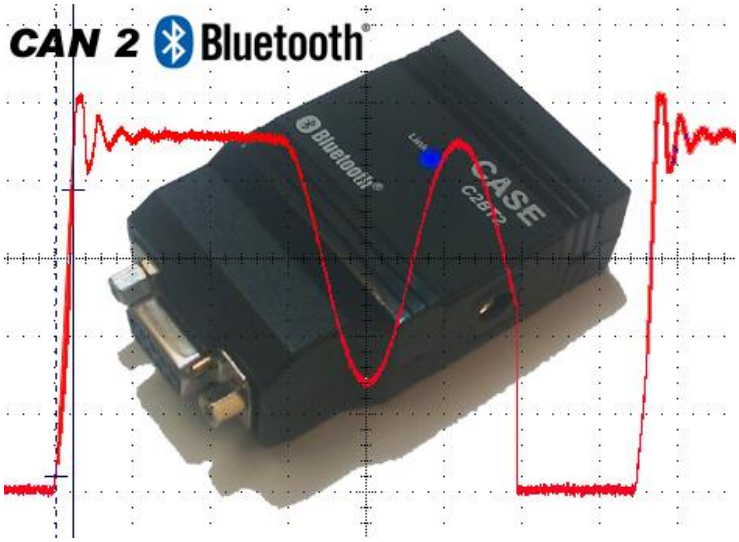
***Bluetooth*® wireless technology Adaptor for CAN-Bus Applications**

C2BT
Bluetooth® wireless technology Adaptor for CAN-Bus Applications



Preliminary Datasheet

Version 3.2 11/2012

Remarks

Trademarks & Logos

Technology

Bluetooth® is a registered trademark of the Bluetooth® Special Interest Group (SIG). As an Adopter Member of the Bluetooth® SIG we are allowed to use the Name Bluetooth® with the dedicated Logos. C2BT is a registered Trademark of CASE GmbH. The C2BT is listed in the End Product Listing (EPL) of the Bluetooth® SIG (Qualification Device ID - QDID: B013784).
CAN / CiA® and CANopen® are registered Community Trademarks of CAN in Automation e.V.

Operating Systems

Android™ is a trademark of Google Inc. The Android Robot can be used, reproduced, and modified freely in marketing communications.
Apple® iOS and MFi Program: The Made for iPod, Made for iPhone, and Made for iPad logos mean that an electronic accessory has been designed to connect specifically to iPod, iPhone, or iPad and has been certified by the developer to meet Apple performance standards.

CASE GmbH
Computer Aided System Engineering

Zum Grenzgraben 23c
D-76698 Ubstadt-Weiher
Tel.: (+49) 07251/9653-0
Fax: (+49) 07251/9653-19
www.case-gmbh.de
support@case-gmbh.de

Index

1 VERSION HISTORY

Version 3.1, 15.05.2012	Document: DS4113GE Translation of the german Datasheet
Version 3.2, 07/11/2012	Document: DS4113HE Additional remark in ASCII Transmission

AUTHOR: THOMAS WINKLER DIPL.-ING (FH).

C2BT – CAN-Bus to Bluetooth® Adapter

2 OVERVIEW

- Wireless transmission of CAN-Bus data via Bluetooth® Class 2
- Usage of the common Serial Port Profiles (SPP)
- Implements CAN V2.0B
- Programmable standard CAN-Bus data rates up to 1 Mb/s
- Supports Standard- and Extended frames (11-bit and 29-bit Identifier)
- Receive buffers, masks and filters:
  - Two receive buffers with prioritized message storage
  - two 29-bit filters
  - two 29-bit masks
- Configuration via Bluetooth®
- Transmission of timestamp (16bit Timer)
- Selectable Echo Function
- Low-power CMOS technology:
  - Operates from 4.7V – 35V
  - 20 mA active current (typical)
  - 4.3 mA idle current (typical)
- Temperature range from -25°C to +85°C
- 9-pin. D-Sub Connector for CAN Connectivity
- Integrated Bluetooth Antennae
- Intended for short range data Transmission, distances up to 15m
- Small Dimensions

The C2BT is capable of transmitting and receiving both standard and extended CAN frames via Bluetooth®. Distances up to 15m may be achieved with the integrated Bluetooth® Antennae. Exact Timing analysis may be done through switchable timestamps according to incoming CAN Messages. CAN data rates, filter- and acceptance mask settings are configurable via Bluetooth. The Power supply ranges from 4.7V to 35V DC. Power is connected either through the 9-pin D-Shell connector or an external 6.5mm power supply connector. The Bluetooth® PIN code for the enumeration process can be changed via Bluetooth.

