

# CDCTL01A Datasheet

## 1 Description

CDCTL01A is an SPI interface UART controller, with the UART port hardcoded to use the CDBUS protocol. (Or simply an SPI interface CDBUS controller.)

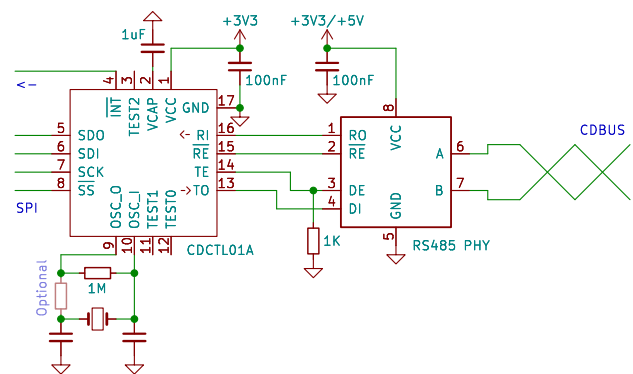
CDBUS is a simple protocol designed for serial ports and serial bus communication, commonly employed in RS-485 buses. It utilizes hardware arbitration and other mechanisms to avoid data conflicts, allowing nodes to freely transmit and receive data packets, thereby overcoming the limitations of single-master polling.

## 2 Features

- In addition to RS-485 communication, this chip also supports M-LVDS, single-wire UART communication, UART expansion, and more.
- Supports multi-master communication, peer-to-peer communication, multicast communication, etc.
- Supports multiple modes such as dual-rate arbitration mode, single-rate high-speed peer-to-peer mode, traditional mode, and full-duplex mode.
- 50Mbps maximum UART baud rate.
- 50MHz maximum SPI clock.
- 8 packet buffers for receive, 2 packet buffers for transmit. (Each buffer is 256 bytes.)
- Small package: QFN16 3x3mm.
- 3.3V ( $\pm 10\%$ ) single power supply.
- Operating temperature range:  $-40 \sim 125$  °C.
- Crystal oscillator or external clock input.
- 4-wire or 3-wire SPI (SDO and SDI connect together).
- 5V tolerant RI input.
- Ultra-wide system clock range: 32KHz  $\sim$  150MHz.
- Lead(Pb)-free / RoHS-compliant.

## 3 Applications

- Industrial Automation
- Robotics
- Automotive
- Smart City
- Consumer Electronics
- IoT



Reference Schematic

(For a crystal, the 1M $\Omega$  resistor is required; the series resistor ( $\sim 240\Omega$ ) is optional for amplitude adjustment.)

## 4 CDBUS Protocol

CDBUS has a 3-byte header: [src\_addr, dst\_addr, data\_len], followed by user data, and concluding with a 2-byte CRC (identical to MODBUS CRC).

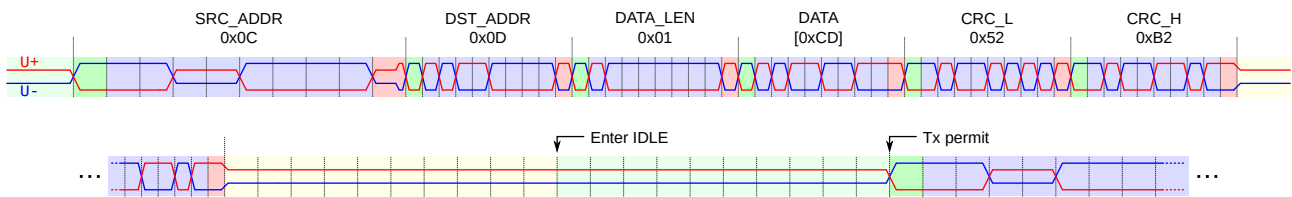
The CDBUS protocol at the byte level can be directly used for traditional serial communication, such as traditional UART, RS-232, RS-485, and USB virtual serial ports.

The full CDBUS protocol at the bit level requires dedicated hardware controllers (or software emulation) to achieve conflict avoidance, higher speeds, and strong real-time performance.

### 4.1 Arbitration Mode (CDBUS-A)

- It introduces an arbitration mechanism, similar to the CAN bus, to automatically avoid conflicts.
- Supporting dual baud rates achieves high-speed communication, with the maximum baud rate during the high-speed phase up to:  $\text{sysclk} \div 3$ . (e.g. 50 Mbps for 150 MHz sysclk.)
- Supports unicast, multicast and broadcast.
- The maximum user data size is 253 bytes.
- Hardware packing, unpacking, verification and filtering, saving your time and CPU usage.
- Compatible with traditional RS-485 hardware (arbitration function remains effective).

