User Manual

CSM430B

隔离 SPI/UART 转 CANFD 芯片

UM01010101 V1.00 Date:2025/10/31

类别	内容
关键词	隔离SPI/UART转CANFD芯片
摘要	CSM430B产品用户手册

ZLG 致远电子

修订历史

版本	日期	原因
V1.00	2025/10/31	创建文档

目 录

1. 功能简介

1.1 概述

CSM430B 支持通过 SPI/UART 扩展出两路 CANFD 接口，它内部采用全隔离设计，隔离耐压 3500VDC，内置 DC-DC 隔离电源、信号隔离和 CANFD 接口电路隔离，当用户控制板上的 CANFD 控制器资源不够时，可以通过 SPI 或 UART 接口扩展出更多的 CANFD 总线接口。

该产品可以很方便地嵌入到具有 SPI 或 UART 接口的设备中，在不需改变原有硬件结构的前提下使设备获得 CANFD 通讯接口，实现 SPI 设备或 UART 设备和 CANFD 总线网络之间的数据通讯。

1.2 产品特性

- 元器件 100%国产化；
- 实现 SPI 或 UART 与 CANFD 的双向数据通信；
- 集成 1 路 SPI 接口，最高可达 6Mbps；
- 集成 2 路 UART 接口，支持多种速率，最高可达 7Mbps；
- 集成 2 路 CANFD 接口，支持多种速率，最高可达 5Mbps；
- 初次级隔离耐压 3500VDC，两路 CANFD 接口隔离耐压 2500VDC；
- 工作温度：-40℃~+105℃；
- 电磁辐射 EME 极低；
- 电磁抗干扰 EMS 极高；

1.3 产品型号

型号	供电	信号电平	SPI 速率	UART 速率	CANFD 速率	封装
CSM430B	3.3V	3.3V	0-6Mbit/s	300-7Mbps	40k-1Mbps 仲裁域 400k-5Mbps 数据域	DFN22

1.4 应用场合

- 储能
- 电池检测
- 充电桩
- 轨道交通
- 重型机械
- 煤矿
- 工业自动化
- 仪器仪表
- ……

2. 硬件说明

2.1 产品外观

产品外观如图 2.1 所示。

图 2.1 产品外观图

2.2 引脚定义

CSM430B 具有 3 种接口。一种是 SPI 接口，一种是 UART 接口，另外一种是 CANFD 接口。产品引脚排列如图 2.2。

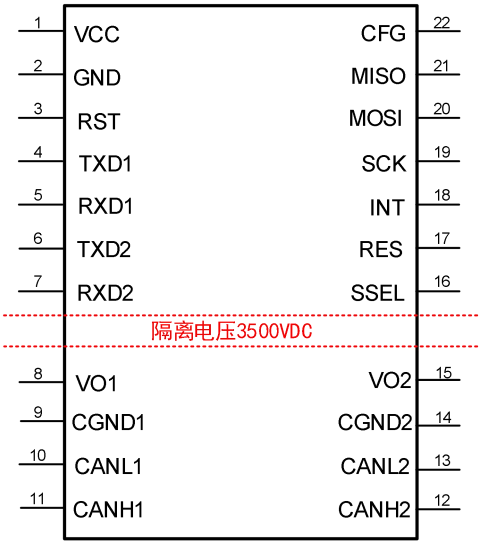

图 2.2 CSM430B 引脚排列

各引脚功能如表 2.1。

表 2.1 引脚功能描述

引脚	名称	功能	引脚	名称	功能
1	VCC	输入电源正	12	CANH2	CANH 脚
2	GND	输入电源地	13	CANL2	CANL 脚
3	RST	复位脚	14	CGND2	隔离输出电源地
4	TXD1	UART 发送脚	15	VO2	隔离输出电源正
5	RXD1	UART 接收脚	16	SSEL	SPI 片选引脚
6	TXD2	UART 发送脚	17	RES	保留引脚
7	RXD2	UART 接收脚	18	INT	从机反馈引脚
8	VO1	隔离输出电源正	19	SCK	SPI SCK 脚
9	CGND1	隔离输出电源地	20	MOSI	SPI MOSI 脚
10	CANL1	CANL 脚	21	MISO	SPI MISO 脚
11	CANH1	CANH 脚	22	CFG	配置引脚

2.3 IO 说明

表 2.2 产品 IO 引脚说明

引脚	名称	类型	说明	引脚	名称	类型	说明
1	VCC	输入	输入电源正	12	CANH2	输入/输出	不支持 5V 容压
2	GND	输入	输入电源地	13	CANL2	输入/输出	不支持 5V 容压
3	RST	输入	低电平复位，不支持 5V 容压	14	CGND2	输出	输出电源地
4	TXD1	输出	TTL 电平，不支持 5V 容压	15	VO2	输出	输出电源正
5	RXD1	输入	TTL 电平，不支持 5V 容压	16	SSEL	输入	不支持 5V 容压
6	TXD2	输出	TTL 电平，不支持 5V 容压	17	RES	输入	不支持 5V 容压
7	RXD2	输入	TTL 电平，不支持 5V 容压	18	INT	输入	不支持 5V 容压
8	VO1	输出	输出电源正	19	SCK	输入	不支持 5V 容压
9	CGND1	输出	输出电源地	20	MOSI	输入	不支持 5V 容压
10	CANL1	输入/输出	不支持 5V 容压	21	MISO	输出	不支持 5V 容压
11	CANH1	输入/输出	不支持 5V 容压	22	CFG	输入	不支持 5V 容压

2.4 应用硬件电路

2.4.1 SPI 转 CANFD1 硬件电路

为保证芯片 CSM430B 输入电源的稳定性，用户使用时务必在输入电源正与地之间并入 4.7μF 的陶瓷电容，输出电源正与地之间并入不小于 22μF 的陶瓷电容。使用 SPI 转 CANFD1 功能时，需要将 CFG 引脚接至高电平。MCU 的 SPI 接口与 CSM430B 的 SPI 接口连接，同时 MCU 需要提供 GPIO 与 RST、INT 引脚连接，实现对 CSM430B 的有效监测与控制。若需要通过 MCU 对 CSM430B 进行配置，则需要额外的 GPIO 与 CFG 引脚连接。图 2.3 是 CSM430B 在 SPI 转 CANFD1 模式下的参考电路。

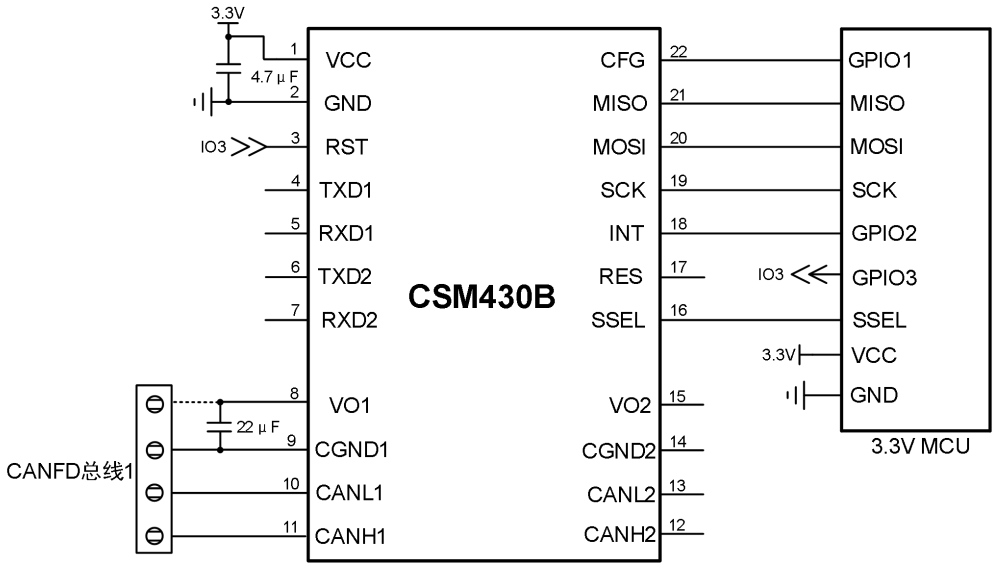

图 2.3　SPI 转 CANFD1 参考电路

2.4.2 SPI 转 CANFD1 和 CANFD2 硬件电路

为保证芯片 CSM430B 输入电源的稳定性，用户使用时务必在输入电源正与地之间并入 4.7μF 的陶瓷电容，输出电源正与地之间并入不小于 22μF 的陶瓷电容。使用 SPI 转 CANFD1 和 CANFD2 功能时，需要将 CFG 引脚接至高电平。MCU 的 SPI 接口与 CSM430B 的 SPI 接口连接，同时 MCU 需要提供 GPIO 与 RST、INT 引脚连接，实现对 CSM430B 的有效监测与控制。若需要通过 MCU 对 CSM430B 进行配置，则需要额外的 GPIO 与 CFG 引脚连接。图 2.4 是 CSM430B 在 SPI 转 CANFD1 和 CANFD2 模式下的参考电路。

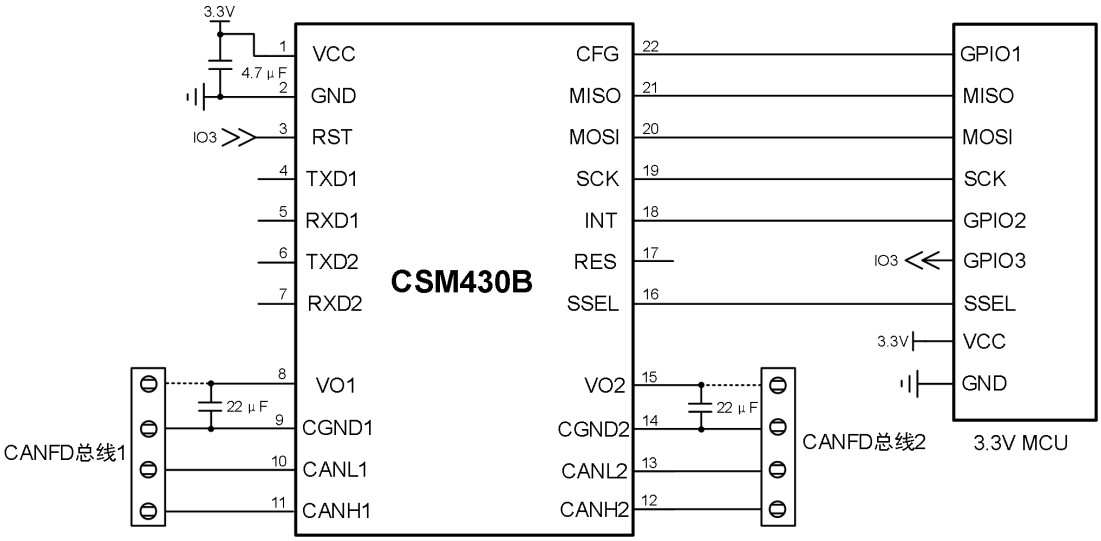

图 2.4　SPI 转 CANFD1 和 CANFD2 参考电路

2.4.3 UART1 转 CANFD1 硬件电路

为保证芯片 CSM430B 输入电源的稳定性，用户使用时务必在输入电源正与地之间并入 4.7μF 的陶瓷电容，输出电源正与地之间并入不小于 22μF 的陶瓷电容。使用 UART1 转 CANFD1 功能时，需要将 CFG 引脚接至高电平。MCU 的 UART 与 CSM430B 的 UART1 接口连接，同时需要一个 GPIO 与 RST 引脚连接。若需要通过 MCU 对 CSM430B 进行配置，则需要额外 GPIO 的与 CFG 引脚连接。图 2.5 是 CSM430B 在 UART1 转 CANFD1 模式下的参考电路。

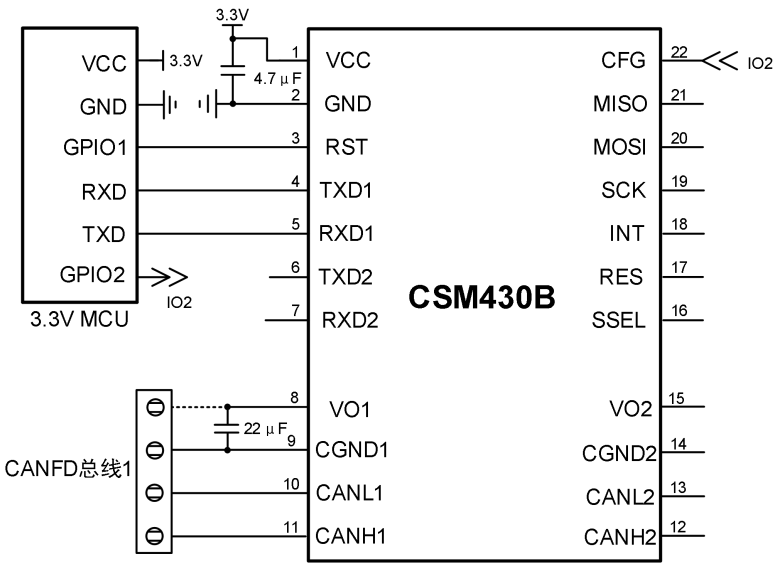

图 2.5　UART1 转 CANFD1 参考电路

2.4.4 UART1 转 CANFD1 和 CANFD2 硬件电路

为保证芯片 CSM430B 输入电源的稳定性，用户使用时务必在输入电源正与地之间并入 4.7μF 的陶瓷电容，输出电源正与地之间并入不小于 22μF 的陶瓷电容。使用 UART1 转 CANFD1 和 CANFD2 功能时，需要将 CFG 引脚接至高电平。MCU 的 UART 与 CSM430B 的 UART1 接口连接，同时需要一个 GPIO 与 RST 引脚连接。若需要通过 MCU 对 CSM430B 进行配置，则需要额外 GPIO 的与 CFG 引脚连接。图 2.6 是 CSM430B 在 UART1 转 CANFD1 和 CANFD2 模式下参考电路。

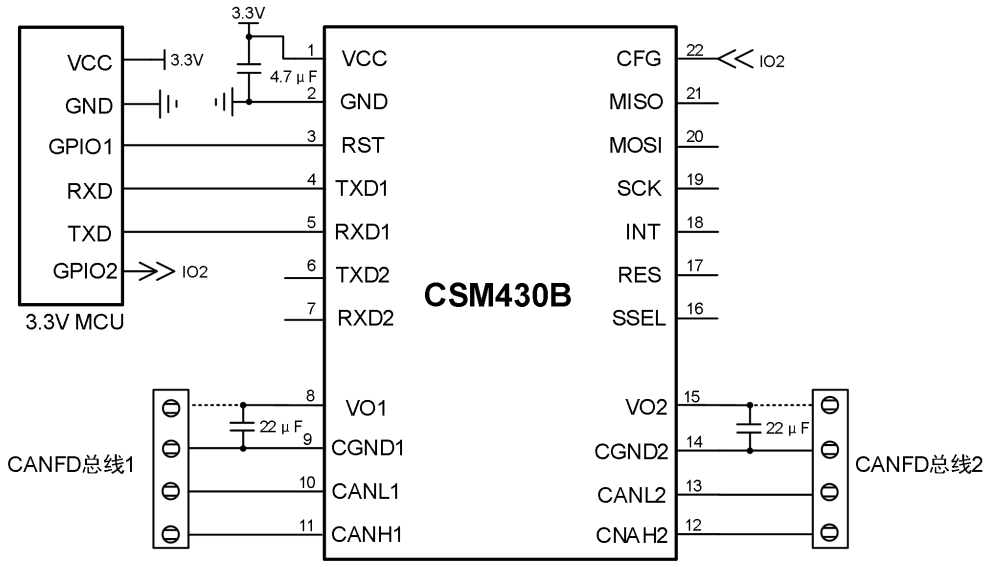

图 2.6　UART1 转 CANFD1 和 CANFD2 参考电路

2.4.5 UART1 和 UART2 转 CANFD1 和 CANFD2 硬件电路

为保证芯片 CSM430B 输入电源的稳定性，用户使用时务必在输入电源正与地之间并入 4.7μF 的陶瓷电容，输出电源正与地之间并入不小于 22μF 的陶瓷电容。使用 UART1 和 UART2 转 CANFD1 和 CANFD2 功能时，需要将 CFG 引脚接至高电平。MCU 的 UART1 与 CSM430B 的 UART1 接口连接，MCU 的 UART2 与 CSM430B 的 UART2 接口连接。同时需要一个 GPIO 与 RST 引脚连接。若需要通过 MCU 对 CSM430B 进行配置，则需要额外 GPIO 的与 CFG 引脚连接。图 2.7 是 CSM430B 在 UART1 和 UART2 转 CANFD1 和 CANFD2 模式下参考电路。

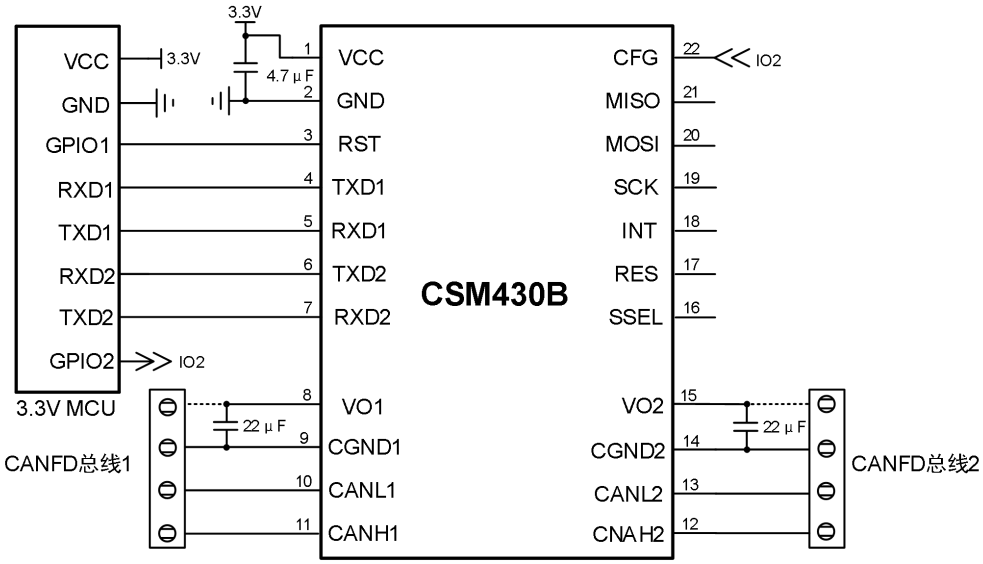

图 2.7　UART1 和 UART2 转 CANFD1 和 CANFD2 参考电路

2.5　外围保护电路

CSM430B 可使用于各种需要使用到 CANFD 总线的场合，如果应用环境比较恶劣(如高压电力、雷击等环境），强烈建议用户增加一定的外围保护电路。保护电路可以有效地吸收恶劣环境下引入到电源或总线上的浪涌，保护产品不被损坏。

图 2.8、图 2.9 提供了两个参考外围保护电路。图 2.8 是配合致远电子 SP00S12 信号浪涌抑制器使用的应用电路图。SP00S12 与 CSM430B 之间连接简单，使用方便，占板面积小，SP00S12 的详细参数请参考产品数据手册。而图 2.9 使用了 2 个 TVS 管与 12 个二极管实现端口差模和共模的保护。表 2.3 的外围保护电路元器件的参数值仅作为参考，请根据实际情况来确定是否需要电路图中的器件，并选取适当的参数值。