Application Examples:

- **Transmission of SOC and Transaction data (State of charge – eBikes)**
- **Machine Configuration and Parameterisation**
- **Wireless CAN Diagnostics and CAN Debugging**
- **Monitoring**

3 CONNECTING THE C2BT

3.1 Physical Connection

The C2BT is connected through 4 wires. The Power can be supplied directly from a vehicle's battery. Care must be taken if the expected jump start voltages exceed the C2BT's absolute maximum ratings. The CAN-Bus is connected directly to CAN_H and CAN_L. We strongly recommend bus termination with resistors on both ends of the Bus lines (about 110 Ohms each). Also the maximum number of CAN Nodes should not exceed 100.

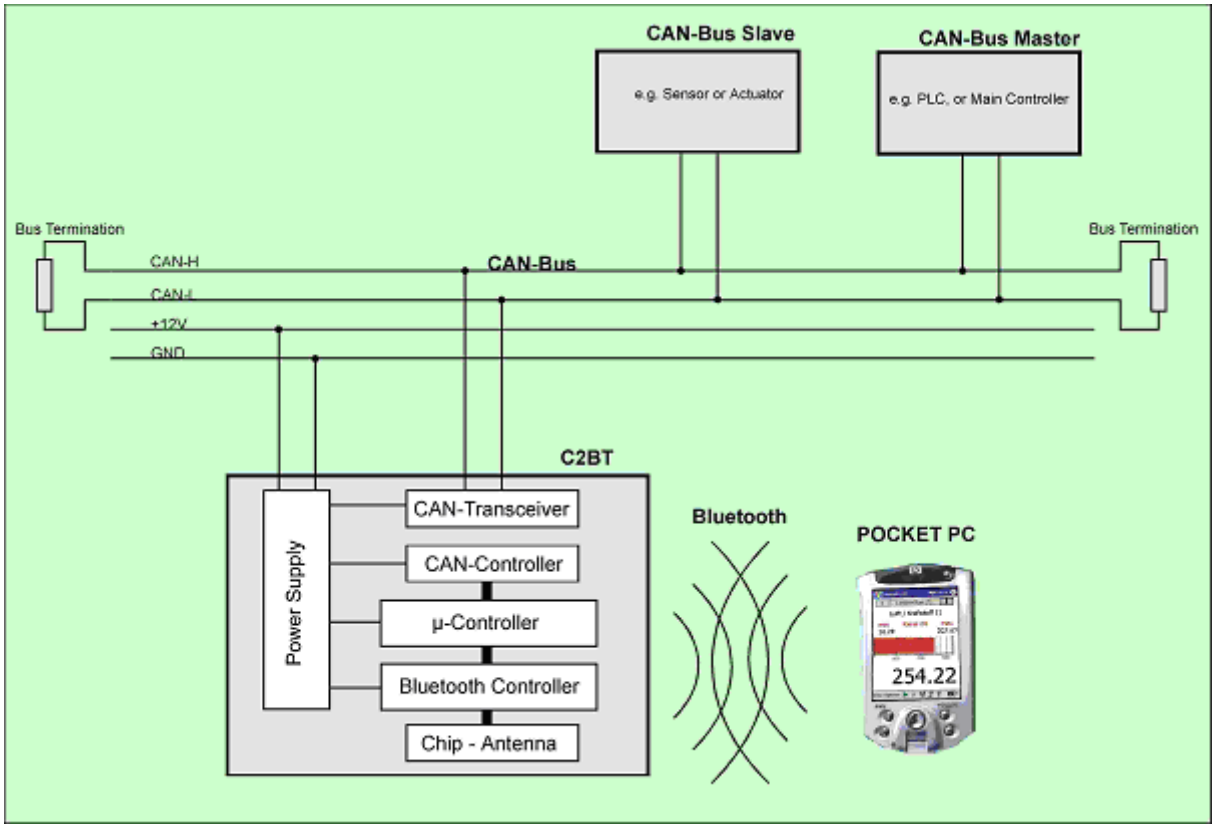


Figure 1 Connection Diagram

The C2BT's CAN Front End is a high-speed CAN, fault-tolerant Transceiver that serves as the interface between the internal CAN protocol controller and the physical CAN bus. It provides differential transmit and receive capability for the CAN protocol controller and is fully compatible with the ISO-11898 standard, including 24V requirements. It will operate at speeds of up to 1 Mb/s.

