Introduction to the CDBUS

DUKELEC

August 14, 2017

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.

Follow tables compares CDBUS to other widely used fieldbus:

<table>
<tr><td>OSI layer</td><td colspan="2">Modbus</td><td>PROFIBUS</td><td colspan="2">Others</td><td>CDBUS</td><td colspan="2">CAN bus</td></tr>
<tr><td>Application</td><td colspan="2">Modbus application protocol</td><td>PROFIBUS

DPV0,1,2</td><td colspan="2">Other application protocols</td><td>Private, Modbus, etc...</td><td colspan="2">CAN application protocol

CANopen-Lift, J1939, etc...</td></tr>
<tr><td>Presentation</td><td colspan="2" rowspan="3">Empty</td><td rowspan="3">Empty</td><td colspan="2" rowspan="3">Almost empty</td><td rowspan="3">Optional</td><td colspan="2" rowspan="3">Optional
e.g.: CANopen</td></tr>
<tr><td>Session</td></tr>
<tr><td>Transport</td></tr>
<tr><td>Network</td><td colspan="2"></td><td></td><td colspan="2"></td><td></td><td colspan="2"></td></tr>
<tr><td rowspan="2">Data Link</td><td colspan="2">Modbus serial line protocol</td><td rowspan="2">PROFIBUS

FDL (token passing)</td><td colspan="2">Other serial line protocols</td><td rowspan="2">CDBUS controller:

CDCTL-XX</td><td rowspan="2">CAN 2.0 controller</td><td rowspan="2">CAN FD controller</td></tr>
<tr><td>HEX (RTU)</td><td>ASCII</td><td>HEX (FF AA ...)</td><td>ASCII</td></tr>
<tr><td>Physical</td><td colspan="5">RS485 transceiver & cable</td><td colspan="2">CAN transceiver & cable</td></tr>
</table>

- Orange color: implemented in software;
- Green color: implemented in hardware.

We don't compares to ethernet-based industrial buses, they are more complex and expensive aim for different markets.

Fieldbus	Max devices	Max speed	Payload	Latency	Multi-master	Decentralised (peer-to-peer)	CPU consumption	Stand alone controller	Easy to use
Modbus RTU	254	depend on RS485 transceiver	252 bytes	high (wait for poll)	no	no	high	no	medium
PROFIBUS DP	126	12 Mbps	244 bytes	medium (wait for token)	yes (token passing)	yes	high	no	hard
Others	--	depend on RS485 transceiver	--	medium or high	may by byte wise arbitration, by frame wise verification, by token or not	may or may not	high	no	medium
CDBUS	255	10 Mbps for typical, support higher if need	253 bytes	low	yes (bitwise arbitration)	yes	low	yes	easy
CAN 2.0	127 for CANopen, or higher for other	1 Mbps	8 bytes	low	yes (bitwise arbitration)	yes	low	yes	hard
CAN FD		12 Mbps, max 3.7 Mbps for typical	64 bytes	low	yes (bitwise arbitration)	yes	low	no yet	hard

1.1 Extra disadvantages for CAN bus

CAN is a message-based protocol, message IDs must be unique on a single CAN bus, that means one message could only send by a single node; e.g. if there are three limit-switchs to stop a servo, we must define three stop messages: "STOP1" for switch1, "STOP2" for switch2 and so on, then the servo getting stoped when receiving any of those three messages;

Entire bus share one message ID allocation space, in order to modify message definition for one device must take care of all other devices.

1.2 Extra disadvantages for data link layer by software

Disadvantages for frame marked by flag like "`ff aa`": the flag is sometimes repeat with the subsequent data, if payload data just also has a "`ff aa`", when the frame error occurs once the data dislocation, the receiver think that the "`ff aa`" in the payload is a flag, and this may cause the error to continue.

Disadvantages for frame marked by IDLE state: we normally can't using FIFO or DMA to receiving data, because frames will stick together in the RAM and can not be separated by time anymore.

