1 Device Requirements for Transceiver Test

1.1 Device documentation

A pre-condition for the execution of the test is the availability of the necessary information as described below.

The customer provides free of charge and in time all necessary:

- data sheets
- additional documentation / information
- available configurations

in order to be able to implement the device.

Note: The availability of a specialist person to discuss problems and to provide further information in case of being necessary would be very helpful.

1.2 Required number of devices

For network testing we need 25 transceivers for testing.

1.3 Hardware Requirements

To connect any device to a host board it must be placed on an adapter and routed to the related pins by customer. Maximum dimensions of the adapter are showed on next image:

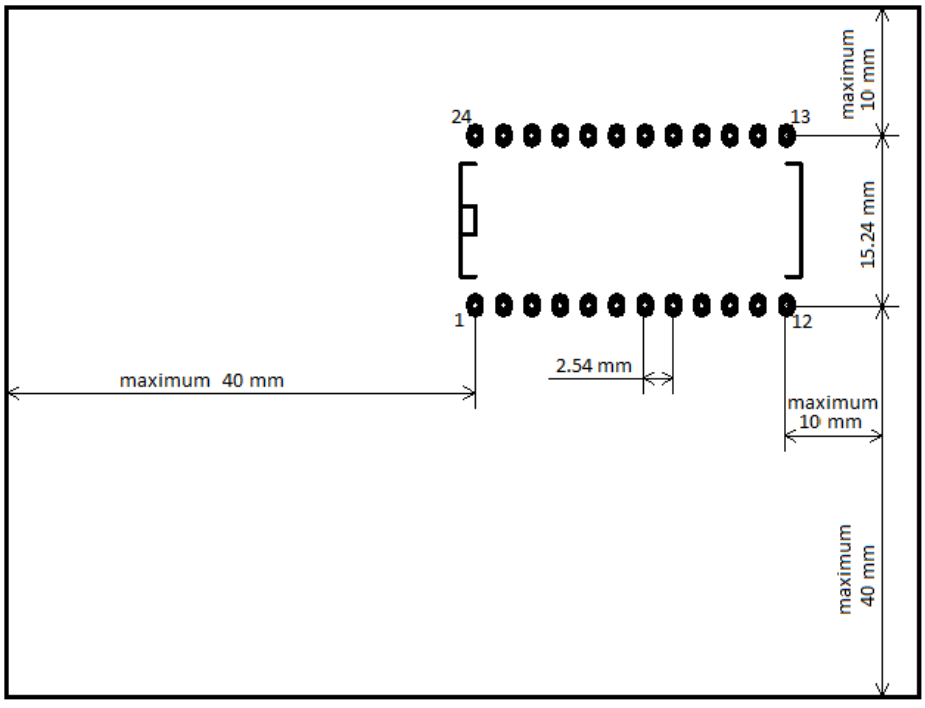


Figure 1.3-1 – Adapter's dimensions

Name	Pin #	Pin #	Name
TxD	1	24	STB
GND	2	23	CANH
Vcc	3	22	CANL
RxD	4	21	SPLIT
VIO	5	20	VBAT
EN	6	19	WAKE
INH	7	18	ERR
SPI_CSN	8	17	VAUX
SPI_MISO	9	16	Input_0 (DUT_Output_0)
SPI_MOSI	10	15	Input_1 (DUT_Output_1)
SPI_SCLK	11	14	Output_2 (DUT_Input_2)
GPIO_SUPPLY	12	13	reserved