3.2 Set up Bluetooth® Connections

When a Bluetooth® device is within the Bluetooth range of the C2BT a connection may be established by the Bluetooth® device (e.g. Handheld / Pocket PC / Tablet or Mobile phone) in the Master/Host role. Connecting the C2BT for the very first time, the master has to initiate the device discovery process (searching for peripheral Bluetooth Devices nearby) and start the pairing with the C2BT. The Bluetooth® device (Master) will at least find one C2BT by its unique MAC address and usually asks for the Remote Name of the C2BT (only if the Master initiates a Remote Name request). In the device discovery process, the master captures the C2BT's Bluetooth profile. The C2BT uses the Serial Port Profile (SPP) for data exchange. Pairing will be completed with a valid PIN code. The Factory setting of the PIN is "0000" and may be modified with configuration commands via Bluetooth®. After PIN code validation, the master provides a new serial port (with a new COM Port Number). The Setup Parameters are then stored in the Master's registry in conjunction with the MAC address. Entering the PIN Code is not necessary for further use if the pairing is pending.
CAN data may be made visible e.g. through a terminal program. If a Program (e.g. a terminal Program or APP) opens, the COM Port of the C2BT changes directly from idle into transparent mode with a minimum of data overhead (according to the Bluetooth Stack, frequency hopping, Cyclical Redundancy Checks etc.). The C2BT interprets and translates CAN Traffic into a serial data stream.

4 COMMUNICATION PROTOCOL

The serial communication between the C2BT and the host may be set up in two ways.

1. The C2BT sends binary data to the host (not readable with a Terminal program).
2. The C2BT sends ASCII data to the host in a readable Hexadecimal format.

Viewing data in ASCII is convenient to start using simple terminal programs. CAN data may be directly be viewed without extensive programs, but using ASCII is uncomfortable in time critical applications.
The data transmission is bi-directional. The host can send data frames to the C2BT while CAN data is transferred from the C2BT to the host. The internal communication Baud rate is 115.2kb/s.
Each data frame is nested in a Start byte, a Configuration byte and ends with a Checksum.

4.1 Data transmission from host to C2BT2

The Length of the Configuration- and CAN- data frames vary depending on the function.

All **Configurations are stored in the internal Flash** and will be restored at start-up.

Example of a data frame assembly to send a CAN Message:

1 Byte	1 Byte	4 Byte	1 Byte	1-8 Byte	1 Byte
Start	Configuration	Identifier	Number of databytes	Data	Checksum

Example of a data frame assembly to set the CAN Baud rate:

1 Byte	1 Byte	1 Byte	1 Byte
Start	Configuration	Baud rate	Checksum

The Start byte is always **0x55h** (hexadecimal).

Start byte:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	1	0	1	0	1

The configuration byte contains the handling information of the succeeding bytes.

Configuration byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0

Bit 7 **CONFIG**: Configuration
1 = New Config data follows
0 = CAN - Bus data follows

Bit 6 **STD/EXT**: Standard or Extended frame
1 = Send data as extended frame
0 = Send data as standard frame

Bit 5 **ECHO**:
1 = On – acknowledge data by repetition of the incoming data stream
0 = Off – no acknowledge

Bit 4 **TRES**: Timestamp -Timer Reset
1 = Reset Timer
0 = No Reset

Bit 3-0 **SCFIG**: Set Configuration
0001 = Baud rate setting
0010 = Filter / Mask setting
0011 = Set timestamp
0100 = Time stamp off

0101 = Set Bluetooth® PIN code

0110 = Set data transmission in binary format

0111 = Set data transmission in ASCII format

CONFIG Bit 7 indicates if Configuration data is pending. If bit 7 = 0, the following data will directly pass through the CAN Bus.

Example: Data frame with CONFIG = 0

1 Byte	1 Byte	4 Byte	1 Byte	1-8 Byte	1 Byte
Start	Configuration	Identifier	Number of databytes	Data	Checksum

If the **CONFIG** bit = 1 the following bytes are interpreted as Configuration.
The bits **STD/EXT**, **ECHO** and **TRES** will only be accepted if the **CONFIG** bit =0.
STD/EXT indicates the type of the CAN Message Identifier (Standard or extended frame). Regardless of the **STD/EXT** bit, the message will be sent as an extended frame if the incoming Identifier is longer than 11 bits.
If the **ECHO** bit is set the incoming Messages will be sent back to the host. (Note: The data throughput will decrease with this Function enabled!)
The **TRES** bit resets the internal 16-bit Timer to 0 and starts the Timer.
The **SCFIG** bits defines the type of Configurations and will only be accepted if **CONFIG** = 1.

4.1.1 Setting the CAN Baud rate

Configuration byte (Value 0x81)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0
1	0	0	0	0	0	0	1

Dataframe

1Byte	1Byte	1Byte	1Byte
Start	Configuration	Baud rate	Checksum

Possible CAN Baud rates:

Value	Baud rate	Complete Example String with Checksum
1	20 kBit	0x55 0x81 0x01 0xD7
2	50 kBit	0x55 0x81 0x02 0xD8
3	100 kBit	0x55 0x81 0x03 0xD9
4	125 kBit	0x55 0x81 0x04 0xDA
5	250 kBit	0x55 0x81 0x05 0xDB
6	500 kBit	0x55 0x81 0x06 0xDC
7	1 Mbit	0x55 0x81 0x07 0xDD

4.1.2 Filter and Mask settings

The C2BT has three transmit and two 29 bit receive buffers, two 29 bit acceptance masks (one for each receive buffer) and two acceptance filters. Figure 2 shows a block diagram of these buffers and their connection to the protocol engine. There is also a separate Message Assembly Buffer (MAB) that acts as a third receive buffer. The MAB is always committed to receiving the next message from the bus.