Without using FIFO and DMA, it's not only increase the CPU consumption, but also requires a higher real-time ability, it will be difficult to specify the interrupt priority order of data reception and main task: if specify the data reception a higher priority, the main task will be disturbed each time when the data comming, but if the receiving priority is lower, then we may always lose data.

Transmission and reception of frame using hardware could avoiding those problems.

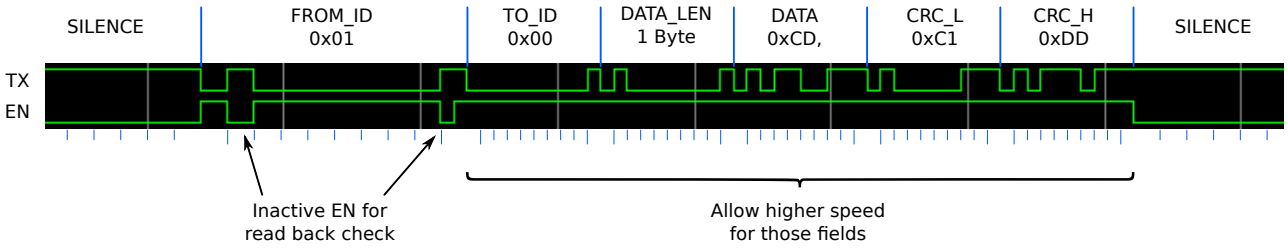
2 Highlights

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

3 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

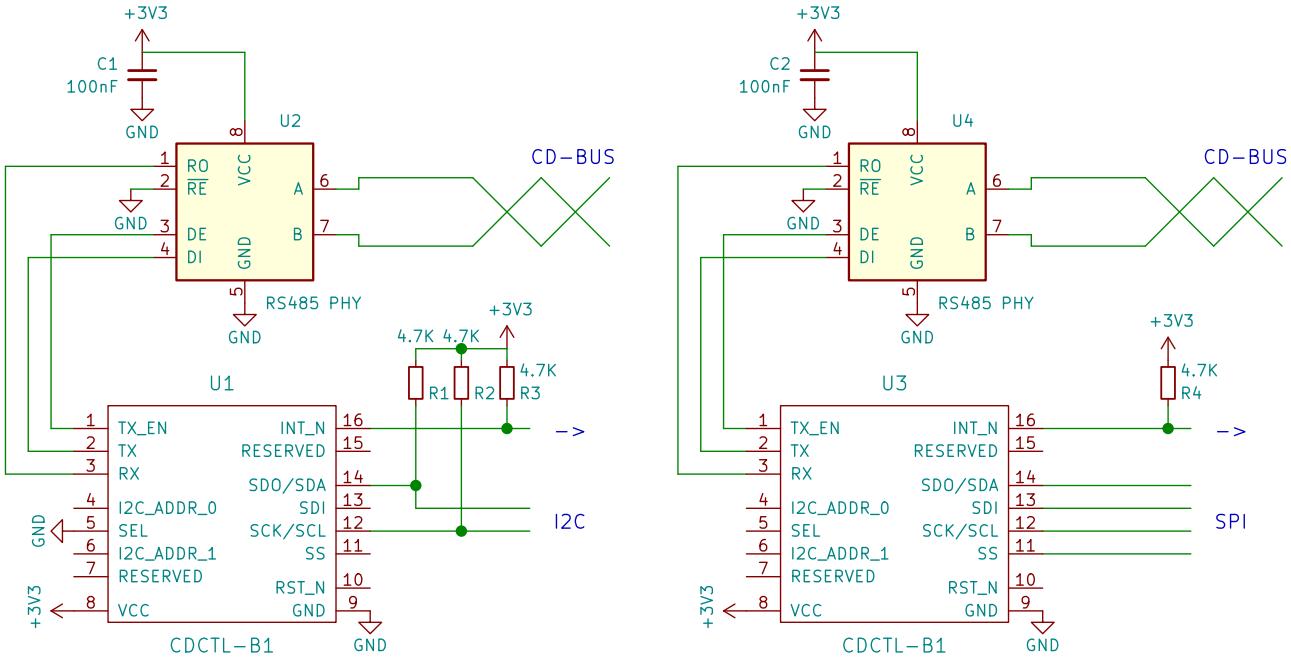
CDBUS protocol only defines the frame format, does not specify the payload data format; Only supports unicast and broadcast, does not support multicast; Only provide hardware arbitration, automatic retransmission after conflict, handshake and error handling are handled by software at upper layer.

