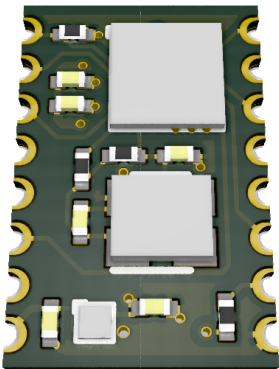


CD485B1 数据手册

Duke Fong

August 6, 2017



Contents

1	功能描述	3
1.1	概述	3
1.2	特性	3
2	CD-BUS 协议	3
3	硬件	4
3.1	电路参考	4
3.2	内部结构	5
3.3	引脚定义	5
3.4	尺寸规格	6
3.5	极限参数	6
3.6	建议工作参数	6
3.7	直流参数	6
4	寄存器列表	7
5	流程图	10
5.1	RX	11
5.2	TX	12

6	设备接口	12
6.1	SPI	13
6.2	I2C	13
7	操作示例	13
7.1	初始化	13
7.1.1	兼容模式和传统模式	14
7.2	TX	14
7.3	RX	14
8	版权说明	15

1 功能描述

1.1 概述

CD-BUS 是一款基於 RS485 的通讯协议，它只定义了 ISO/OSI 模型的数据链路层。  
CD-BUS 协议由 DUKELEC 公司於 2009 年设计，以便捷、多主对等、高速通讯为目标。

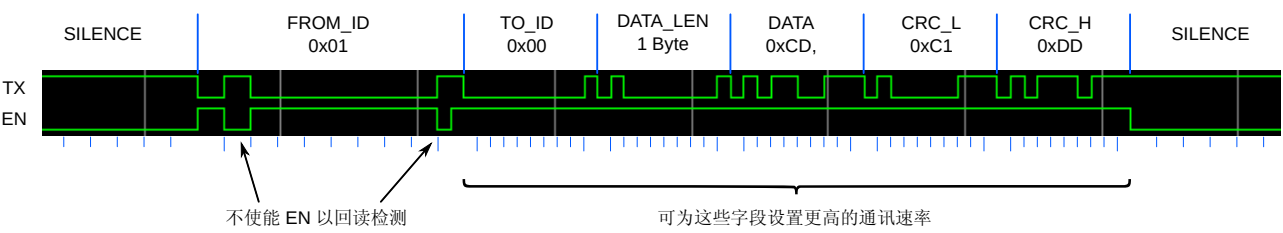
1.2 特性

CD485B1 型号模块支持：

- 支持 CD-BUS 多主对等通讯协议，使用发送方地址按位仲裁
- 每个数据帧可装载 253 字节数据
- 8 个接收缓冲页，2 个发送缓冲页，每个页 256 字节
- 16 位硬件 CRC 校验
- 波特率范围 412 bps 至 9 Mbps（如果需要可以支持更高）
- 仲裁字段和后续数据可设定不同波特率
- 可兼容传统 RS485 总线设备
- 支持 SPI 和 I2C 接口
- 配置和使用简单

