BUSMUST

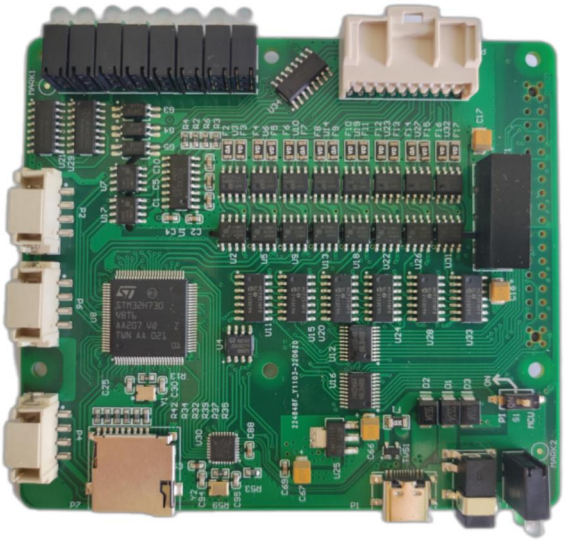
产 品 说 明 书

Product User Guide

产品名称：CANFD 接口卡

产品型号：CANFD-X8PI

发布日期：2022.07.08



一、产品特点

★ 全面支持 CAN Classic 2.0以及CANFD 协议，最高支持 5000kbps 数据波特率;
★ 支持双路RS232端口，支持 115200kbps 数据波特率，可用作协议网关;
★ 支持树莓派4B的标准40Pin控制接口，可通过树莓派控制6路CANFD；
★ 调试接口引出，公开引脚定义，可作为开发板，通过自定义固件实现高级功能;
★ 单设备八通道同时收发，支持多设备，最多可通过Busmaster同时连接 16 个通道，方便批量测试&控制;
★ USB2.0 HS 接口高速通信，吞吐量高，同时兼容 FS 接口，适应性强;
★ 创新的三级帧缓冲设计，500kbps 等经典场景下 24 小时总线满载测试不丢帧;
★ 业内领先的上位机功能，支持 DBC 加载/录制回放/信号解析/曲线绘制/节点仿真/UDS/自动测试等企业级功能;
★ 支持加载 UDS 27 安全算法 DLL 自动解锁 ECU（兼容 CANoe 安全 DLL 接口）；
★ 支持二次开发，支持 Python-can/Linux/树莓派，丰富实用的示例程序助您快速自主定制企业级上层应用;
★ 每个通道 2 个硬件级实时接收过滤器，独家支持报文负载内容过滤;
★ 每个通道 1 个硬件级定时发送任务，自增/随机等模式随心配置;
★ 每个通道内置 120Ω终端电阻，帮您免除复杂的总线连接，降低故障率;
★ BusMaster支持波特率和终端电阻自动侦测功能，帮您免除复杂的软件配置，提升工作效率;
★ 电脑 USB 接口过流保护，强力保障您的电脑安全;
★ 总线 ESD 防护，减少信号干扰导致的总线故障;
★ 工业级电气隔离，总线前端与数字后端独立供电;

二、应用范围

- 汽车行业
- 航空航天、航海
- 过程工业
- 纺织机械
- 机器人
- 自动控制
- 医疗器械
- 机械工业
- 农用机械
- 数控机床

三、参数表

参数名称	规格	备注
PCB尺寸	87mm * 99mm	
PCBA高度	17mm	
PCBA重量	<100g	
外壳尺寸	无外壳	
供电方式	USB Type-C 5V 2A	
USB协议	USB2.0 480Mbps	
指示灯	18个	每通道2个LED，各指示CAN收发状态和终端电阻状态；另有PWR和IO指示
信号端子	22P Molex 5018762240	每路CANFD引出CANH和CANL，另有双路RS232
UART端子	2个 2.54mm*4 单排通用端子	3.3V电平规范
SWD端子	1个 2.54mm*4 单排通用端子	3.3V电平规范
协议兼容性	CAN2.0 CANFD(ISO/Bosch) RS232	
CANFD波特率	5kbps - 5Mbps	
CANFD接收帧率	>16000fps [1]	仅当总线波特率和负载足够高时可达到标称规格
CANFD发送帧率	>4000fps [2]	仅当总线波特率和负载足够高时可达到标称规格
终端电阻	120Ohm / Disabled	
前端电气隔离	支持	
终端电阻控制	程控	
外部DC输入	5V	
TF卡插槽	压弹式	
报文路由功能	无	可选：支持在上位机中创建路由表，根据报文ID与类型转发报文至指定端口
固件二次开发接口	无	可选：支持在KeilMDK中编程，进行自定义的报文解析/转发等
离线报文发送功能	无	可选：支持在上位机中创建不超过16个定时发送任务，可发送网络管理报文等
离线文件录制功能	无	可选：支持将CANFD端口收到的报文离线录制到板载TF卡
离线文件播放功能	无	可选：支持将板载TF卡中的报文文件播放至CANFD端口