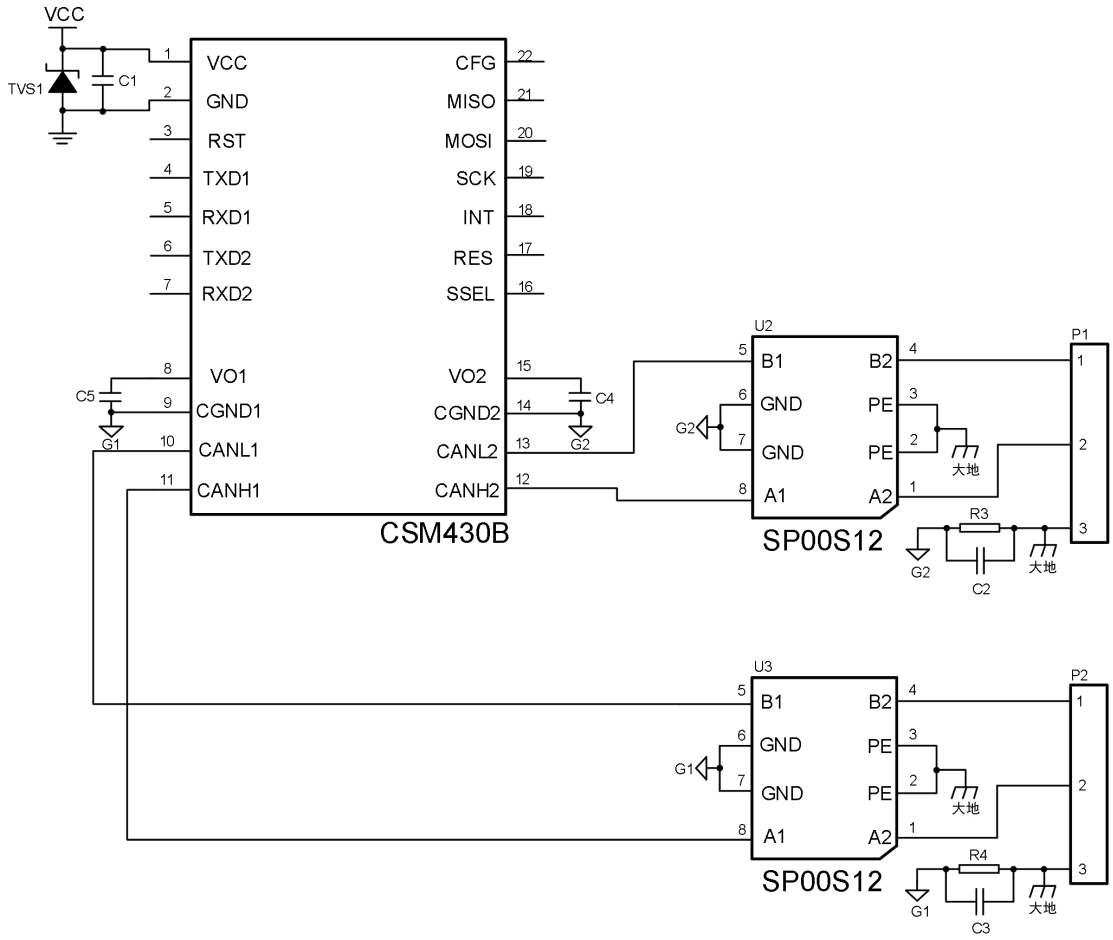
VCC
TVS1
C1
1
2
3
4
5
6
7
8
9
10
11
VCC
GND
RST
TXD1
RXD1
TXD2
RXD2
VO1
CGND1
CANL1
CANH1
22
21
20
19
18
17
16
15
14
13
12
CFG
MISO
MOSI
SCK
INT
RES
SSEL
VO2
CGND2
CANL2
CANH2
C5
G1
C4
G2
CSM430B
U2
B1
GND
GND
A1
B2
PE
PE
A2
G2
SP00S12
P1
大地
R3
C2
U3
G1
C3
R4
P2

图 2.8　外围保护电路 1

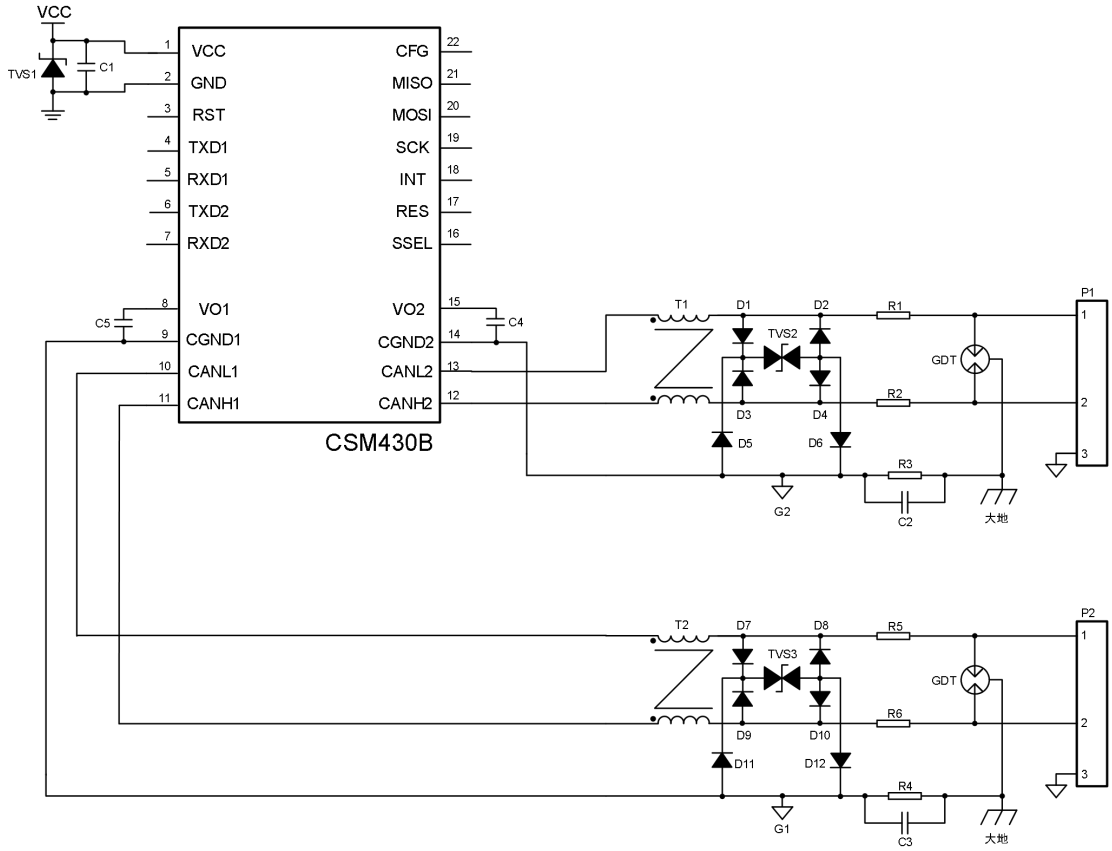

图 2.9　外围保护电路 2

表 2.3　外围保护电路 1 和电路 2 推荐参数表

标号	型号	标号	型号
C1	4.7μF，25V	TVS1	SMBJ5.0A
R1，R2，R5，R6	2.7Ω，2W	TVS2，TVS3	P6KE15CA
R3，R4	1MΩ，1206	GDT	B3D090L
C2，C3	102，2kV	T1，T2	B82793S0513N201
D1~D12	1N4007	U2，U3	SP00S12

2.6 推荐组网方式

CAN/CANFD 总线一般使用直线型布线方式，总线节点数可达 110 个。布线推荐使用屏蔽双绞线，CANH、CANL 与双绞线线芯连接，CGND 与屏蔽层连接，最后屏蔽层单点接地。无论总线长短，总线两端都需要连接终端电阻，一般推荐值为 120Ω。电阻大小可根据实际布线进行调整，图 2.10 给出了推荐组网示意图。

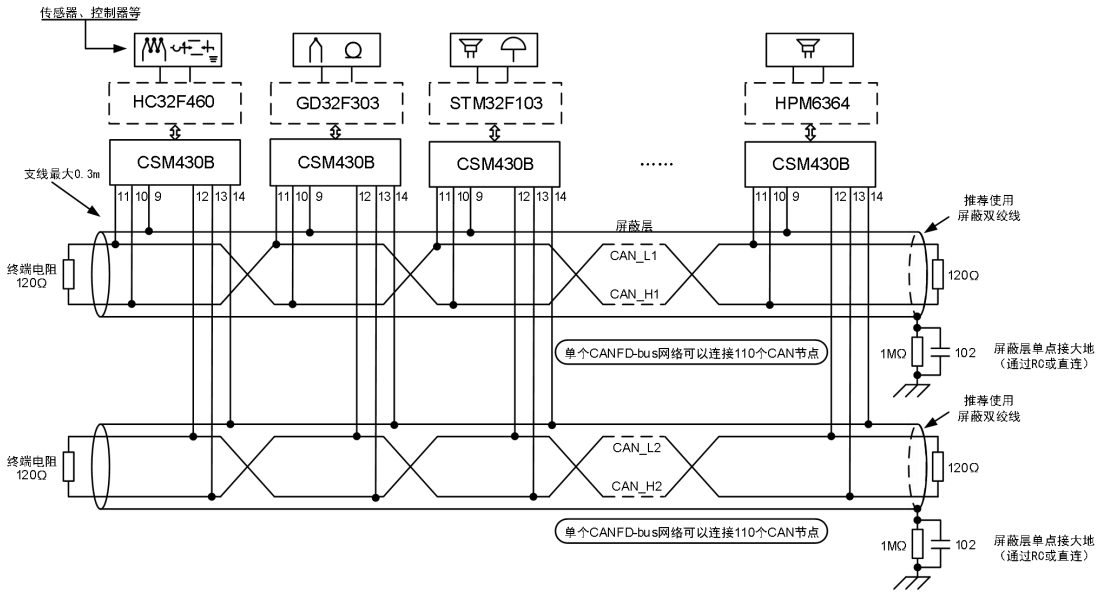

图 2.10 推荐组网示意图

3. 产品应用

3.1 名词解释

1. SPI

SPI 是串行外设接口(Serial Peripheral Interface）的缩写。SPI 是一种高速的、全双工、同步的通信总线。

2. UART

UART 是通用异步接收/发送装置(Universal Asynchronous Receiver/Transmitter）的缩写。UART 是一种通用串行数据总线，可实现全双工的串行异步通信。

3. CAN 总线

CAN 是控制器局域网（Controller Area Network）的缩写。CAN 总线属于现场总线的范畴，是一种有效支持分布式控制或实时控制的串行通信网络。

4. CANFD 总线

CANFD 是控制器局域网(CAN with Flexble Data Rate）的缩写。是 CAN 总线的一种升级版。

5. 串行帧

即串行总线帧，是 SPI 总线通信帧(下文简称 SPI 帧)、UART 总线通信帧(下文简称 UART 帧）的统称。

6. CAN 帧

即 CAN 总线帧，是 CAN 接口标准帧、扩展帧的统称。

7. CANFD 帧

即 CANFD 总线帧，是 CANFD 接口标准帧、扩展帧的统称。

8. 标准帧

CAN/CANFD 帧的类型，标准帧的帧 ID 共 11 位，范围为：0x0-0x7FF。

9. 扩展帧

CAN/CANFD 帧的类型，扩展帧的帧 ID 共 29 位，范围为：0x00000000-0x1FFFFFFF。

10. 配置反馈机制

当使用 UART1 接口对 CSM430B 进行配置时，向 RXD1 引脚每发送一帧配置帧，CSM430B 会通过 TXD1 引脚发出一帧返回帧来告知用户上一帧配置帧是否正确成功。在返回帧中，其中一个字节用来标识上一帧配置帧正确与否。如有错误，可以通过该字节确认错误类型。当使用 SPI 接口对 CSM430B 进行配置，主机每发送一帧配置帧，CSM430B 会将 INT 引脚拉低并保持约 130us，通知主机上一帧配置帧是否正确。主机检测到低电平后，向 CSM430B 发送 7 个字节无效数据，同时 CSM430B 通过 MISO 引脚发出返回帧。

11. 错误反馈机制

在传输模式下，CSM430B 检测到 UART1、UART2 和 SPI 帧长度错误、通道错误、帧 CRC 校验错误、帧类型错误、帧 ID 错误，CANFD 错误计数(计数到 128 以上），用户可以使用当前通讯接口，向 CSM430B 发送读取错误命令帧，CSM430B 接收到该命令后，通过相同通讯接口发出错误信息回应帧。错误信息回应帧发出后部分寄存器值自动清零。

12. 自定义协议转换

CSM430B 的一种数据传输方式，自定义协议转换方式下，串行帧必须符合规定的帧格式。有效的串行帧由帧头、帧长度、命令、通道、帧类型、帧 ID、数据域组成。

13. 自定义带校验协议转换

CSM430B 的一种数据传输方式，在自定义协议转换的基础上增加了 CRC 校验字节，

用于判断 SPI/UART 端数据是否正确。串行帧必须符合规定的帧格式，有效的串行帧由帧头、帧长度、命令、通道、帧类型、帧 ID、数据域、CRC 校验域组成。

3.2 工作模式

3.2.1 工作模式切换说明

CSM430B 上电后，会进入默认配置。若需要切换产品的工作模式，发送对应配置命令后，必须对产品进行复位，才能使其进入设定的工作模式。需要注意的是，为保证产品成功复位，复位保持时间至少为 100us，复位后，产品初始化等待时间至少为 200ms，待产品初始化完成后，才能进行正常操作如图 3.1 所示，。

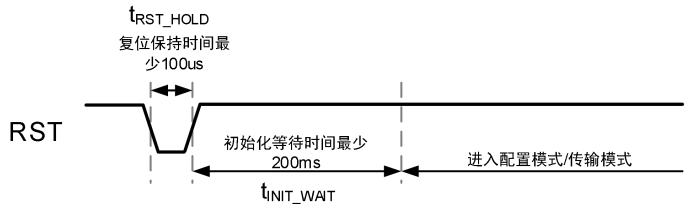

图 3.1 复位时序示意图

3.2.2 SPI 转 CANFD1 模式

在此工作模式下，CSM430B 始终作为 SPI 从机，SPI 限定工作在模式 3(CPOL、CPHA 均为 1），数据长度最大为 128 字节，MSB 高位先传输。SPI 的最高速率可达 6Mbps。SPI 主机可以发送数据至 CAN1 总线端，且可接收 CAN1 总线端收到的数据。此时 UART1、UART2 和 CAN2 接口无效，CSM430B 不会处理出现在这些接口的任何数据，也不会将 CAN1 总线端接收到的数据转换至 UART1 和 UART2。

1. SPI 帧

从接收到第一个数据开始，直到收到第 n 个字符(此参数由用户定义，包括帧头+长度+命令+通道+帧类型+帧 ID+数据域+CRC 域总字节数）后为止，这些数据定义为一帧数据。帧与帧之间读写缓冲区数据应有 50us 的时间间隔，如图 3.2 所示。

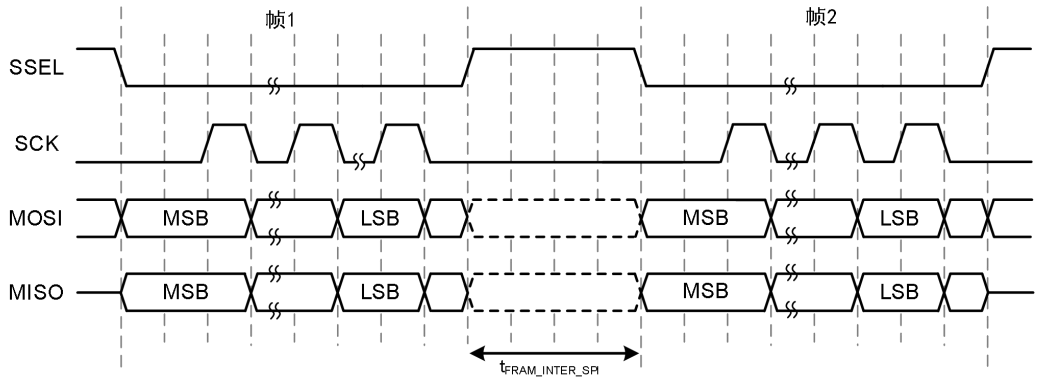

图 3.2　SPI 帧间隔示意图

2.　主机读取从机数据

当 CAN1 接口接收到数据时，CSM430B 会将接收到的数据转换成 SPI 帧并放进缓存区内，待主机来读取。主机需要读取从机数据时，要进行以下两步操作：

1.主机发送读取从机数据量的命令如表 3.1 所示。主机继续发送 11 个字节给从机，从机同时会返回一帧数据帧如表 3.2 所示给主机。该帧中的数据字段代表从机缓存区中有 n 个字节的待读取数据；

2.主机发送读取从机数据命令如表 3.3 所示。主机继续发送(n+1)个字节给从机，同时从机将缓存区中的数据帧按帧格式如表 3.4 一次性返回给主机；

表 3.1　读取从机数据量命令帧构成

字节数	含义	代码(0x)	描述	发送顺序
1	帧头	AC	通过 SPI 接口发送和接收数据	从上至下依次发送
1	长度	00	该命令帧帧不涉及帧类型、帧 ID、数据域和 CRC 字段	
1	命令	07	该帧为主机读取从机数据量命令帧	
1	通道	FF	该命令帧不涉及通道	

表 3.2　从机缓存区待读取数据量数据帧构成

字节数	含义	代码(0x)	描述	发送顺序
1	帧头	AC	通过 SPI 接口发送和接收数据	从上至下依次发送
1	长度	以实际为准	该帧中数据字段和 CRC 字段的总字节数	
1	命令	07	与主机读取从机数据量命令一样	
1	通道	FF	该命令帧不涉及通道	
4	数据	以实际为准	缓存区中待读取的数据量	
2	CRC	以实际为准	自定义带校验协议转换模式才有这个字段， 自定义协议转换模式没有该字段	

表 3.3　读取从机数据命令帧构成

字节数	含义	代码(0x)	描述	发送顺序
1	帧头	AC	通过 SPI 接口发送和接收数据	从上至下依次发送
1	长度	00	该命令帧不涉及帧类型、帧 ID、数据域和 CRC 字段	
1	命令	03	该帧为主机读取从机数据命令帧	
1	通道	FF	该命令帧不涉及通道字段	

表 3.4　从机缓存区中每一帧数据帧构成

字节数	含义	代码(0x)	描述	发送顺序
1	帧头	AC	通过 SPI 接口发送和接收数据	从上至下依次发送
1	长度	以实际为准	该帧中数据字段和 CRC 字段的总字节数	
1	命令	03	与主机读取从机数据命令一样	
1	通道	04/05	CAN1/CAN2 接口收到数据帧	
以实际为准	数据	以实际为准	缓存区中主机待读取的数据	
2	CRC	以实际为准	自定义带校验协议转换模式才有该字段 自定义协议转换模式没有该字段	

3.　反馈机制

CSM430B 只能作为 SPI 从机，不能主动地控制其他 SPI 总线设备，如果 CSM430B 每次收到 CAN1 端发送过来的数据时，都需要主机来查询状态的话，会使得整个通讯过程的效率很低，因此我们为其添加了一个反馈机制。

CSM430B 硬件上有一个 INT 反馈引脚，此引脚与主机连接，出现以下两种情况时，INT 引脚会由高电平变成低电平，通知主机进行读数据操作(为避免数据丢失，建议主机使用下降沿触发方式检测）：

(1)　CAN1 缓冲区中的 CAN/CANFD 帧数达到设置的触发点时

当 CAN1 总线端接收缓冲区接收到的 CAN/CANFD 帧数达到触发点时，INT 引脚电平置低约 130us 后，INT 引脚才会恢复高电平。用户可以在获得 INT 信号之后查询 CSM430B 的状态，获取可读字节数，然后读取缓冲区中数据。

(2)　CAN1 缓冲区数据帧数少于触发帧数，且在设定时间内主机未读取时

当 CAN1 缓冲区有数据帧但少于触发帧数时，若总线长时间未有新增数据，且主机未进行读取操作时，缓冲区的数据将有可能长期得不到处理，这就导致数据的实时性不高。为了解决少量数据的实时性问题，CSM430B 内部设置了一个计时器，若 CAN1 缓冲区中的数据在一定时间内未被读取，将触发 INT 引脚置低约 130us 后，通知主机读取数据。CSM430B 在接收到第一帧数据时，计时器启动，主机进行读取操作时复位计时器。

3.2.3 SPI 转 CANFD1 和 CANFD2 模式

在此工作模式下，CSM430B 始终作为 SPI 从机，SPI 限定工作在模式 3(CPOL、CPHA 均为 1），数据长度最大为 128 字节，MSB 高位先传输，SPI 的最高速率可达 6Mbps。SPI 主机可以发送数据至 CAN1 总线端和 CAN2 总线端，且可接收来自 CAN1 总线端和 CAN2 总线端的数据。此模式下 CSM430B 不会处理出现在 UART1 和 UART2 接口的数据，也不会将 CAN1 总线端和 CAN2 总线端接收到的数据转换至 UART1 和 UART2。

1. SPI 帧

从接收到第一个数据开始，直到收到第 n 个字符(此参数由用户定义，包括帧头+长度+命令+通道+帧类型+帧 ID+数据域+CRC 域总字节数）后为止，这些数据定义为一帧数据。帧与帧之间读写缓冲区数据应有 50us 的时间间隔，如图 3.2。