After a valid message reception in the Message assembly buffer the Message will be transferred into the receiver Buffer 0. The next Message will roll over to RXBuf 1 if RxBuf 0 is full. The Message will pass if the Message Identifier meets the acceptance filter criteria.

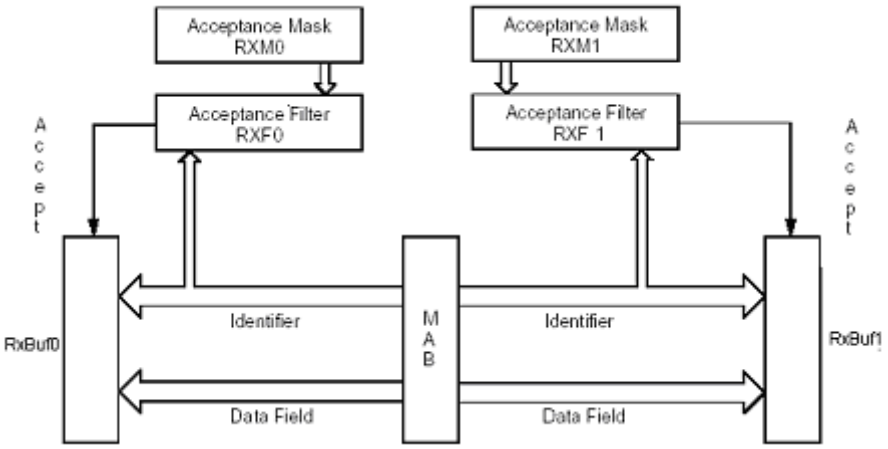


Figure 2: Filter Configuration

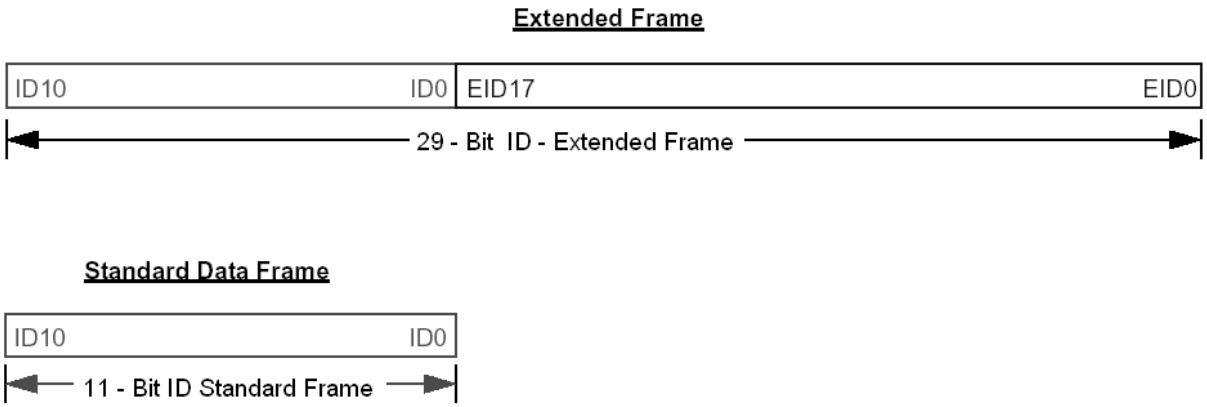


Figure 3: Active Masks and Filters for Standard and Extended frames

The RXM bits set special receive modes. Normally, these bits are cleared to 00 to enable reception of all valid messages as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the RFXnSIDL.EXIDE bit in the acceptance filter register. If the RXM bits are set to 01 or 10, the receiver will only accept messages with standard or extended identifiers, respectively. If an acceptance filter has the RFXn-

SIDL.EXIDE bit set such that it does not correspond with the RXM mode, that acceptance filter is rendered useless. These two modes of RXM bits can be used in systems where it is known that only standard or extended messages will be on the bus. If the RXM bits are set to 11, the buffer will receive all messages, regardless of the values of the acceptance filters. Also, if a message has an error before the EOF, that portion of the message assembled in the MAB before the error frame will be loaded into the buffer. This mode has some value in debugging a CAN system and would not be used in an actual system environment.