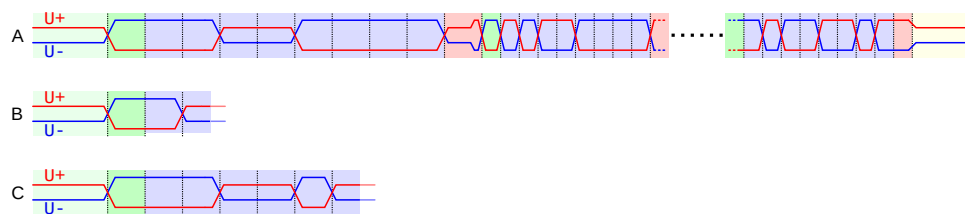
The protocol timing example, consisting of only one byte of user data:  
(You can configure the lengths of idle time and transmission permit time.)



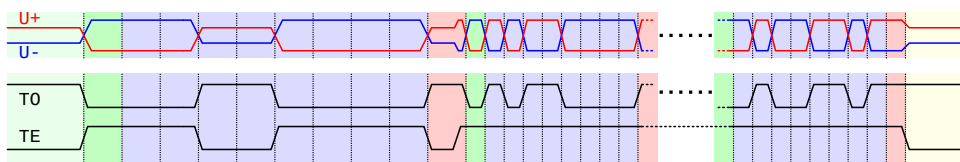
Tips:

- When a high-priority node needs to send unimportant data, we can dynamically increase the transmission permit time (TX\_PERMIT\_LEN).

Arbitration example:



Example waveforms for TO and TE pins:

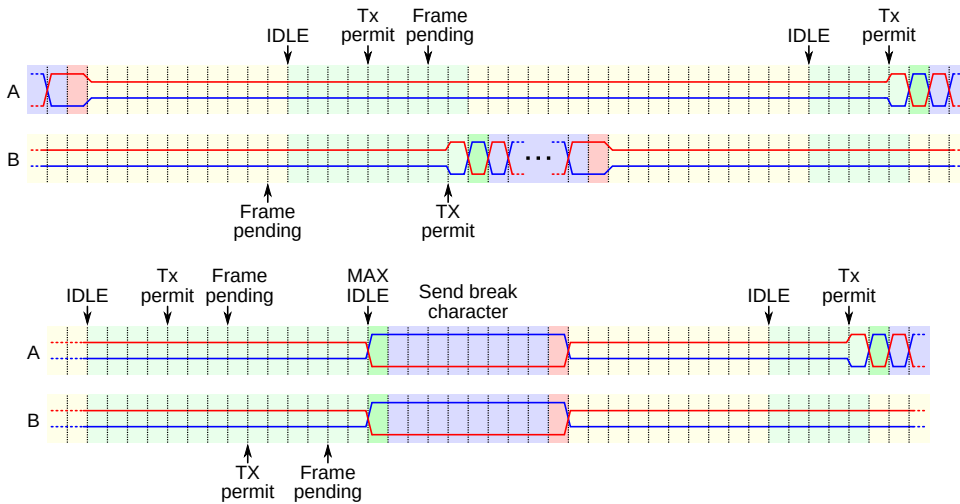


The RX data sampling point for reception is at 1/2 bit; for TX readback, it's at 3/4 bit.

## 4.2 Break Sync Mode (CDBUS-BS)

In CDBUS-A mode, if the low-speed portion takes longer, it can become a bottleneck for communication efficiency. To address this, single-rate peer-to-peer bus communication can be achieved using CDBUS-BS mode:

- Different nodes have different configurations for the TX\_PERMIT\_LEN parameter, requiring sufficiently large differences to avoid conflicts.
- If any node has pending data frames waiting to be sent before the transmission permit point, transmission starts from that point.
- Otherwise, wait until the idle time exceeds MAX\_IDLE\_LEN. When there are pending data frames waiting for transmission, first send a break character to bring the bus out of the idle state.