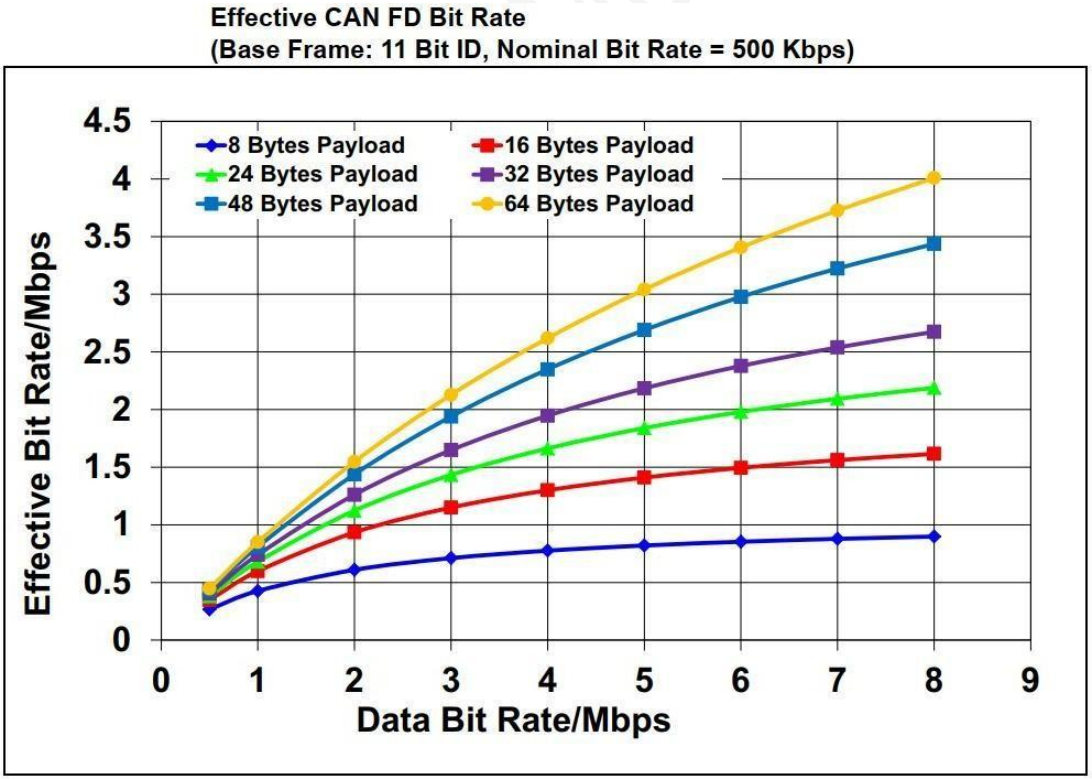
注：

[1][2] 通道1和通道2可独立实现标称性能，通道3-8由于奇数通道和偶数通道复用SPI接口，故当奇偶通道同时开启时，吞吐量会有不超过50%的下降。

四、典型性能

- 支持RS232总线（需二次开发）
- 支持 CAN-FD 比特率切换（BRS）
- 支持 CAN-FD 超大 64 字节报文负载
- 64 字节*2000 包/秒报文收发不丢帧
- 每一台分析仪出厂前均通过 CAN-FD 仲裁段 500kbps+数据段 2000kbps，经典车载场景测试