Table 1.3-1 – Pinout

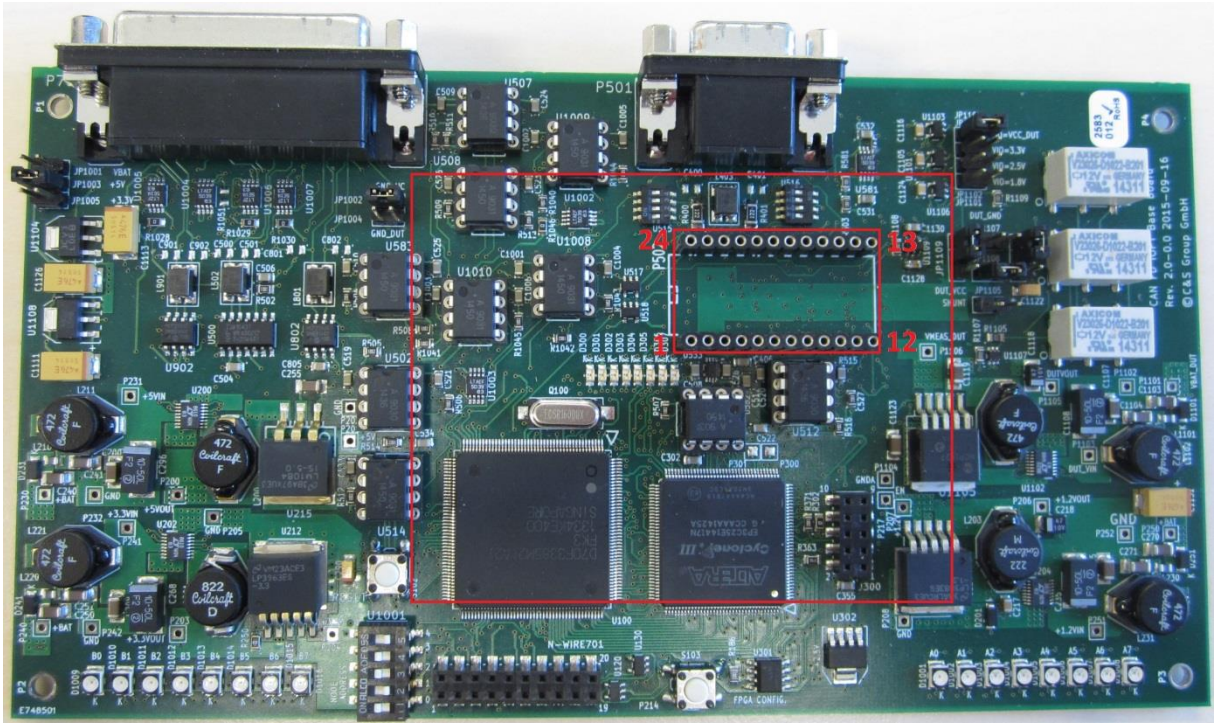
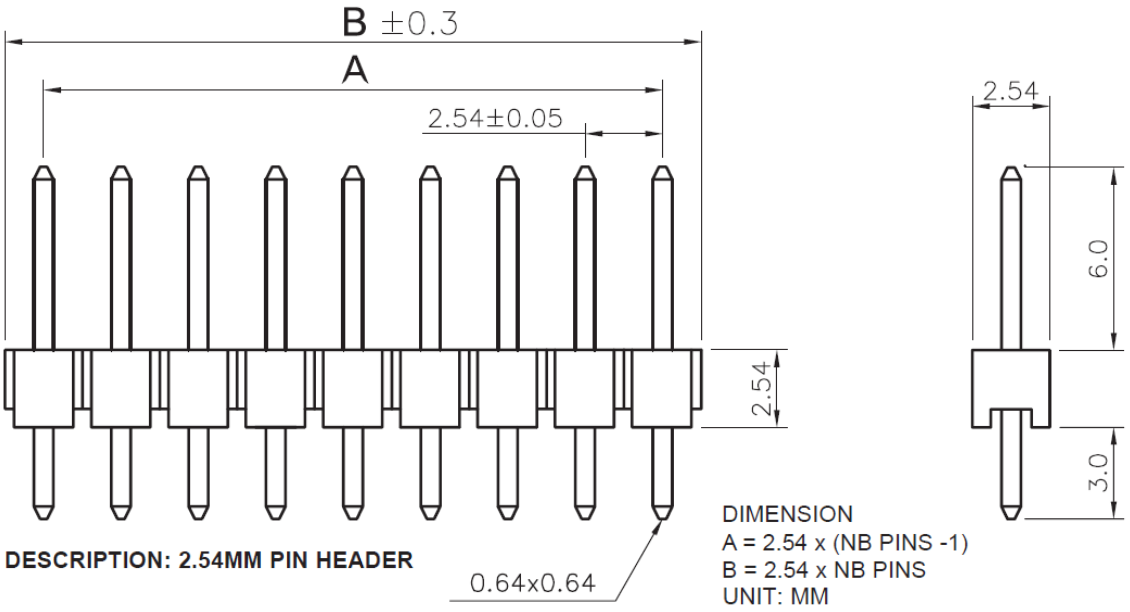


Figure 1.3-2 – Adapter's dimensions

Note:

Pins must be 0.64mm square.



The pins must be mounted on the back of the circuit board.

Only components up to a height of 2mm may be mounted on the back of the circuit board.

Missing signals INH, ERR, EN and STB can be left open.

Inverted signals for INH, ERR, EN, STB or WAKE will be adapted by test software.

1.4 SPI driver for SBC devices

Devices controlled by SPI commands require an additional software implementation. This driver has to be delivered by the customer.

The driver has to be written in c-code (ANSI C). Source-code has to be delivered for eventual necessary adaptations.

The test software expects following functions combined in TRX_API.h file.