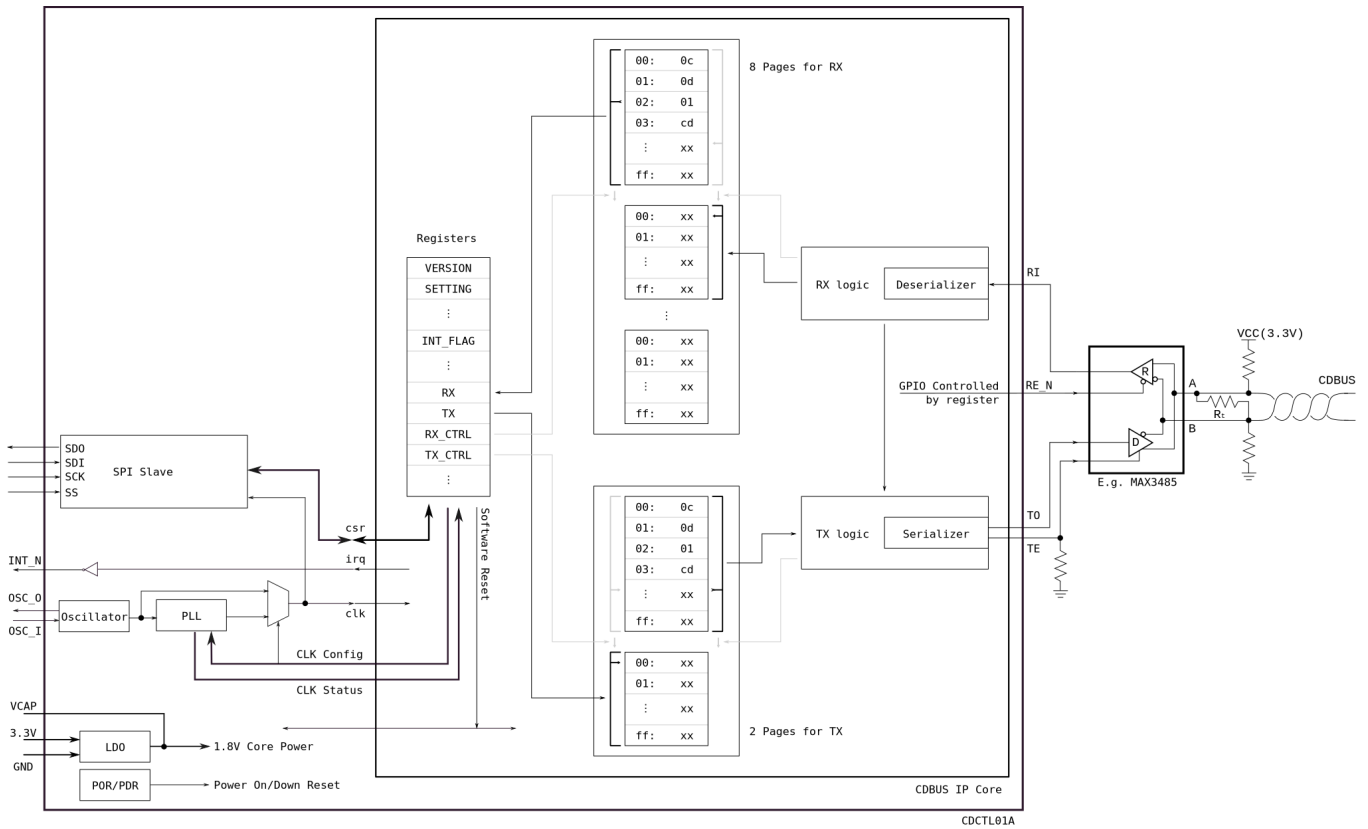


The CDBUS-BS mode is suitable for high-speed applications with few nodes, and it is also suitable for software implementation.

