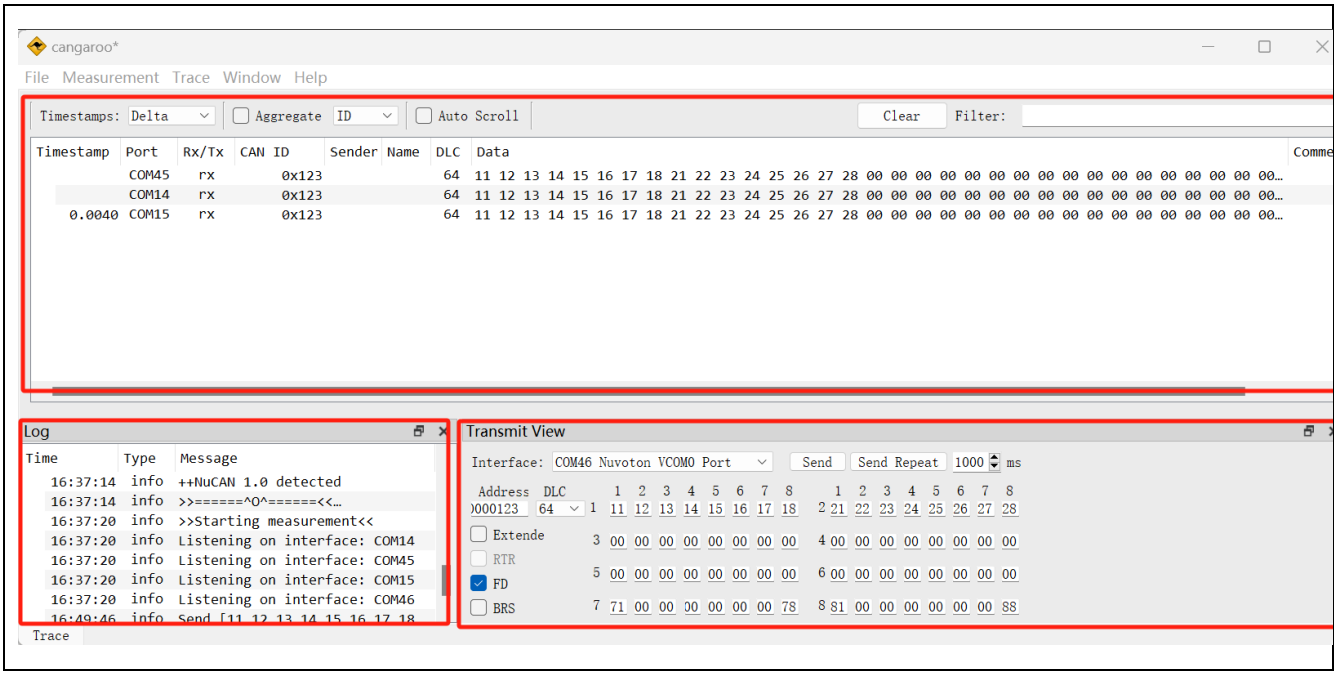

圖 2–3 CAN 調試助手窗口

圖 2–3 最上方的為監聽視窗，左下角的小視窗為日誌視窗，右下角的為訊息發送視窗。

2.3 程式簡介

USB 虛擬 VCOM，虛擬 4 路 VCOM 由 *descriptors.c* 描述符文件描述 USB 的設備的類型和節點資訊。每一個 VCOM 都有各自的描述，如下文。

```
//------------------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 文件負責 USB 與電腦交握和 USB 資料的收發。如下是 VCOM0 的發送中斷 *EPA_Handle()* 和接收中斷 *EPB_Handle()*。發送中斷提示 USB 快取資料已發送完成，可進行下一筆封包資料的發送。接收中斷負責將 USB 快取資料複製到 RAM 快取。

```
//-----------------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 介面的初始化和 RAM 快取資料收發處理函數。資料的收發主要用到類似 *putchar()* 和 *getchar()* 函數的方式，將資料丟入暫存區和從暫存區取數據。此文件包含從 USB 快取複製資料到 RAM 快取、RAM 快取複製到 USB 快取、SLCAN 快取複製到 RAM 快取、RAM 快取複製到 SLCAN 快取，4 個快取操作函數。如下是本範例 COM 埠的 *com_putchar()* 和 *com_getchar()* 函數。

```
/**
 * @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;
}
```