- 需要使用 2000kbps 以上超高波特率的客户请注意：
- 请使用总长不超过 1.5m 的优质低电容双绞线
- 双绞线两端各外接一个 120 欧终端电阻
- 双绞线请保持良好电磁屏蔽
- 请合理设置 CANFD 分析仪的采样点位置（推荐 75%）



4.1 吞吐量

本产品的单通道吞吐量指标如下：

波特率	1 字节	8 字节	64 字节
500kbps	> 8700 fps	**> 4000 fps** **(车载CAN 网络经典场景)**	N/A
1Mbps	> 17500 fps	> 8600 fps @ RX > 4000 fps @ TX	N/A
1Mbps+2Mbps	不建议	> 12600 fps @ RX > 4000 fps @ TX	> 2800 fps > 2000 fps @ TX
1Mbps+5Mbps	不建议	> 16000 fps @ RX > 4000 fps @ TX	> 5000 fps @ RX > 2000 fps @ TX

注：

- 以上测试数据源为进口 PeakCAN-USB-Pro（支持 CANFD）在 PcanView 中发送产生，测试工况接近满载
- 以上测试的帧率信息来自本产品提供的BusMaster 企业级上位机中的Network statistics 中的自动接收统计功能
- 在 64 字节长报文模式下，虽然帧率低于8字节CAN标准报文模式，但是其吞吐量更高
- 在总线帧率接近产品性能极限时，可能会有丢包现象，但此时仍可保证收发数据的准确性

4.2 CPU占用率

在嵌入式平台上，由于处理器性能相对较差，需要关注BMAPI在处理报文时的CPU占用率。

本章节以树莓派4B为参考平台，针对以下几种典型场景对CPU占用率进行实测评估，供参考及交叉对比。

测试条件：

CPU	树莓派4B
开发语言	C++
控制方式	BMAPI（USB）
接收方式	多通道同时侦听Notification，然后BM_ReadCanMessage
发送方式	未发送报文
极限CPU占用率评估	运行100s，使用htop观察占用率最高的核的**CPU%** （期间占用率一直波动，记录最高值）
平均CPU占用率评估	运行100s，使用htop观察占用率最高的核的**TIME+** （TIME+除以100s即为此间的平均占用率）