Truth table of Filter and Masks

Mask bit	Filter Bit	Identifier Bit	Accept / decline
0	X	X	Accept
1	0	0	Accept
1	0	1	Decline
1	1	0	Decline
1	1	1	Accept

4.1.3 Communication Protocol for filter- and mask Settings

Configuration Byte (Value 0x82)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0
1	0	0	0	0	0	1	0

Dataframe

1 Byte	1 Byte	8 Byte	8 Byte	1 Byte
Start	Configuration	Filter	Masks	Checksum

Filter table:

Byte order for Filter #	Byte name	Description
1	RXF0SIDL	Receive Filter 0, Standard Identifier, Low byte
2	RXF0SIDH	Receive Filter 0, Standard Identifier, High byte
3	RXF0EID0	Receive Filter 0, Extended Identifier, Low byte
4	RXF0EID8	Receive Filter 0, Extended Identifier, High byte
5	RXF1SIDL	Receive Filter 1, Standard Identifier, Low byte
6	RXF1SIDH	Receive Filter 1, Standard Identifier, High byte
7	RXF1EID0	Receive Filter 1, Extended Identifier, Low byte
8	RXF1EID8	Receive Filter 1, Extended Identifier, High byte

Mask table

Byte order for Mask #	Byte name	Description
1	RXM0SIDL	Receive Maske 0, Standard Identifier, Low byte
2	RXM0SIDH	Receive Maske 0, Standard Identifier, High byte
3	RXM0EID0	Receive Maske 0, Extended Identifier, Low byte
4	RXM0EID8	Receive Maske 0, Extended Identifier, High byte
5	RXM1SIDL	Receive Maske 1, Standard Identifier, Low byte
6	RXM1SIDH	Receive Maske 1, Standard Identifier, High byte
7	RXM1EID0	Receive Maske 1, Extended Identifier, Low byte
8	RXM1EID8	Receive Maske 1, Extended Identifier, High byte

For more information please contact our Support team: support@case-gmbh.de.

4.1.4 Timestamp Settings

The C2BT uses a 16bit Counter with a 1ms Timebase interval. The Counter value may be added to the incoming CAN Message and will be transmitted as exact timestamp. The Value of the Counter can be set to a pre-defined starting value or may be reset. The Counter ranges from 1ms to 59.999ms (1Minute) and rolls over to 0ms. The transmission of the Counter value is initiated by a Counter reset or a Counter preset.

4.1.4.1 Resetting the Timebase

After a reset, the timer initially starts with 0ms.

Example for a Reset:

Explanation:

- **Send CAN Data with Standard Identifier,**
- **Echo = off**
- **Timer will be Reset – this Activates the timestamp**

Configuration byte (Value 0x10)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0
0	0	0	1	0	0	0	0

Dataframe to send:

1 Byte	1 Byte	4 Byte	1 Byte	1-8 Byte	1 Byte
Start	Configuration	Identifier	Number of databytes	Data	Checksum

Example String:

0x55 0x10 0x00 0x00 0x04 0x33 0x08 0x21 0x32 0x43 0x11 0x9E 0xF4 0x87 0x77 0xDB

4.1.4.2 Setting the timestamp

To synchronize the C2BT clock with the timebase of a host, the Timebase may be preset by the host.

Configuration byte (Value 0x83)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0
1	0	0	0	0	0	1	1

Dataframe to preset the Timebase

1 Byte	1 Byte	2 Byte	1 Byte
Start	Configuration	Timestamp	Checksum

Timestamp byte order	Byte name	Description
1	TSTH	Timestamp High byte
2	TSTL	Timestamp Low byte

Example String: Set Timebase to 1234ms

0x55 0x83 0x12 0x34 0x1E

After receiving the preset data incoming CAN messages will be automatically transmitted with the timestamp.

4.1.4.3 Deactivation of the timestamp

The timestamp function can be disabled as follows:

Configuration byte (Value 0x84)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0
1	0	0	0	0	1	0	0

Dataframe

1 Byte	1 Byte	1 Byte
Start	Configuration	Checksum

Example String:

0x55 0x84 0xD9

4.1.5 Changing the Bluetooth® PIN code

The Factory Setting of the PIN code is „0000“. Bluetooth® data exchange is only possible with the correct PIN. After Changing the PIN code, the C2BT adaptor will be reset and starts automatically with the new PIN code content from the internal flash. There is no possibility for the user to restore factory defaults!

Caution: For the PIN code, every character between 0x00 to 0xFF is allowed to be programmed. Many Operating Systems allow only characters in the range from 0x30 to 0x39 (ASCII characters ‘0’…’9’) for visibility reasons. Therefore, it is recommended to use chars within this range.