協議轉換主要由 *slcan.c* 文件將 CAN 接收到的資料轉換成 SLCAN 協議後發送給 VCOM，並且將 VCOM 發送過來的 SLCAN 資料轉換成 CAN 訊息發送到匯流排上。*slcan.c* 同時負責解析 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 運行實例

實例使用相同的串列傳輸速率設置，測試 1 路 CAN 發送，另外 3 路 CAN 接收。COM46 負責發送資料，COM45，COM14，COM15 三個口負責接收資料。

運行前需要先確認 VCOM 設備的驅動名稱已按照新唐 VCOM 名稱修改好，可參考範例 *EC_M460_HSUSBD_VCOM_MultiPort_V1.00* 說明文件。

打開上位機設置 CAN 串列傳輸速率的運行日誌：

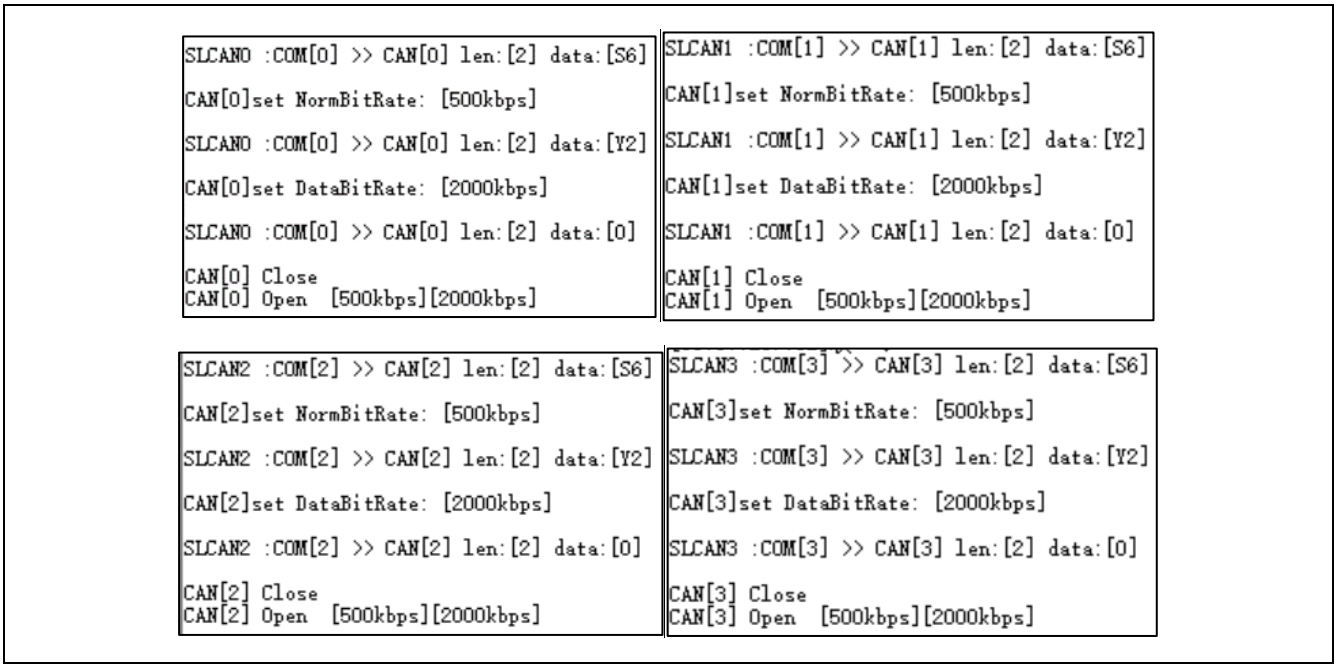

圖 2–4 CAN 埠串列傳輸速率設置日誌

4 路 CAN 仲裁域的通訊串列傳輸速率為 500 kbps，變速幀資料段的串列傳輸速率為 2 Mbps。

上位機選擇 COM46 Nuvoton VCOM0 Port 發送一幀資料：

CAN FD 標準幀，64 位元組長度，非變速率幀，資料收發如圖 2–5:

