I2C RS232 Adapter
(SLAVE)

Manual

COPTONIX

Luxemburger Str. 31
D – 13353 Berlin
Phone: +49 – (0)30 – 61 74 12 48
Fax: +49 – (0)30 – 61 74 12 47
www.coptonix.com
support@coptonix.com

1. Description

The I2C-RS232-Adapter is a universal usable I2C to RS232 converter. With the Adapter as I2C-Slave, several RS232 devices (e.g. PC, microcontroller) can be connected to one I2C bus, or several I2C-Devices (Master) can be connected to one RS232 device.

The slave address can be programmed freely over the serial interface. The default slave address is ***0xFE*** (***1111 1110***). The data of a master are temporarily stored in the I2C-RS232-Adapter and then transmitted through the serial interface. A data packet can be maximal 128 bytes.

The data, which a serial device receives (e.g. PC, microcontroller), can be processed and then sent back to the I2C-RS232-Adapter. Here the data packet can be also maximally 128 bytes. So that the I2C-RS232-Adapter can get the function of a master, the I2C-RS232-Adapter controls two Interrupt outputs. An Interrupt output can be selected over the serial interface while sending data to the adapter.

Two types of this adapter are available:

- I2C-RS232-Adapter **RS** (Part No. #020101):
  This type is controlled only via the serial interface (RS232). The communication is done through ASCII commands (see page 15).

- I2C-RS232-Adapter **IC** (Part No. #020102):
  This type is controlled only via the I2C-Bus. The communication is done through I2C commands (see page 8). The RS232 interface is transparent.

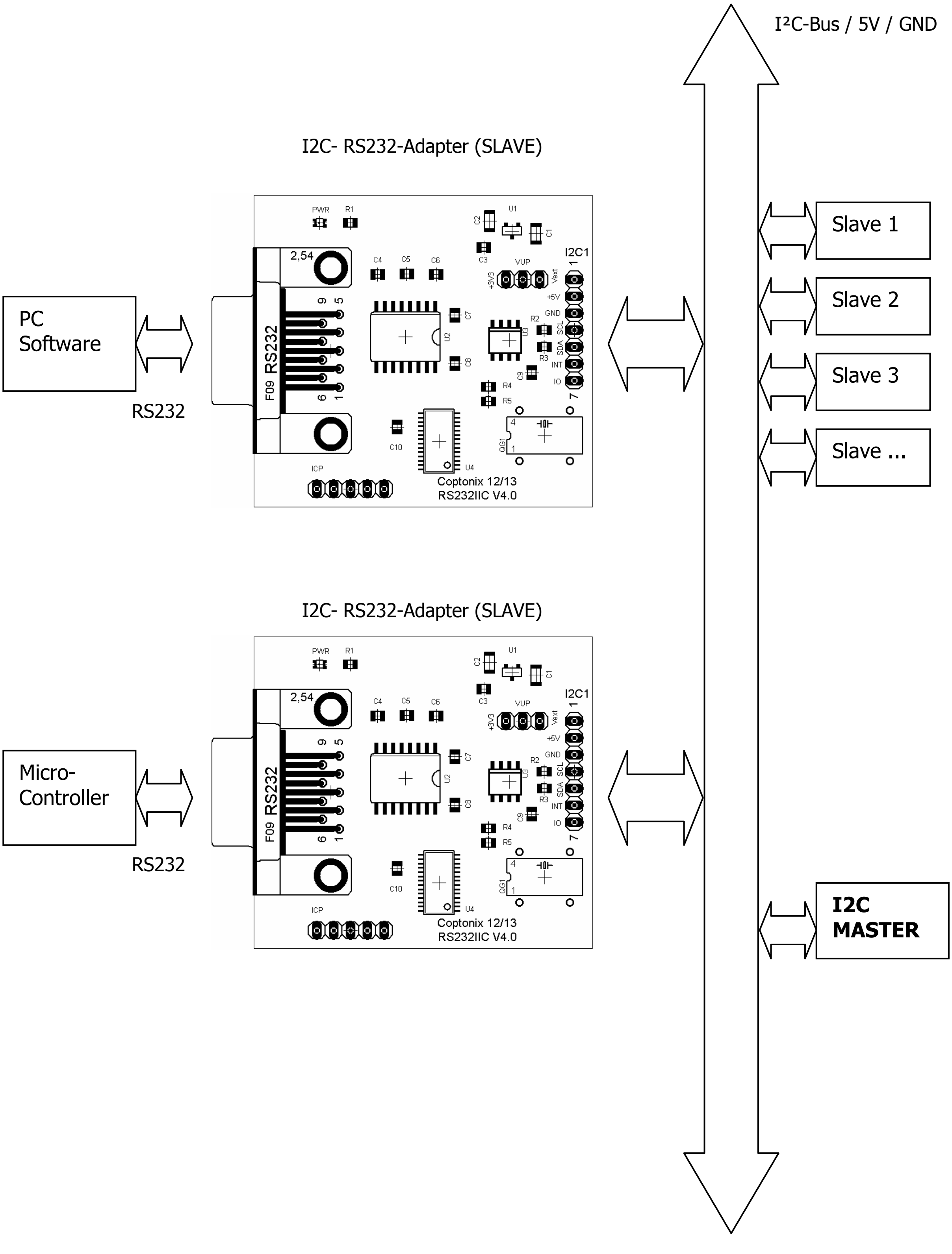
With the many possibilities of the Adapter it is very simple to define/program own I2C-Slaves when connected to a PC or a microcontroller.

