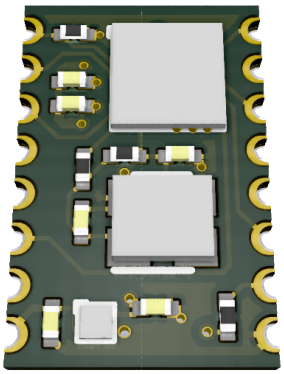
CDCTL-B1 Data Manual

DUKELEC

August 14, 2017



Contents

1 Functional Description

1.1 Overview

CDBUS is a communication protocol based on RS485, it only covers the data link layer in ISO/OSI model.

CDBUS protocol was designed by DUKELEC in 2009 for simple, multi-master and high-speed communication in mind.

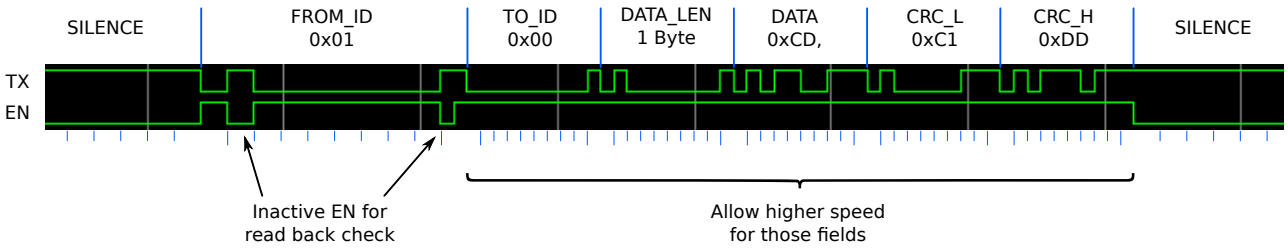
1.2 Highlights

The CDCTL-B1 module supports:

- Support multiple master on CDBUS, bitwise arbitration by sender ID
- 253 bytes data payload per frame
- 8 buffer pages for RX purpose, 2 buffer pages for TX purpose, each page is 256 bytes
- 16 bit hardware CRC generation and verification
- Baud rate from 458 bps to 10 Mbps (support higher if need)
- Separate baud rate setting for arbitration byte and follow data
- Backward-compatible with traditional RS485 bus
- Support SPI and I2C peripheral interface
- Easy configuration and operation

2 CDBUS Protocol

Timing example of CDBUS:



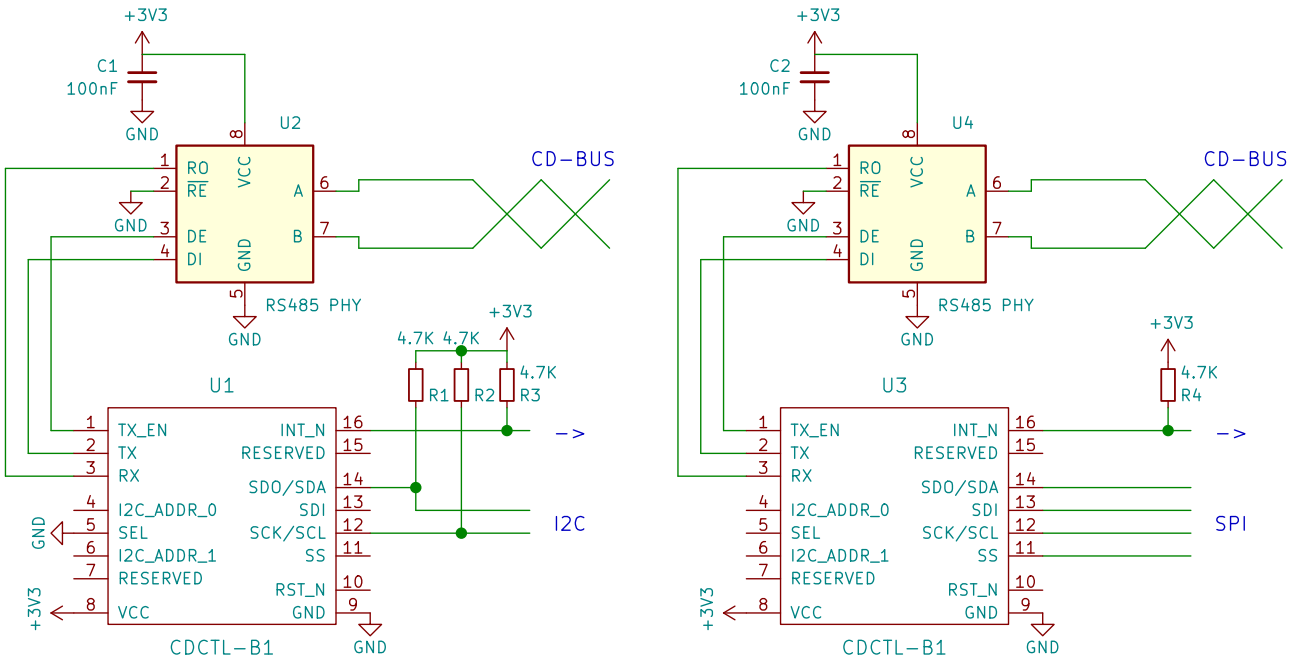
Field name	Length (bytes)	Purpose
SILENCE	0~25.5 Default: 2 (20 bits)	The separator between frames Wait for the end of any frame on the bus and bus keep logic 1 for SILENCE bits of time, then bus enter IDLE mode. Allow receiving when bus in IDLE mode. Allow sending after bus kept in IDLE mode for a period of time (10 bits by default).