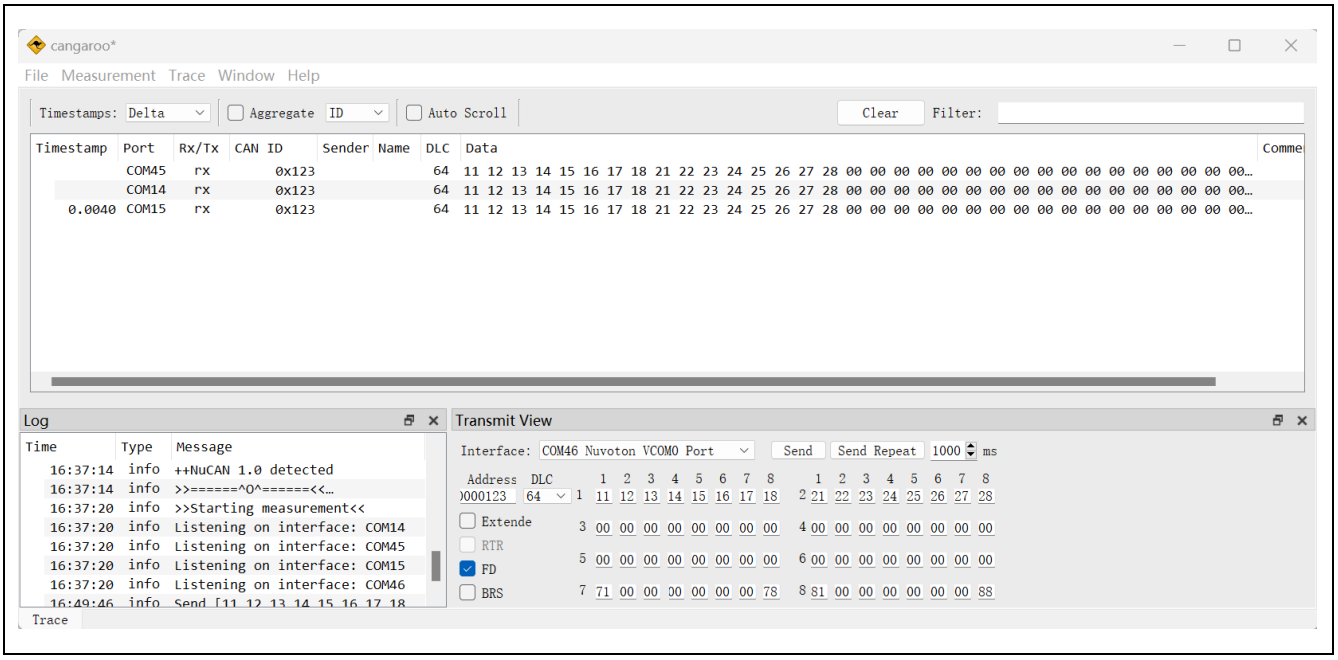

圖 2–5 上位機資料收發圖

由上圖可以看到 COM45，COM14，COM15 3 個 Port 各收到一幀資料，資料內容和 COM46 發送的資料一樣。

資料收發的日誌如下圖：

```
SLCAN0 :COM[0] >> CAN[0] len:[134] data:
[b123F111213141516171821222324252627280000000000000000000000000000000000000000000000000000000000000000000000000000007100
000000000078810000000000088]

SLCAN2 :COM[2] << CAN[2] len:[134] data:
[b123F111213141516171821222324252627280000000000000000000000000000000000000000000000000000000000000000000000000000007100
000000000078810000000000088]

SLCAN3 :COM[3] << CAN[3] len:[134] data:
[b123F111213141516171821222324252627280000000000000000000000000000000000000000000000000000000000000000000000000000007100
000000000078810000000000088]

SLCAN1 :COM[1] << CAN[1] len:[134] data:
[b123F111213141516171821222324252627280000000000000000000000000000000000000000000000000000000000000000000000000000007100
000000000078810000000000088]
```