Therefore the adapter is suitable for developers, who would like to develop and/or test own I2C-circuits. The Adapter can be a PCF8574, PCF8574A, temperature sensor or any other slave with any (7-bit) slave address.

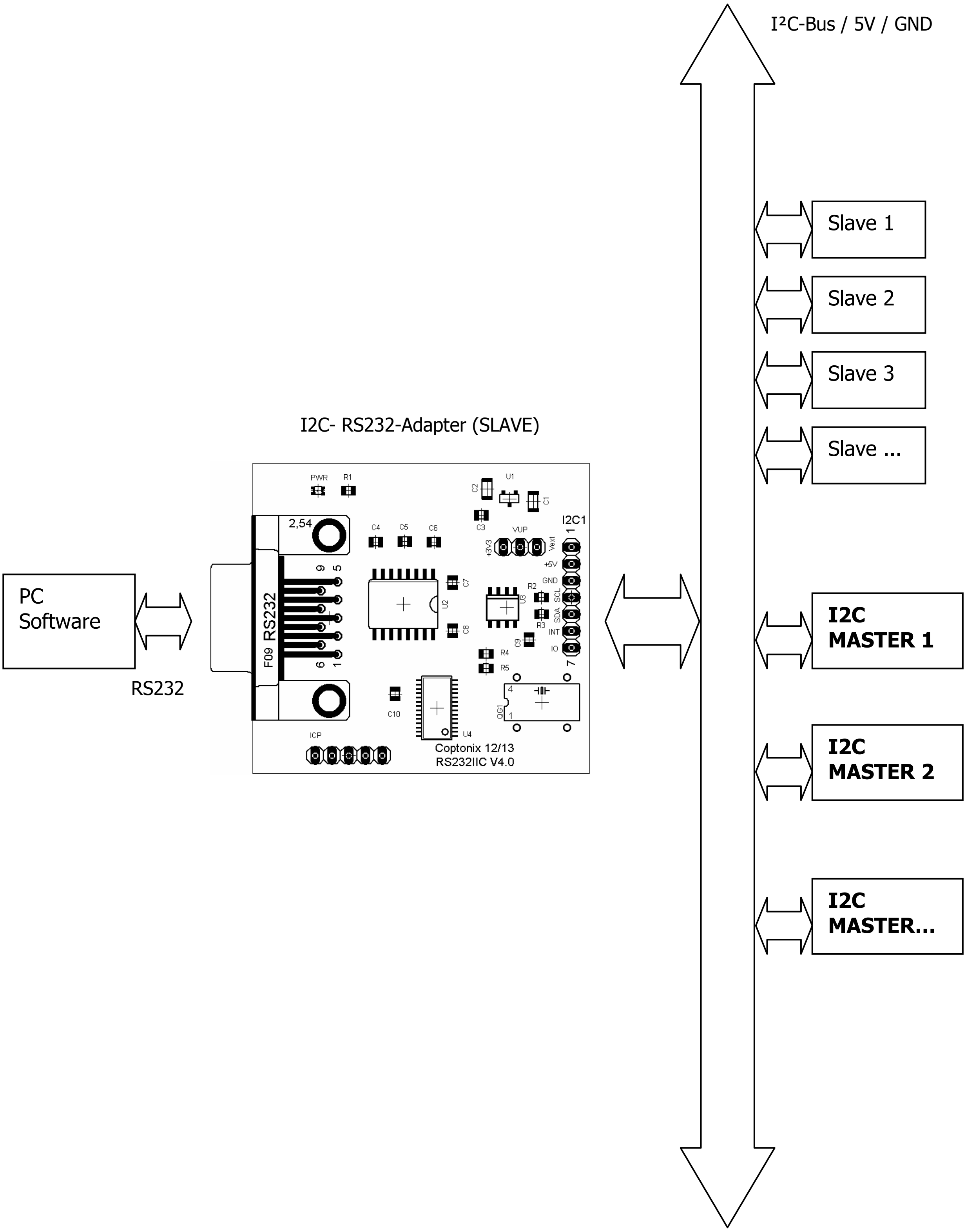
With a PC or microcontroller the Adapter can be used also to control devices with I2C-Interface (Master). For software development, a Delphi component, a DLL, examples and data transmission protocol of the Adapter are available for developer. When using the software interfaces, the Adapter can be integrated very simply into own applications