Changing the PIN code:

Configuration byte (Value 0x85)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0
1	0	0	0	0	1	0	1

Dataframe

1 Byte	1 Byte	4 Byte	1 Byte
Start	Configuration	PIN code	Checksum

PIN code table

PIN code byte order	Byte name	Description
1	PIN3	PIN code - Most significant Char
2	PIN2	PIN code
3	PIN1	PIN code
4	PIN0	PIN code - Least significant Char

Example String for PIN code ‚1‘ ‚2‘ ‚3‘ ‚4‘:

Char Range:

PIN3 = 1 (0x31)

PIN2 = 2 (0x32)

PIN1 = 3 (0x33)

PIN0 = 4 (0x34)

Complete sample String:

0x55 0x85 0x31 0x32 0x33 0x34 0xA4

4.1.6 Binary and ASCII format

The data exchange between the C2BT and the host is selectable between a readable ASCII format or binary format. ASCII Transmission is only possible from the C2BT to the host to ease the viewing of CAN messages with a simple terminal program. That is the Default Factory setting to easily get started. All other Commands from the host to the C2BT must be in binary format. The C2BT internally handles all messages in binary format and calculates the ASCII characters before transmission to the host. For better visibility of the complete frame, “blank” characters will additionally be inserted. To transmit one binary byte in ASCII it is necessary to split one byte into 4-bit nibbles, convert the nibbles and send 2 ASCII characters. This lowers the data throughput and is not recommended to use in time critical applications.

Data transmission in binary format:

Configuration byte (Value 0x86)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0
1	0	0	0	0	1	1	0

Dataframe

1 Byte	1 Byte	1 Byte
Start	Configuration	Checksum

Example String:

0x55 0x86 0xDB

Data transmission in ASCII format:

Configuration Byte (Value 0x87)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	ECHO	TRES	SCFIG3	SCFIG2	SCFIG1	SCFIG0
1	0	0	0	0	1	1	1

Dataframe

1 Byte	1 Byte	1 Byte
Start	Configuration	Checksum

Example String:

0x55 0x87 0xDC

4.1.7 Function Summary of the Configuration bytes

Configuration	Description
0x00	Send data as Standard frame, Echo Off, No TRES
0x10	Send data as Standard frame, Echo Off, Timer Reset and Start
0x20	Send data as Standard frame, Echo On, No TRES
0x30	Send data as Standard frame, Echo On, Timer Reset and Start
0x40	Send data as Extended frame, Echo Off, No TRES
0x50	Send data as Extended frame, Echo Off, Timer Reset and Start
0x60	Send data as Extended frame, Echo On, No TRES
0x70	Send data as Extended frame, Echo On, Timer Reset and Start
0x81	Set CAN Baud rate
0x82	Set Filter and Masks
0x83	Set timestamp
0x84	Timestamp Off
0x85	Set Bluetooth® PIN code
0x86	Datatransmission in Binary format
0x87	Datatransmission in readable ASCII format

Configuration	Byte 1	Byte 2	Data	Data	Data	Byte
0x00	Start	Configuration	4 byte ID	1 byte Num	Data	Checksum
0x10	Start	Configuration	4 byte ID	1 byte Num	Data	Checksum
0x20	Start	Configuration	4 byte ID	1 byte Num	Data	Checksum
0x30	Start	Configuration	4 byte ID	1 byte Num	Data	Checksum
0x40	Start	Configuration	4 byte ID	1 byte Num	Data	Checksum
0x50	Start	Configuration	4 byte ID	1 byte Num	Data	Checksum
0x60	Start	Configuration	4 byte ID	1 byte Num	Data	Checksum
0x70	Start	Configuration	4 byte ID	1 byte Num	Data	Checksum
0x81	Start	Configuration	1 byte Baud rate	X	X	Checksum
0x82	Start	Configuration	8 byte Filter	8 byte Mask	X	Checksum
0x83	Start	Configuration	2 byte timestamp	X	X	Checksum
0x84	Start	Configuration	X	X	X	Checksum
0x85	Start	Configuration	4 byte PIN code	X	X	Checksum
0x86	Start	Configuration	X	X	X	Checksum
0x87	Start	Configuration	X	X	X	Checksum

4.2 Data transmission from the C2BT2 to the host

Every message sent from the C2BT to the host starts with Start byte 0xAA for synchronization purposes. Messages will be sent with variable length, depending on the configuration and the received CAN frames.

Example of a typical Data frame after a valid CAN Message reception:

1 Byte	1 Byte	2 Byte	4 Byte	1 Byte	1-8 Byte	1 Byte
Start	Configuration	Timestamp	Identifier	Number of databytes	Data	Checksum

Remark: The two bytes of the Time stamp will only be sent if bit TRES is set.

The Configuration byte holds information's about the internal Register and Bus status. The configuration byte will only be sent after a valid CAN Reception.