2 CD-BUS 协议

CD-BUS 示例时序：



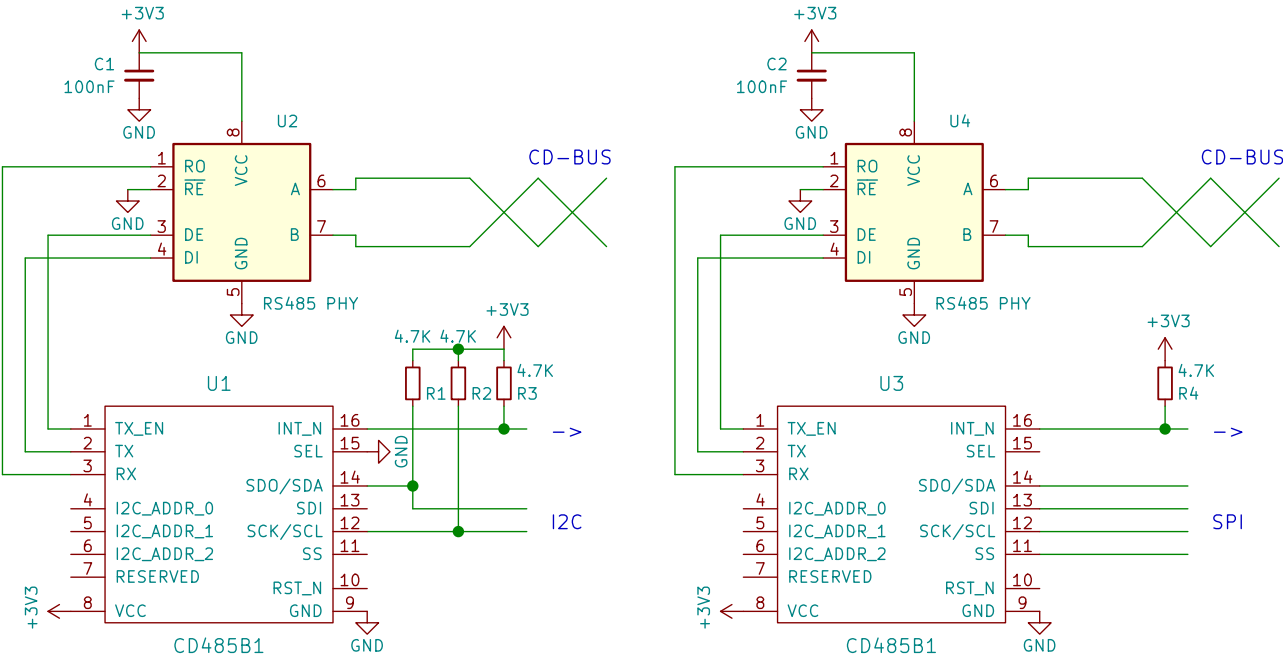
字段	长度（字节）	用途
SILENCE	0~25.5 默认： 2 (20 bits)	分隔数据帧 总线在数据帧结束后继续保持 SILENCE 位长的时间为 1, 总线便进入空闲模式。 当总线为空闲模式才允许接收。 当总线持续空闲一段时间（默认 10 bit）后才开始发送。

FROM_ID	1	发送方 ID 在发送此字段时，所有的 1 不使能 TX_EN 脚，从而回读总线状态以判断是否有更高优先级节点同时在发送。若有，该节点立即停止并推后发送；若无，在最后一次回读后使能 TX_EN 并保持到数据帧结束。此字段会在所有数据 1 的中间位置进行回读，因为发送与接收存在延时，所以通常设置低于 1 Mbps。
TO_ID	1	接收方 ID, 255 为广播帧。
DATA_LEN	1	装载的数据长度，范围：0~253 字节，每个缓存页是 256 字节，最前 3 字节被 FROM_ID、TO_ID 和 DATA_LEN 使用。
DATA	0~253	装载的数据
CRC_L	1	CRC 低 8 位，与 Modbus RTU 使用相同的 CRC 标准。
CRC_H	1	CRC 高 8 位

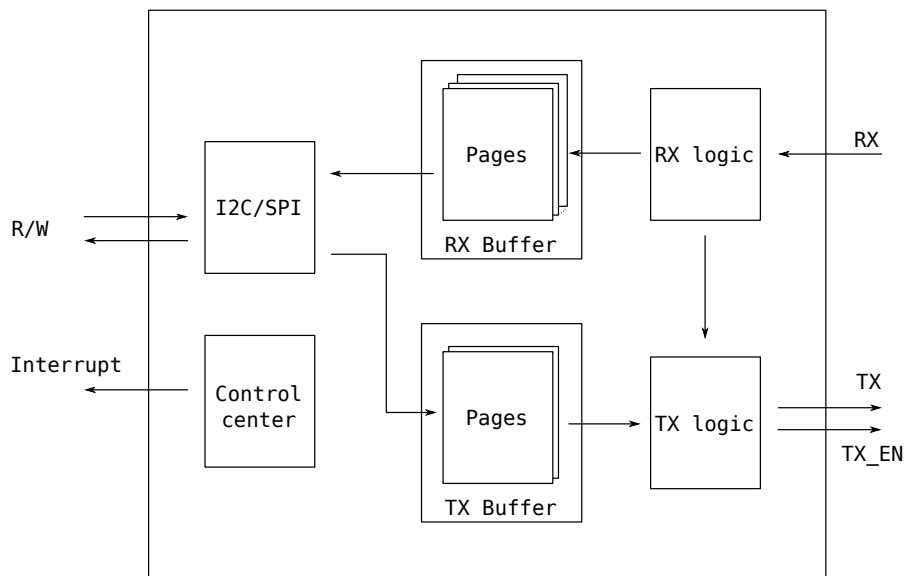
CD-BUS 协议只定义数据帧格式，不规定所装载数据格式；只支持单播和广播，不支持多播；只提供硬件避让、避让后自动重传，而应答及出错处理则由上层软件负责。