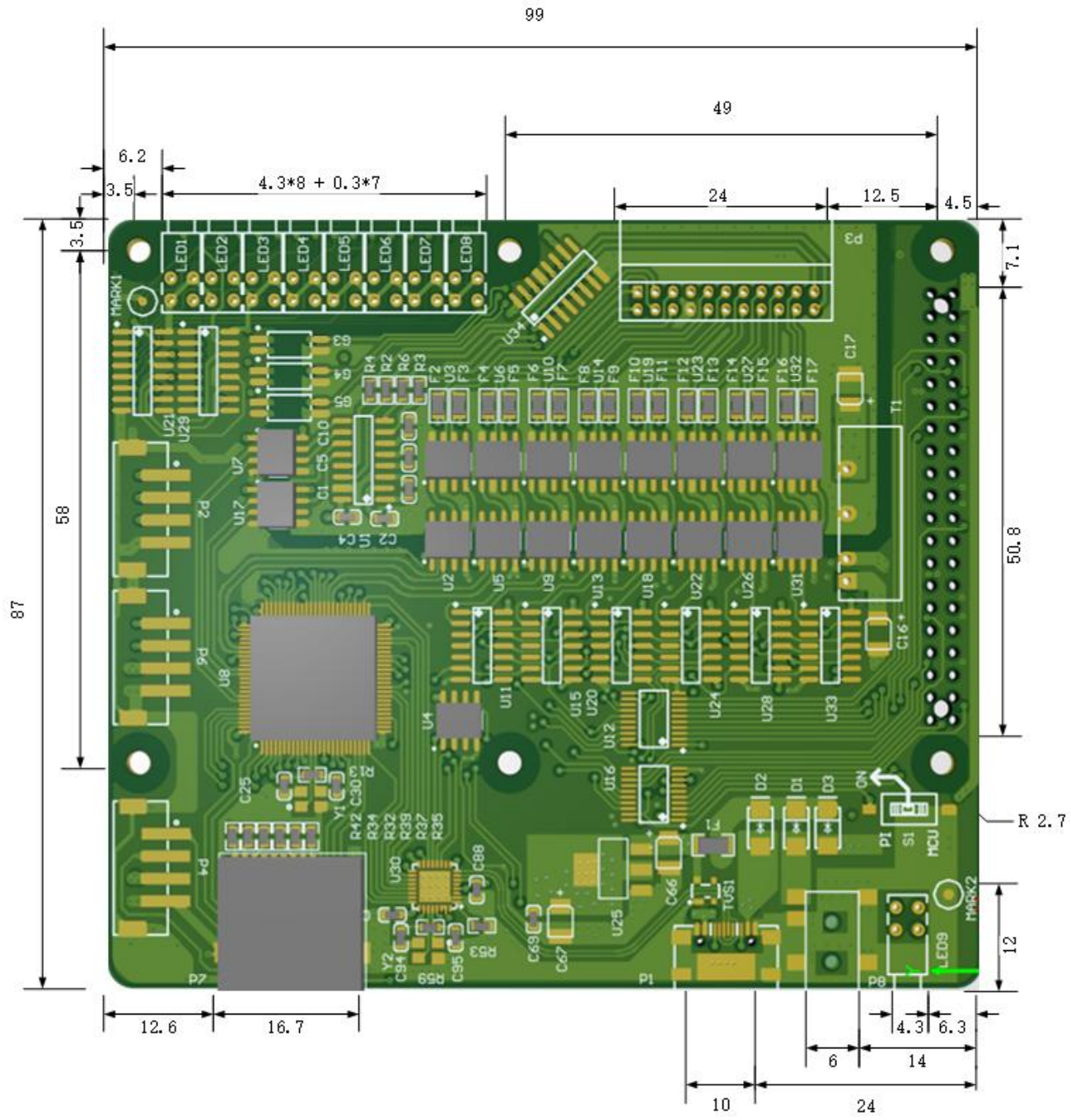
测试结果：

波特率	负载长度	单通道包速率	平均CPU占用率 （仅统计占用最高的单核）	极限CPU占用率 （仅统计占用最高的单核）
1Mbps + 5Mbps	8	10000	34%	41%
1Mbps + 5Mbps	8	5000	33%	40%
1Mbps + 5Mbps	8	2000	22%	26%
1Mbps + 5Mbps	8	1000	13%	16%
1Mbps + 5Mbps	48	4000	33%	39%
1Mbps + 5Mbps	48	3000	32%	37%
1Mbps + 5Mbps	48	2000	25%	32%
1Mbps + 5Mbps	48	1000	17%	19%

注：

- Shell连接状态下，命令行打印对CPU占用率的影响较大，本测试每5秒仅打印一次统计信息以降低对平均CPU占用率的影响，但在打印信息的瞬间仍然可以观测到明显的CPU占用率提升
- 以上测试i结果使用C++语言，O3优化，配合异步通知机制才能达到，在需要较高吞吐量性能的情况下，不推荐使用Python进行编程开发
- 以上测试使用BMAPI 1.8.0版本测试，该版本相较1.7版本而言，有约二倍的性能提升（仅针对二次开发包中的multichannel_rx_cpp例程）

五、外形尺寸图(单位：mm)