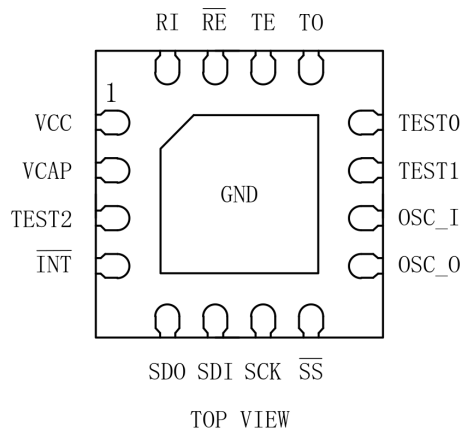
## 5 CDBUS IP Core

Chip source code and details: <https://cdbus.org>

## 5.1 Block Diagram



## 6 Pin Definition

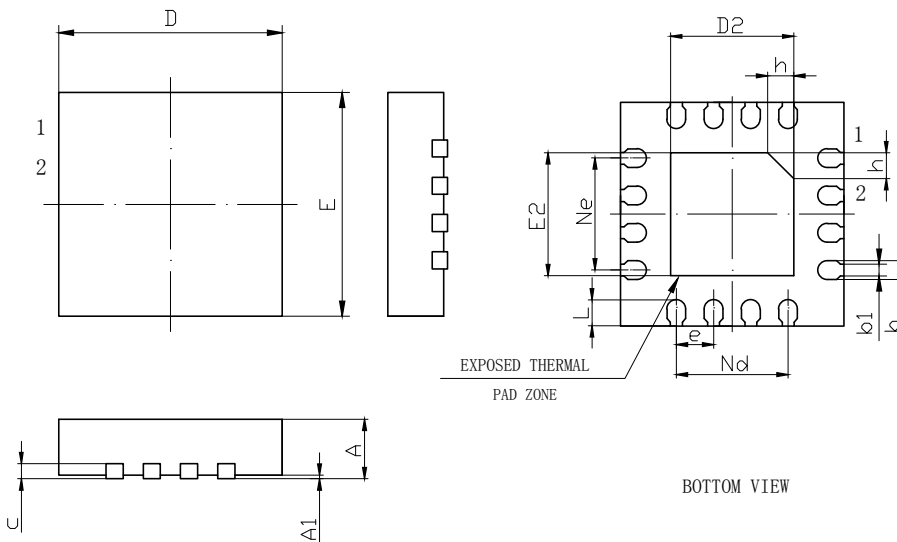


No	Name	I/O	Internal Pull	Description
17	GND			Ground
1	VCC			Supply voltage, 3.3V ( $\pm 10\%$ ), bypass with 100nF ceramic capacitor to ground
2	VCAP			Internal 1.8V ( $\pm 10\%$ ) 100mA LDO output, bypass with 1uF ceramic capacitor to ground

3, 11, 12	TESTx	I	Down	Left empty or connect to ground
4	INT	O	Up	Interrupt pin, open-drain (default) or push-pull output
5	SDO	O	-	SPI MISO
6	SDI	I	-	SPI MOSI
7	SCK	I	-	SPI clock
8	SS	I	-	SPI chip select
9	OSC_O	O	-	OSC output (left empty when using external clock input)
10	OSC_I	I	-	OSC input or external clock input
13	TO	O	-	TX output, connect to transmit pin of RS-485 PHY
14	TE	O	-	TX enable, connect to transmit enable pin of RS-485 PHY
15	RE	O	-	GPIO push-pull output, typically used to control the receive enable pin of the RS-485 PHY
16	RI	I	-	RX input, connect to receive pin of RS-485 PHY (5V tolerant)

## 7 Specifications

### 7.1 Mechanical Specifications



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.80	0.85	0.90
A1	—	0.02	0.05
b	0.18	0.25	0.30
b1	0.11	0.16	0.21
c	0.10	0.15	0.20
D	2.90	3.00	3.10
D2	1.55	1.65	1.75
e	0.50BSC		
Ne	1.50BSC		
Nd	1.50BSC		
E	2.90	3.00	3.10
E2	1.55	1.65	1.75
L	0.30	0.35	0.40
h	0.30	0.35	0.40

### 7.2 Absolute Maximum Ratings

Parameter	Min.	Max.
Storage Temperature (Ambient)	-55 °C	150 °C
Electrostatic Discharge Human Body Model (ESD-HBM) <sup>1</sup>	-	±2000 V
Electrostatic Discharge Charged Device Model (ESD-CDM) <sup>2</sup>	-	±750 V
High Temperature Latch-Up <sup>3</sup>	-	±200 mA / +1.5 VccMax