3 硬件

3.1 电路参考



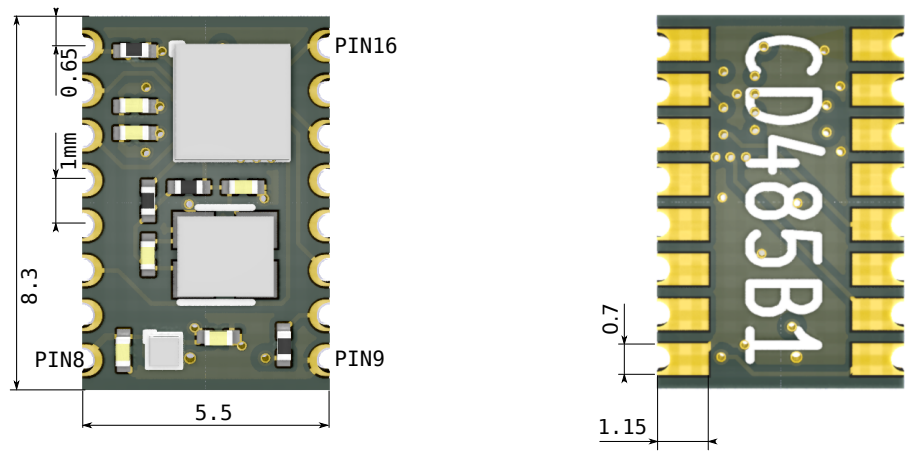
### 3.2 内部结构



### 3.3 引脚定义

引脚	定义	I/O	上下拉	描述
1	TX_EN	O	下拉 (10 kOhm)	使能脚，连接 RS485 收发器
2	TX	O	-	发送脚，连接 RS485 收发器
3	RX	I	-	接收脚，连接 RS485 收发器
4..6	I2C_ADDR	I	上拉 (40 kOhm)	设置 I2C 地址
7	RESERVED			留空
8	VCC			电源
9	GND			地
10	RST_N	I	上拉 (10 kOhm)	复位， $\geq 200$ ns 低脉冲复位（可选）
11	SS	I	上拉 (40 kOhm)	SPI 片选
12	SCK/SCL	I	-	SPI/I2C 时钟
13	SDI	I	-	SPI MOSI
14	SDO/SDA	I/O	-	SPI MOSI / I2C SDA
15	SEL	I	上拉 (40 kOhm)	输入高选择 SPI 模式；低为 I2C 模式
16	INT_N	O	-	中断，低有效，开漏输出

3.4 尺寸规格



3.5 极限参数

参数	最小	最大
VCC 电压	-0.5 V	3.60 V
环境温度	-65 °C	150 °C
节温 (T <sub>J</sub> )	-65 °C	125 °C

3.6 建议工作参数

参数	最小	最大
VCC 电压	3.14 V	3.46 V
节温	-40 °C	100 °C
上电速度	0.6 V/ms	10 V/ms

3.7 直流参数

参数	最小	典型	最大
V <sub>IL</sub>	-0.3 V	-	0.8 V
V <sub>IH</sub>	2.0 V	-	VCC + 0.2 V
V <sub>OL</sub>	0.2 V	-	0.4 V
V <sub>OH</sub>	VCC - 0.4 V	-	VCC - 0.2 V
I <sub>OL</sub>	-	-	8 mA
I <sub>OH</sub>	-	-	-8 mA
Input 或 I/O 漏电流	-	-	+/-10 uA

I/O 寄生电容 (25°C, 1.0 MHz)	-	6 pF	-
器件功耗	-	-	15 mW
V <sub>PORUP</sub> (上电复位电压阈值)	0.7 V	-	1.6 V
V <sub>PORDN</sub>	-	-	1.6 V