2. 主机读取从机数据

当 CAN1、CAN2 任意一个或者两个接口都接收到数据时，CSM430B 会将接收到的数据转换成 SPI 帧并放进缓存区内，待主机来读取。主机需要读取从机数据时，要进行以下两步操作：

1.主机发送读取从机数据量的命令如表 3.1 所示。主机继续发送 11 个字节给从机，从机同时会返回一帧数据帧如表 3.2 所示给主机。该帧中的数据字段代表从机缓存区中有 n 个字节的待读取数据；

2.主机发送读取从机数据命令如表 3.3 所示。主机继续发送(n+1)个字节给从机，同时从机将缓存区中的数据帧按表 3.4 帧格式一次性返回给主机；

3. 反馈机制

CSM430B 只能作为 SPI 从机，不能主动地控制其他 SPI 总线设备，如果 CSM430B 每次收到 CAN1 端或者 CAN2 端发送过来的数据时，都需要主机来查询状态的话，会使得整个通讯过程的效率很低，因此我们为其添加了一个反馈机制。

CSM430B 硬件上有一个 INT 反馈引脚，此引脚与主机连接，出现以下两种情况时，INT 引脚会由高电平变成低电平，通知主机进行读数据操作(为避免数据丢失，建议主机使用下降沿触发方式检测）：

(1) CAN1 和 CAN2 缓存区中的 CAN/CANFD 帧数总和达到设置的触发点时

当 CAN1 总线端和 CAN2 总线端缓冲区接收到的 CAN/CANFD 帧数总和达到触发点时，INT 引脚电平置低约 130us 后，INT 引脚才会恢复高电平。用户可以在获得 INT 信号之后查询 CSM430B 的状态，获取可读字节数，然后读取缓冲区 CANFD 数据。

(2) CAN1 和 CAN2 缓存区中的 CAN/CANFD 帧数总和少于触发帧数，且在设定时间内主机未进行读取时

当 CAN1 总线端和 CAN2 总线端缓存区中的 CAN/CANFD 帧数总和少于触发帧数时，若总线长时间未有新增数据，且主机未进行读取操作时，CAN1 和 CAN2 接收缓冲区的数据将有可能长期得不到处理，这就导致数据的实时性不高。为了解决少量数据的实时性问题，CSM430B 内部设置了一个计时器，若 CAN1、CAN2 缓冲区的数据在一定时间内未被读取，将触发 INT 引脚置低约 130us，通知主机读取数据。CSM430B 在接收到第一帧数据时，计时器启动，主机进行读取操作时复位计时器。

3.2.4 UART1 转 CANFD1 模式

在此模式下，CSM430B 只能通过 UART1 向 CAN1 总线端发送或接收数据。UART 通信格式固定为：1 起始位，8 数据位，1 停止位，不可更改。UART 的通信速率范围为 300bps~7Mbps。此模式下不会处理出现在 SPI、UART2 和 CAN2 的任何数据，也不会将 CAN1 总线端接收到的数据转换至 SPI 和 UART2。

1. UART 帧

UART 从接收到第一个数据开始，直到收到第 n 个字符(此参数由用户定义，包括帧头+长度+命令+通道+帧类型+帧 ID+数据域+CRC 域总字节数）后为止，这些数据定义为一帧

数据。上一帧最后一个字节发送后到下一帧的第一字节发送前，这段时间定义为“帧间隔”，UART 通信帧格式如图 3.3。

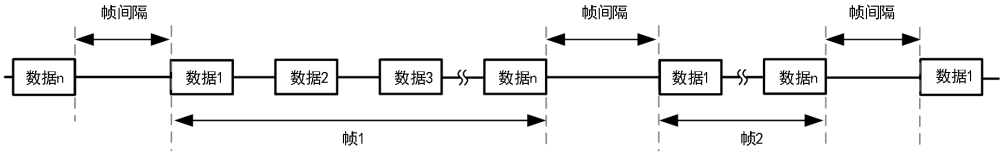

图 3.3　UART 通信帧格式

3.2.5 UART1 转 CANFD1 和 CANFD2 模式

在此模式下，CSM430B 只能通过 UART1 向 CAN1 总线端和 CAN2 总线端发送或接收数据。UART 通信格式固定为：1 起始位，8 数据位，1 停止位，不可更改。UART 的通信速率范围为 300bps~7Mbps。此模式下不会处理出现在 SPI 和 UART2 接口的任何数据，也不会将 CAN1 总线端和 CAN2 总线端接收到的数据转换至 SPI 和 UART2。

1. UART 帧

UART 从接收到第一个数据开始，直到收到第 n 个字符(此参数由用户定义，包括帧头+长度+命令+通道+帧类型+帧 ID+数据域+CRC 域总字节数）后为止，这些数据定义为一帧数据。上一帧最后一个字节发送后到下一帧的第一个字节发送前，这段时间定义为“帧间隔”，UART 通信帧格式如图 3.3。

3.2.6 UART1 和 UART2 转 CANFD1 和 CANFD2 模式

在此模式下，CSM430B 只能通过 UART1 向 CAN1 总线端发送或接收数据，通过 UART2 向 CAN2 总线端发送或接收数据。UART 通信格式固定为：1 起始位，8 数据位，1 停止位，不可更改。UART1 和 UART2 的通信速率范围均为 300bps~7Mbps。此模式下不会处理出现在 SPI 的任何数据，也不会将 CAN1 总线端和 CAN2 总线端接收到的数据转换至 SPI。

1. UART 帧

UART 从接收到第一个数据开始，直到收到第 n 个字符(此参数由用户定义，包括帧头+长度+命令+通道+帧类型+帧 ID+数据域+CRC 域总字节数）后为止，这些数据定义为一帧数据。上一帧最后一个字节发送后到下一帧的第一字节发送前，这段时间定义为“帧间隔”，UART 通信帧格式如图 3.3。

3.2.7 SPI 配置模式

在此模式下，CSM430B 处于等待配置状态，无法向 CAN1 总线端和 CAN2 总线端发送或从该接口接收数据。此模式下仅能通过 SPI 接口进行配置，具体配置说明请参考第 4.4.1 节。

3.2.8 UART1 配置模式

在此模式下，CSM430B 处于等待配置状态，无法向 CAN1 总线端和 CAN2 总线端发送或从该接口接收数据。此模式下仅能通过 UART1 接口进行配置，具体配置说明请参考第 4.4.1 节。

3.3 数据转换方式

CSM430B 的数据转换方式，指串行总线和 CAN/CANFD 总线数据之间转换的基本规则。同一时间内，产品只能工作于一种数据转换方式，若需要改变数据转换方式，则需要更改配置。CSM430B 的数据转换方式有两种：自定义协议转换、自定义带校验协议转换。

3.3.1 自定义协议转换

自定义协议转换方式下，串行帧必须符合规定的帧格式。有效的串行帧由帧头、帧长度、命令、通道、帧类型、帧 ID、数据域。

帧头，是一个串行帧的开始字节，如表 3.5 所示。

表 3.5 帧头字段代码定义

帧头字段代码	代码定义
0xAA	使用 UART1 发送和接收数据
0xAB	使用 UART2 发送和接收数据
0xAC	使用 SPI 发送和接收数据

帧长度，指帧类型、帧 ID、数据域总共包含的字节数。

命令，不同的命令代表不同的操作，如表 3.6 所示。

表 3.6 命令字段代码定义

命令段代码	代码定义
0x1	配置模式下读寄存器配置参数
0x2	配置模式下写寄存器
0x3	传输模式下读数据
0x4	传输模式下写数据
0x5	配置模式读全部寄存器
0x6	传输模式下查询错误信息
0x7	传输模式读当前数据长度

通道，传输模式下数据流通道，如表 3.7 所示。

表 3.7 通道字段代码定义

通道字段代码	代码定义
0x1	UART1 ⟷ CAN1
0x2	UART1 ⟷ CAN2
0x3	UART2 ⟷ CAN2
0x4	SPI ⟷ CAN1
0x5	SPI ⟷ CAN2

帧类型，指发送 CAN/CANFD 帧的帧类型，如表 3.8 所示。

表 3.8　帧类型字段代码定义

帧类型字段代码	代码定义
0x0	CAN 标准帧
0x8	CAN 扩展帧
0x80	CANFD 标准帧
0x88	CANFD 扩展帧

帧 ID，指发送 CAN/CANFD 帧的帧 ID。由 4 个字节组成，低字节先发。CAN/CANFD 标准帧，其范围为 0x0-0x7FF，CAN/CANFD 扩展帧范围为 0x00000000~0x1FFFFFFF。

数据域，是需要发送的数据，转换至 CAN/CANFD 帧的数据域。

当用户发送的串行帧完全符合定义的格式时，CSM430B 才会接收串行帧的数据并进行转发，否则不作任何处理直接丢弃。

1.　串行帧转 CAN/CANFD 帧(SPI/UART1/UART2→CAN/CANFD）

串行接口接收到有效的串行帧后，“帧类型”确定将要发送 CAN/CANFD 帧的帧类型，“帧 ID”作为 CAN/CANFD 帧 ID，“数据域”填充至 CAN/CANFD 帧数据域。

当帧类型为 CAN 标准帧/扩展帧时，串行帧数据域字节数小于等于 8，数据全部通过一个 CAN 帧转发；当帧类型为 CANFD 标准帧/扩展帧，串行帧数据域字节数小于等于 64，数据全部通过一个 CANFD 帧转发，如图 3.4 所示。

当帧类型为 CAN 标准帧/扩展帧时，串行帧数据域字节数大于 8 小于等于 128，数据通过多个 CAN 帧转发；当帧类型为 CANFD 标准帧/扩展帧，串行帧数据域字节数大于 64 小于等于 128，数据通过多个 CANFD 帧转发，如图 3.5 所示。

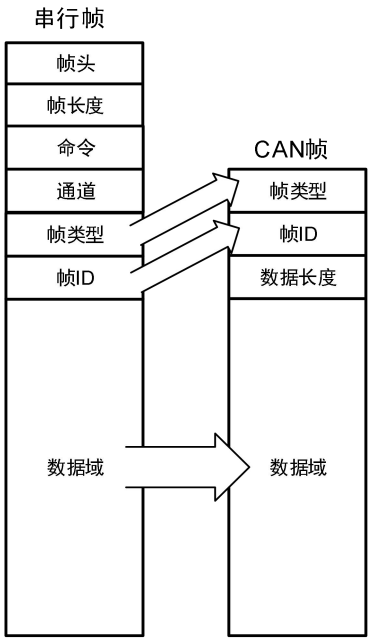

图 3.4　串行帧转 CAN/CANFD 帧 1(自定义协议转换）

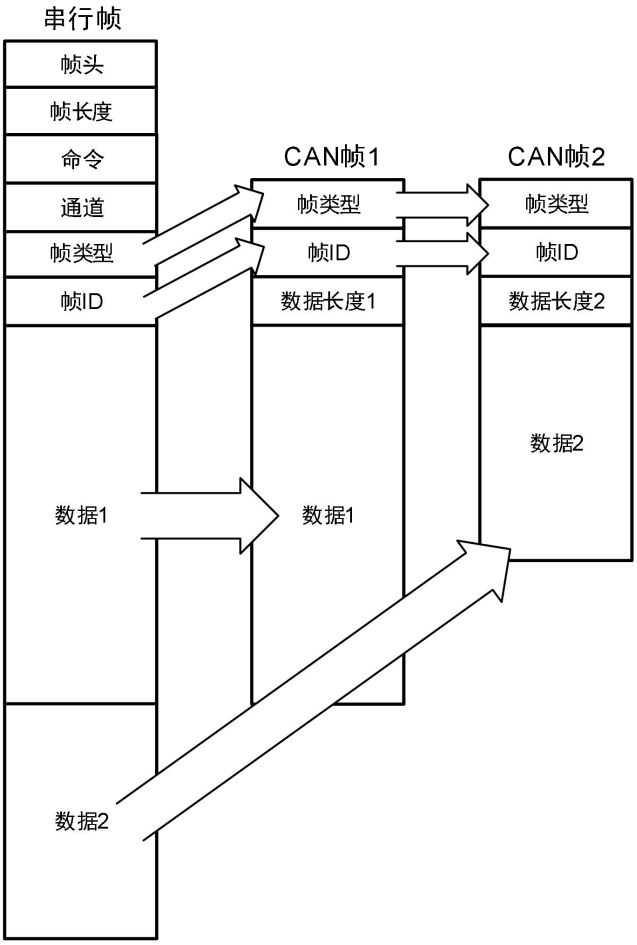

图 3.5　串行帧转 CAN/CANFD 帧 2(自定义协议转换）

转换实例

例 1：假设用户配置的串行帧头为 0xAA。用户发送帧类型为 CAN 标准帧(0x0），帧 ID 为 0123，通道为 01，数据{0x1，0x2，0x3，0x4，0x5，0x6，0x7，0x8，0x9，0xA，0xB}，则帧长度为 0x10。转换示例如图 3.6。

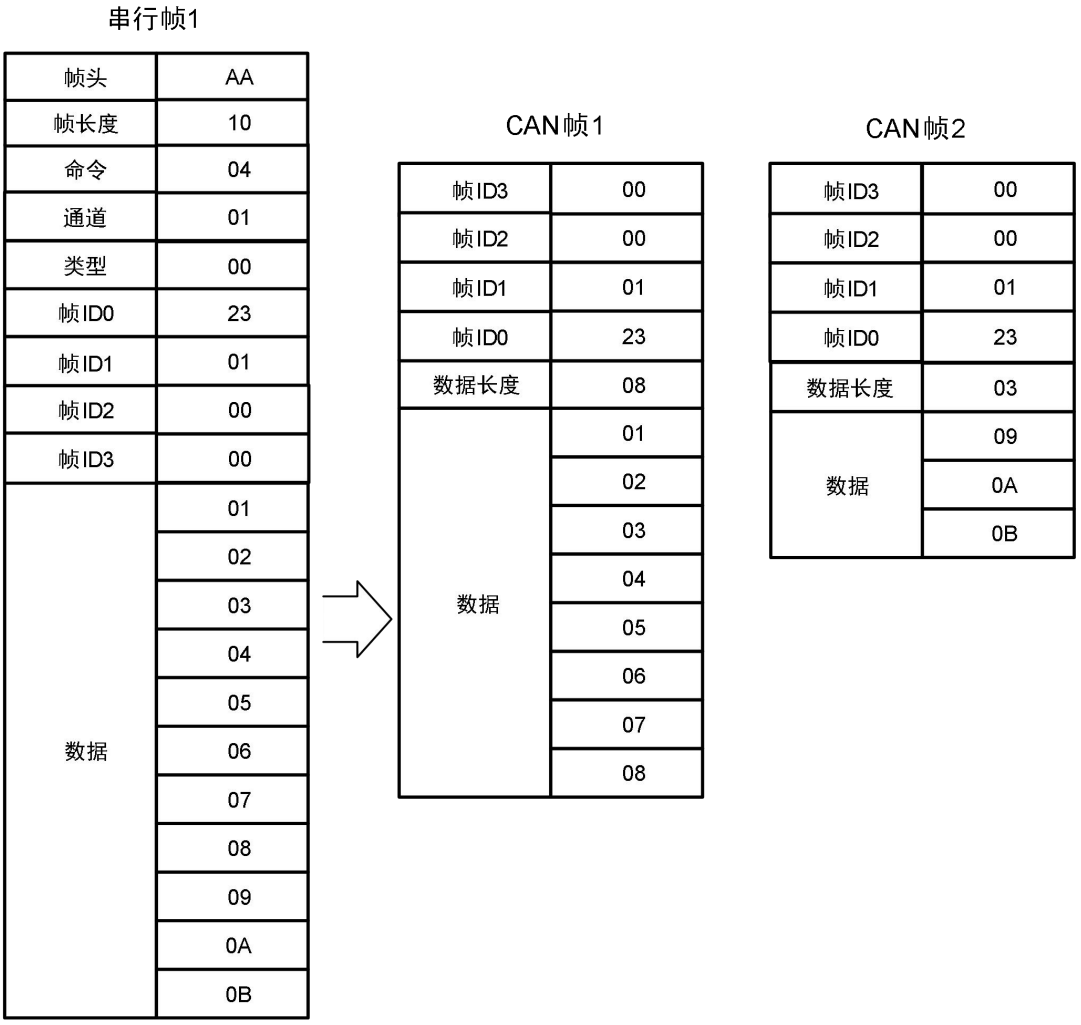

图 3.6　串行帧转 CAN 帧示例(自定义协议转换）

2.　CAN/CANFD 帧转 UART 帧

CAN/CANFD 总线收到一帧 CAN/CANFD 帧立即转换一帧。数据格式对应如图 3.7 所示。转换时，CAN/CANFD 帧类型转换到 UART 帧类型字节，帧 ID 转换到 UART 帧 ID 字节，CAN/CANFD 帧数据域中的数据依序全部转换到串行帧数据域中。

为了保证串行帧的完整性，在 CAN/CANFD 帧转换为串行帧时，帧头和通道与用户设置一致，这时的串行帧完全与串行帧转 CAN/CANFD 帧时的帧格式一致。

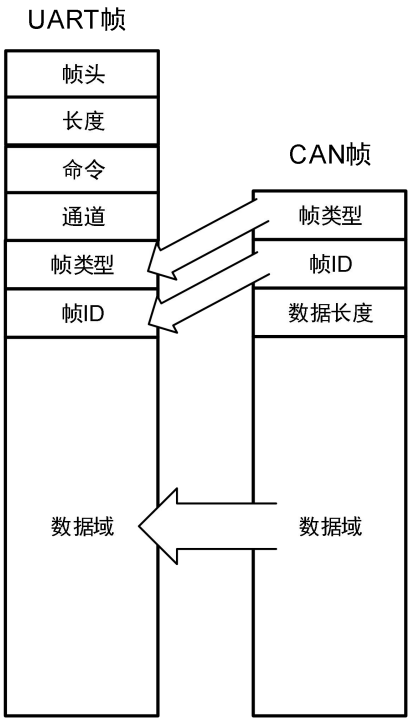

图 3.7　CAN/CANFD 帧转 UART 帧(自定义协议转换）

转换实例

例 1：假设用户配置的串行帧头为 0xAA，通道为 0x1。转换示例如图 3.8。

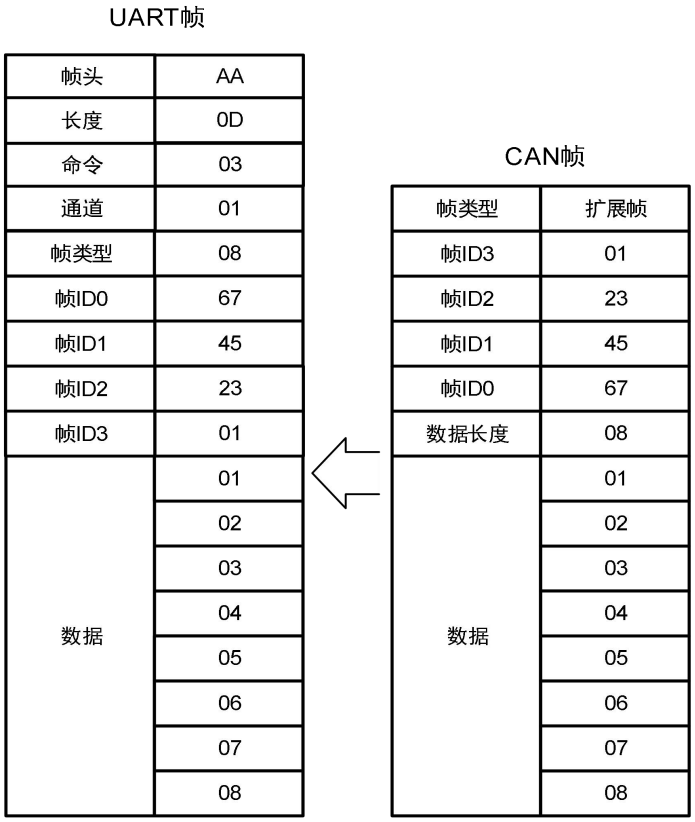

图 3.8　CAN 帧转 UART 帧示例(自定义协议转换）

3. CAN/CANFD 帧转 SPI 帧(CAN/CANFD→SPI）

CSM430B 作为 SPI 从机，无法主动控制 SPI 外设，当产品从 CAN/CANFD 总线收到一帧数据后，只能立即存储到接收缓冲区。当缓冲区的 CAN/CANFD 帧数达到反馈触发帧数，或触发时间，INT 引脚输出低电平并保持约 130us 通知 SPI 主机进行数据读取。