FROM_ID	1	Sender ID TX_EN pin inactive for all logic 1 during this field, allow the sender read back bus state to check if there are any other node start sending at same time, if so, the lower priority node immediately stops and defer sending, or enable TX_EN at the end of last check. Hardware read back bus state at middle of logic 1 during this field, because of the delay exist between TX and RX, the baud rate for this field should normally less than 1 Mbps.
TO_ID	1	Receiver ID, 255 for broadcast.
DATA_LEN	1	Payload data length, range: 0~253 bytes, each buffer page is 256 bytes, the first 3 bytes occupied by FROM_ID, TO_ID and DATA_LEN.
DATA	0~253	Payload data
CRC_L	1	Low 8 bits of CRC, Use the same CRC standard as Modbus RTU.
CRC_H	1	High 8 bits of CRC

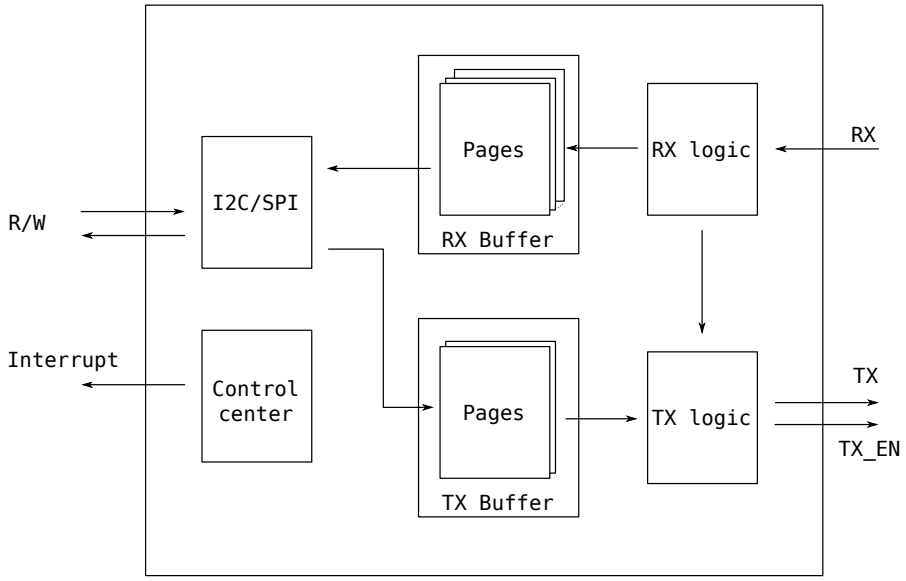
CDBUS protocol only defines the frame format, does not specify the payload data format; Only supports unicast and broadcast, does not support multicast; Only provide hardware arbitration, automatic retransmission after conflict, handshake and error handling are handled by software at upper layer.