## 4 寄存器列表

地址	名称	R/W	说明												
0x00	VERSION	R	硬件版本，当前为：0x01												
0x01	SETTING	RW	<div>Bits:</div> <table><tr><td>bit0</td><td>TX_PUSH_PULL 如果关闭，TX 为开漏输出，且 TX_EN 悬空不使用。</td></tr><tr><td>bit1</td><td>TX_INVERT 如果设置，TX 将会反向输出。</td></tr><tr><td>bit2</td><td>USER_CRC 关闭硬件 CRC 如果关闭：用户需要自行计算两字节 CRC 并追写在数据之后；在读完数据之后再读两字节 CRC 数据以便自行校验。用户数据最大长度将会降至 251 字节。</td></tr><tr><td>bit3</td><td>NO_DROP 如果设置，当接收出错置位 RX_ERROR 标志时会同时保留出错的数据帧。 通过 RX_PAGE_FLAG 判断当前 RX 缓存页中的数据帧是否有错。</td></tr><tr><td>bit[5:4]</td><td>TX_EN_ADVANCE（仅 NO_ARBITRATE 置位时有效） TX_EN 提前 TX 使能的位长（额外加上 1 个系统时钟周期）。</td></tr><tr><td>bit6</td><td>NO_ARBITRATE 关闭仲裁功能，输出时 TX_EN 一直有效。</td></tr></table> <div>默认：x0010000 (x: 不关心，写 0)</div>	bit0	TX_PUSH_PULL 如果关闭，TX 为开漏输出，且 TX_EN 悬空不使用。	bit1	TX_INVERT 如果设置，TX 将会反向输出。	bit2	USER_CRC 关闭硬件 CRC 如果关闭：用户需要自行计算两字节 CRC 并追写在数据之后；在读完数据之后再读两字节 CRC 数据以便自行校验。用户数据最大长度将会降至 251 字节。	bit3	NO_DROP 如果设置，当接收出错置位 RX_ERROR 标志时会同时保留出错的数据帧。 通过 RX_PAGE_FLAG 判断当前 RX 缓存页中的数据帧是否有错。	bit[5:4]	TX_EN_ADVANCE（仅 NO_ARBITRATE 置位时有效） TX_EN 提前 TX 使能的位长（额外加上 1 个系统时钟周期）。	bit6	NO_ARBITRATE 关闭仲裁功能，输出时 TX_EN 一直有效。
bit0	TX_PUSH_PULL 如果关闭，TX 为开漏输出，且 TX_EN 悬空不使用。														
bit1	TX_INVERT 如果设置，TX 将会反向输出。														
bit2	USER_CRC 关闭硬件 CRC 如果关闭：用户需要自行计算两字节 CRC 并追写在数据之后；在读完数据之后再读两字节 CRC 数据以便自行校验。用户数据最大长度将会降至 251 字节。														
bit3	NO_DROP 如果设置，当接收出错置位 RX_ERROR 标志时会同时保留出错的数据帧。 通过 RX_PAGE_FLAG 判断当前 RX 缓存页中的数据帧是否有错。														
bit[5:4]	TX_EN_ADVANCE（仅 NO_ARBITRATE 置位时有效） TX_EN 提前 TX 使能的位长（额外加上 1 个系统时钟周期）。														
bit6	NO_ARBITRATE 关闭仲裁功能，输出时 TX_EN 一直有效。														
0x02	SILENCE_LEN	RW	总线在数据帧结束后继续保持 SILENCE 位长的时间为 1, 总线便进入空闲模式，默认 20 (bits)												
0x03	TX_DELAY	RW	当总线进入空闲并保持此段时间，才允许发送，默认 10 (bits) 可以为越高优先级节点设置越低的值，但至少要保留 1 bit, 以确保所有节点都有足够时间检测到总线 IDLE 状态。												