注：结构设计需要更多尺寸信息可参考3D图纸。

六、接插件定义

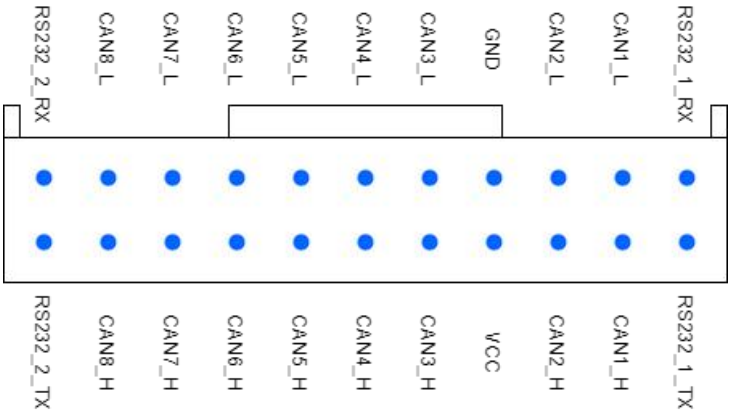


图 1 molex 5018762240 pin-out

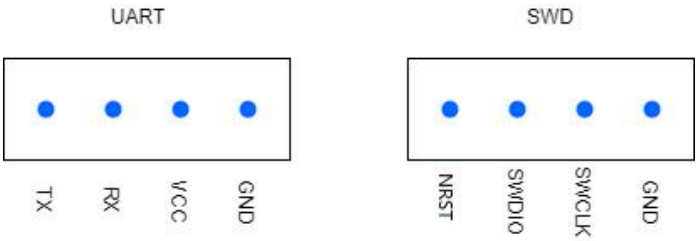


图 2 ttl-uart接口和SWD接口定义

5.1.1 GPIO Pin Assignments

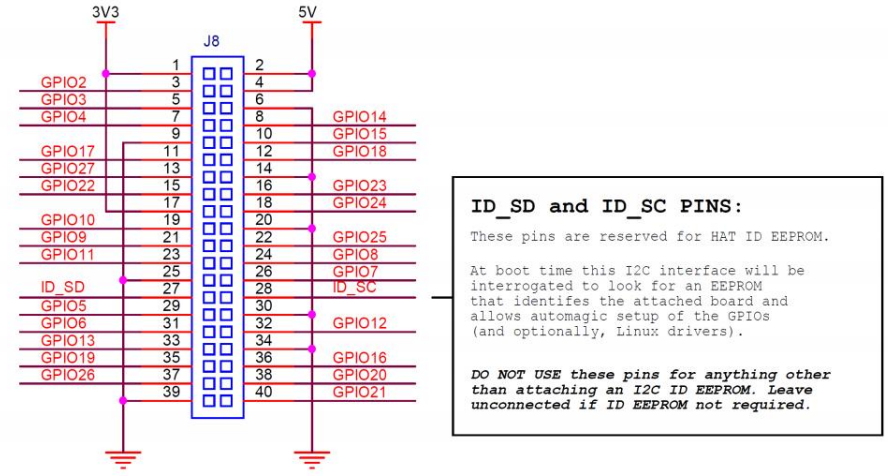


Figure 3: GPIO Connector Pinout

图 3 树莓派引脚与GPIO编号对应图（即图4中的BCM编码）

5.1.2 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

图 4 树莓派GPIO分配表（见红色方框标记）

七、引脚资源分配

以下内容仅用于二次开发，若您使用Busmaster或者BMAPI驱动本扩展卡，可跳过本章节。本扩展卡使用了一颗LQFP100封装的STM32H730VBT6芯片作为主控制器，内置了同时支持CANFD和RS232外设的固件，其中主控制器的引脚资源分配情况如下表。

表 1 引脚分配表

引脚	功能	说明
PE2	SPI4_SCK	
PE3	CAN6_INT1	可选GPIO输入信号
PE4	SPI4_NSS1	
PE5	SPI4_MISO	
PE6	SPI4_MOSI	
PC14	LED1	PWR LED：H = 熄灭；L=点亮
PC15	LED2	IO LED：H = 熄灭；L=点亮
晶振	PH0	24MHz
晶振	PH1	24MHz
PC0	USB_OTG_HS_ULPI_STP	
PC1		
PC2	USB_OTG_HS_ULPI_DIR	
PC3	USB_OTG_HS_ULPI_NXT	
PA0	UART4_TX	
PA1	UART4_RX	
PA2	PI_EN	树莓派检测：H = 已禁用；L=已启用
PA3	USB_OTG_HS_ULPI_D0	
PA4	SPI1_NSS1	
PA5	USB_OTG_HS_ULPI_CK	
PA6	SPI1_MISO	
PA7	SPI1_MOSI	
PC4	CAN4_INT0	可选GPIO输入信号
PC5	CAN3_INT0	可选GPIO输入信号
PB0	USB_OTG_HS_ULPI_D1	
PB1	USB_OTG_HS_ULPI_D2	
PE7	CAN3_INT1	可选GPIO输入信号
PE12	SPI1_4MHZ_CLK_1	此引脚输出4MHz时钟给CAN3&CAN4的MCP2518FD
PE13	SPI4_4MHZ_CLK_3	此引脚输出4MHz时钟给CAN7&CAN8的MCP2518FD
PE14	SPI2_4MHZ_CLK_2	此引脚输出4MHz时钟给CAN5&CAN6的MCP2518FD
PE15	CAN4_INT1	可选GPIO输入信号
PB10	USB_OTG_HS_ULPI_D3	
PB11	USB_OTG_HS_ULPI_D4	
PB12	USB_OTG_HS_ULPI_D5	
PB13	USB_OTG_HS_ULPI_D6	
PB14	SPI2_MISO	
PB15	SPI2_MOSI	