1. Refer to AEC-Q100-002. Zap 1 pulse with 0.3 sec interval.

2. Refer to AEC-Q100-011.

3. Refer to JESD78F:2022, Ta=+125°C.

### 7.3 Operating Conditions

Parameter	Min.	Max.
Operation Temperature	-40 °C	125 °C

## 7.4 DC Electrical Characteristics

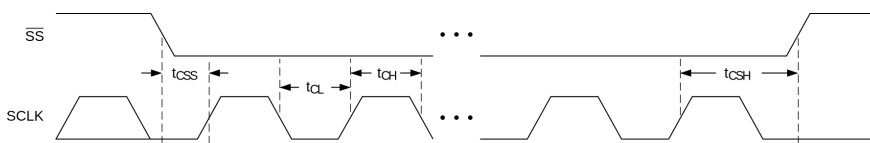
Parameter	Min.	Typ.	Max.
Supply Voltage VCC	-10%	3.3 V	+10%
V <sub>IL</sub>	-0.3 V	-	0.8 V
V <sub>IH</sub>	2.0 V	-	3.6 V
V <sub>IH</sub> (RI pin only)	2.0 V	-	5.5 V
V <sub>OL</sub>	-	-	0.4 V
V <sub>OH</sub>	2.4 V	-	-
I <sub>OL</sub> , I <sub>OH</sub>	-	12 mA	-
I <sub>OL</sub> , I <sub>OH</sub> (OSC_O pin only)	-	2 mA	-
Input or I/O Leakage	-	-	+/-10 uA
I <sub>VCC</sub> (F <sub>sys</sub> = 32KHz)	-	140 uA	-
I <sub>VCC</sub> (F <sub>sys</sub> = 12MHz, pll off)	-	2.8 mA	-
I <sub>VCC</sub> (F <sub>sys</sub> = 12MHz, pll on)	-	8.7 mA	-
I <sub>VCC</sub> (F <sub>sys</sub> = 60MHz)	-	10.6 mA	-
I <sub>VCC</sub> (F <sub>sys</sub> = 150MHz)	-	14.3 mA	-

## 7.5 Timing Specifications

Symbol	Parameter	Min.	Max.
F <sub>CRYSTAL</sub>	Crystal frequency	4 MHz	32 MHz
F <sub>EXT</sub>	External clock input	32 KHz	32 MHz
F <sub>SYS</sub>	System clock frequency	32 KHz	150 MHz
F <sub>SCK</sub>	SPI clock frequency	-	50 MHz
F <sub>UART</sub>	Baud rate	-	50 Mbps

Additional restrictions on RX and TX register reads and writes:  $F_{SCK} \leq F_{SYS} \times 80\%$

Writing to any register requires:  $t_{CSH} \geq \frac{1}{F_{SYS} \times 80\%}$



## 8 Register Reference

Register Name	Addr	Access	Default	Description (8-bit width by default if not specified)
VERSION	0x00	RD	0x10	Hardware version
CLK_CTRL	0x01	RD/WR	0x00	Clock Control
SETTING	0x02	RD/WR	0x10	Configs
IDLE_WAIT_LEN	0x04	RD/WR	0x0a	Waiting time to enter idle
TX_PERMIT_LEN_L	0x05	RD/WR	0x14	Waiting time to allows sending (10 bits)
TX_PERMIT_LEN_H	0x06	RD/WR	0x00	
MAX_IDLE_LEN_L	0x07	RD/WR	0xc8	Max idle waiting time in BS mode (10 bits)
MAX_IDLE_LEN_H	0x08	RD/WR	0x00	

TX_PRE_LEN	0x09	RD/WR	0x01	Enable TE how long ahead than TO output (2 bits)
FILTER	0x0b	RD/WR	0xff	Local address
DIV_LS_L	0x0c	RD/WR	0x67	Low-speed rate setting (16 bits)
DIV_LS_H	0x0d	RD/WR	0x00	
DIV_HS_L	0x0e	RD/WR	0x67	High-speed rate setting (16 bits)
DIV_HS_H	0x0f	RD/WR	0x00	
INT_FLAG	0x10	RD	n/a	Status
INT_MASK	0x11	RD/WR	0x00	Interrupt mask
RX	0x14	RD	n/a	Read RX page
TX	0x15	WR	n/a	Write TX page
RX_CTRL	0x16	WR	n/a	RX control
TX_CTRL	0x17	WR	n/a	TX control
RX_ADDR	0x18	RD/WR	0x00	RX page read pointer (rarely used)
RX_PAGE_FLAG	0x19	RD	n/a	RX page flag
FILTER_M0	0x1a	RD/WR	0xff	Multicast filter0
FILTER_M1	0x1b	RD/WR	0xff	Multicast filter1
PLL_ML	0x30	RD/WR	0x12	PLL M[7:0] (M: 9 bits)
PLL_OD_MH	0x31	RD/WR	0x20	PLL OD and M[8]
PLL_N	0x32	RD/WR	0x00	PLL N (5 bits)
PLL_CTRL	0x33	RD/WR	0x01	PLL Control
PIN_INT_CTRL	0x34	RD/WR	0x00	$\overline{\text{INT}}$ pin Control
PIN_RE_CTRL	0x35	RD/WR	0x00	$\overline{\text{RE}}$ pin Control
CLK_STATUS	0x36	RD	0x01	Clock status