当 CAN1 或者 CAN2 总线端接收到数据时，CSM430B 会将接收到的数据转换成 SPI 帧并放进缓存区内，待主机来读取。主机需要读取从机数据时，要进行两步操作：1.主机发送读取从机数据量的命令，CSM430B 会发送一帧数据帧，告知主机当前缓存区中有多少数据；2.主机发送读取从机数据命令，CSM430B 会将缓存区中的数据帧发送给主机。

CAN1、CAN2 接收缓冲区存储的是多个有效格式字节段的集合。CAN1 或者 CAN2 端接收到一个 CAN/CANFD 数据帧时，将其转换为符合用户定义格式的字节段放置到缓冲区。为避免出现缓存区数据量超过 140 帧后无法读取第一帧数据情况，建议用户可以在读取 140 帧数据后，再接收新的数据。

SPI 主机读取 CAN1、CAN2 接收缓冲区可读数据量大小后，可以一次读取出所有可读数据帧。其转换数据格式如图 3.9。

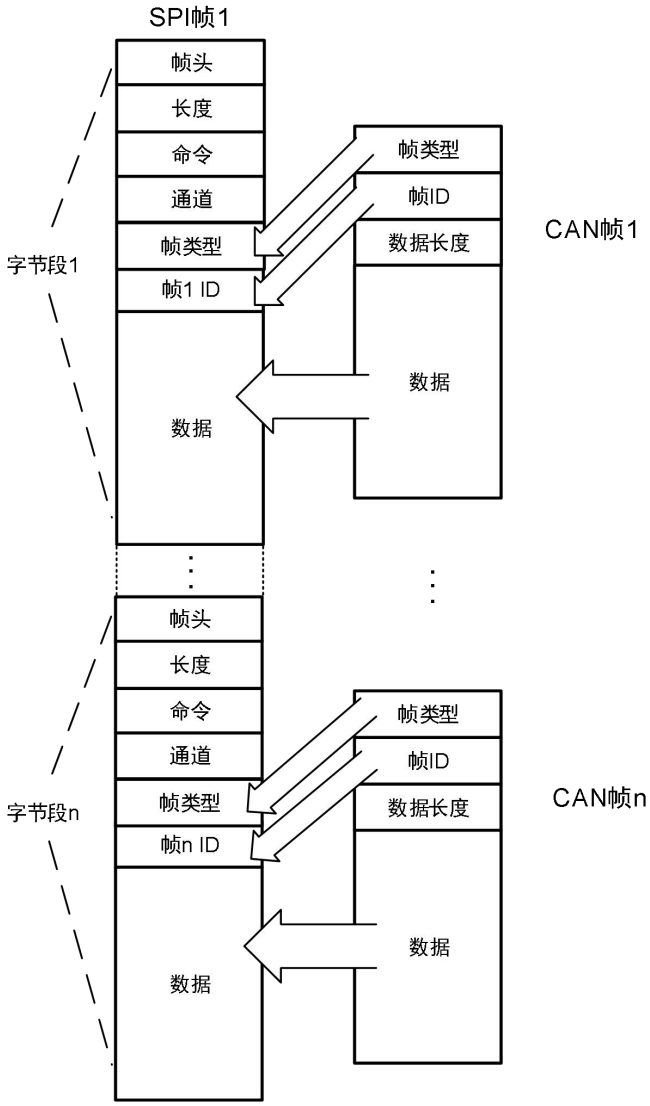

图 3.9 CAN/CANFD 帧转 SPI 帧(自定义协议转换）

转换实例

例 1：假设用户配置的串行帧头为 0xAC，通道是 0x4。转换示例如图 3.10。

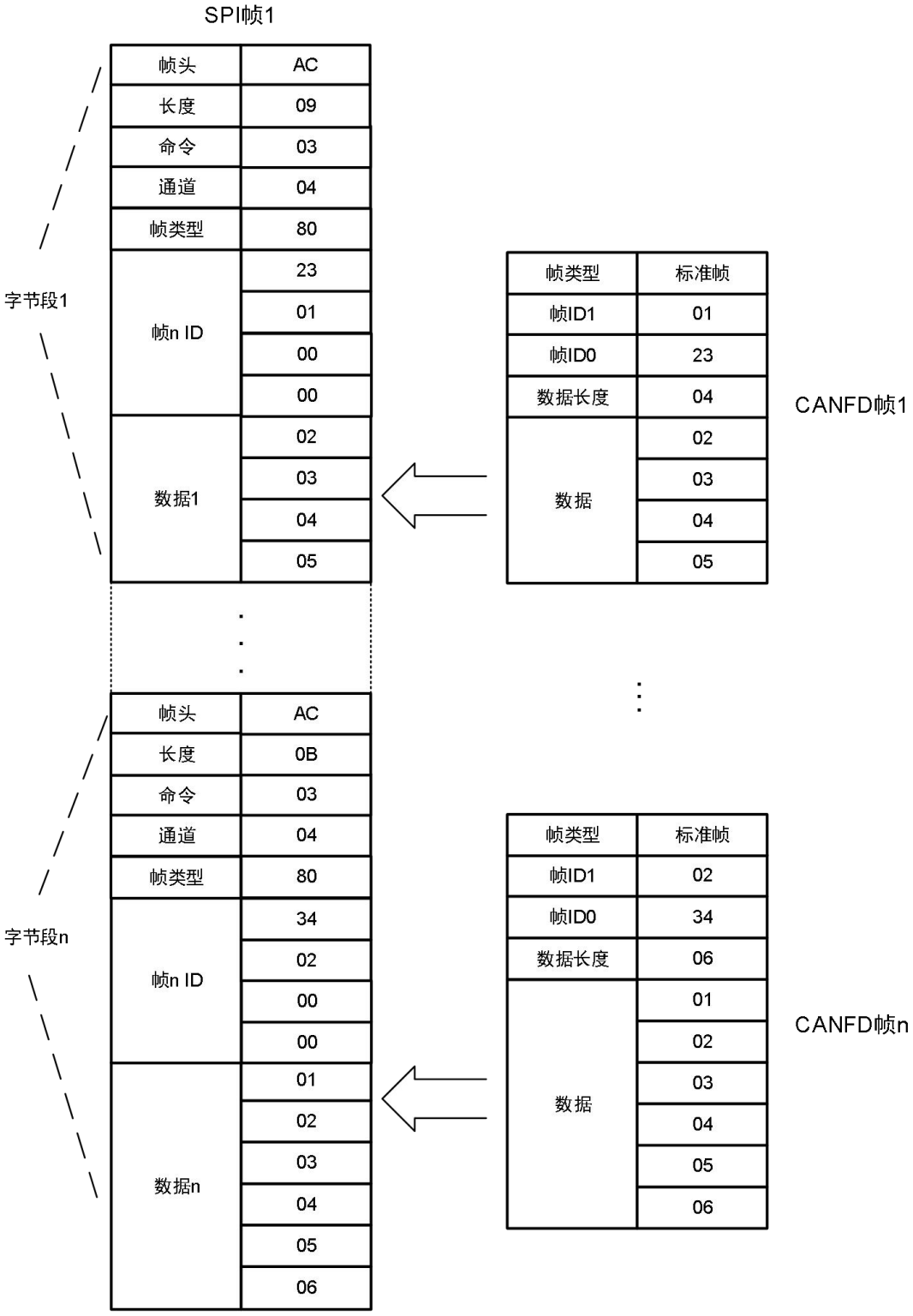

图 3.10　CANFD 帧转 SPI 帧示例(自定义协议转换）

3.3.2 自定义带校验协议转换

自定义协议转换方式下，串行帧必须符合规定的帧格式。有效的串行帧由帧头、帧长度、命令、通道、帧类型、帧 ID、数据域和 CRC 校验域。

帧头，是一个串行帧的开始字节，如表 3.5 所示。

帧长度，指帧类型、帧 ID、数据域和 CRC 校验域总共包含的字节数。

命令，不同的命令代表不同的操作。如表 3.6 所示。

通道，传输模式下数据流通道，如表 3.7 所示。

帧类型，指发送 CAN 帧的帧类型，表 3.8 所示。

帧 ID，指发送 CAN/CANFD 帧的帧 ID。由 4 个字节组成，低字节先发。CAN/CANFD 标准帧，其范围为 0x0-0x7FF，CAN/CANFD 扩展帧范围为 0x00000000~0x1FFFFFFF。

数据域，是需要发送的数据，转换至 CAN/CANFD 帧的数据域。

CRC 校验，指对帧头、帧长度、命令、通道、帧类型、帧 ID、数据域进行 CRC 校验的 2 个字节，低字节先发。CRC 采用 CRC-16-X25 算法，具体参考表 3.9。在该 CRC 算法下将数据{0xAA 0x47 0x4 0x1 0x88 0xFF 0xFF 0xFF 0x1F 0x0 0x1 0x10 0x11 0x2 0x20 0x22 0x3 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x0 0x1 0x10 0x11 0x2 0x20 0x22 0x3 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x0 0x1 0x10 0x11 0x2 0x20 0x22 0x3 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x0 0x1 0x10 0x11 0x2 0x20 0x22 0x3 0xFF 0xFF 0xFF0x FF 0xFF 0xFF 0xFF 0xFF}进行校验结果为 0x7E 0x2A。

表 3.9　CRC 校验说明

参数模型	宽度位数	多项式	初始值	结果异或值	输入输出反转性
X16+X12+X5+1	16	0x1021	0xFFFF	0xFFFF	均需反转

当用户发送的串行帧完全符合定义的格式时，CSM430B 才会接收串行帧的数据并进行转发，否则不作处理直接丢弃。

1.　串行帧转 CAN/CANFD 帧(SPI/UART→CAN/CANFD）

串行接口接收到有效的串行帧后，“帧类型”确定将要发送 CAN/CANFD 帧的帧类型，“帧 ID”作为 CAN/CANFD 帧 ID，“数据域”填充至 CAN/CANFD 帧数据域。

当帧类型为 CAN 标准帧/扩展帧时，串行帧数据域字节数小于等于 8，数据全部通过一个 CAN 帧转发；当帧类型为 CANFD 标准帧/扩展帧，串行帧数据域字节数小于等于 64，数据全部通过一个 CANFD 帧转发，如图 3.11 所示。

当帧类型为 CAN 标准帧/扩展帧时串行帧数据域字节数大于 8 小于等于 128，数据通过多个 CAN 帧转发；当帧类型为 CANFD 标准帧/扩展帧串行帧数据域字节数大于 64 小于等于 128，数据通过多个 CANFD 帧转发，如图 3.12 所示。

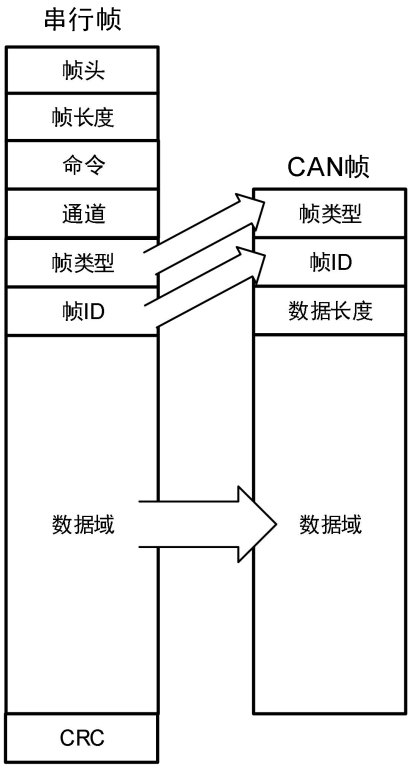

图 3.11　串行帧转 CAN/CANFD 帧 1(自定义带校验协议转换）

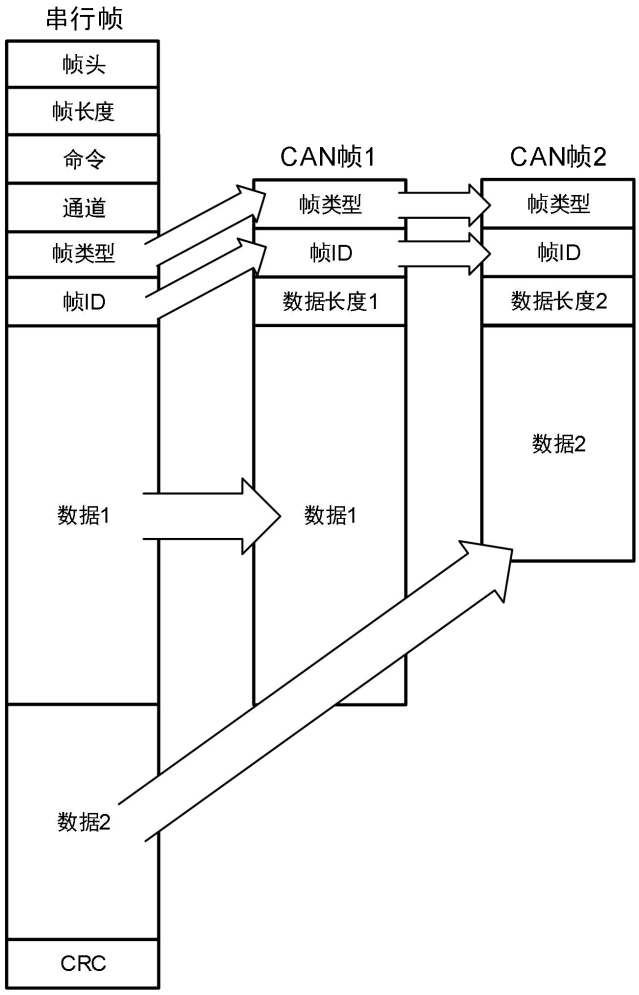

图 3.12　串行帧转 CAN/CANFD 帧 2(自定义带校验协议转换）

转换实例

例 1：假设用户配置的串行帧头为 0xAA，通道为 0x1，命令为 0x4，类型为 0x0。用户发送帧类型为 CAN 标准帧(0x0），帧 ID 为 0123，数据为{0x1，0x2，0x3，0x4，0x5，0x6，0x7，0x8，0x9，0xA，0xB}，则帧长度为 0x12。转换示例如图 3.13。

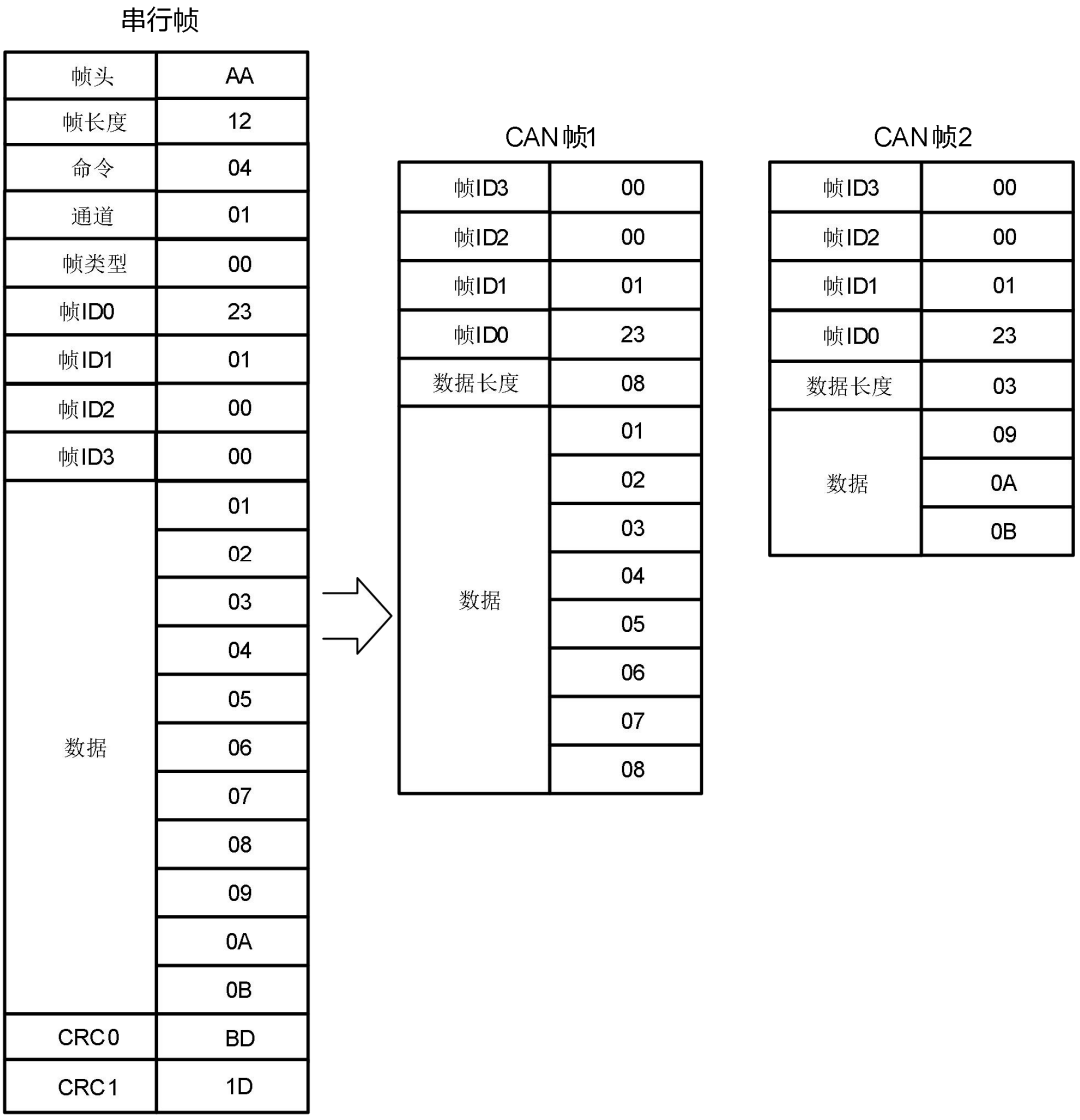

图 3.13　串行帧转 CAN/CANFD 帧示例(自定义带校验协议转换）

2.　CAN/CANFD 帧转 UART 帧

CAN/CANFD 总线收到一帧 CAN/CANFD 帧立即转换一帧。数据格式对应如图 3.14 所示，转换时，CAN/CANFD 帧类型转换到 UART 帧类型字节，帧 ID 转换到 UART 帧 ID 字段，CAN/CANFD 帧数据域中的数据依序全部转换到串行帧数据域中，CRC 校验域将自动计算后填充。

为了保证串行帧的完整性，在 CAN/CANFD 帧转换为串行帧时，帧头和通道与用户设置一致，这时的串行帧完全与串行帧转 CAN/CANFD 帧时的帧格式一致。

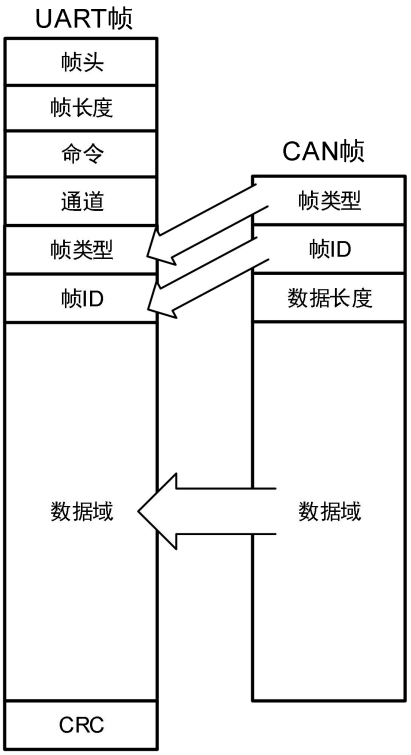

图 3.14　CAN/CANFD 帧转 UART 帧(自定义带校验协议转换）

转换实例

例 1：假设用户配置的串行帧头为 0xAA，通道 0x1。转换示例如图 3.15。

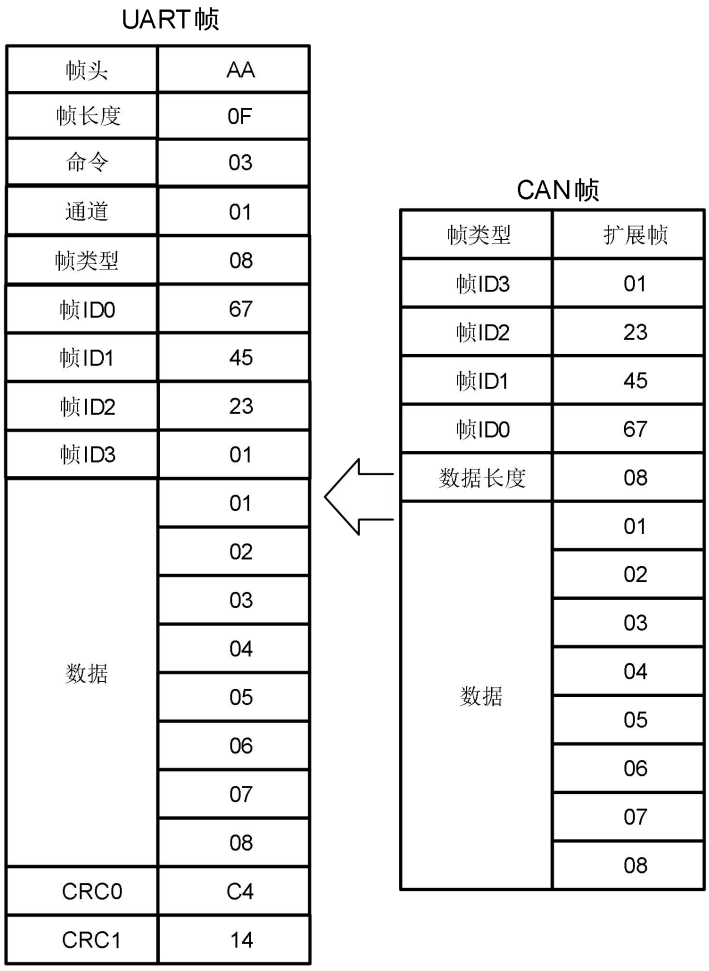

图 3.15　CAN 帧转 UART 帧示例(自定义带校验协议转换）

3.　CAN/CANFD 帧转 SPI 帧(CAN/CANFD→SPI）

CSM430B 作为 SPI 从机，无法主动控制 SPI 外设，当产品从 CAN/CANFD 总线收到一帧数据后，只能立即存储到 CANFD 接收缓冲区。当 CANFD 缓冲区的 CAN/CANFD 帧数达到反馈触发帧数，或触发时间，INT 引脚输出低电平并保持 130us 通知 SPI 主机进行数据读取。