2. Applications

2.1 One I2C-Master – multiple RS232 devices

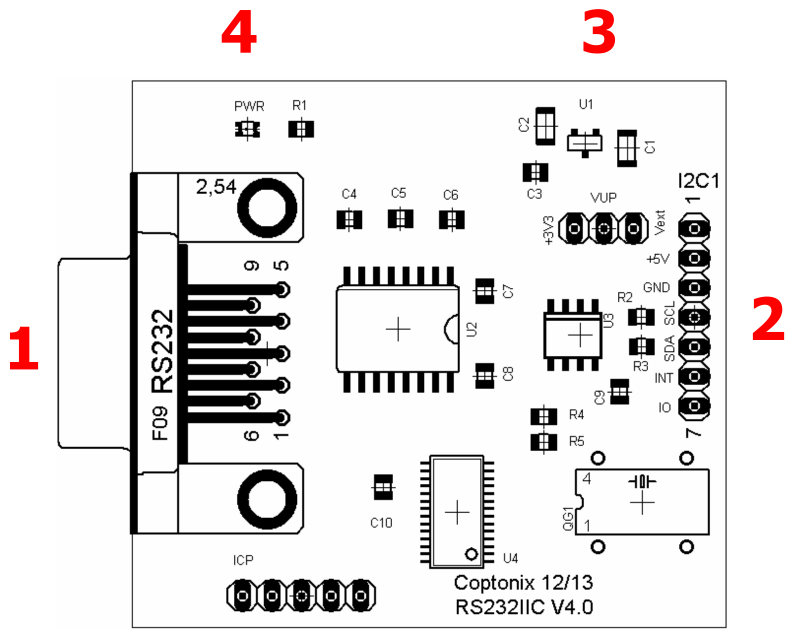


2.1 Multiple I2C-Master – one RS232 device

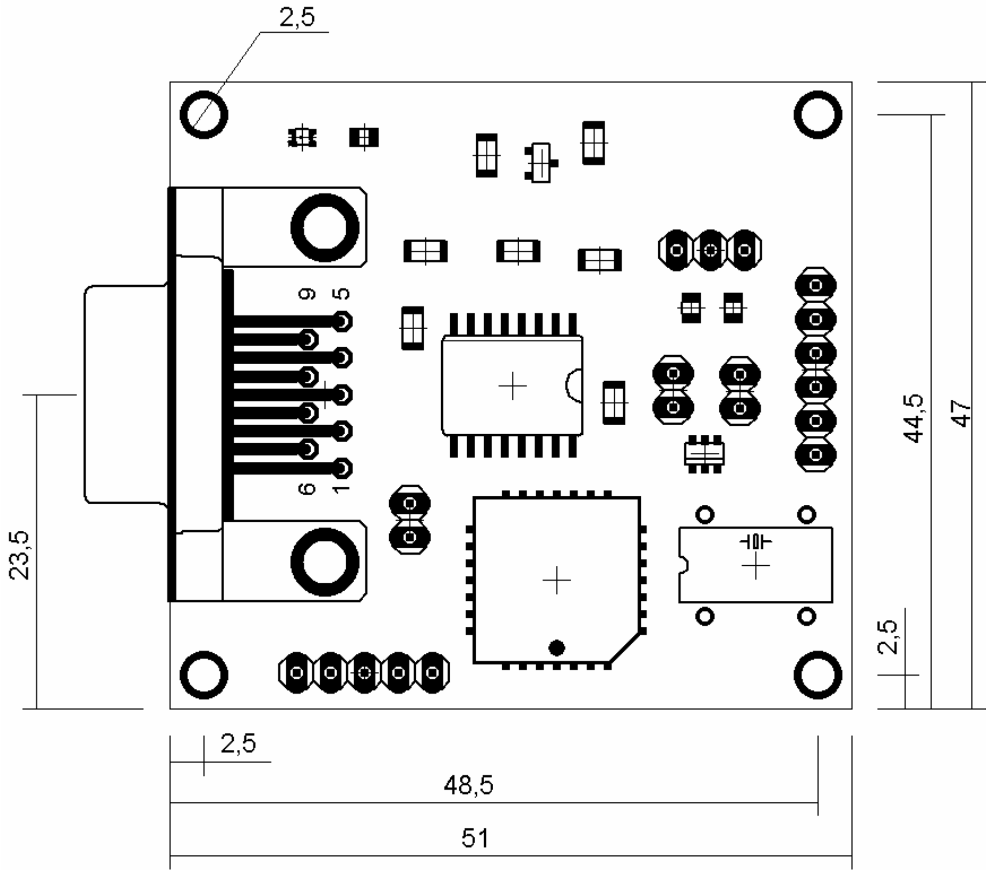


3. Interface

- **1** RS232 port for communication with a PC (RxD, TxD, GND)
- **2** I^{2}C-Interface
  Pin 1: External pull-up voltage V_{ext} (2V – 15V)
  Pin 2: Supply voltage +5V
  Pin 3: Ground
  Pin 4: I2C – SCL
  Pin 5: I2C –SDA
  Pin 6: Interrupt – Output 1
  Pin 7: Interrupt – Output 2
  Header strip / 2.54 pitch
- **3** Jumper VUP – $V_{pull\text{-}up}$ Pull-up voltage
  Pin 1: +3.3V
  Pin 2: $V_{pull\text{-}up}$; connected to pull-up resistors (4K7)
  Pin 3: External pull-up voltage
  Position 1-2: connects pull-up voltage to internal +3.3V
  Position 3-2: connects pull-up voltage to external pull-up voltage.
- **4** LED Power-ON



4. Dimension (mm)



5. Characteristics

	Min.	**Typ**	**Max.**	**Unit**
Power-Supply				
Supply Voltage		5.0		V
Supply Current		12	15	mA
I2C-Bus pins (SCL, SDA)				
V_{ext} External Pull-up Voltage	2	-	15	V
V_{IH} High-State Input Voltage	$0.58V_{pull-up}$	-	-	V
V_{IL} Low-State Input Voltage	-	-	$0.42V_{pull-up}$	V
Limiting values				
Interrupt pins				
Input Voltage	0	-	5.5	V
Output Voltage	0	-	$V_{DD(3,3V)}$	V
Power-Supply				
Supply Voltage	4.0	5.0	6.0	V
Temperature				
operating temperature	-20	-	+70	°C

6. I2C Commands (Only I2C-controlled devices / Part No. #020102)

6.1 Command Overview

Command	Description	Value
TransmitData	Transmit data bytes (from I2C) over UART	0x74
ReadData	Read data bytes received from UART	0x72
ReadCount	Read count of received bytes (from UART)	0x63
SetSlvAdr	Set a new slave address to temporary register	0x41
ReadSlvAdr	Read slave address from temporary register	0x42