圖 2–6 資料收發運行日誌

- COM[0] >> CAN[0]表示資料從 VCOM0 發送到 CAN0 匯流排。
- COM[2] << CAN[2]表示 CAN2 從匯流排上接收到資料並發送到 VCOM2 埠。
- COM[3] << CAN[3]表示 CAN3 從匯流排上接收到資料並發送到 VCOM3 埠。
- COM[1] << CAN[1]表示 CAN1 從匯流排上接收到資料並發送到 VCOM1 埠。

協議解析：（更多命令參考圖 1–1）

len:[134]表示 SLCAN 協議資料長度為 134 Bytes。

134 Bytes SLCAN 協議內容包含：

命令碼： b -> CAN FD 標準幀資料（參考圖 1–1）

幀 ID： 123 -> 0x123

資料長度： F -> 64 bytes（參考圖 1–4）

幀數據(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

日誌解析的資料和收發的資料一致，測試完成。

3. 軟體與硬體需求

3.1 軟體需求

- 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 硬體需求

- 電路元件
  - 3-In-1 Analyzer Tool
- 線路示意圖
  - 將 UART0 TX pin 腳連接到 PC UART RX，以顯示範例程式的執行結果。
  - 將 HSUSB 連接到 PC，並打開 Cangaroo 上位機，以顯示收發的 CAN 訊息。

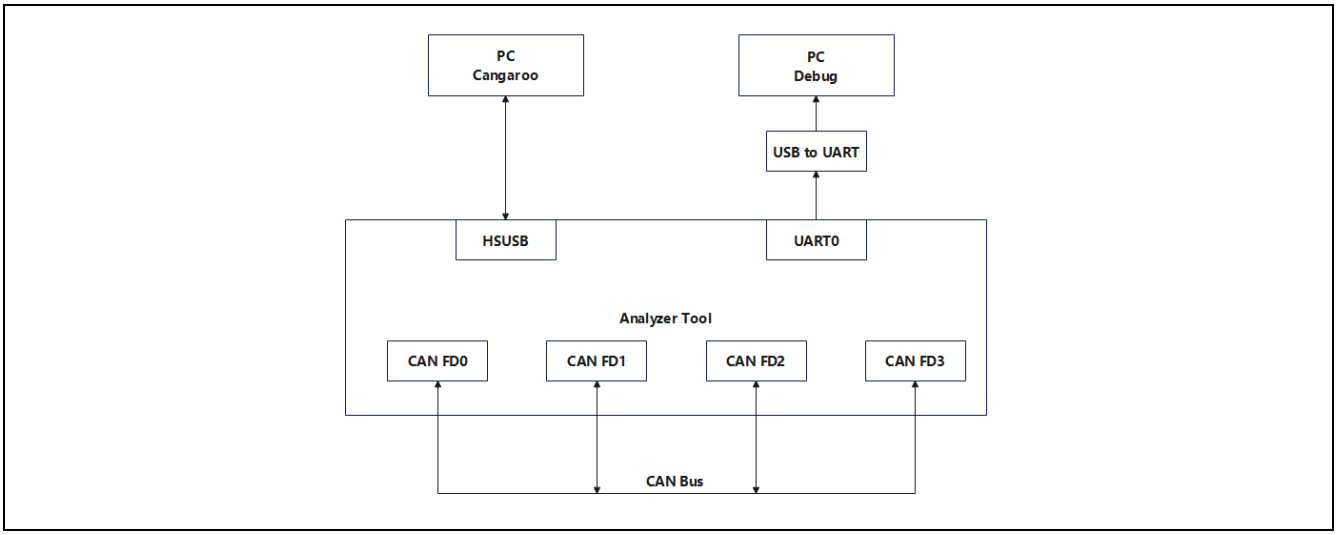

圖 3–1 線路示意圖

USB-C
UART0
UART2
UART3
UART4
CAN3
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
RS485
CAN0
CAN1
CAN2

圖 3–2 實物接線

4. 目錄資訊

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

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 *4_CAN_FD_to_SLCAN_by_VCOM .uvprojx*。
2. 進入編譯模式介面
   - 編譯
   - 下載程式至記憶體
   - 進入 / 離開除錯模式
3. 進入模擬模式介面
   - 執行程式

6. 修訂紀錄

Date	Revision	Description
2024.06.18	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.