当 CAN1 或者 CAN2 总线端接收到数据时，CSM430B 会将接收到的数据转换成 SPI 帧并放进缓存区内，待主机来读取。主机需要读取从机数据时，要进行两步操作：1.主机发送读取从机数据量的命令，CSM430B 会发送一帧数据帧，告知主机当前缓存区中有多少数据；2.主机发送读取从机数据命令，CSM430B 会将缓存区中的数据帧发送给主机。

CAN1、CAN2 接收缓冲区存储的是多个有效格式字节段的集合。CAN1 或者 CAN2 总线端接收到一个 CAN/CANFD 数据帧时，将其转换为符合用户定义格式的字节段后填充到缓冲区。为避免出现缓存区数据量超过 140 帧后无法读取第一帧数据情况，建议用户可以在读取 140 帧数据后，再接收新的数据。

SPI 主机获取 CANFD 接收缓冲区可读数据量大小后，可以一次读取出所有可读数据帧。其转换数据格式如图 3.16。

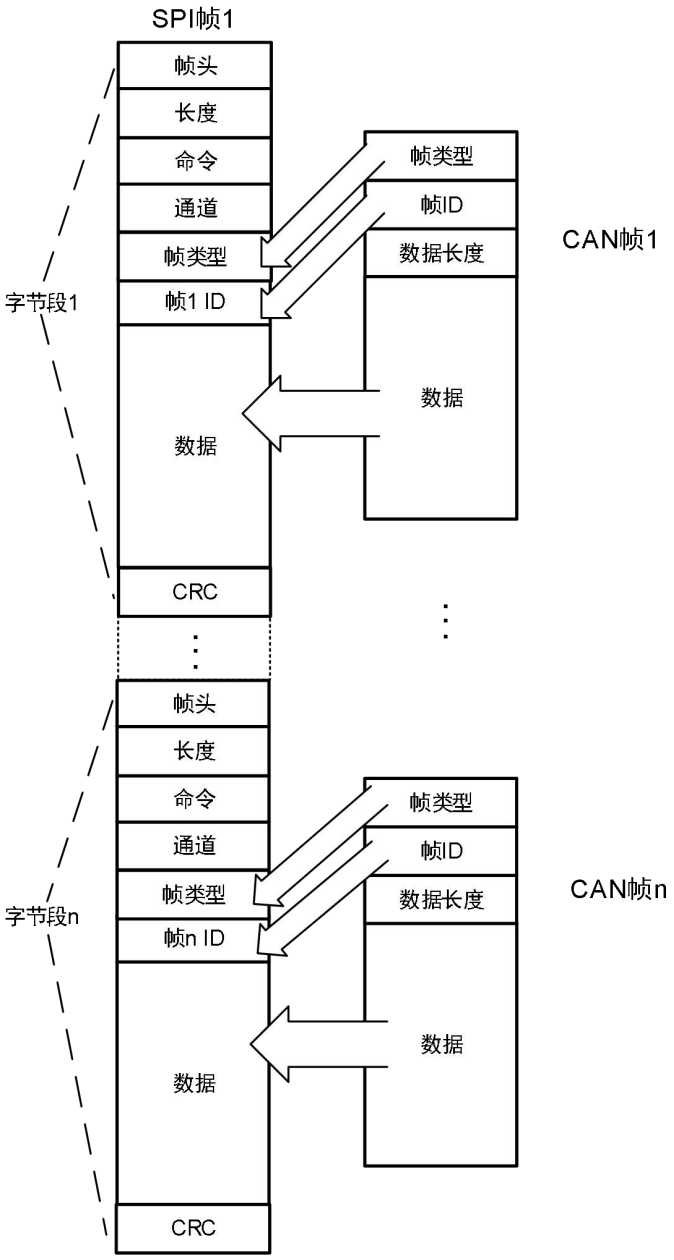

图 3.16　CAN/CANFD 帧转 SPI 帧(自定义带校验协议转换）

转换实例

例 1：假设用户配置的串行帧头为 0xAC，帧类型为 0x80，通道为 0x4。转换示例如图 3.17。

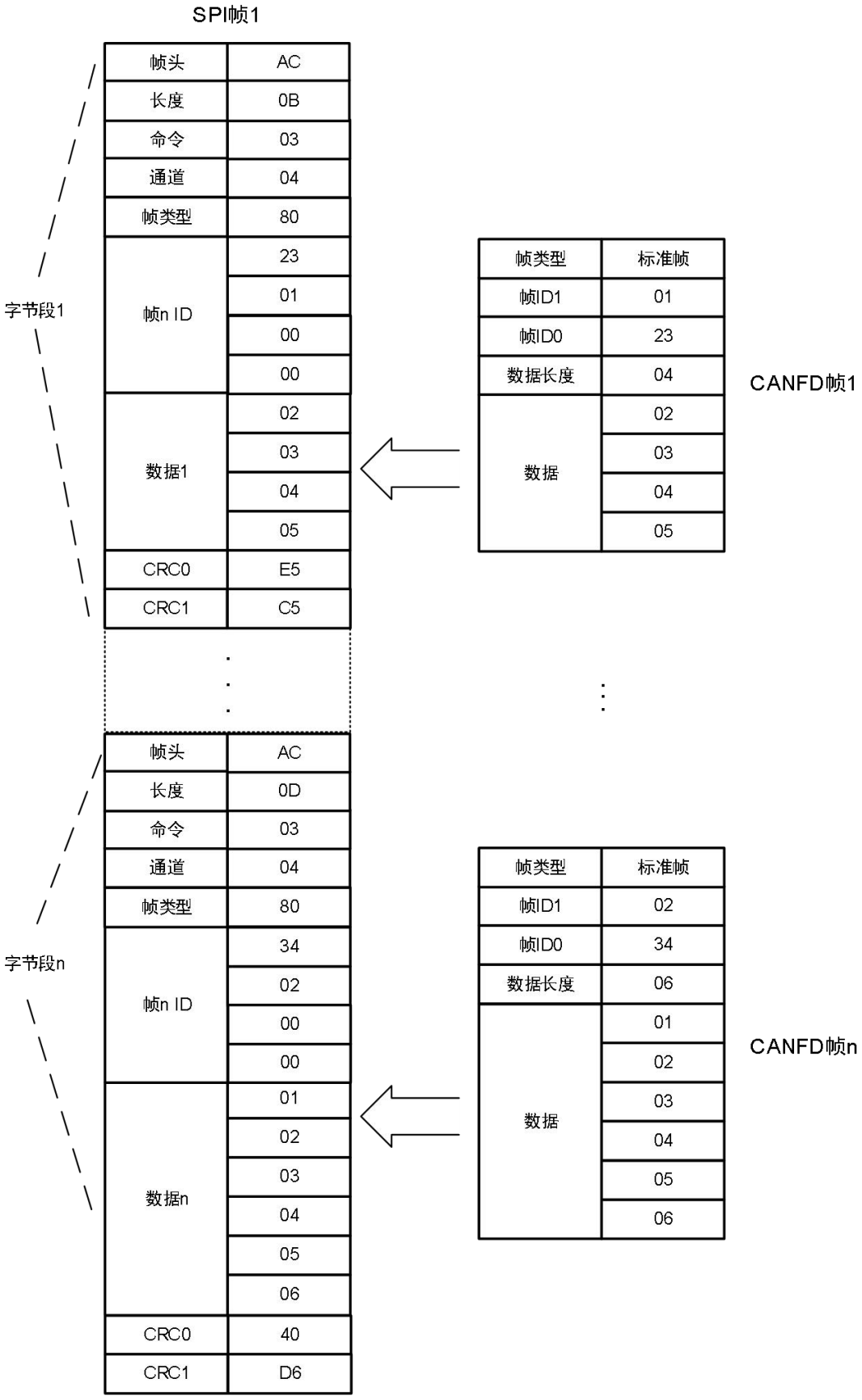

图 3.17　CANFD 帧转 SPI 帧示例(自定义带校验协议转换)

3.4 错误反馈机制

CSM430B 具备错误反馈机制，用于帮助用户了解通信过程中的错误信息及读取 CANFD 错误计数值。

错误检测范围为每个串行端(UART1、UART2 和 SPI）的长度错误、通道错误、CRC 校验错误、CAN 帧类型错误、CANFD1 计数错误和 CANFD2 计数错误。用户需要主动发送读错误命令读取到错误寄存器中的信息。

3.4.1 获取错误信息

1. 读错误命令帧格式

CSM430B 读错误命令帧格式如表 3.10 和表 3.11 所示。

表 3.10 自定义协议转换模式读错误命令帧格式

帧头	长度	命令	通道
1 字节	1 字节	1 字节	1 字节
0xAA	0x0	0x6	0xFF
0xAB			
0xAC			

表 3.11 自定义带校验协议转换模式读错误命令帧格式

帧头	长度	命令	通道	CRC 校验字
1 字节	1 字节	1 字节	1 字节	2/0 字节
0xAA	0x2	0x6	0xFF	0x5D, 0x6C
0xAB				0xE6, 0x70
0xAC	0x0			

说明：

帧头：1 字节，0xAA 通过 UART1 查询错误计数、0xAB 通过 UART2 查询错误计数、0xAC 通过 SPI 查询错误计数。

长度：1 字节，自定义转换模式下为 0x0；自定义带校验协议转换模式下为 0x2。

命令：1 字节，读错误命令 0x6。

通道：1 字节，固定为 0xFF。

校验字：自定义协议转换模式下，没有 CRC 校验字；自定义带校验协议转换模式下，使用 UART1 和 UART2 发送命令时，为 2 字节，为前面所有字节的异或；使用 SPI 发送命令时，为 0 字节，无需 CRC 校验字。

3.4.2 错误信息回应帧

CSM430B 收到主机发送的读错误命令后，将通过原端口发出通信错误信息，错误信息格式见表 3.12。

表 3.12　错误信息回应帧格式

<table>
<tr><td rowspan="2">帧头</td><td rowspan="2">长度</td><td rowspan="2">命令</td><td rowspan="2">通道</td><td colspan="5">串行端错误计数</td></tr>
<tr><td>帧长度错误计数</td><td>通道错误计数</td><td>CRC 错误计数</td><td>帧类型错误计数</td><td>帧 ID 错误计数</td></tr>
<tr><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>2 字节</td><td>1 字节</td><td>1 字节</td></tr>
<tr><td>0xAA
0xAB
0xAC</td><td>0xE/0x10</td><td>0x6</td><td>0xFF</td><td>0x0</td><td>0x0</td><td>0x0,
0x0</td><td>0x0</td><td>0x0</td></tr>
<tr><td colspan="4">CANFD1 总线错误计数</td><td colspan="4">CANFD2 总线错误计数</td><td rowspan="2">CRC 校验字</td></tr>
<tr><td>接收错误计数</td><td>发送错误计数</td><td>被动错误计数</td><td>BUSOFF 计数</td><td>接收错误计数</td><td>发送错误计数</td><td>被动错误计数</td><td>BUS OFF 计数</td></tr>
<tr><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>1 字节</td><td>0/2 字节</td></tr>
<tr><td rowspan="3">0x0</td><td rowspan="3">0x0</td><td rowspan="3">0x0</td><td rowspan="3">0x0</td><td rowspan="3">0x0</td><td rowspan="3">0x0</td><td rowspan="3">0x0</td><td rowspan="3">0x0</td><td>0x30,
0x9B</td></tr>
<tr><td>0xD7,
0x63</td></tr>
<tr><td>0x40,
0x99</td></tr>
</table>

说明：

帧头：1 字节，依次为 0xAA、0xAB 和 0xAC，分别代表通过 UART1、UART2 和 SPI 发送错误信息返回帧。

长度：1 字节，自定义协议转换时为 0xE、自定义带校验协议转换时为 0x10。

命令：读错误命令 0x6。

通道：1 字节，固定为：0xFF

帧长度错误计数：1 字节，此字节存放串行端帧长度错误信息计数值，最大累加至 FF，用户读取后清 0。

通道错误计数：1 字节，此字节存放串行端通道错误信息计数值，最大累加至 FF，用户读取后清 0。

CRC 错误计数：2 字节，此字节存放串行端错误信息计数值，最大累加至 FF FF，用户读取后清 0。

帧类型错误计数：1 字节，此字节存放串行端帧类型错误信息计数值，最大累加至 FF，用户读取后清 0。

帧 ID 错误计数：1 字节，此字节存放串行端帧 ID 错误信息计数值，最大累加至 FF，用户读取后清 0。

CANFD1 的接收错误计数：1 字节，此字节存放 CANFD 接收错误计数值，最大累加至 7F。

CANFD1 的发送错误计数：1 字节，此字节存放 CANFD 发送错误计数值，最大累加至 FF。

CANFD1 的被动错误计数：当发送错误计数值或接收错误计数值大于 127 时，被动错误计数将加 1，最大累加至 FF，用户读取后清 0。

CANFD1 的 BUSOFF 计数：当发送错误计数值大于 255 时，BUSOFF 错误计数加 1，最大累加至 FF，用户读取后清 0。

CANFD2 的接收错误计数：1 字节，此字节存放 CANFD 接收错误计数值，最大累加至 7F。

CANFD2 的发送错误计数：1 字节，此字节存放 CANFD 发送错误计数值，最大累加至 FF。

CANFD2 的被动错误计数：当发送错误计数值或接收错误计数值大于 127 时，被动错误计数将加 1，最大累加至 FF，用户读取后清 0。

CANFD2 的 BUSOFF 计数：当发送错误计数值大于 255 时，BUSOFF 错误计数加 1，最大累加至 FF，用户读取后清 0。

校验字节：自定义转换模式下，无需 CRC 校验字；自定义带校验协议转换模式下，为前面所有字节的异或。

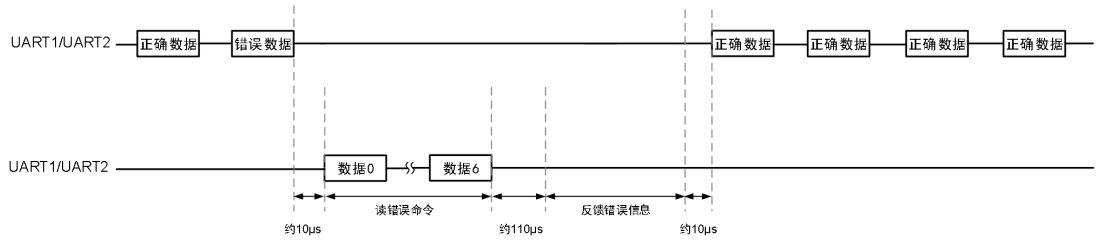

图 3.18　UART1/UART2 错误反馈机制时序图

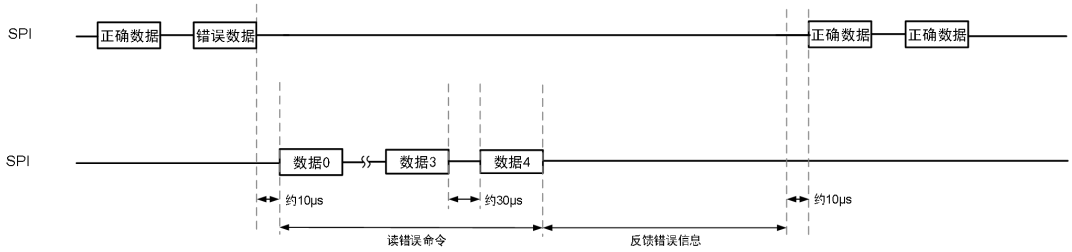

图 3.19　SPI 错误反馈机制时序图

4. 产品配置

4.1 配置参数

4.1.1 转换参数

1. 转换方式

CSM430B 的数据转换方式，指串行总线和 CANFD 总线数据之间转换的基本规则。同一时间内，产品只能工作于一种数据转换方式，若需要改变数据转换方式，则需要更改配置。产品配置好一种转换方式时，将同时作用于五种工作模式中。

数据转换方式有两种：自定义协议转换、自定义带校验协议转换。

自定义协议转换，串行帧必须符合规定的帧格式。有效的串行帧由帧头、帧长度、命令、通道、帧类型、帧 ID、数据域组成。此方式的详细说明见 3.3.1 小节。

自定义带校验协议转换，串行帧必须符合规定的帧格式。有效的串行帧由帧头、帧长度、命令、通道、帧类型、帧 ID、数据域、CRC 校验域组成。此方式的详细说明见 3.3.2 小节。

2. 转换方向

转换方向，指数据的允许转换方向。CSM430B 有三种转换方向：双向、仅 SPI/UART 转 CAN/CANFD、仅 CAN/CANFD 转 SPI/UART。

3. 允许串行帧信息转换到中 CAN/CANFD

SPI/UART 转 CAN/CANFD 时，SPI/UART 帧的帧信息同时转换至 CAN/CANFD，详见 3.3.1 小节。

4. 允许 CAN/CANFD 帧 ID 转换到串行帧中

CAN/CANFD 转 SPI/UART 时，CAN/CANFD 帧的帧 ID、帧信息同时转换至 SPI/UART，详见 3.3.1 小节。

5. 帧头

用于设置串行帧的帧开始及区分不同串行端口，详见 3.3.1 小节。

6. CRC 校验

该校验字节校验范围为帧头+帧长度+命令+通道+帧类型+帧 ID+帧数据，详见 3.3.2 小节。

7. 错误反馈机制

此功能默认是使能状态，无需用户配置也能正常使用，该功能描述详见 3.4 节。

4.1.2 SPI 参数

1. 反馈触发帧数

此配置参数仅在 SPI 转 CANFD1 模式、SPI 转 CANFD1 和 CANFD2 模式下有效。由于 CSM430B 作为 SPI 从机，无法主动向主机发送数据，故当 CSM430B 接收到一定数量的 CANFD 帧数据后，需要通过 INT 引脚通知主机获取数据。反馈触发帧数以接收到的 CANFD 帧为单位，当 CANFD 缓冲区接收到设定数量的 CANFD 帧后，触发反馈。

2. 反馈触发时间

此配置参数仅在 SPI 转 CANFD1 模式、SPI 转 CANFD1 和 CANFD2 模式有效。由于

CSM430B 作为 SPI 从机，无法主动向主机发送数据。当缓冲区接收到的 CAN/CANFD 帧数未达到反馈触发帧数，并在反馈触发时间内未被读取时，通过 INT 引脚通知主机获取数据。反馈触发时间以 100ms 为单位，当反馈触发时间到达设定值后，触发反馈。