PD8	SDMMC1_CD	
PD9	CAN5_INT0	可选GPIO输入信号
PD10	SPI1_NSS2	
PD11	SPI2_NSS2	
PD12	FDCAN3_RX	
PD13	FDCAN3_TX	
PD14	TRES595_DATA	使用1颗74HC595来控制8个通道终端电阻，低电平使能
PD15	TRES595_LATCH	
PC6	TRES595_CLK	
PC7	LED595_DATA	使用2颗74HC595来控制16个通道状态指示LED，低电平点亮
PC8	SDMMC1_D0	
PC9	SDMMC1_D1	
PA8	UART7_RX	
PA9	SPI2_SCK	
PA10	LED595_LATCH	
PA11	SPI2_NSS1	
PA12	LED595_CLK	
PA13	SWDIO	
PA14	SWCLK	
PA15	UART7_TX	
PC10	SDMMC1_D2	
PC11	SDMMC1_D3	
PC12	SDMMC1_CK	
PD0	FDCAN1_RX	
PD1	FDCAN1_TX	
PD2	SDMMC1_CMD	
PD3	CAN5_INT1	可选GPIO输入信号
PD4	CAN6_INT0	可选GPIO输入信号
PD5	USART2_TX	
PD6	USART2_RX	
PD7	CAN7_INT0	可选GPIO输入信号
PB3	SPI1_SCK	
PB4	SPI4_NSS2	
PB5	USB_OTG_HS_ULPI_D7	
PB6	USART1_TX	
PB7	USART1_RX	
PB8	CAN8_INT1	可选GPIO输入信号
PB9	CAN8_INT0	可选GPIO输入信号
PE0	CAN7_INT1	可选GPIO输入信号
PE1	USB_RESET	USB PHY复位信号，高电平复位有效

八、片内外设资源分配

以下内容仅用于二次开发，若您使用Busmaster或者BMAPI驱动本扩展卡，可跳过本章节。本扩展卡使用了一颗LQFP100封装的STM32H730VBT6芯片作为主控制器，内置了同时支持CANFD和RS232外设的固件，其中主控制器的片内资源分配情况如下表。

表 2 片内外设分配

扩展板功能	片内外设
CAN1	FDCAN1
CAN2	FDCAN3
CAN3	SPI1
CAN4	SPI1
CAN5	SPI2
CAN6	SPI2
CAN7	SPI4
CAN8	SPI4
RS2321	USART1
RS2322	USART2
UART1	UART4
UART2	UART7
SDMMC	SDMMC1
USB	USB1
通道状态LED	GPIO+74HC595
通道终端电阻	GPIO+74HC595

九、PCB丝印

为了方便识别板载资源，以下列出本板卡上的核心资源对应的PCB丝印标记，供查阅。

表 3 PCB丝印名称

功能	丝印
CAN1-CAN8 & RS2321-RS2322端子	P3
树莓派开关	S1
USB端子	P1
树莓派端子	P5
UART1端子	P4
UART2端子	P6
SWD仿真器端子	P2
TF卡座	P7
5V 4A电源输入端子	P8
CAN1 TRES(黄) TX/RX(绿) LED	LED8
CAN2 TRES(黄) TX/RX(绿) LED	LED7

CAN3 TRES(黄) TX/RX(绿) LED	LED6
CAN4 TRES(黄) TX/RX(绿) LED	LED5
CAN5 TRES(黄) TX/RX(绿) LED	LED4
CAN6 TRES(黄) TX/RX(绿) LED	LED3
CAN7 TRES(黄) TX/RX(绿) LED	LED2
CAN8 TRES(黄) TX/RX(绿) LED	LED1
PWR(黄)/IO(绿) LED	LED9