3 Hardware

3.1 Circuit Reference



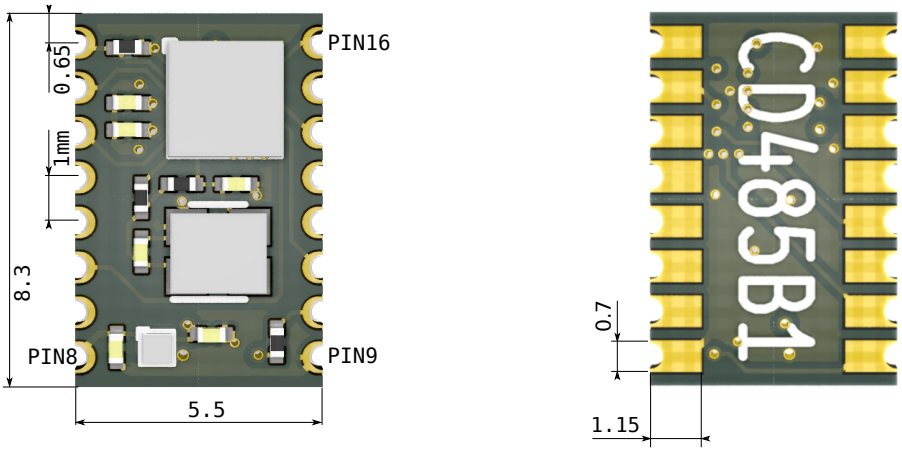
3.2 Internal Block



3.3 Pin Definition

No	Name	I/O	Pull	Description
1	TX_EN	O	Down (10 kOhm)	Transmit enable pin to RS485 PHY
2	TX	O	-	Transmit pin to RS485 PHY
3	RX	I	-	Receive pin from RS485 PHY
4, 6	I2C_ADDR	I	Up (40 kOhm)	Set I2C address
5	SEL	I	Up (40 kOhm)	High select SPI mode, low select I2C mode
7, 15	RESERVED			Left empty
8	VCC			Supply voltage
9	GND			Ground
10	RST_N	I	Up (10 kOhm)	Reset pin, $\geq$ 200 ns low pulse width required to reset (optional)
11	SS	I	Up (40 kOhm)	SPI chip select
12	SCK/SCL	I	-	SPI/I2C clock
13	SDI	I	-	SPI MOSI
14	SDO/SDA	I/IO	-	SPI MOSI / I2C SDA
16	INT_N	O	-	Interrupt pin, open-drain output

3.4 Mechanical Specifications



3.5 Absolute Maximum Ratings

Parameter	Min.	Max.
Supply Voltage VCC	−0.5 V	3.60 V
Storage Temperature (Ambient)	−65 ℃	150 ℃
Junction Temperature (T_J)	−65 ℃	125 ℃

3.6 Recommended Operating Conditions

Parameter	Min.	Max.
Supply Voltage VCC	3.14 V	3.46 V
Junction Temperature Operation	−40 ℃	100 ℃
Power supply ramp rate	0.6 V/ms	10 V/ms

3.7 DC Electrical Characteristics

Parameter	Min.	Typ.	Max.
V_{IL}	-0.3 V	-	0.8 V
V_{IH}	2.0 V	-	VCC + 0.2 V
V_{OL}	0.2 V	-	0.4 V
V_{OH}	VCC - 0.4 V	-	VCC - 0.2 V
I_{OL}	-	-	8 mA
I_{OH}	-	-	-8 mA
Input or I/O Leakage	-	-	+/-10 uA

I/O Capacitance (25℃, 1.0 MHz)	-	6 pF	-
Power Consumption	-	-	15 mW
V_{PORUP} (Power-On-Reset threshold)	0.7 V	-	1.6 V
V_{PORDN}	-	-	1.6 V