4.1.3 UART 参数

1. 波特率

指通信串口和错误反馈机制串口的工作波特率。有效串口波特率如表 4.7。

2. 帧间隔

指 UART 通信帧之间的时间间隔，详见 3.2.4 小节 UART 帧的定义。

4.1.4 CAN/CANFD 参数

1. 波特率

指 CAN/CANFD 的工作波特率。有效 CAN/CANFD 波特率如表 4.7。

2. CANFD 接口工作模式

指 CAN 接口的工作模式，包括三种：CAN 模式、CANFD 模式和 CAN+CANFD 模式。CAN 模式下，只能收发 CAN 帧；CANFD 模式下，只能收发 CANFD 帧；CAN+CANFD 模式下可以收发 CAN/CANFD 帧。

3. 发送帧类型

指发送的 CANFD 帧类型，包括四种：CAN 标准帧、CAN 扩展帧、CANFD 标准帧和 CANFD 扩展帧。

4. 接收过滤模式

该项选择分为标准帧滤波、扩展帧滤波、标准帧和扩展帧滤波三种模式。如果仅想接收扩展帧格式的 CAN/CANFD 帧，则应该选择扩展帧滤波；如果仅想接收标准帧格式的 CAN/CANFD 帧，则应该选择标准帧滤波。如果需要接收 CAN/CANFD 标准帧和扩展帧，则应该选择标准帧、扩展帧滤波。

5. 屏蔽码

屏蔽码用来管理“验收码”，按照位管理。当屏蔽码某位值为 1 时，则该位对应的验收码会被“使能”，被“使能”的“验收码”和产品要接收的 CAN/CANFD 帧的“帧 ID”相同，该 CAN/CANFD 帧才会被接收到接收缓冲区。当“屏蔽码”的位值为 0 时，验收码不起作用，相应位的帧标识为任何值都可被接收。填充数据格式为 16 进制，每个 8 位的字节间用“空格符”隔开。

6. 验收码

验收码有验收码 0~验收码 3，共 4 组。

接收 CAN/CANFD“帧 ID”时的比较值，和“屏蔽码”按照位的关系相对应。在“屏蔽码”设定为 1 时，只有接收“帧 ID”和“验收码”相同时才会将该帧数据收到接收缓冲区中，否则不接收。填充数据格式为 16 进制，每个 8 位的字节间用“空格符”隔开。表 4.1 给出了屏蔽位、验收位过滤帧 ID 的真值关系。

7. 自定义采样点

该项选择用来管理 CAN1 端口和 CAN2 端口的采样点。在默认配置中，CAN1 和 CAN2 端口的仲裁域在所有可配置波特率采样点均为 80%。数据域除了波特率为 5Mbps 时采样点

为 75%，其他可配置波特率下的采样点均为 80%。通过配置对应的寄存器，可以改变 CANFD 接口时钟分频值、仲裁域预分频值、仲裁域 TSEG1 值、仲裁域 TSEG2 值、数据域预分频值、数据域 TSEG1 值和数据域 TSEG2 值，进而改变 CAN/CANFD 接口的采样点。用户可以通过 CSMCfgTools 配置软件进行计算，得出对应寄存器的配置参数后再进行配置。

表 4.1　滤波、屏蔽码真值表

屏蔽位	验收位	帧 ID	接收或拒绝位
0	X	X	接收
1	0	0	接收
1	0	1	拒绝
1	1	0	拒绝
1	1	1	接收

4.2　出厂默认配置

表 4.2　出厂默认配置参数

参数	默认值	说明	备注
数据转换方式	00 00 00 01	自定义带校验协议转换	--
工作模式	00 00 00 03	UART1 和 UART2 转 CANFD1 和 CANFD2	--
UART1 波特率	115200 bps	通信串口波特率	--
UART1 数据位	8	串口数据位，固定为 8	无法更改
UART1 停止位	1	串口停止位，固定为 1	
UART1 校验位	0	串口校验位，固定为 0	
UART2 波特率	115200 bps	通信串口波特率	--
UART2 数据位	8	串口数据位，固定为 8	无法更改
UART2 停止位	1	串口停止位，固定为 1	
UART2 校验位	0	串口校验位，固定为 0	
CAN1 波特率	1Mbps(仲裁域） 5Mbps(仲裁域）	CAN1 接口工作波特率	--
数据域加速	使能	CAN1 数据域波特率加速	--
CAN1 工作模式	CAN+CANFD 模式	CAN1 接口工作模式	--
接收过滤模式	标准帧和扩展帧滤波	CAN1 接口验收滤波方式选择	--
采样点自定义	禁能	仲裁域 1Mbps，采样点 80% 仲裁域 5Mbps，采样点 75%	--
CAN1 标准帧屏蔽码	00 00 00 00	CAN1 标准帧屏蔽码	--
CAN1 标准帧验收码 0	00 00 00 00	CAN1 标准帧验收码 0	--
CAN1 标准帧验收码 1	00 00 00 00	CAN1 标准帧验收码 1	--

续上表

参数	默认值	说明	备注
CAN1 标准帧验收码 2	00 00 00 00	CAN1 标准帧验收码 2	--
CAN1 标准帧验收码 3	00 00 00 00	CAN1 标准帧验收码 3	--
CAN1 扩展帧屏蔽码	00 00 00 00	CAN1 扩展帧屏蔽码	--
CAN1 扩展帧验收码 0	00 00 00 00	CAN1 扩展帧验收码 0	--
CAN1 扩展帧验收码 1	00 00 00 00	CAN1 扩展帧验收码 1	--
CAN1 扩展帧验收码 2	00 00 00 00	CAN1 扩展帧验收码 2	--
CAN1 扩展帧验收码 3	00 00 00 00	CAN1 扩展帧验收码 3	--
CAN2 波特率	1Mbps(仲裁域） 5Mbps(仲裁域）	CAN2 接口工作波特率	--
数据域加速	使能	CANFD2 数据域波特率加速	--
CAN2 工作模式	CAN+CANFD 模式	CAN2 接口工作模式	--
接收过滤模式	标准帧、扩展帧 滤波	CAN2 接口验收滤波方式选择	--
自定义采样点	禁能	仲裁域 1Mbps，采样点 80% 仲裁域 5Mbps，采样点 75%	--
CAN2 标准帧屏蔽码	00 00 00 00	CAN2 标准帧屏蔽码	--
CAN2 标准帧验收码 0	00 00 00 00	CA2 标准帧验收码 0	--
CAN2 标准帧验收码 1	00 00 00 00	CAN2 标准帧验收码 1	--
CAN2 标准帧验收码 2	00 00 00 00	CAN2 标准帧验收码 2	--
CAN2 标准帧验收码 3	00 00 00 00	CAN2 标准帧验收码 3	--
CAN2 扩展帧屏蔽码	00 00 00 00	CAN2 扩展帧屏蔽码	--
CAN2 扩展帧验收码 0	00 00 00 00	CAN2 扩展帧验收码 0	--
CAN2 扩展帧验收码 1	00 00 00 00	CAN2 扩展帧验收码 1	--
CAN2 扩展帧验收码 2	00 00 00 00	CAN2 扩展帧验收码 2	--

续上表

参数	默认值	说明	备注
CAN2 扩展帧验收码 3	00 00 00 00	CAN2 扩展帧验收码 3	--
反馈触发帧数	10 帧	CAN 缓冲区反馈触发帧数	只适用于 SPI 转 CANFD1 模式、 SPI 转 CANFD1 和 CANFD2 模式
反馈触发时间	1000ms	CAN 缓冲区反馈触发时间	
数据位长度	8 位	SPI 数据长度固定为 8 位	无法更改
位传输方式	MSB	位传输按高位优先	
CPOL	1	SPI 工作模式参数 CPOL(固定为 1）	
CPHA	1	SPI 工作模式参数 CPHA(固定为 1）	

4.3 配置通信协议

对 CSM430B 进行配置，首先需要将产品复位并进入“SPI 配置模式”或“UART1 配置模式”。用户向 CSM430B 的 SPI 接口或 UART1 接口发送相应命令帧，产品接收到配置命令后，对自身进行操作，并返回回应帧。

需要注意的是，在“UART1 配置模式”下，产品接收到数据，处理完毕后会主动向 UART1 返回回应帧。但产品处于“SPI 配置模式”时，CSM430B 作为从机无法主动返回回应帧，因此，SPI 主机发送命令帧，CSM430B 在正确接收并处理完成后，会将 INT 引脚置低，通知主机处理完成，可以读取回应帧，等待时间如表 4.3。

表 4.3 命令帧处理等待时间

命令帧类型	等待处理时间
写配置	3ms
读配置	200us

1. 命令帧

命令帧由主控制端(上位机、MCU 等）向被控制端 CSM430B 发送，CSM430B 接收到命令帧后进行相应操作。

2. 回应帧

回应帧指 CSM430B 接收到命令帧后，由 CSM430B 返回的，对主控制端的回应信息。

4.3.1 写配置参数

1. 写配置命令帧

写配置命令帧，用于对 CSM430B 的参数进行配置，其包含配置 CSM430B 所需要的所有数据。写配置命令帧的格式如表 4.4。

表 4.4　写配置命令帧格式

帧头	长度	命令	通道	寄存器地址	寄存器值	校验字
1 字节	1 字节	1 字节	1 字节	1 字节	4 字节	2 字节
0xAA, 0xAC	0x7	0x2	0xFF	定义如表 4.5	定义如表 4.6	前面所有字节 均参与校验

说明：

帧头：1 字节，依次为 0xAA 或者 0xAC，只能通过 UART1 或者 SPI 进行配置。

长度：1 字节，固定为 0x7。

命令：1 字节。固定为 0x2。

通道：1 字节，固定为 0xFF。

寄存器地址：1 字节，根据寄存器表选择对应需要配置的寄存器，如表 4.5。

寄存器值：4 字节，每个寄存器对应的配置信息，如表 4.6。

校验字：2 字节，为前面所有字节均参与校验，校验采用 CRC-16-X25 算法。

表 4.5　寄存器地址及对应说明

地址	位	说明	默认
0x0	[31:0]	转换方式寄存器	0x1
0x1	[31:0]	转换模式寄存器	0x3
0x2	[31:0]	UART1 配置寄存器	0xA
0x3	[31:0]	UART2 配置寄存器	0xA
0x4	[31:0]	SPI 配置寄存器	0x909
0x5	[31:0]	CAN1 配置寄存器	0x1A060A
0x6	[31:0]	CAN2 配置寄存器	0x1A060A
0x7	[31:0]	CAN1 标准帧屏蔽码配置	0x0
0x8	[31:0]	CAN1 标准帧验收码 0 配置	0x0
0x9	[31:0]	CAN1 标准帧验收码 1 配置	0x0
0xA	[31:0]	CAN1 标准帧验收码 2 配置	0x0
0xB	[31:0]	CAN1 标准帧验收码 3 配置	0x0
0xC	[31:0]	CAN1 扩展帧屏蔽码配置	0x0
0xD	[31:0]	CAN1 扩展帧验收码 0 配置	0x0
0xE	[31:0]	CAN1 扩展帧验收码 1 配置	0x0
0xF	[31:0]	CAN1 扩展帧验收码 2 配置	0x0
0x10	[31:0]	CAN1 扩展帧验收码 3 配置	0x0
0x11	[31:0]	CAN2 标准帧验屏蔽码配置	0x0
0x12	[31:0]	CAN2 标准帧验验收码 0 配置	0x0
0x13	[31:0]	CAN2 标准帧验验收码 1 配置	0x0
0x14	[31:0]	CAN2 标准帧验验收码 2 配置	0x0
0x15	[31:0]	CAN2 标准帧验验收码 3 配置	0x0
0x16	[31:0]	CAN2 扩展帧屏蔽码配置	0x0
0x17	[31:0]	CAN2 扩展帧验收码 0 配置	0x0

续上表

地址	位	说明	默认
0x18	[31:0]	CAN2 扩展帧验收码 1 配置	0x0
0x19	[31:0]	CAN2 扩展帧验收码 2 配置	0x0
0x1A	[31:0]	CAN2 扩展帧验收码 3 配置	0x0
0x1B	[31:0]	CANFD 时钟分频选择	0x40004
0x1C	[31:0]	CANFD 仲裁域预分配配置	0x10001
0x1D	[31:0]	CANFD 仲裁域 SEG1 配置	0x200020
0x1E	[31:0]	CANFD 仲裁域 SEG2 配置	0x80008
x1F	[31:0]	CANFD 数据域预分配配置	0x10001
0x20	[31:0]	CANFD 数据域 SEG1 配置	0x60006
0x21	[31:0]	CANFD 数据域 SEG2 配置	0x20002
0x22	[31:0]	IC 固件版本	0xC185001

表 4.6 每个寄存器对应配置值

寄存器地址	位	数值范围	说明
0x0	[1:0]	00~01	00：自定义协议转换； 01: 自定义带校验协议转换；
	[31:2]	0x0	保留位默认为 0
0x1	0	0~1	0：使用 CANFD1 接口 1：使用 CANFD1 和 CANFD2 接口
	[2:1]	00~11	00: 使用 UART1 接口； 01: 使用 UART1 和 UART2 接口； 10: 使用 SPI 接口；
	[31:3]	0x0	保留位默认为 0
0x2	[7:0]	0x0~0x18	UART1 波特率代码如表 4.7
	8	0	数据位固定为 8 位
	9	0	停止位固定为 1 位
	10	0	校验位默认为 0
	[31:11]	0x0	保留位默认为 0
0x3	[7:0]	0	UART2 波特率代码如表 4.7
	8	0x0~0x18	数据位固定为 8 位
	9	0	停止位固定为 1 位
	10	0	校验位默认为 0
	[31:11]	0x0	保留位默认为 0
0x4	[7:0]	0x0~0x63	SPI 反馈触发帧数代码如表 4.8
	[15:8]	0x0~0xFF	SPI 反馈触发时间代码如表 4.8
	16	0	SPI 传输方式：CPOL1 CPHA1
	17	0	位传输方式：MSB
	18	0	数据位长度默认为 8

续上表

寄存器地址	位	数值范围	说明
	[31:19]	0x0	保留位默认为 0
0x5	[7:0]	0x0~0xA	CAN1 数据域/CANFD1 仲裁域 波特率代码如表 4.7
	[15:8]	0x0~0x6	CANFD1 数据域波特率代码如表 4.7
	[17:16]	00~11	CAN1 模式： 00：CAN 模式 01：CAN FD 模式 10：CAN + CAN FD 模式 11：无功能
	[19:18]	00~11	CAN1 滤波模式： 00：标准帧滤波； 01：扩展帧滤波 10: 标准帧、扩展帧滤波 11：无功能
	20	0~1	CANFD1 数据域是否加速 0：不加速； 1：加速
	21	0~1	0：不使能自定义采样点功能 1：使能自定义采样点功能
	[31:22]	0x0	保留位默认为 0
0x6	[7:0]	0x0~0xA	CAN2 数据域/CANFD2 仲裁域 波特率代码表 4.7
	[15:8]	0x0~0x6	CANFD2 数据域波特率代码表 4.7
	[17:16]	00~11	CANFD2 模式： 00：CAN 模式 01：CAN FD 模式 10：CAN + CAN FD 模式 11：无功能
	[19:18]	00~11	CAN2 滤波模式： 00：标准帧滤波； 01：扩展帧滤波 10: 标准帧、扩展帧滤波 11：无功能
	20	0~1	CANFD2 数据域是否加速 0：不加速； 1：加速
	21	0~1	0：不使能自定义采样点功能 1：使能自定义采样点功能
	[31:22]	0x0	保留位默认为 0

续上表

寄存器地址	位	数值范围	说明
0x7	[10:0]	0x0~0x7FF	CAN1 标准帧屏蔽码
	[31:11]	0x0	保留位默认为 0
0x8	[10:0]	0x0~0x7FF	CAN1 标准帧验收码 0
	[31:11]	0x0	保留位默认为 0
0x9	[10:0]	0x0~0x7FF	CAN1 标准帧验收码 1
	[31:11]	0x0	保留位默认为 0
0xA	[10:0]	0x0~0x7FF	CAN1 标准帧验收码 2
	[31:11]	0x0	保留位默认为 0
0xB	[10:0]	0x0~0x7FF	CAN1 标准帧验收码 3
	[31:11]	0x0	保留位默认为 0
0xC	[28:0]	0x0~0x1FFFFFFF	CAN1 扩展帧屏蔽码
	[31:29]	0	保留位默认为 0
0xD	[28:0]	0x0~0x1FFFFFFF	CAN1 扩展帧验收码 0
	[31:29]	0	保留位默认为 0
0xE	[28:0]	0x0~0x1FFFFFFF	CAN1 扩展帧验收码 1
	[31:29]	0	保留位默认为 0
0xF	[28:0]	0x0~0x1FFFFFFF	CANFD1 扩展帧验收码 2
	[31:29]	0	保留位默认为 0
0x10	[28:0]	0x0~0x1FFFFFFF	CAN1 扩展帧验收码 3
	[31:29]	0	保留位默认为 0
0x11	[10:0]	0x0~0x7FF	CAN2 标准帧屏蔽码
	[31:11]	0x0	保留位默认为 0
0x12	[10:0]	0x0~0x7FF	CAN2 标准帧验收码 0
	[31:11]	0x0	保留位默认为 0
0x13	[10:0]	0x0~0x7FF	CAN2 标准帧验收码 1
	[31:11]	0x0	保留位默认为 0
0x14	[10:0]	0x0~0x7FF	CAN2 标准帧验收码 2
	[31:11]	0x0	保留位默认为 0
0x15	[10:0]	0x0~0x7FF	CAN2 标准帧验收码 3
	[31:11]	0x0	保留位默认为 0
0x16	[28:0]	0x0~0x1FFFFFFF	CAN2 扩展帧屏蔽码
	[31:29]	0	保留位默认为 0
0x17	[28:0]	0x0~0x1FFFFFFF	CAN2 扩展帧验收码 0
	[31:29]	0	保留位默认为 0
0x18	[28:0]	0x0~0x1FFFFFFF	CAN2 扩展帧验收码 1
	[31:29]	0	保留位默认为 0
0x19	[28:0]	0x0~0x1FFFFFFF	CAN1 扩展帧验收码 2
	[31:29]	0	保留位默认为 0

续上表

寄存器地址	位	数值范围	说明
0x1A	[28:0]	0x0~0x1FFFFFFF	CAN1 扩展帧验收码 3
	[31:29]	0	保留位默认为 0
0x1B	[15:0]	0x0~0x6	CAN1 时钟分频如表 4.9
	[31:16]	0x0~0x6	CAN2 时钟分频如表 4.9
0x1C	[15:0]	0x1~0x200	CAN1 数据域/CANFD1 仲裁域 预分频值如表 4.10
	[31:16]	0x1~0x200	CAN2 数据域/CANFD2 仲裁域 预分频值如表 4.10
0x1D	[15:0]	0x3~0x101	CAN1 数据域/CANFD1 仲裁域 TSEG1 值如表 4.11
	[31:16]	0x3~0x101	CAN2 数据域/CANFD2 仲裁域 TSEG1 值如表 4.11
0x1E	[15:0]	0x2~0x80	CAN1 数据域/CANFD1 仲裁域 TSEG2 值如表 4.12
	[31:16]	0x2~0x80	CAN2 数据域/CANFD2 仲裁域 TSEG2 值如表 4.12
0x1F	[15:0]	0x1~0x2	CANFD1 数据域预分频值如表 4.13
	[31:16]	0x1~0x2	CANFD2 数据域预分频值如表 4.13
0x20	[15:0]	0x2~0x21	CANFD1 数据域 TSEG1 值如表 4.14
	[31:16]	0x2~0x21	CANFD2 数据域 TSEG1 值如表 4.14
0x21	[15:0]	0x1~0x10	CANFD1 数据域 TSEG2 值如表 4.15
	[31:16]	0x1~0x10	CANFD2 数据域 TSEG2 值如表 4.15
0x22	[7:0]	0x0~0x9	固件主版本号
	[11:8]	0x0~0xF	固件副版本号 0
	[15:12]	0x0~0xF	固件副版本号 1
	[23:16]	0x18~0x21	固件发布日期年份
	[31:24]	0x1~0xC	固件发布日期月份

表 4.7　串口和 CANFD 波特率代码表

16 进制代码	串口波特率(bps)	CANFD 仲裁域波特率(bps)	CANFD 数据域波特率(bps)
0x0	300	40k	400k
0x1	600	50k	500k
0x2	1200	80k	800k
0x3	2400	100k	1M
0x4	4800	125k	2M
0x5	9600	200k	4M
0x6	14400	250k	5M
0x7	19200	400k	--
0x8	38400	500k	--
0x9	57600	800k	--