1.4.1 Transceiver Application Programming Interface

Function description:

```
/* definition of constants for transceiver */

#define TRX_SLEEP                    0x00
#define TRX_NORMAL                   0x01
#define TRX_STANDBY                  0x02
#define TRX_FRAME_DETECT             0x03
#define TRX_RECEIVEONLY              0x04

#define TRX_FCT_ERROR                0xFF

#define TRX_INH_STATUS_ON            0x01
#define TRX_INH_STATUS_FLOATING      0x00
```

```
/*****************************************************************************************
**
 *  unsigned char TRX_API_Init (void)
 *
 *  FILENAME:    TRX_API.H
 *
 *  PARAMETERS:         --
 *
 *  DESCRIPTION: use this function to initialize your driver
 *
 *
 *  RETURNS:     Error value
 *               TRX_FCT_ERROR    -> error during execution
 *               0 no error
 *               > 0 error e.g. error code !!!!!!!!!!values to be defined !!!!!!!!!!!!
 *

*****************************************************************************************
*/
 unsigned char TRX_API_Init (void);
```

```
/*****************************************************************************************
**
 *  unsigned char TRX_GetError (void)
 *
 *  FILENAME:   TRX_API.H
 *
 *  PARAMETERS:        --
 *
 *  DESCRIPTION: read the error information from the transceiver
 *
 *
 *  RETURNS:     Error value
 *               TRX_FCT_ERROR    -> error during execution
 *               0 no error
 *               > 0 error e.g. error code !!!!!!!!!!values to be defined !!!!!!!!!!!!
 *
*****************************************************************************************
*/
 unsigned char TRX_GetError (void);



/*****************************************************************************************
*
 ** unsigned char TRX_SetOperationMode (unsigned char mode)
 *
 *  FILENAME:   TRX_API.H
 *
 *  PARAMETERS:        mode to be set
 *               TRX_NORMAL       -> set TRX to normal mode
 *               TRX_SLEEP        -> set TRX to sleep mode
 *               TRX_STANDBY      -> set TRX to standby mode
 *               TRX_FRAME_DETECT -> set TRX to sleep mode with wake up frame detect
 *               TRX_RECEIVEONLY  -> set TRX to receive only mode
 *               other values will be ignored
 *
 *  DESCRIPTION: set the transceiver in the provided mode
 *
 *
 *  RETURNS:     function results
 *               TRX_FCT_ERROR    -> error during execution
 *               0 mode is set
 *               1 wrong parameter or mode could not be set
 *               all other values not valid
 *
*****************************************************************************************
/
 unsigned char TRX_SetOperationMode (unsigned char mode);


NOTE:  For TRX_FRAME_DETECT mode
       Following global variables will be used for wake up frame configuration

extern uint8  SelectiveWakeUpID[4];         /* Wake up frame ID */
extern uint8  SelectiveWakeUpDLC;           /* Wake up frame DLC */
extern uint8  SelectiveWakeUpIDMask[4];     /* Wake up frame ID Mask */
extern uint8  SelectiveWakeUpPayLoadMask[8];/* Wake up frame Payload Mask */
```

```
/*****************************************************************************************
*
 *  unsigned char TRX_GetOperationMode (void)
 *
 *  FILENAME:   TRX_API.H
 *
 *  PARAMETERS:         ---
 *
 *
 *  DESCRIPTION: get the transceiver operation mode
 *
 *  RETURNS:            transceiver operation mode
 *                TRX_NORMAL       -> set TRX to normal mode
 *                TRX_SLEEP        -> set TRX to sleep mode
 *                TRX_STANDBY      -> set TRX to standby mode
 *                TRX_FRAME_DETECT -> set TRX to wake up frame detect mode
 *                TRX_RECEIVEONLY  -> set TRX to receive only mode
 *                other values are not valid
 *
 *

*****************************************************************************************
/
 unsigned char TRX_GetOperationMode (void);


/*****************************************************************************************
*
 *  unsigned char TRX_GetINHState (void)
 *
 *  FILENAME:   TRX_API.H
 *
 *  PARAMETERS:         --
 *
 *  DESCRIPTION: read the status of the INH bit
 *
 *
 *  RETURNS:     Error value
 *                TRX_FCT_ERROR            -> error during execution of function
 *                TRX_INH_STATUS_ON        -> INH == 1 -> voltage regulator on
 *                TRX_INH_STATUS_FLOATING  -> INH == 0 -> voltage regulator on
 *                > 0 error e.g. error code !!!!!!!!!!values to be defined !!!!!!!!!!!!
 *
 *

*****************************************************************************************
/
 unsigned char TRX_GetINHState (void);
```

1.5 Requirements for test according to ISO 16845-2

Additionally to the general device documentation described above (cp. chapter 1.1) complete and detailed descriptions of:

- the SPI interface and all related registers
- the states and state changes (text or graph)

are absolutely necessary to be provided.

Name all bits to be set. Describe read back values.

Describe all bits to check for correct mode change:

- Normal Mode
- Sleep Mode
- Sleep Mode with Wake up Frame detect

Name all bits to be checked for error analysis:

- Failure in Frame detect configuration
- Timeout / Receive error counter overflow
- Under voltage

Furthermore source code examples related to the mode changes shall be provided.

An example for our SPI programming routines could be provided.