4 Register Reference

Addr	Name	R/W	description
0x00	VERSION	R	Hardware version, value: 0x01.
0x01	SETTING	RW	Bits: bit0: TX_PUSH_PULL If not set, TX is open-drain output and TX_EN is not used (float). bit1: TX_INVERT If set, TX is active low. bit2: USER_CRC Disable hardware CRC generation and verification. If set: user should write two addition CRC bytes followed by data; and read two more bytes for CRC after data, the data length reduced to 251 bytes in this situation. bit3: NO_DROP If set, not drop frame which set RX_ERROR flag. Determine whether the frame is correct or not in current RX page by RX_PAGE_FLAG. bit[5:4]: TX_EN_ADVANCE (only used if NO_ARBITRATE is set). Advance TX_EN for bits of time before TX sending (with 1 system clock period in addition). bit6: NO_ARBITRATE Disable auto arbitration and always active TX_EN pin during sending. Default value: x0010000 (x: not care, write by 0).
0x02	SILENCE_LEN	RW	Bus enter IDLE when bus keep logic 1 for SILENCE bits of time after end of any frame, default: 20 (bits).
0x03	TX_DELAY	RW	Allow TX after bus kept in IDLE mode for this period of length, default: 10 (bits). You could lower the value for higher priority node, retain at least 1 bit to make sure all node have enough time to detect the bus's IDLE state.

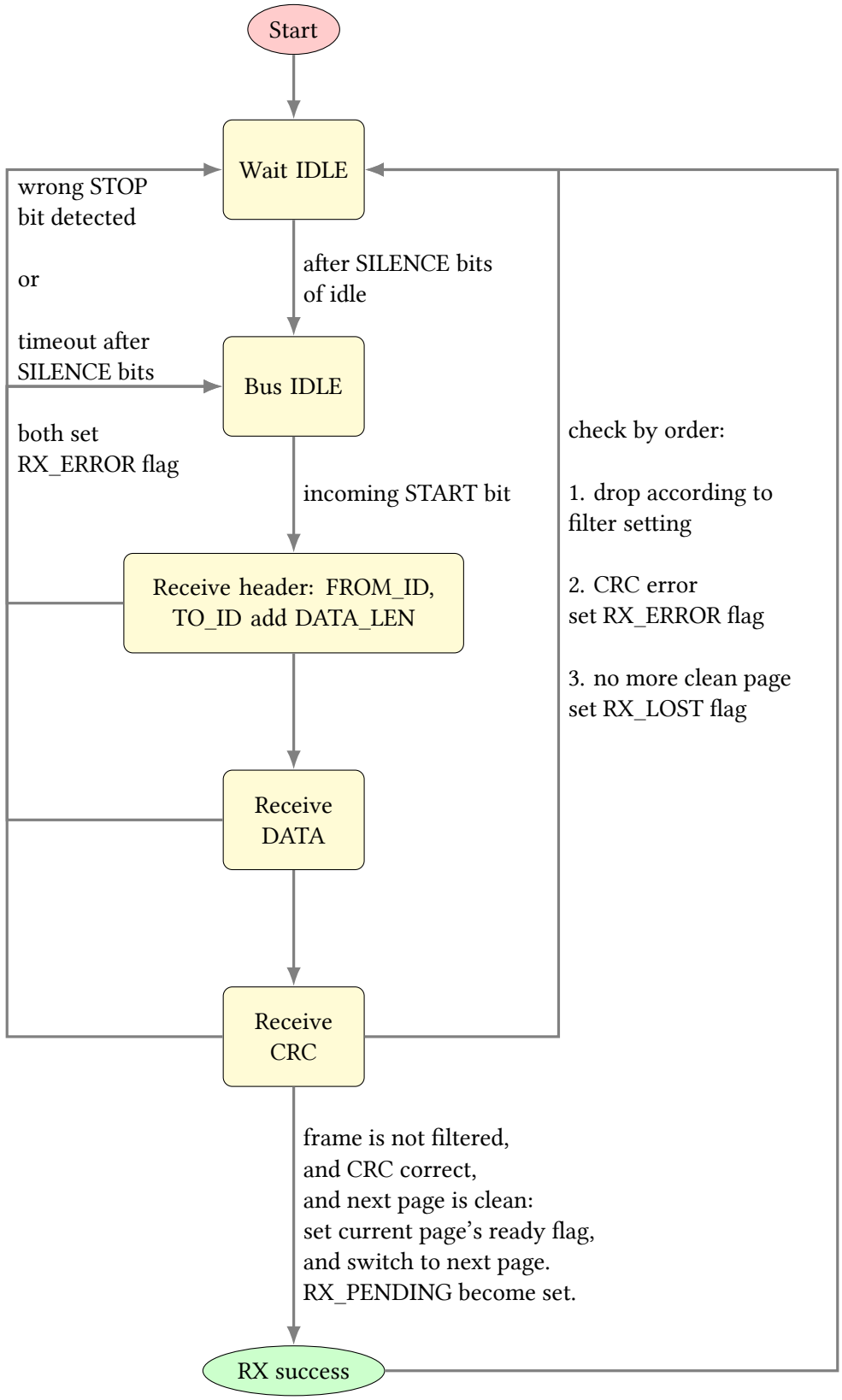
<table>
<tr><td>0x04</td><td>SELF_ID</td><td>RW</td><td>Only used for RX frame filtering: (match from top to bottom)
<table><tr><th>FROM_ID</th><th>TO_ID</th><th>SELF_ID</th><th>Receive or drop</th></tr><tr><td>not care</td><td>not care</td><td>255</td><td>receive (sniffer mode)</td></tr><tr><td>= SELF_ID</td><td>not care</td><td>!= 255</td><td>drop (avoid loopback)</td></tr><tr><td>!= SELF_ID</td><td>255</td><td>not care</td><td>receive (broadcast)</td></tr><tr><td>!= SELF_ID</td><td>!= 255</td><td>= TO_ID</td><td>receive (P2P)</td></tr><tr><td>not care</td><td>!= 255</td><td>!= TO_ID</td><td>drop</td></tr></table>Default: 255.</td></tr>
<tr><td>0x05</td><td>PERIOD_LS_L</td><td>RW</td><td>Low byte of PERIOD_LS (LS: short for Low Speed)
Bond rate corresponding divisor factor for SILENCE, TX_DELAY and FROM_ID (for EN_ADVANCE also).
Formula: $factor = sysclock \div bond\_rate - 1$
E.g. the system clock is 30 MHz, set factor to 259 to get bond rate close to 115200 bps (default: 259).</td></tr>
<tr><td>0x06</td><td>PERIOD_LS_H</td><td>RW</td><td>High byte of PERIOD_LS, 16 bits in total.</td></tr>
<tr><td>0x07</td><td>PERIOD_HS_L</td><td>RW</td><td>Low byte of PERIOD_HS (HS: short for High Speed, default: 259)
Bond rate corresponding divisor factor for TO_ID, DATA_LEN, DATA and CRC_L/H.</td></tr>
<tr><td>0x08</td><td>PERIOD_HS_H</td><td>RW</td><td>High byte of PERIOD_HS, 16 bits in total.</td></tr>
</table>