0x04	SELF_ID	RW	仅用做接收过滤：（由上至下进行匹配）																								
			<table><tr><th>FROM_ID</th><th>TO_ID</th><th>SELF_ID</th><th>接收或丢弃</th></tr><tr><td>not care</td><td>not care</td><td>255</td><td>接收（嗅探模式）</td></tr><tr><td>= SELF_ID</td><td>not care</td><td>!= 255</td><td>丢弃（避免环路）</td></tr><tr><td>!= SELF_ID</td><td>255</td><td>not care</td><td>接收（广播）</td></tr><tr><td>!= SELF_ID</td><td>!= 255</td><td>= TO_ID</td><td>接收（点对点）</td></tr><tr><td>not care</td><td>!= 255</td><td>!= TO_ID</td><td>丢弃</td></tr></table>	FROM_ID	TO_ID	SELF_ID	接收或丢弃	not care	not care	255	接收（嗅探模式）	= SELF_ID	not care	!= 255	丢弃（避免环路）	!= SELF_ID	255	not care	接收（广播）	!= SELF_ID	!= 255	= TO_ID	接收（点对点）	not care	!= 255	!= TO_ID	丢弃
FROM_ID	TO_ID	SELF_ID	接收或丢弃																								
not care	not care	255	接收（嗅探模式）																								
= SELF_ID	not care	!= 255	丢弃（避免环路）																								
!= SELF_ID	255	not care	接收（广播）																								
!= SELF_ID	!= 255	= TO_ID	接收（点对点）																								
not care	!= 255	!= TO_ID	丢弃																								
			默认：255																								
0x05	PERIOD_LS_L	RW	PERIOD_LS 低 8 位 (LS: Low Speed) 为 SILENCE、TX_DELAY 和 FROM_ID 字段设置波特率（也包括 EN_ADVANCE）。 计算公式 $factor = sysclock \div bond\_rate - 1$ 举例：当前系统时钟是 27 MHz，设置 233 最接近 115200 bps（默认 233）																								
0x06	PERIOD_LS_H	RW	PERIOD_LS 高 8 位，共 16 位（默认 0）																								
0x07	PERIOD_HS_L	RW	PERIOD_HS 低 8 位 (LS: Low Speed, 默认 233) 为 TO_ID、DATA_LEN、DATA 和 CRC_L/H 字段设置波特率。																								
0x08	PERIOD_HS_H	RW	PERIOD_HS 高 8 位，共 16 位（默认 0）																								