### 8.1 CLK\_CTRL:

FIELD	DESCRIPTION
[7]	Soft reset: write 1 to reset device
[0]	Clock select: 0: OSC input, 1: PLL output

### 8.2 SETTING:

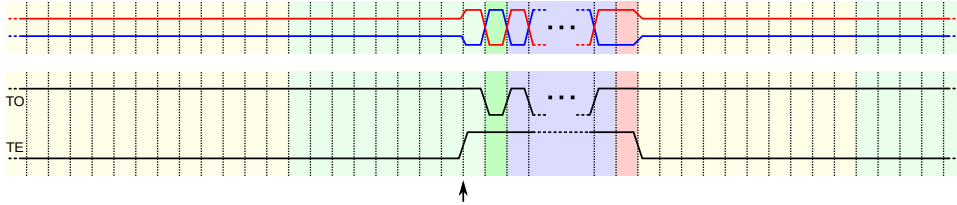
FIELD	DESCRIPTION
[6]	Full duplex mode
[5]	Break Sync mode
[4]	Enable arbitration
[3]	Save broken frame
[2]	CRC maintained by user
[1]	Invert TO pin output
[0]	Enable push-pull output for TO and TE pin

The TO pin defaults to an open-drain output and the TE pin defaults to a high-resistance output.

[6]	[5]	[4]	DESCRIPTION
0	0	0	Traditional half-duplex mode
0	0	1	CDBUS-A mode (default)
0	1	0	CDBUS-BS mode
1	0	0	Full-duplex mode

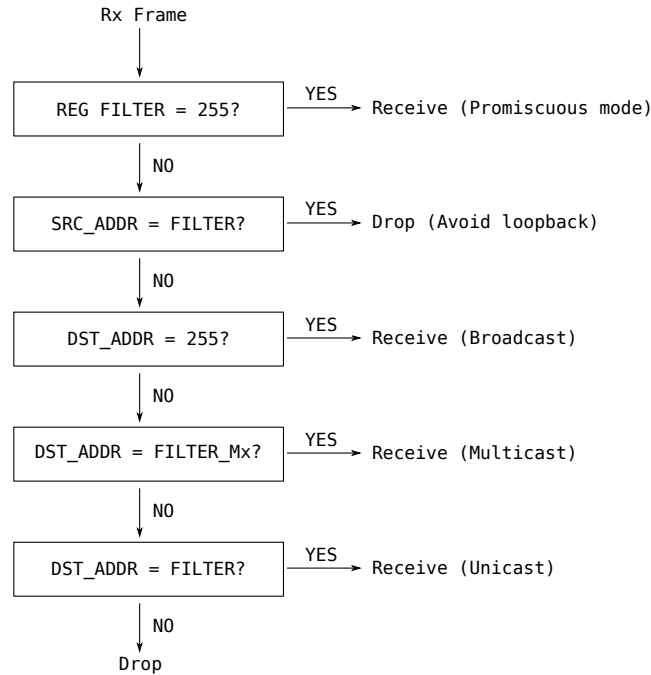
### 8.3 TX\_PRE\_LEN:

Example waveforms for TO and TE pins (TX\_PRE\_LEN = 1 bit):



Unused for Arbitration mode and the break character automatically generated by BS mode.

#### 8.4 FILTERS:



The default value of FILTER\_Mx (0xff) indicates it is disabled. Any address other than 0xff can be used as a multicast address.

#### 8.5 DIV\_xx\_x:

Baud rate divider value:

$$div\_xx[15 : 0] = \frac{sysclk}{baudrate} - 1$$

The minimum value is 2.

For single rate, DIV\_HS needs to be set to the same value as DIV\_LS.