<table>
<tr><td>0x09</td><td>INT_FLAG</td><td>R</td><td>Interrupt flag:
<table><tr><td>bit0</td><td>BUS_IDLE
Indicate whether the bus is in IDLE.</td></tr><tr><td>bit1</td><td>RX_PENDING
Indicate whether any page of the RX buffer is ready for read
Clear current page's ready flag by write 1 to RX_CTRL [CLR_RX_PENDING] bit.</td></tr><tr><td>bit2</td><td>RX_LOST
Auto set when incoming frame is correct, but droped due to no more page is clean for next RX.
Clear by write 1 to RX_CTRL[CLR_RX_LOST] bit.</td></tr><tr><td>bit3</td><td>RX_ERROR
Auto set when incoming frame is incorrect: stop bit error, timeout or CRC error.
Clear by write 1 to RX_CTRL[CLR_RX_ERROR] bit.</td></tr><tr><td>bit4</td><td>TX_BUF_CLEAN
Indicate whether each page of the TX buffer is clean.</td></tr><tr><td>bit5</td><td>TX_CD
Auto set when collision detected during TX,
Clear by write 1 to TX_CTRL[CLR_TX_CD] bit.
This bit is for debug purpose.</td></tr><tr><td>bit6</td><td>TX_ERROR
After collision detected, the hardware re-send the frame when bus idle for specified time, if collision continue for further 3 times, the transmition will be canceled, and this flag will be set.
Clear by write 1 to TX_CTRL[CLR_TX_ERROR] bit.</td></tr></table></td></tr>
<tr><td>0x0A</td><td>INT_MASK</td><td>RW</td><td>Interrupt mask
INT_N output low if INT_FLAG & INT_MASK != 0, ouput Hi-Z otherwise (default: 0x00).</td></tr>
<tr><td>0x0B</td><td>RX</td><td>R</td><td>Read data from internal RX buffer page, address pointer auto increase
There are 8 buffer pages for RX purpose with each 256 bytes.
On hardware side, when receive a wanted frame: if next page is clean for next receive, set current page's ready flag and switch to the next page; else drop the frame and set RX_LOST bit.
On user side, the RX_PENDING bit indicated that there are some pages ready for read, write 1 to CLR_RX_PENDING bit clear current page's ready flag and switch to next page. write 1 to RST_RX bit clear all page's ready flag by reset both RX buffer and RX logic.
RX_PENDING bit is clean if all page's ready flag are clean.</td></tr>
</table>