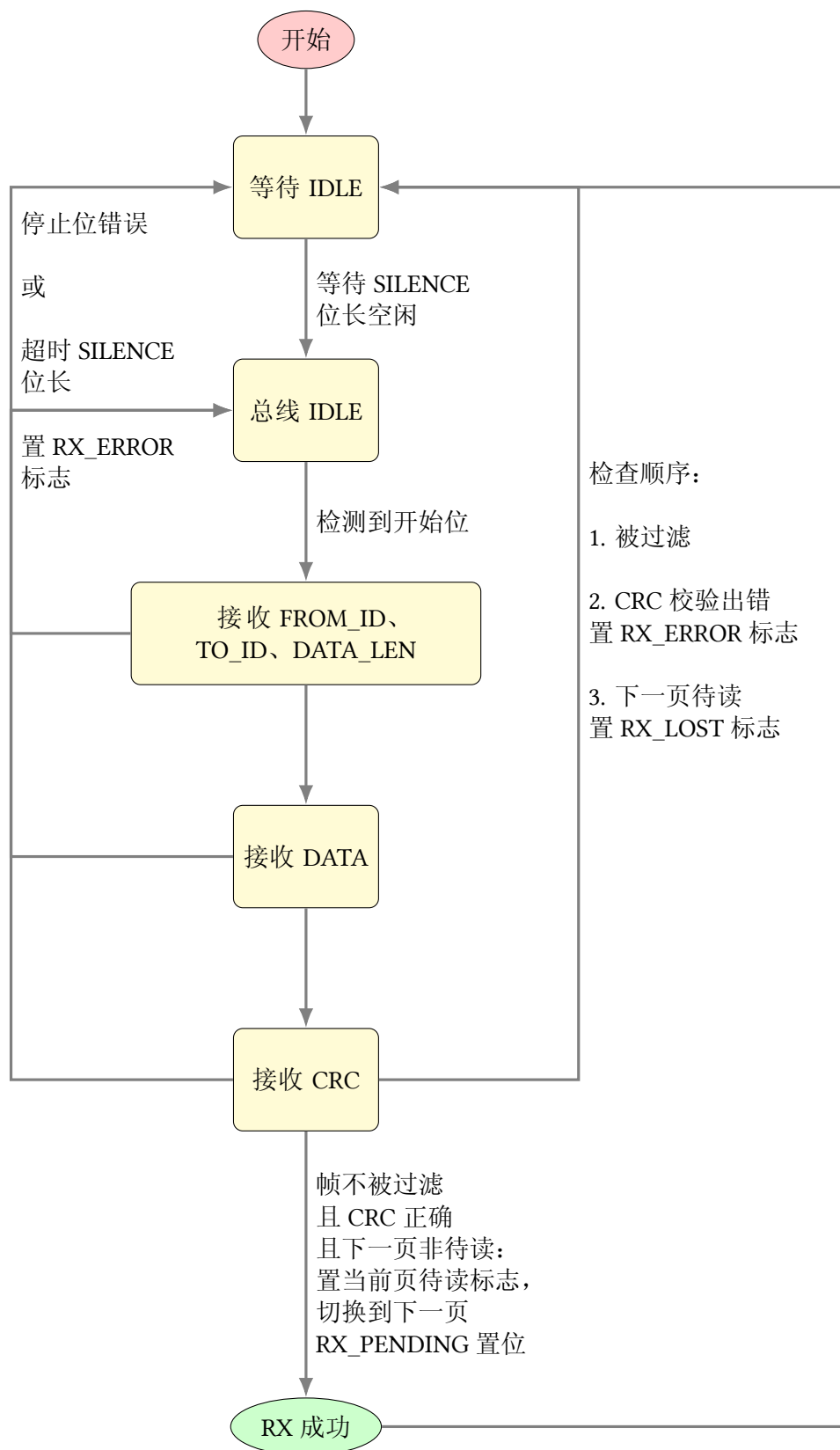


0x09	INT_FLAG	R	中断标志：  <div> <div>bit0</div> <div>BUS_IDLE</div> <div>指示总线是否进入 IDLE 模式。</div> </div> <div> <div>bit1</div> <div>RX_PENDING</div> <div>指示 RX 缓存是否有页待读。 写 1 到 RX_CTRL[CLR_RX_PENDING] 清除当前页待读标志。</div> </div> <div> <div>bit2</div> <div>RX_LOST</div> <div>当一个帧正确抵达且不被过滤，但却因为没有更多页用做下一次接收而被丢弃，此标志置位。 写 1 到 RX_CTRL[CLR_RX_LOST] 清此标志。</div> </div> <div> <div>bit3</div> <div>RX_ERROR</div> <div>当一个不被过滤的帧停止位错误、超时或校验错误，此标志置位。 写 1 到 RX_CTRL[CLR_RX_ERROR] 清此标志。</div> </div> <div> <div>bit4</div> <div>TX_BUF_CLEAN</div> <div>指示是否所有 TX 缓存页都未标记为待发送。</div> </div> <div> <div>bit5</div> <div>TX_CD</div> <div>当检测到有更高优先级节点时推后发送并置此标志。 写 1 到 TX_CTRL[CLR_TX_CD] 清此标志。 此位用作调试使用。</div> </div> <div> <div>bit6</div> <div>TX_ERROR</div> <div>检测到冲突后，当总线再次空闲超过设定时间硬件会自动重发，但如果连续重发 3 次都发生冲突，则取消发送，并置位此标志。 写 1 到 TX_CTRL[CLR_TX_ERROR] 清此标志。</div> </div>
0x0A	INT_MASK	RW	中断允许 当 INT_FLAG & INT_MASK != 0 时 INT_N 输出低，否则输出高阻（默认 0x00）
0x0B	RX	R	读 RX 缓存页数据，地址自动增加 共有 8 个 RX 缓存页，每一页 256 字节。 当硬件端成功接收到不被过滤的帧：如果下一页未标记为待读（可用作下一次接收），将当前页标记为待读并切换到下一页；否则丢弃该帧并置位 RX_LOST。 RX_PENDING 位指示用户端当前页待读，写 1 到 CLR_RX_PENDING 清除当前页的待读标志并切换到下一页。写 1 到 RST_RX 清除所有页的待读标志，并复位接收逻辑。