续上表

16 进制代码	串口波特率(bps)	CANFD 仲裁域波特率(bps)	CANFD 数据域波特率(bps)
0xA	115200	1M	--
0xB	128000	--	--
0xC	256000	--	--
0xD	460800	--	--
0xE	921600	--	--
0xF	1M	--	--
0x10	1.5M	--	--
0x11	2M	--	--
0x12	2.5M	--	--
0x13	3M	--	--
0x14	3.5M	--	--
0x15	4M	--	--
0x16	5M	--	--
0x17	6M	--	--
0x18	7M	--	--

表 4.8　SPI 反馈触发帧数和反馈触发时间代码表

16 进制代码	SPI 反馈触发帧数	SPI 反馈触发时间 (*100ms)
0x0	1	1
0x1	2	2
0x2	3	3
0x3	4	4
0x4	5	5
0x5	6	6
0x6	7	7
0x7	8	8
0x8	9	9
0x9	10	10
0xA	11	11
0xB	12	12
0xC	13	13
0xD	14	14
0xE	15	15
0xF	16	16

续上表

16 进制代码	SPI 反馈触发帧数配置	SPI 反馈触发时间 (*100ms）配置
0x10	17	17
0x11	18	18
0x12	19	19
...	...	...
0x63	100	100
...	--	...
0xFF	--	256

表 4.9　CANFD1 和 CANFD2 时钟分频代码表

16 进制代码	时钟分频值
0x0	2
0x1	3
0x2	4
0x3	5
0x4	6
0x5	7
0x6	8

表 4.10　CANFD1 和 CANFD2 仲裁域预分频值代码表

16 进制代码	仲裁域预分频值(BRP)
0x1	1
0x2	2
0x3	3
0x4	4
0x5	5
0x6	6
0x7	7
...	...
0x200	512

表 4.11　CANFD1 和 CANFD2 仲裁域 TSEG1 值代码表

16 进制代码	TSEG1 值
0x3	3
0x4	4
0x5	5
0x6	6
0x7	7
0x8	8
0x9	9
...	...
0x101	257

表 4.12　CANFD1 和 CANFD2 仲裁域 TSEG2 值代码表

16 进制代码	TSEG2 值
0x2	2
0x3	3
0x4	4
0x5	5
0x6	6
0x7	7
0x7	8
...	...
0x80	128

表 4.13　CANFD1 和 CANFD2 数据域预分频值代码表

16 进制代码	数据域预分频值(BRP)
0x1	1
0x2	2

表 4.14　CANFD1 和 CANFD2 数据域 TSEG1 值代码表

16 进制代码	TSEG1 值
0x2	2
0x3	3
0x4	4
0x5	5
0x6	6
0x7	7
0x8	8
...	...
0x21	33

表 4.15　CANFD1 和 CANFD2 数据域 TSEG2 值代码表

16 进制代码	TSEG2 值
0x1	1
0x2	2
0x3	3
0x4	4
0x5	5
0x6	6
0x7	7
...	...
0x10	16

2.　写配置回应帧

CSM430B 在收到写配置参数命令帧后，会依据收到的数据对当前配置进行更新，操作完成后，无论配置更新成功与否，都会对主控制端发送回应帧进行应答。写配置回应帧的格式如表 4.16。

表 4.16　写配置参数回应帧格式

帧头	长度	命令	通道	数据域	校验字
1 字节	1 字节	1 字节	1 字节	1 字节	2 字节
0xAA 0xAC	0x3	0x2	0xFF	如表 4.17	前面所有字节的异或

说明：

帧头：1 字节，依次为 0xAA，0xAC，分别代表使用 UART1 和 SPI 进行配置。

长度：1 字节，数据域和校验字段总字节数，即 0x3。

命令：1 字节，0x2 表示配置模式下写寄存器。

通道：1 字节，固定为 0xFF。

数据域：1 字节,具体参考下表 4.17。

校验字：2 字节，前面所有字节的异或。

表 4.17　数据域命令符代码及定义说明

16 进制代码	说明
0x80	配置正确
0x81	寄存器地址错误
0x82	寄存器数据错误
0x83	CRC 校验错误
0x84	长度错误
0x85	命令错误

4.3.2 读配置参数

1. 读单个寄存器配置参数命令帧

读配置参数命令帧，可用于获取 CSM430B 当前寄存器配置参数。读配置参数命令帧的格式如表 4.18。

表 4.18 读单个配置参数命令帧格式

帧头	长度	命令	通道	数据域	校验字
1 字节	1 字节	1 字节	1 字节	1 字节	2 字节
0xAA 0xAC	0x3	0x1	0xFF	寄存器地址如表 4.5	前面所有字节的异或

说明：

帧头：1 字节，依次为 0xAA，0xAC，分别代表使用 UART1 和 SPI 进行配置。

长度：数据域和校验字段字节数总和，即 0x3。

命令：1 字节，0x1 表示配置模式下读寄存器配置参数。

通道：1 字节，固定为 0xFF。

数据域：1 字节，寄存器地址，具体参考表 4.5。

校验字：2 字节，前面所有字节的异或。

2. 读单个寄存器配置参数回应帧

CSM430B 在收到读单个寄存器配置参数命令帧后，会获取自身当前该寄存器配置参数信息，并通过回应帧返回到主控制端。读配置参数回应帧的格式如表 4.19。

表 4.19 读单个寄存器配置参数回应帧格式

帧头	长度	命令	通道	数据域	校验字
1 字节	1 字节	1 字节	1 字节	4 字节	2 字节
0xAA 0xAC	0x6	0x1	0xFF	寄存器配置参数值如表 4.6	前面所有字节的异或

说明：

帧头：1 字节，依次为 0xAA，0xAC，分别代表使用 UART1 和 SPI 进行配置。

长度：数据域和校验字段字节数总和，即 0x6。

命令：1 字节，0x1 表示配置模式下读寄存器配置参数。

通道：1 字节，固定为 0xFF。

数据域：4 字节，寄存器配置参数值，具体参考表 4.6。

校验字：2 字节，前面所有字节的异或。

3. 读全部寄存器配置参数命令帧

读全部寄存器配置参数命令帧，可用于获取 CSM430B 当前所有寄存器配置参数。读全部配置参数命令帧的格式如表 4.20。

表 4.20　读全部寄存器配置参数命令帧格式

帧头	长度	命令	通道	校验字
1 字节	1 字节	1 字节	1 字节	2 字节
0xAA 0xAC	0x2	0x5	0xFF	前面所有字节的异或

说明：

帧头：1 字节，依次为 0xAA，0xAC，分别代表使用 UART1 和 SPI 进行配置。

长度：校验字段字节数总和，即 0x2。

命令：1 字节，0x5 表示配置模式下读全部寄存器配置参数。

通道：1 字节，固定为 0xFF。

校验字：2 字节，前面所有字节的异或。

4.　读全部寄存器配置参数回应帧

CSM430B 在收到读全部寄存器配置参数命令帧后，会获取自身当前所有寄存器配置参数信息，并通过回应帧返回到主控制端。读配置参数回应帧的格式如表 4.21。

表 4.21　读全部寄存器配置参数回应帧格式

帧头	长度	命令	通道	数据域	校验字
1 字节	1 字节	1 字节	1 字节	140 字节	2 字节
0xAA 0xAC	0x8E	0x5	0xFF	全部寄存器配置参数	前面所有字节的异或

说明：

帧头：1 字节，依次为 0xAA，0xAC，分别代表使用 UART1 和 SPI 进行配置。

长度：数据域和校验字段总字节数，即 0x8E。

命令：1 字节，0x5 表示配置模式下读全部寄存器配置参数。

通道：1 字节，固定为 0xFF。

数据域：140 字节，从 0x0 寄存器开始到 0x22 寄存器，每四个字节代表一个寄存器的配置参数。

校验字：2 字节，前面所有字节的异或。

4.4 配置方式

4.4.1 MCU 配置方式

1. 通过 SPI 配置

当用户使用 MCU 的 SPI 接口与 CSM430B 进行连接时，可以通过 SPI 接口对产品参数进行配置，硬件连接图可参考图 2.3。

配置方法如下：

(1) 上电后各引脚电平可以保持稳定，此时 CFG 置低至少 50us，控制 RST 复位至少 100us，等待至少 200ms，产品进入 SPI 配置模式（复位过程中，INT 引脚会自行置低并保持，复位结束后，INT 引脚会持续约 110ms 后自行置高）；

(2) SSEL 置低，MCU 发送写配置命令帧，然后 SSEL 置高，等待至少 3ms，或检测到 INT

(3) 引脚至低约 130us；

(4) SSEL 置低，主机发送 7 个字节无效数据，接收产品写配置回应帧，然后 SSEL 置高，延时 50us；

(5) MCU 确认写配置回应帧内容，判断配置是否成功；

(6) 配置完成后，CFG 引脚置高等待至少 200ms，产品进入 SPI 转 CANFD 模式。

通过 SPI 写配置命令的时序图如图 4.1，此时序同样适用于读配置命令操作，只需要更改 SPI 读写数据，以及写配置命令后等待时间。

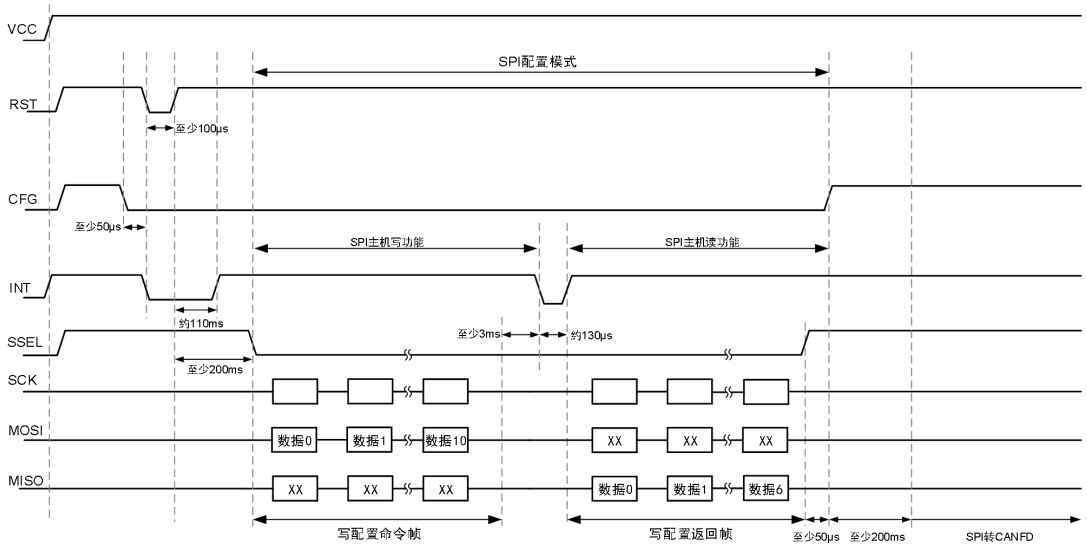

图 4.1 通过 SPI 写配置命令的时序图

2. 通过 UART 配置

当用户使用 MCU 的 UART 接口与 CSM430B 进行连接时，可以通过 UART 接口对产品参数进行配置，硬件连接图可参考图 2.5。其中，配置时的串口波特率固定为 9600bps，无法更改。若使用波特率不正确，则无法配置成功。

配置方法如下：

(1) 上电后各引脚电平可以保持稳定，此时 CFG 置低，等待至少 50us 后，RST 低电平保持至少 100us 后，等待至少 200ms，产品进入 UART 配置模式；

(2) MCU 发送写配置命令帧(产品 RXD 接收数据）；

(3) 产品收到配置命令帧后，主动发送写配置回应帧至 MCU(产品 TXD 发送数据）；

(4) MCU 确认写配置回应帧内容，判断配置是否成功；

(5) 配置完成至少等待 50us 后，CFG 引脚置高等待至少 200ms，产品进入 UART 转 CANFD 模式。

通过 UART 写配置命令的时序图如图 4.2，此时序同样适用于读配置命令操作，只需要更改 MCU 发送的命令帧。

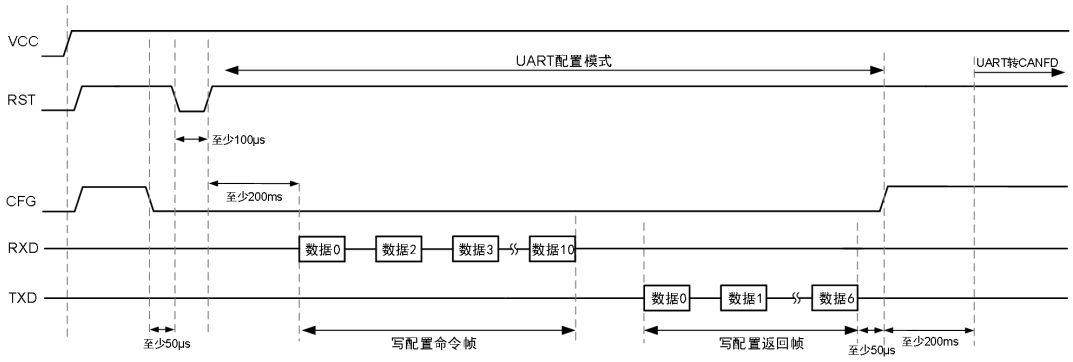

图 4.2 通过 UART 写配置命令时序图

4.4.2 上位机配置方式

除了用 MCU 对 CSM430B 进行配置的方式外，用户也可以先使用上位机配置方式对产品进行，配置完成后再使用。通过上位机进行配置，需要使用到 CSMCfgTools 配置软件、CSM430B-Eval 评估板，详细配置使用说明请参考 CSM430B-Eval 评估板说明文档《CSM 系列辅助开发工具手册》。

5. SPI 转 CANFD 在 Linux 系统中的使用方法

本章以 RK3568 为例，在 Linux 系统环境下如何适配及使用 CSM430B 的 SPI 驱动。

5.1 设备树配置

设备树配置如下：

```
&spi0 {
        pinctrl-names = "default";
        max-freq = <48000000>;
        pinctrl-0 = <&spi0m1_cs0 &spi0m1_pins>;
        status = "okay";

        csm400b_spi: csm400b-spi@0{
                compatible = "zhiyuan,csm400b-spi";
                reg = <0>;
                gpios = <&gpio3 21 0                    /* for rst */
                        &gpio3 20 0>;                   /* for cfg */
                interrupt-parent = <&gpio3>;
                interrupts = <RK_PC3 IRQ_TYPE_LEVEL_LOW>;      /* for interrupt GPIO */
                spi-max-frequency = <3000000>;
                gid = <0>;          /* if gid set 0 , use two can*/
                status = "okay";
        };
};
&pinctrl {
        can-uart{
                canrst{
                        rockchip,pins = <3 RK_PC5 RK_FUNC_GPIO &pcfg_pull_down>;
                };
                cancfg{
                        rockchip,pins = <3 RK_PC4 RK_FUNC_GPIO &pcfg_pull_down>;
                };
                canint{
                        rockchip,pins = <3 RK_PC3 RK_FUNC_GPIO &pcfg_pull_down>;
                };
        };
};
```

将 gid 设置为 0 时，驱动设置为 SPI 转 CANFD1 和 CANFD2 模式；gid 设置为 1 时，驱动则设置为 SPI 转 CANFD1 模式。

5.2 驱动移植

将 csm400b_spi.c、csm400b_spi.h 拷贝到 drivers/net/can 下，修改 drivers/net/can 下的 Makefile，添加以下内容：

```
obj-$(CONFIG_CAN_CSM400B_SPI)                    += csm400b_spi.o
```

修改修改 drivers/net/can 下的 Kconfig，添加以下内容：

```
config CAN_CSM400B_SPI
    tristate "zlg csm400b to can device"
    help
    Support for ZLG SPI to CAN interfaces.
    This is a CAN device driver which registers as an spi adapter
    and gpiochip to expose these functions of the CSM400B.
```

修改完成后在 menuconfig 开启或者在 defconfig 设置即可，本文直接在 defconfig 设置，如下：

```
CONFIG_CAN_CSM400B_SPI = m
```

设置完后重新编译即可。

5.3 驱动使用

输入以下命令加载驱动：

```
insmod csm400b_spi.ko
```

加载完后 demesg 出现 spi check ok 则说明驱动正常，如果异常检测硬件连接状态及设备树配置。

```
[root@M3568:~/output]# insmod csm400b_spi.ko
[    694.000644] csm400b-spi spi0.0: max_speed_hz = 3000000.
[    694.001053] csm400b-spi spi0.0: GID = 0, spi checking...
[    694.630512] csm400b-spi spi0.0 can0: GID = 0, spi check ok.
使用 ip link show 可以看到两个 can 设备
[root@M3568:~]# ip link show
1: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
link/can
2: can1: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
link/can
```

使用 ip 工具设置 can：如下：

```
ip link set can0 up type can bitrate 1000000 dbitrate 5000000 fd on
    ip link set can1 up type can bitrate 1000000 dbitrate 5000000 fd on
```

设置完后即可进行收发测试，使用 cansend 进行数据发送, candump 进行数据接收，如图 5.1 所示。

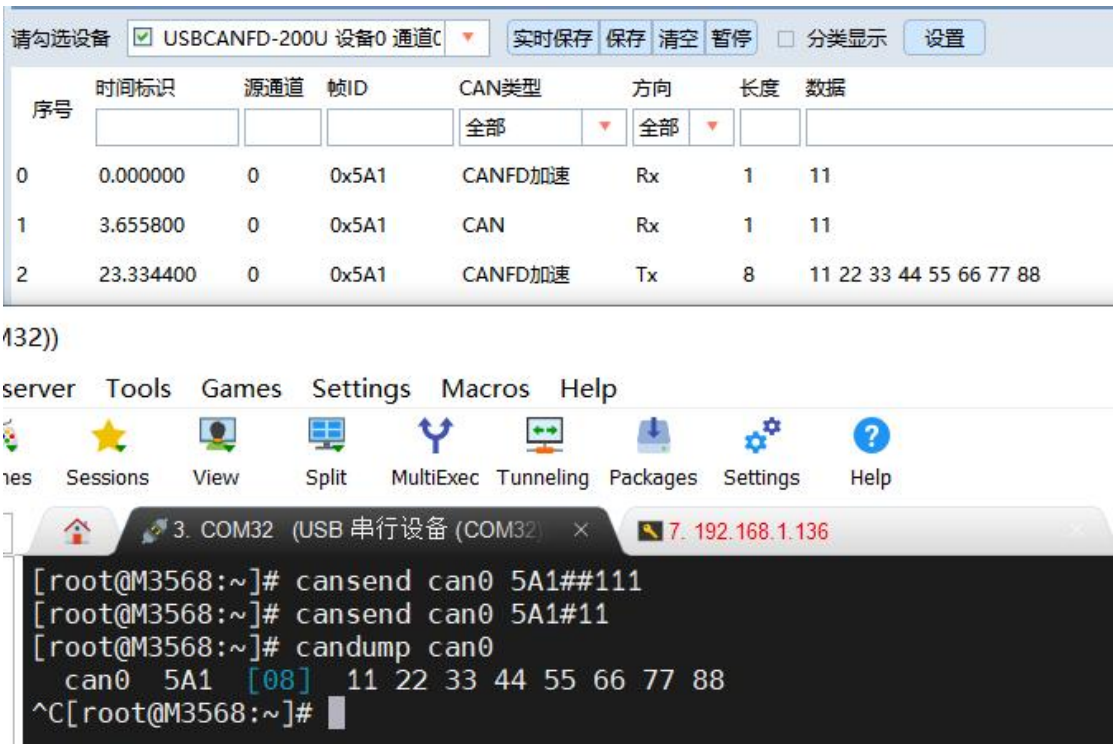

图 5.1　使用 SPI 驱动在 SPI 转 CANFD1 数据流通道进行数据收发

5.4　其他说明

由于 CSM430B 的 SPI 模式固定为模式 3，故在驱动中将 RK3568 的 SPI 模式也设置为模式 3。在此模式下 RK3568 的 SPI 最大速率为 3Mbps，而 CSM430B 的 SPI 最大支持速率为 6Mbps，如果使用其他主控芯片在模式 3 下的 SPI 速率可支持 6Mbps 时，在设备树设置中可以将 SPI 速率设置为 6Mbps，进而提升 CAN/CANFD 的收发效率。