Configuration byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG	STD/EXT	BUSOFF	BUS128	BUS96	TBFULL	RXBOV	TSRACK

Bit 7 **CONFIG**: Configuration Acknowledge
1 = New configuration data was received, configuration is done
0 = No new configuration data received

Bit 6 **STD/EXT**: Standard or Extended frame
1 = CAN Message is an Extended frame
0 = CAN Message is a Standard frame

Bit 5 **BUSOFF**:
1 = BUS OFF - Condition detected
0 = BUS ON - Condition detected

Bit 4 **BUS128**: Bus Heavy
1 = more than 128 Bus Failures detected
0 = less than 128 Bus Failures detected

Bit 3 **BUS96**: Bus Light

1 = more than 96 Bus Failures detected

0 = less than 96 Bus Failures detected

Bit 2 **TBFULL**: Transmit Buffer Full

1 = Transmit Buffer Overflow – unresolved Messages pending

0 = Tx Buffer ready for next Transmission

Bit 1 **RXBOV**: Receiver Buffer Overflow

1 = Valid CAN-Message could not be resolved

0 = No Overflow – Receiver Buffer is ready for new CAN data

Bit 0 **TSRACK**: Time Stamp Reset Acknowledge

1 = The 16 bit Counter was reset.

0 = No Reset of the Counter

4.2.1 Echo Mode

In Echo Mode the C2BT responds every received Char and appends the Checksum for validation.

Example of an Echo frame:

1 Byte	n Bytes	1 Byte
Start (0xAA)	Reply of **all** received bytes (checksum in-cluded)	Calculated Checksum

4.2.2 Format of the timestamp

The Time stamp consists of two bytes as unsigned integer (16bit). It holds the Register Content of the 1ms Timebase counter in correlation to the CAN Message reception. The range lasts from 0ms to 59.999ms.

Time stamp byte Order	Bytename	Description
1	TSTH	Timestamp High byte
2	TSTL	Timestamp Low byte

4.3 Data frame in ASCII transmission

For comfortable debugging and viewing the ASCII data String is divided into logical Portions using white space characters. The String is terminated by Carriage Return and Line Feed (\r\n or in hex: 0x0D, 0x0A). The termination and whitespace Chars are not considered in checksum calculation!

ASCII Transmission is only possible for viewing the CAN-Bus Data in ASCII Format. All Data to the C2BT must be sent in Binary Format. It is not possible to send any configuration Data in ASCII Format to the C2BT.

Example:

C2BT2 -> Host: C2BT sends an error free CAN Message with standard ID 0x685 containing 3 data bytes (0x03 0xA6 0xFC) with additional timestamp @ 0x0A13 (2,759 sec)

AA 00 0A13 0685 03 03 A6 FC F7

4.4 Calculation of the Checksum

The Checksum is the overall sum of the transmitted bytes including the Start byte.

Example1:

Baud rate Setting:

Start byte = 0x55

Configuration byte = 0x86

Checksum = 0x55 + 0x86 = 0xDB

String to send:

0x55 0x86 0xDB

Example2:

Send a Standard frame

- @ Address 433
- With 8 data bytes
- byte values: 0x21 0x32 0x43 0x11 0x9E 0xF4 0x87 0x77
- and start timestamp

String to send:

0x55 0x10 0x00 0x00 0x04 0x33 0x08 0x21 0x32 0x43 0x11 0x9E 0xF4 0x87 0x77 0xDB

4.5 Factory Settings

The C2BT is in default factory reset when delivered. The settings are:

CAN Baud rate:	500kBit/s	
Bluetooth® PIN code:	'0000'	
Data transmission:	ASCII-format	
Filter:	0xFFFFFFFFFFFFFFFF	(No filter - open for all CAN-ID's)
Masks:	0x0000000000000000	(No Masks - open for all CAN-ID's)
Device Name:	Inquiry – Remote Name Request: 'C2BT2 www.case.eu'	
Bluetooth Profile:	Serial Port Profile (SPP – Link Establishment)	

5 TECHNICAL DATA

5.1 Electrical characteristics

General Specification

Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 1. Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
Vbat	Supply Voltage range	-30	+35	V
Ts	Storage Temperature	-40	+85	°C
	ESD Human Body Model		2000	V

Table 2. Recommended Ratings

Symbol	Parameter	Min	Max	Einheit
Vbat	Supply Voltage range	+4.7	+30	V
To	Environment Temperature	-20	+85	°C
	Humidity		80	%RH

Table 3. Current consumption under different conditions

Symbol	Parameter	@Vbat=9V	@Vbat=12V	@Vbat=24V	Einheit
Ir	Ir without Bluetooth® Connection	10,5	7,5	4,3	mA
Is	Ir with Bluetooth® connection and **Little** CAN Traffic @ 500kbps	16	11,7	6,8	mA
It	Ir with Bluetooth® connection and **Heavy** CAN Traffic @ 500kbps	20	15,5	9	mA