#### 8.6 INT\_FLAG:

FIELD	DESCRIPTION
[7]	1: TX error: TX is 0, but RX is sampled as 1
[6]	1: TX collision detected
[5]	1: TX page released by hardware
[4]	1: RX error: frame broken
[3]	1: RX lost: no empty page for RX



- [2] 1: Break character received
- [1] 1: RX page ready for read
- [0] 1: Bus in IDLE mode

Reading this register will automatically clear bit7, bit6, bit4, bit3 and bit2.

### 8.7 INT\_MASK:

```
irq = ((INT_FLAG & INT_MASK) != 0)
int_n = !irq
```

### 8.8 RX\_CTRL:

FIELD	DESCRIPTION
[4]	Reset RX block
[1]	Switch RX page
[0]	Reset RX page read pointer

### 8.9 TX\_CTRL:

FIELD	DESCRIPTION
[5]	Send break character
[4]	Abort TX
[1]	Switch TX page
[0]	Reset TX page write pointer

### 8.10 RX\_PAGE\_FLAG:

A value of zero indicates that the frame in the current RX page is correct;  
Non-zero indicates the pointer to the last received byte of the disrupted frame, including CRC.  
Always zero if 'save broken frame' is not enabled.

### 8.11 PLL\_OD\_MH:

FIELD	DESCRIPTION
[5:4]	PLL OD
[0]	PLL M[8]

### 8.12 PLL\_CTRL:

FIELD	DESCRIPTION
[4]	PLL enable: 0: disable pll, 1: turn on pll
[0]	PLL sleep: 1: sleep, 0: run

Reserved bits in this register must remain zero.

PLL output clock:

$$pll\_output = \frac{osc\_input}{pll\_n + 2} \times (pll\_m + 2) \times \frac{1}{2^{(pll\_od[1] + pll\_od[0])}}$$

When the OSC input is equal to 12MHz, the default PLL parameters correspond to an output of 60MHz.

The PLL parameters needs to meet the following conditions:

$$1MHz \leq \frac{osc\_input}{pll\_n + 2} \leq 15MHz$$

$$100MHz \leq \frac{osc\_input}{pll\_n + 2} \times (pll\_m + 2) \leq 500MHz$$

Switch to using the PLL clock:

- Change PLL N, M and OD values if need
- Write 0x10 to PLL\_CTRL
- Write 0x01 to CLK\_CTRL

### 8.13 PIN\_INT\_CTRL:

FIELD	DESCRIPTION
[4]	Output mode: 0: open-drain, 1: push-pull

### 8.14 PIN\_RE\_CTRL:

FIELD	DESCRIPTION
[4]	Output mode: 0: disabled (high impedance), 1: push-pull
[0]	Output value: 0: low, 1: high

### 8.15 CLK\_STATUS:

FIELD	DESCRIPTION
[2]	Clock switching status: 1: completed, 0: in progress
[1]	System clock is PLL output
[0]	System clock is OSC input

## 9 SPI Interface

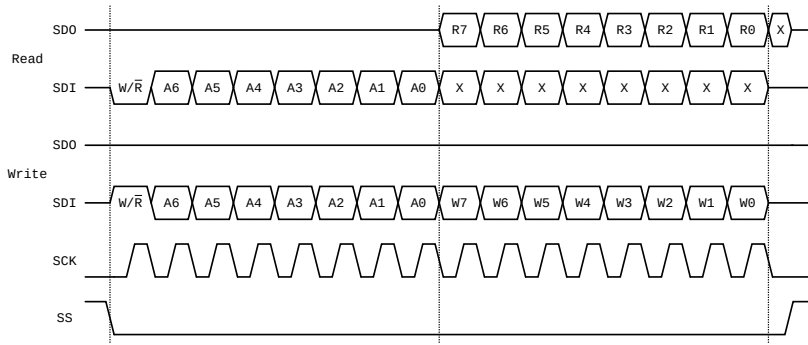
Burst read and write are useful for accessing REG\_RX and REG\_TX.