6. UART 转 CANFD 在 Linux 系统中的使用方法

本章以 RK3568 为例，在 Linux 系统环境下如何适配及使用 CSM430B 的 UART 驱动。

6.1 UART1 转 CANFD1 和 CANFD2 模式

6.1.1 设备树配置

设备树配置如下：

```
&uart8 {
        status = " okay ";
        pinctrl-names = "default";
        pinctrl-0 = <&uart8m0_xfer>;
        dma-names = "tx", "rx";                         /* open dma */
        csm400b_u1: csm400b-u1{
                compatible = "zhiyuan,csm400b-uart";
                gpios = <&gpio3 21 0                    /* for rst */
                        &gpio3 20 0>;                   /* for cfg */
                gid = <0>;              /* using the same group id for the same csm400b */
                uid = <1>;              /* 1:u1    2:u2 */
                status = "okay";
        };
};
&pinctrl {
        can-uart{
            canrst{
                rockchip,pins = <3 RK_PC5 RK_FUNC_GPIO &pcfg_pull_down>;
            };
            cancfg{
                rockchip,pins = <3 RK_PC4 RK_FUNC_GPIO &pcfg_pull_down>;
            };
            canint{
                rockchip,pins = <3 RK_PC3 RK_FUNC_GPIO &pcfg_pull_down>;
            };
        };
};
```

6.1.2 驱动移植

将 csm400b.c、csm400b.h 拷贝到 drivers/net/can 下，修改 drivers/net/can 下的 Makefile，

添加以下内容：

```
obj-$(CONFIG_CAN_CSM400B)                    += csm400b.o
```

修改修改 drivers/net/can 下的 Kconfig，添加以下内容：

```
config CAN_CSM400B
    tristate "zlg csm400b to can device"
    depends on TTY && SERIAL_DEV_CTRL_TTYPORT && SERIAL_DEV_BUS
    help
    Support for ZLG UART to CAN interfaces.
    This is a CAN device driver which registers as an spi adapter
    and gpiochip to expose these functions of the CSM400B.
```

修改完成后在 menuconfig 开启或者在 defconfig 设置即可，本文直接在 defconfig 设置，如下：

```
CONFIG_CAN_CSM400B = m
```

设置完后重新编译即可。

6.1.3 驱动使用

输入以下命令加载驱动：

```
insmod csm400b.ko
```

加载完后 demesg 出现 uart check ok 则说明驱动正常，如果异常检测硬件连接状态及设备树配置。

```
# insmod csm400b.ko
csm400b-uart serial0-0: GID = 0, uart checking...
dw-apb-uart fe6c0000.serial: got rx and tx dma channels
csm400b-uart serial0-0 (unnamed net_device) (uninitialized): registers_write ok.
dw-apb-uart fe6c0000.serial: got rx and tx dma channels
csm400b-uart serial0-0 (unnamed net_device) (uninitialized): registers_write ok.
dw-apb-uart fe6c0000.serial: got rx and tx dma channels
csm400b-uart serial0-0 (unnamed net_device) (uninitialized): registers_write ok.
dw-apb-uart fe6c0000.serial: got rx and tx dma channels
csm400b-uart serial0-0 (unnamed net_device) (uninitialized): GID = 0, uart check ok.
```

使用 ip link show 可以看到 CAN/CANFD 设备。

```
[root@M3568:~]# ip link show
1: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
link/can
```

使用 ip 工具设置 CAN/CANFD 波特率：如下：

```
ip link set can0 up type can bitrate 1000000 dbitrate 5000000 fd on
```

设置完后即可进行收发测试，使用 cansend 进行发送, candump 进行接收，默认使用 CSM430B 的 CAN1 进行数据收发，如下图 6.1 所示：

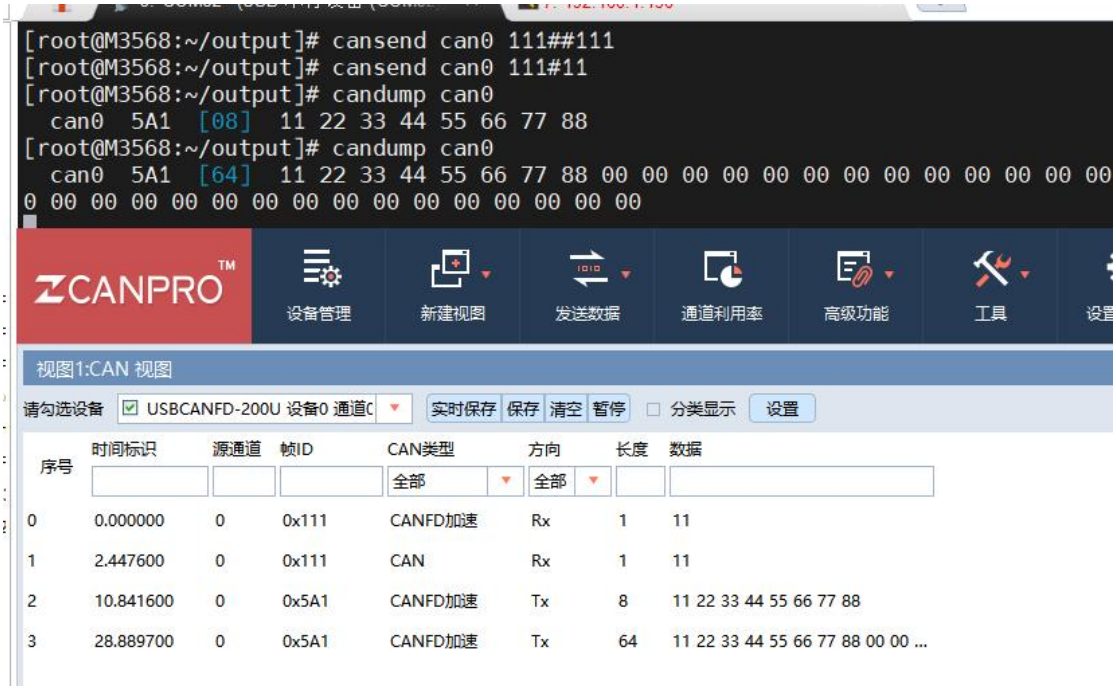

图 6.1　使用 UART1 转 CANFD1 和 CANFD2 驱动

在 UART1 转 CANFD1 数据流通道进行数据收发

切换数据流通道，在 UART1 转 CANFD2 数据流通道进行数据收发测试，如下：

```
echo 2 > /sys/bus/serial/devices/serial0-0/channel
#注：切换回 can1  则 echo 1 > /sys/bus/serial/devices/serial0-0/channel
```

切换完后使用 cansend 进行发送, candump 进行接收，如图 6.2 所示：

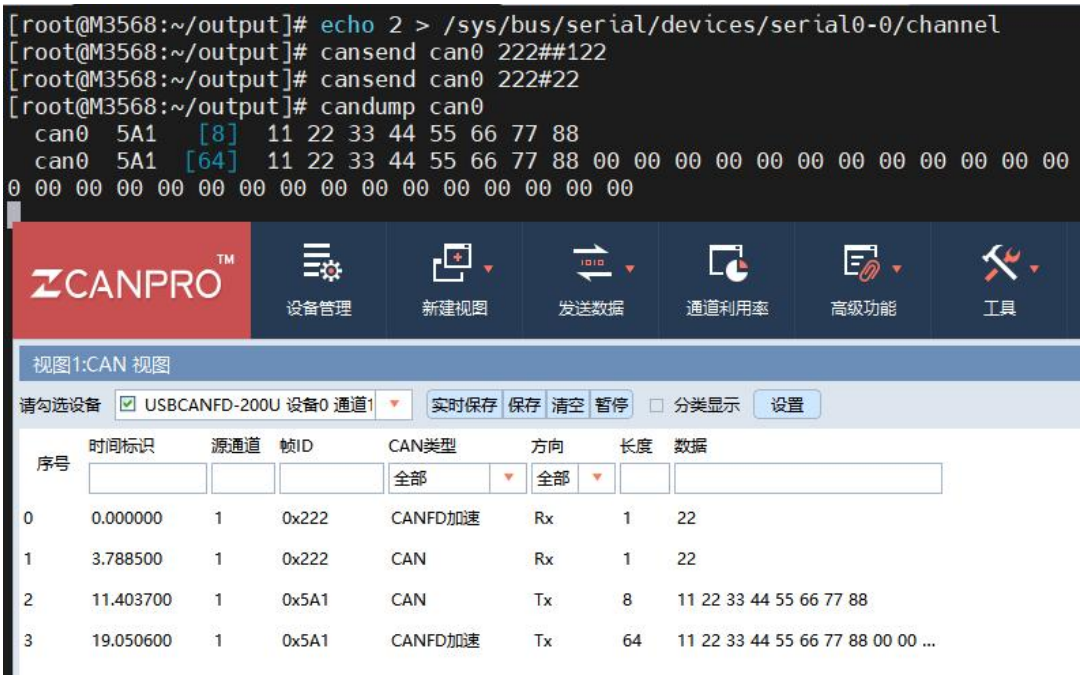

图 6.2　使用 UART1 转 CANFD1 和 CANFD2 驱动

在 UART1 转 CANFD2 数据流通道进行数据收发

6.1.4 其他说明

受限于 RK3568 的性能，驱动中 UART 最大速率设置为 4Mbps，而 CSM430B 的 UART1 最大速率支持 7Mbps，如果使用其他主控芯片的 UART 速率可支持 7Mbps 时，在设备树设置中可以将 UART 速率设置为 7Mbps，进而提升 CAN/CANFD 的收发效率。

6.2 UART1 和 UART2 转 CANFD1 和 CANFD2 模式

6.2.1 设备树配置

设备树配置如下：

```
&uart8 {
        status = " okay ";
        pinctrl-names = "default";
        pinctrl-0 = <&uart8m0_xfer>;
        dma-names = "tx", "rx";                         /* open dma */
        csm400b_u1: csm400b-u1{
                compatible = "zhiyuan,csm400b-uart";
                gpios = <&gpio3 21 0                    /* for rst */
                        &gpio3 20 0>;                   /* for cfg */
                gid = <0>;              /* using the same group id for the same csm400b */
                uid = <1>;              /* 1:u1    2:u2 */
                status = "okay";
        };
};
&uart9 {
        status = " okay ";
        pinctrl-names = "default";
        pinctrl-0 = <&uart9m1_xfer>;
        dma-names = "tx", "rx";                         /* open dma */
        csm400b_u1: csm400b-u1{
                compatible = "zhiyuan,csm400b-uart";
                gid = <0>;              /* using the same group id for the same csm400b */
                uid = <2>;              /* 1:u1    2:u2 */
                status = "okay";
        };
};

&pinctrl {
        can-uart{
                canrst{
                        rockchip,pins = <3 RK_PC5 RK_FUNC_GPIO &pcfg_pull_down>;
                };
                cancfg{
```

```
                rockchip,pins = <3 RK_PC4 RK_FUNC_GPIO &pcfg_pull_down>;
            };
            canint{
                rockchip,pins = <3 RK_PC3 RK_FUNC_GPIO &pcfg_pull_down>;
            };
        };
    };
```

6.2.2 驱动移植

将 csm400b.c、csm400b.h 拷贝到 drivers/net/can 下，修改 drivers/net/can 下的 Makefile，添加以下内容：

```
obj-$(CONFIG_CAN_CSM400B)                    += csm400b.o
```

修改修改 drivers/net/can 下的 Kconfig，添加以下内容：

```
config CAN_CSM400B
    tristate "zlg csm400b to can device"
    depends on TTY && SERIAL_DEV_CTRL_TTYPORT && SERIAL_DEV_BUS
    help
    Support for ZLG UART to CAN interfaces.
    This is a CAN device driver which registers as an spi adapter
    and gpiochip to expose these functions of the CSM400B.
```

修改完成后在 menuconfig 开启或者在 defconfig 设置即可，本文直接在 defconfig 设置，如下：

```
CONFIG_CAN_CSM400B = m
```

设置完后重新编译即可。

6.2.3 驱动使用

输入以下命令加载驱动：

```
insmod csm400b.ko
```

加载完后 demesg 出现 uart check ok 则说明驱动正常，如果异常检测硬件连接状态及设备树配置。

```
# insmod csm400b.ko
csm400b-uart serial0-0: GID = 0, uart checking...
dw-apb-uart fe6c0000.serial: got rx and tx dma channels
csm400b-uart serial0-0 (unnamed net_device) (uninitialized): registers_write ok.
dw-apb-uart fe6c0000.serial: got rx and tx dma channels
csm400b-uart serial0-0 (unnamed net_device) (uninitialized): registers_write ok.
dw-apb-uart fe6c0000.serial: got rx and tx dma channels
csm400b-uart serial0-0 (unnamed net_device) (uninitialized): registers_write ok.
```

```
dw-apb-uart fe6c0000.serial: got rx and tx dma channels
csm400b-uart serial0-0 (unnamed net_device) (uninitialized): GID = 0, uart check ok.
```

使用 ip link show 可以看到 CAN/CANFD 设备。

```
[root@M3568:~]# ip link show
1: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
link/can
2: can1: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
link/can
```

使用 ip 工具设置 CAN/CANFD：如下：

```
ip link set can0 up type can bitrate 1000000 dbitrate 5000000 fd on
ip link set can1 up type can bitrate 1000000 dbitrate 5000000 fd on
```

设置完后即可进行收发测试，使用 cansend 进行发送, candump 进行接收，如下图 6.3 和所示：

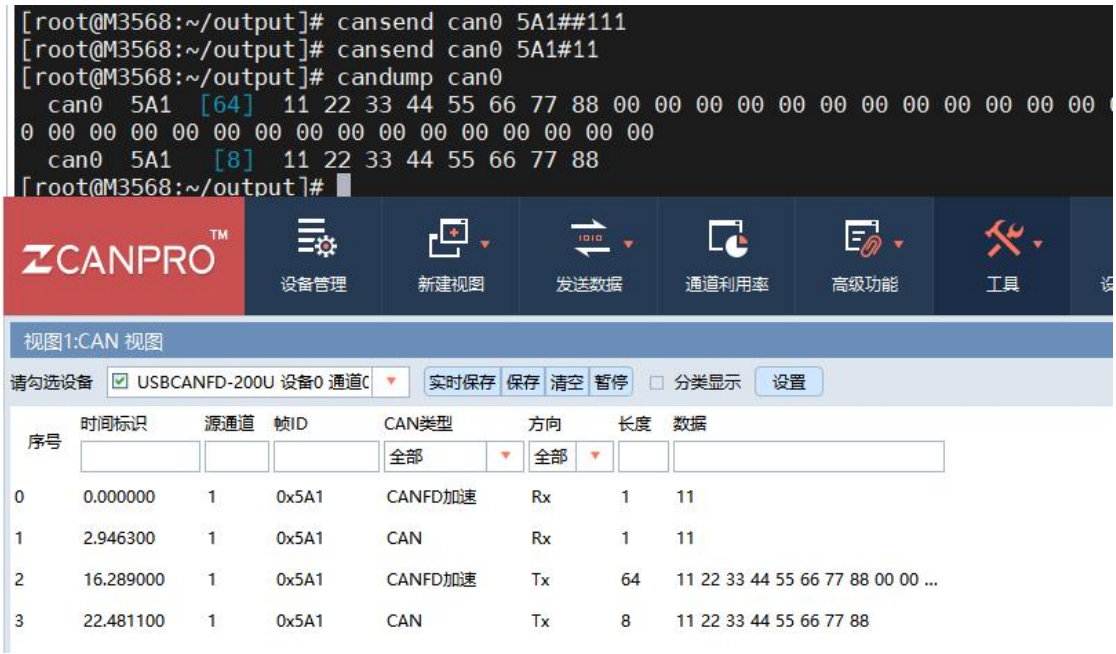

图 6.3　使用 UART1 和 UART2 转 CANFD1 和 CANFD2 驱动

在 UART1 转 CANFD1 数据流通道进行数据收发

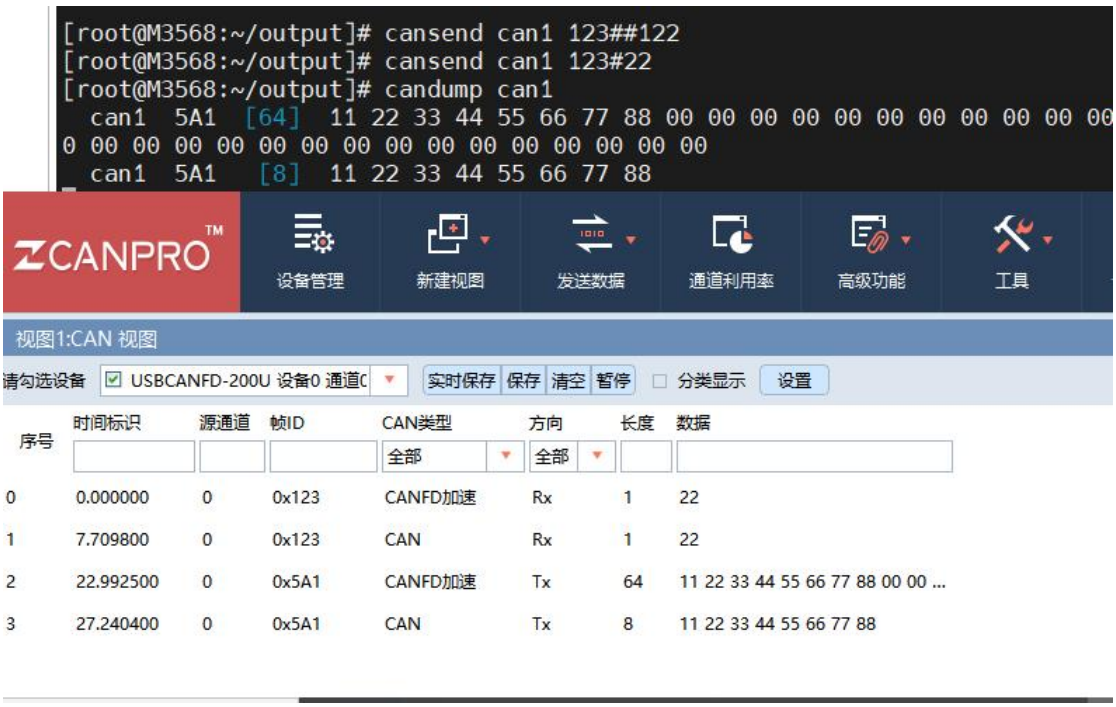

图 6.4　使用 UART1 和 UART2 转 CANFD1 和 CANFD2 驱动

在 UART2 转 CANFD2 数据流通道进行数据收发

6.2.4 其他说明

受限于 RK3568 的性能，驱动中 UART 最大速率为 4Mbps，而 CSM430B 的 UART1 和 UART2 最大速率均支持 7Mbps，如果使用其他主控芯片的 UART 速率可支持 7Mbps 时，在设备树设置中可以将 UART 速率设置为 7Mbps，进而提升 CAN/CANFD 的收发效率。

7. 产品使用注意事项

- 不支持热插拔；
- 未使用引脚请悬空处理；
- 产品为 ESD 敏感器件，请做好防静电措施；
- 产品供电电压切勿超过允许范围，以免损坏产品。CSM430B 标准 3.3V 供电，非 CANFD 总线信号接口均为 3.3V 电平标准；
- 若需要更深入了解 CSM430B 产品的电气参数，请参考《CSM430B 数据手册》；
- 若需要更深入了解辅助开发工具，请参考《CSM 系列辅助开发工具手册》。
- 生产注意事项：产品经来料检后，需放入干燥柜进行存储；产品上机贴片前应检查包装的完整性；产品在车间使用的时间长短需按 MSL3 等级管控，在车间使用寿命内，拆封后未使用完产品，须重新放入新的湿度指示卡和干燥剂进行真空包装，然后放入干燥柜存储；对于超出车间使用寿命未使用完产品，需烘烤后再使用；产品回流焊最高温度需≤245 ℃。以上生产注意事项详细内容与回流曲线请查阅《SiP 产品使用说明》。

8. 免责声明

隔离 SPI/UART 转 CANFD 收发器芯片 CSM430B 版权属广州致远电子股份有限公司所有，其产权受国家法律绝对保护，未经本公司授权，其它公司、单位、代理商及个人不得非法使用和拷贝，否则将会承担相关的法律责任。

广州致远电子股份有限公司保留随时修改上述内容的权利，产品用户手册更新时恕不另行通知，如需查看最新版本的信息，请访问我司官方网站或联系我司人员获取。

诚信共赢，持续学习，客户为先，专业专注，只做第一

广州致远电子股份有限公司
更多详情请访问 www.zlg.cn
欢迎拨打全国服务热线 400-888-4005