0x0C	TX	W	<p>写 TX 缓存页数据，地址自动增加 共有 2 个 TX 缓存页，每一页 256 字节。 当用户写完数据，需要等待 TX_BUF_CLEAN 置位，然后才可以通 过 START_TX 置位当前页的待发送标志，并自动切换到下一页（否 则什么都不会发生）。 当页的待发送标志被置上，硬件将会启动发送，当发送完毕，硬件 端清页的待发送标志并切换到下一页。</p>
0x0D	RX_CTRL	W	<p>RX 控制:</p> <hr/> <p>bit0 RST_RX_POINTER 写 1 归零当前 RX 缓存页的读指针</p> <hr/> <p>bit1 CLR_RX_PENDING (自动包含 bit0)</p> <hr/> <p>bit2 CLR_RX_LOST</p> <hr/> <p>bit3 CLR_RX_ERROR</p> <hr/> <p>bit4 RST_RX (自动包含 bit0, 2, 3)</p>
0x0E	TX_CTRL	W	<p>TX 控制:</p> <hr/> <p>bit0 RST_TX_POINTER 写 1 归零当前 TX 缓存页的写指针</p> <hr/> <p>bit1 START_TX (自动包含 bit0)</p> <hr/> <p>bit3 CLR_TX_CD</p> <hr/> <p>bit4 CLR_TX_ERROR</p>
0x0F	RX_ADDR	RW	读写当前 RX 缓存页的读指针
0x10	RX_PAGE_FLAG	R	<p>（仅 NO_DROP 置位时使用） 0 代表当前 RX 缓存页中的数据帧正确； 非 0 表示数据帧错误，其值指示最后接收到的字节地址，包含 CRC 字段。</p>