十、通过BMAPI使用本转接板

本板卡支持BMAPI通用软件编程接口，用户可通过BMAPI 1.8.1.21或者更高的版本来控制：

- CAN1-CAN8的报文收发
- CAN1-CAN8的终端电阻
- CAN1-CAN8的波特率配置等

具体使用步骤如下：

1. 将板卡的树莓派开关S1拨至OFF（MCU一侧）
2. 使用USB Type-C数据线将板卡连接至电脑
3. 检查设备管理器是否正确识别本设备（如设备管理器未识别或者识别出黄色叹号，请安装最新驱动后重试）
4. 使用任意编程语言，调用BM_OpenCan()函数打开对应的端口
5. 使用任意编程语言，调用BM_SetBitrate()函数来设置总线波特率，采样点等参数
6. 使用任意编程语言，调用BM_SetTerminalResistor()函数来控制终端电阻开关
7. 使用任意编程语言，调用以下任一API实现报文收发：
   (1) BM_ReadCanMessage
   (2) BM_WriteCanMessage
   (3) BM_ReadMultipleCanMessage
   (4) BM_WriteMultipleCanMessage
   (5) BM_ReadIsotp
   (6) BM_WriteIsotp
8. 不再需要进行报文收发后，调用 BM_Close()函数关闭对应的端口，释放资源

以上仅为概要说明，具体API定义请参考BMAPI SDK的doc/bmapi.chm说明文档，或者参考example文件夹下的各个例程。

如果在软件操作的过程中板卡检测到异常（例如无法收发报文），且通过BM_Close + BM_OpenEx组合无法从异常状态中恢复，可尝试调用BM_ResetDevice来重启板卡（及USB连接），或者直接通过SWD端子上的NRST引脚来执行硬件复位。

十一、通过树莓派使用本转接板

本板卡支持通过树莓派的SPI接口来控制CAN3-CAN8对应的6颗MCP2518FD，包括：

- CAN3-CAN8的报文收发
- CAN3-CAN8的波特率配置等

具体使用步骤如下：

1. 将板卡的树莓派开关S1拨至ON（PI一侧）

2. 使用USB Type-C数据线将树莓派连接至5V 3A电源，树莓派将通过40P接口给本板卡供电，在特殊情况下（例如树莓派电源驱动能力不足），本板卡也可以再外界一个5V电源独立供电，两电源间互不干扰

3. 打开树莓派根文件系统的/boot/config.txt，追加以下文本，从而在树莓派上开启SPI4，SPI5，SPI6三个片内外设（树莓派原生操作系统默认不使能这三个外设）：

```
dtoverlay=spi0-1cs
dtoverlay=spi3-2cs
dtoverlay=spi4-2cs
dtoverlay=spi5-2cs
dtoverlay=spi6-2cs
```

4. 在树莓派上编写C语言程序，通过ioctl接口实现SPI总线数据读写：

```
    struct spi_ioc_transfer spi;

    memset(&spi, 0, sizeof(spi));
    spi.tx_buf = (unsigned long)SpiTxData;    //transmit from "data"
    spi.rx_buf = (unsigned long)SpiRxData;    //receive into "data"
    spi.len = spiTransferSize;
    spi.delay_usecs = 0;
    spi.speed_hz = spi_speed;
    spi.bits_per_word = spi_bitsPerWord;
    spi.cs_change = 0;                        //0=Set CS high after a transfer
    int retVal = ioctl(spi_cs_fd, SPI_IOC_MESSAGE(1), &spi);
```

5. 在树莓派上编写C语言程序，通过SPI总线数据读写来操作MCP2518FD，实现：

   (1) 写入控制寄存器配置波特率，采样点等

   (2) 读写报文RAM

   (3) 读取状态寄存器查询是否有RX报文

   (4) 写入控制寄存器通知有TX报文

6. 在树莓派上运行编写的C语言程序

以上仅为概要说明，具体MCP2518FD寄存器操作，请阅读器件手册等。

另外，本板卡附带了一个spitest例程，用于演示对MCP2518FD的RAM读写测试操作。