If we set the two baud rates to the same value, CDBUS node can communicate with the traditional RS485 node while maintaining the arbitration mechanism: the traditional node priority is set higher than CDBUS nodes, when collision is detected, the traditional node takes precedence. Of course, you can also turn off the arbitration function, to fully use the traditional communication mode.

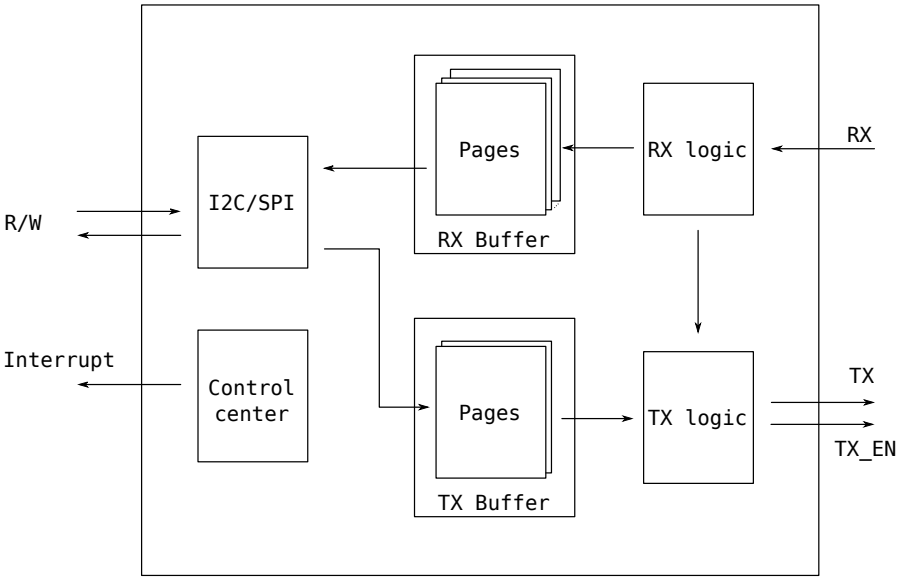
CDBUS and the CDCTL-XX controller could be used for physical medium other than RS485, e.g.: single wire UART bus.

4 Hardware

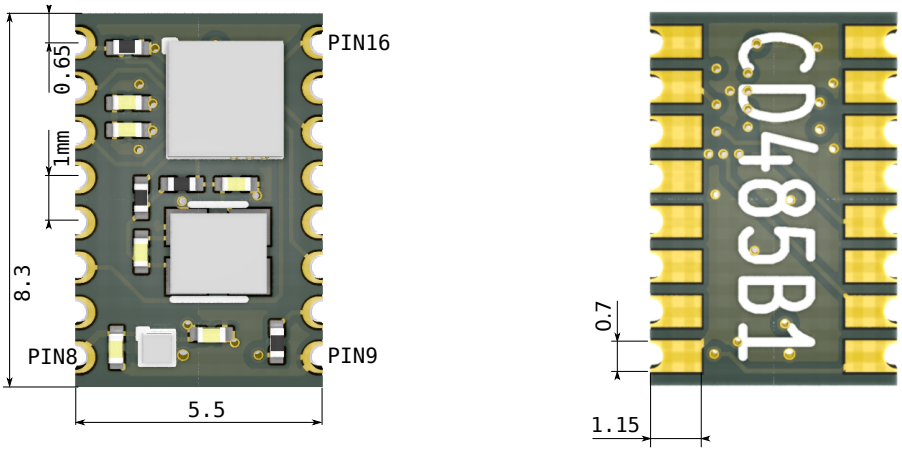
4.1 Circuit Reference



4.2 Internal Block



4.3 Mechanical Specifications



5 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