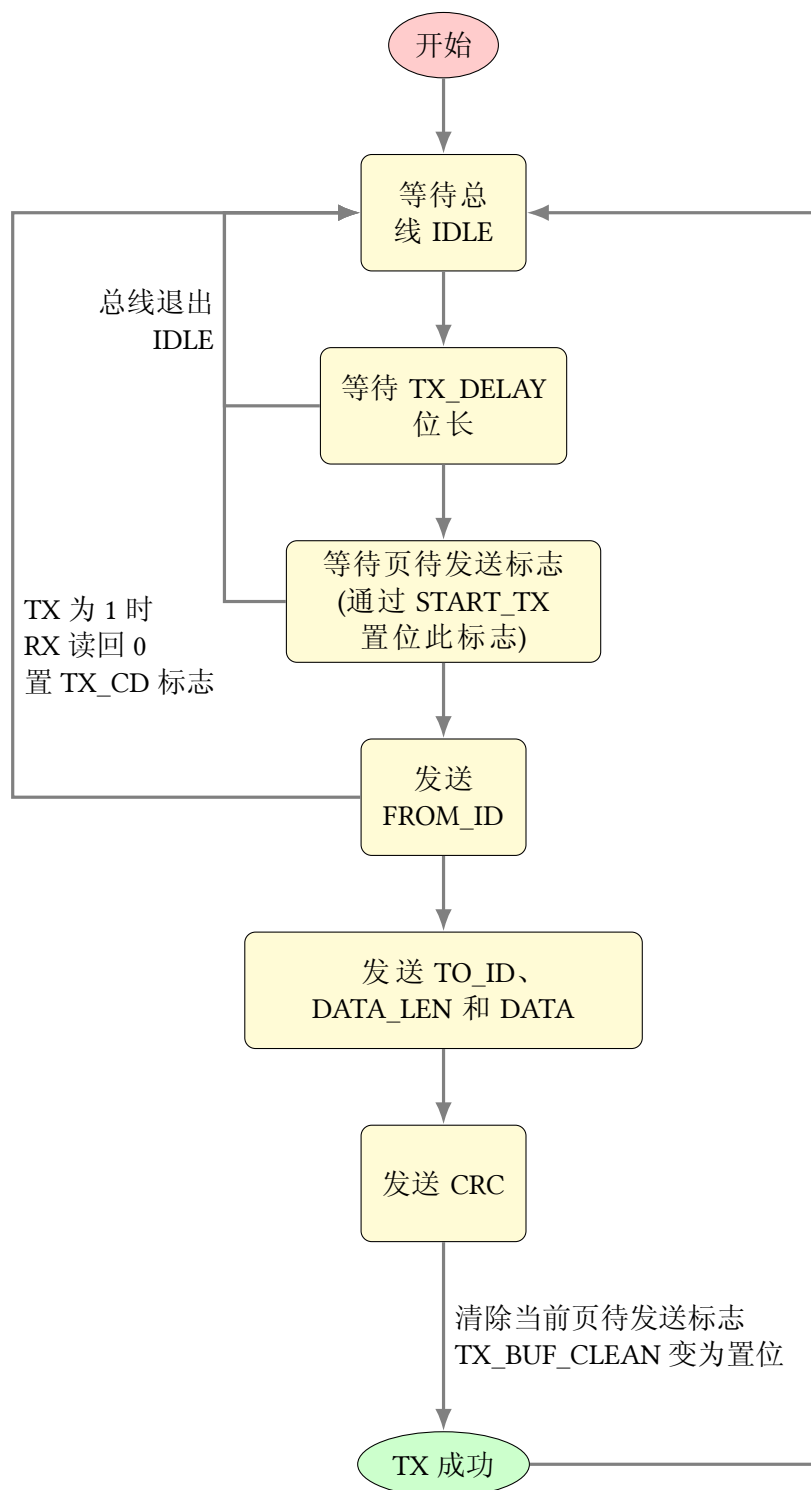
## 5 流程图

## 5.1 RX



如果当前帧接收不够两个字节，或者将会被丢弃则不会设置 RX\_ERROR 标志。

## 5.2 TX



## 6 设备接口

SPI 和 I2C 频率低于  $sysclock \div 10$ .

除了 RX 和 TX, 其余寄存器通常只读写 1 字节。

## 6.1 SPI

读写:

```
start (NSS = 0)
Write reg address with bit7: 0: read, 1: write
Read or write arbitrary length of data
stop (NSS = 1)
```

## 6.2 I2C

```
Write address: 0xc0 | (I2C_ADDR << 1)
Read address: 0xc1 | (I2C_ADDR << 1)
```

I2C\_ADDR is the value of I2C\_ADDR\_n pins.

写:

```
start
write the write address
write 1 byte reg address
write arbitrary length of data
stop
```

读:

```
start
write the write address
write 1 byte reg address
restart (or stop + start)
write the read address
read arbitrary length of data, ACK all bytes except last byte
stop
```

# 7 操作示例

## 7.1 初始化

```
// enable OUTPUT
CD485_write(REG_SETTING, TX_PUSH_PULL);

// set SELF_ID
CD485_write(REG_SELF_ID, 0xcd);

// set bondrate, PERIOD_XX_H default 0
CD485_write(REG_PERIOD_LS_L, 35); // 750000 bps
CD485_write(REG_PERIOD_HS_L, 2); // 9 Mbps

// clean RX buffer
```

```
CD485_write(REG_RX_CTRL, RST_RX);

// enable interrupt
// CD485_write(REG_INT_MASK, TX_ERROR | RX_ERROR | RX_LOST | RX_PENDING);
```

### 7.1.1 兼容模式和传统模式

PERIOD\_LS 和 PERIOD\_HS 设置相同为兼容模式。

进一步置位 NO\_ARBITRATE 进入传统模式：



## 7.2 TX

```
header_buf[0] = 0xcd; // FROM_ID
header_buf[1] = 0x02; // TO_ID
header_buf[2] = 12;   // DATA_LEN

CD485_write_chunk(REG_TX, header_buf, 3);           // write HEADER
CD485_write_chunk(REG_TX, data_buf, header_buf[2]); // write DATA

while (CD485_read(REG_INT_FLAG) & TX_BUF_CLEAN == 0); // make sure TX_BUF_CLEAN is set
CD485_write(REG_TX_CTRL, START_TX); // sent frame
```

## 7.3 RX

```
while (CD485_read(REG_INT_FLAG) & RX_PENDING == 0);

CD485_read_chunk(REG_RX, header_buf, 3);           // read HEADER
CD485_read_chunk(REG_RX, data_buf, header_buf[2]); // read DATA

CD485_write(REG_RX_CTRL, CLR_RX_PENDING);          // release page
```

## 8 版权说明

CD-BUS（又或 CD485）是一个相当开放的协议，硬件实现也相对简单，除了芯片生产商需要支付少量版权费，其余任何人都可以免费使用此协议及其变种，只需要在产品说明中保留原始的版权信息。

联络：[info@dukelec.com](mailto:info@dukelec.com)