0x0C	TX	W	Write data to internal TX buffer page, address pointer auto increase There are 2 buffer pages for TX purpose with each 256 bytes. On user side, when finish writing frame to current page, user should wait for TX_BUF_CLEAN bit be set, then trigger START_TX bit to set current page's ready flag and move to next page (nothing happen if TX_BUF_CLEAN bit is clean). On hardware side, the hardware gonna start TX if page's ready flag was set, clear the flag and waiting for next page when transmit completed. TX_BUF_CLEAN bit is set if all page's ready flag are clean.
0x0D	RX_CTRL	W	RX control: bit0 RST_RX_POINTER Write 1 to reset the read pointer of current RX page. bit1 CLR_RX_PENDING (auto select bit0) bit2 CLR_RX_LOST bit3 CLR_RX_ERROR bit4 RST_RX (auto select bit0, 2, 3)
0x0E	TX_CTRL	W	TX control: bit0 RST_TX_POINTER Write 1 to reset the write pointer of current TX page. bit1 START_TX (auto select bit0) bit3 CLR_TX_CD bit4 CLR_TX_ERROR
0x0F	RX_ADDR	RW	Set and get the read pointer of current RX page.
0x10	RX_PAGE_FLAG	R	(only used if NO_DROP is set). Value zero indicate the frame in current RX page is correct; Non-zero indicate the pointer of last received byte of the disturbed frame, include CRC.