5.2 Dimensions

Dim in Inch / [mm]

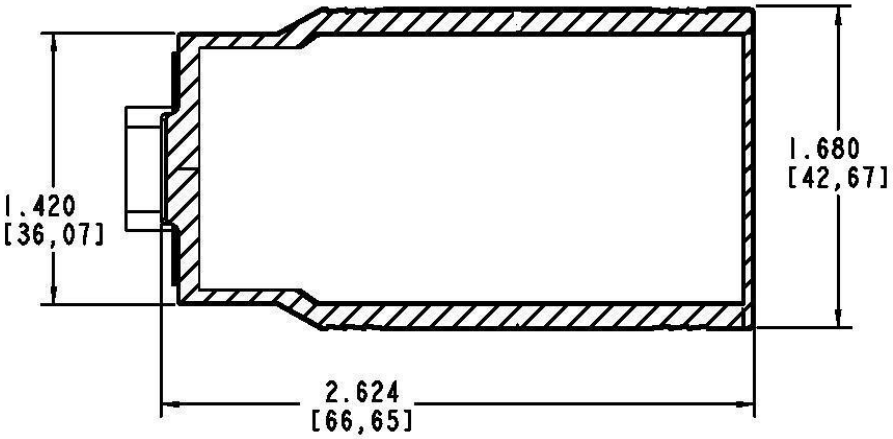


Figure 4: Top view X,Y

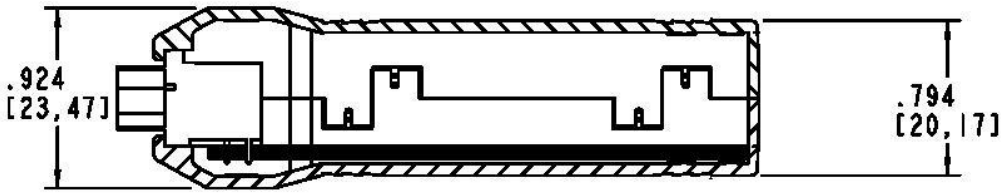
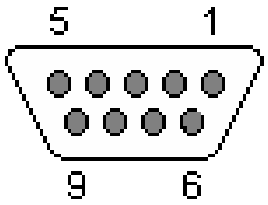


Figure 5: Side View Z

5.3 Pin Connections



Pin1:	N.C.
Pin2:	CAN Low
Pin3:	GND
Pin4:	N.C.
Pin5:	N.C.
Pin6:	GND
Pin7:	CAN High
Pin8:	N.C.
Pin9:	+Vbat

6 ANDROID™- APPLE™- iOS AND MS-WINDOWS™ APPLICATIONS

Remark: The C2BT Series supports the entire Bluetooth® Class 2.0 specification. The integrated Bluetooth® Module is certified by the Bluetooth® Special Interest Group and complies to the SPP (Serial Port Profile).

Microsoft

No known problems. Tested Plattforms:

1. OS-PC: Windows 98, XP, VISTA, Windows 7
2. OS-Pocket PC / PDA: Windows CE, Windows Mobile Pocket PC 6.x

Android™ Applications

Known Issues:

- up to Version 2.2 (FROYO 05.2010) connecting the C2BT was not possible.
- some problems up to Android Version 2.3 to 2.3.7 (Gingerbread) - depending on several Mobile phone manufacturers and their Bluetooth stack version.
- No known issues since Version 3.0 (Honeycomb 02.2011) with SAMSUNG Galaxy and ACER Tablet devices. Problems still exist with HTC Flyer and Desire (SPP is not yet supported).

Developer

Connection with iOS is not possible. There is no encryption Chip on the C2BT Series that is needed for Communication with Apple devices via SPP.

7 DEVICE SUPPORT AND TERMINAL PROGRAMS

For a quick start with the C2BT a Terminal Program is a very useful Tool. There are several Terminal Program suppliers we can recommend.

Plattform:	Description
Windows PC: Windows 98 Windows XP Windows VISTA Windows 7	Name: **Hyperterminal -** Microsoft Existing Standard Program on each PC with OS up to Win XP http://technet.microsoft.com/en-us/library/cc736511(WS.10).aspx Name: **Hterm -** Tobias Hammer Free Terminalprogram to send and receive data in ASCII and binary http://www.der-hammer.info/terminal/ Name: **Docklight –** FuH Test Version with enhanced Macro Functions http://www.docklight.de/
Windows Mobile: 2002/2003/2003SE/5/6	Name **mToken** - Choung Networks http://www.choung.net/products/discontinued/
Android: Version 2.3.x (Gingerbread) & higher	Name: **bTerm** - SENA Free APP in the Android Market Info: www.sena.com/download/manual_bterm/overview.html