Read or write depend by bit W/ $\bar{R}$ :

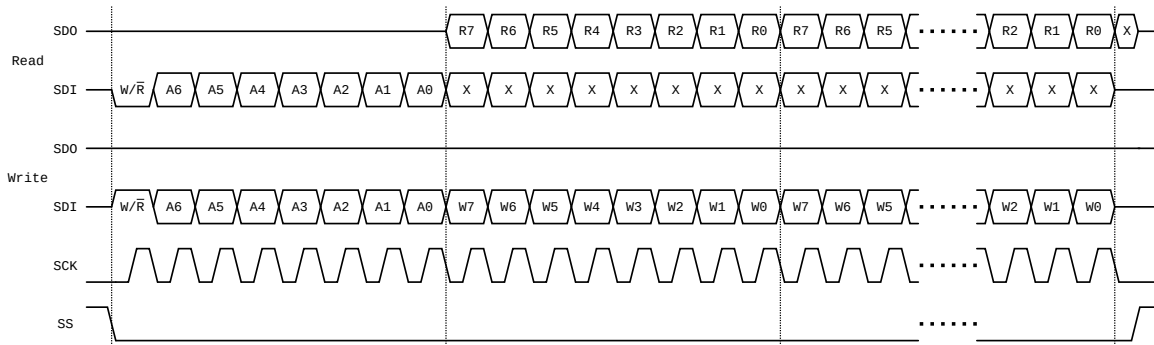
- 0: Read  
1: Write

FIELD	DESCRIPTION
Ax	Register address
Wx	Write data
Rx	Read data
X	Don't care

Read or write single byte:



Burst read or write:



## 10 Operate Demonstration

### 10.1 Init

```
cd_write(REG_CLK_CTRL, 0x80);           // Soft reset
cd_write(REG_PIN_RE_CTRL, 0x10);        // Set RE_N pin to low
cd_write(REG_SETTING, 0x11);             // Enable push-pull output
cd_write(REG_FILTER, 0x0c);             // Set FILTER

// Set baudrates, LS and HS should be identical for single-rate
cd_write(REG_DIV_LS_L, 11);              // 1 Mbps @ 12MHz sysclk
cd_write(REG_DIV_LS_H, 0);
cd_write(REG_DIV_HS_L, 2);               // 4 Mbps @ 12MHz sysclk
cd_write(REG_DIV_HS_H, 0);

// Reset RX module to ignore errors received before baud rate is set (optional)
// cd_write(REG_RX_CTRL, 0x11);

// Enable interrupts (optional, MCU can also poll INT_FLAG periodically)
// cd_write(REG_INT_MASK, BIT_FLAG_TX_ERROR | BIT_FLAG_RX_ERROR \
// | BIT_FLAG_RX_LOST | BIT_FLAG_RX_PENDING);
```

Allow the chip to complete its startup after power-up. Wait 50ms or until the correct VERSION is read before initializing.

### 10.2 TX

```
uint8_t tx_buf[] = {
    0x0c, 0x0d, 0x02,           // src_addr, dst_addr, data_len
```

```
    0x01, 0x00                // 2-byte example data, no CRC
};

cd_write_chunk(REG_TX, tx_buf, tx_buf[2] + 3); // Write CDBUS frame without CRC
while (!(cd_read(REG_INT_FLAG) & 0x20));      // Wait for prior TX frame to complete
cd_write(REG_TX_CTRL, 0x03);                  // Submit frame to start TX
```

The FILTER register is used only for receive filtering and does not affect transmission. For transmission, the sender should use its own address as the frame's `src_addr`, non-own addresses are typically used for loopback tests to bypass filtering.

### 10.3 RX

```
while (!(cd_read(REG_INT_FLAG) & 0x02));      // Wait for RX page ready
cd_read_chunk(REG_RX, rx_buf, 3);              // Read frame header
cd_read_chunk(REG_RX, rx_buf + 3, rx_buf[2]);  // Read frame data
cd_write(REG_RX_CTRL, 0x03);                  // Release RX page
```

Reading the packet header and payload can also be done within a single  $\overline{SS}$  low period.

## 11 Copyright Statement

The CDBUS protocol is royalty-free for everyone except chip manufacturers.  
Copyright (c) 2026 DUKELEC, All rights reserved.

## 12 Contact Information

- Sales and customer support: [sales@dukelec.com](mailto:sales@dukelec.com)
- Technical support: [support@dukelec.com](mailto:support@dukelec.com)
- Business corporation: [info@dukelec.com](mailto:info@dukelec.com)
- Website: <https://dukelec.com>

## 13 Change History

- 20260123 (v1.3): Add filtering flowchart and sample code notes.
- 20240304 (v1.2): Modify the introduction description.
- 20231125 (v1.1): Add ESD and Latch-Up related data.
- 20230802 (v1.0): Init.