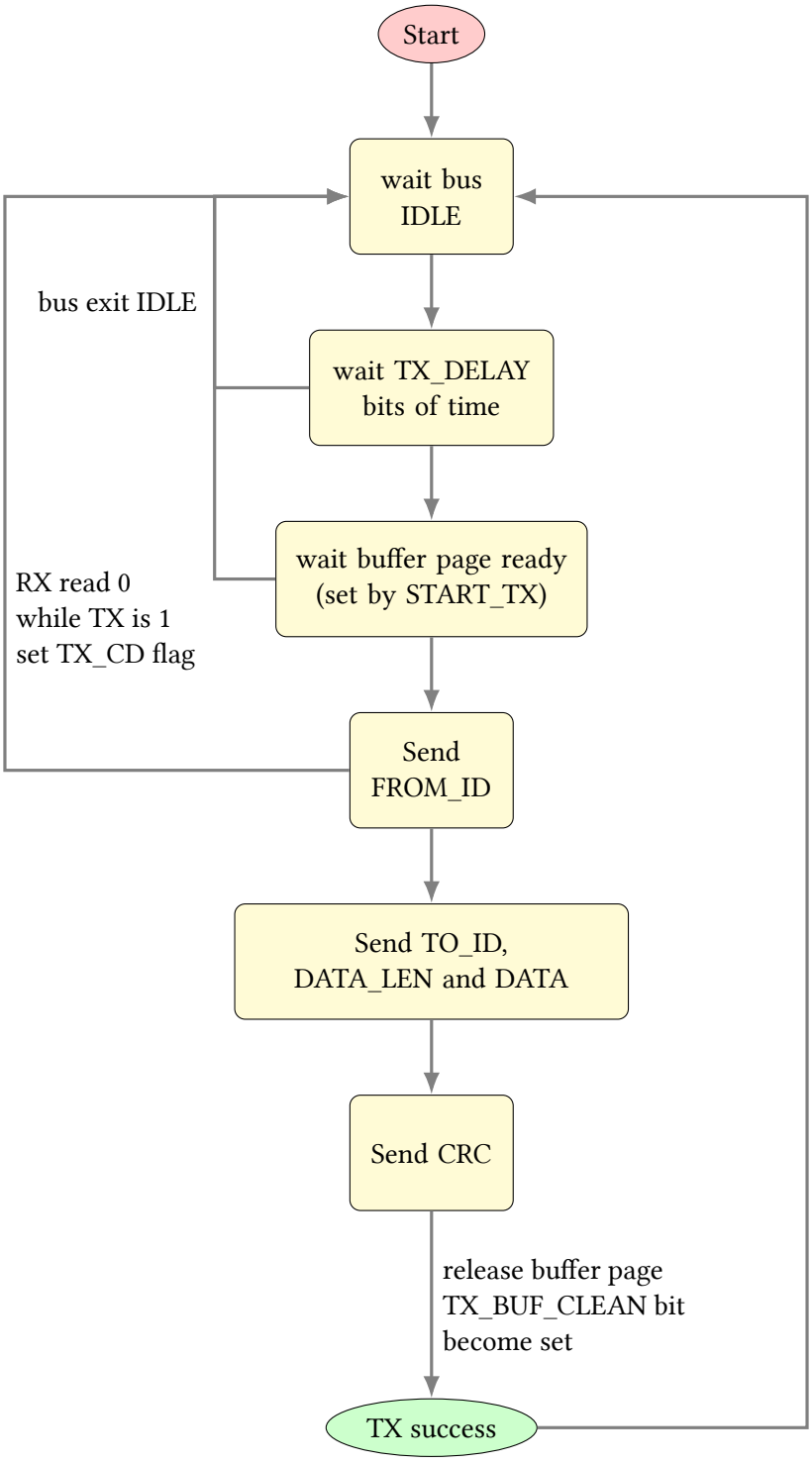
5 Workflow

5.1 RX



Not set RX_ERROR flag if current frame is gonna filtered or less than two bytes received.

5.2 TX



6 Peripheral Interface

Both SPI and I2C frequency should less than $sysclock \div 10$.

We usually read or write only 1 byte except reg RX and TX.

6.1 SPI

Read and write:

```
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.

Write:

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

Read:

```
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 Operate Demonstration

7.1 Init

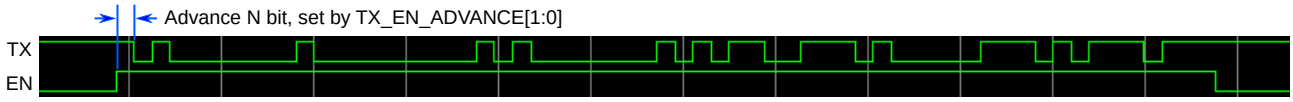
```
// enable OUTPUT
cd_write(REG_SETTING, TX_PUSH_PULL);

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

// set bondrate
```

```
cd_write(REG_PERIOD_LS_L, 39); // 750000 bps
cd_write(REG_PERIOD_LS_H, 0);
cd_write(REG_PERIOD_HS_L, 2);  // 10 Mbps
cd_write(REG_PERIOD_HS_H, 0);

// clean RX buffer
cd_write(REG_RX_CTRL, RST_RX);

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

7.1.1 Compatible and Conventional Mode

Set same value to reg PERIOD_LS and PERIOD_HS for compatible mode.

Set NO_ARBITRATE bit further for conventional mode:



7.2 TX

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

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

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

7.3 RX

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

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

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

8 Copyright Statement

CDBUS is a fairly open protocol, hardware implementation is relatively simple, in addition to chip manufacturers need to pay a small amount of royalties, the rest of anyone can use this protocol and its variants for free, only need to retain the original copyright information in the product manual.

Contact: info@dukelec.com