SaveSlvAdr	Save and initialise new slave address	0x43
SetBaud	Set, save and initialise new baud rate	0x44
ReadBaud	Read the currently adjusted baud rate	0x45
ResetBuffers	Reset data buffers to initial state	0x66
ReadVersion	Read adapter's version	0x76
SetUARTMode	Set UART operation mode	0x46
GetUARTMode	Get UART operation mode	0x47
SetETXChar	Set UART ETX Character	0x48
GetETXChar	Get UART ETX Character	0x49
SetTimeOut	Set UART TimeOut	0x4A
GetTimeOut	Get UART TimeOut	0x4B

6.2 Command Description

6.2.1 TransmitData

In order to transmit an array of data bytes (I2C -> UART) issue the following command. The buffer size is limited to 128 bytes per message.

TransmitData command (0x74)									
Byte	Content	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	TransmitData	0	1	1	1	0	1	0	0
2	Data 1	1. data byte							
…	…	2. to n-1 byte							
n+1	Data n	n data byte (up to 128 data bytes)							

6.2.2 ReadData

This command is used for reading data that were received from UART and temporary written into internal RAM.

ReadData command (0x72)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	ReadData	0	1	1	1	0	0	1	0

ReadData Response									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	1
1	Slave Address	Slave address of the selected device							
2	Byte 1	1. Byte							
…	...	…							
n+1	Byte n	n Bytes (up to 128 bytes; n = Length)							

When the adapter receives a message from UART, then output Pin5 is set low. This Pin remains low until the ReadData command is issued, then Pin5 ist set high.

6.2.3 ReadCount

This command returns the number of bytes received from the UART interface.

ReadCount command (0x63)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (W)	A7	A6	A5	A4	A3	A2	A1	0
1	ReadCount	0	1	1	0	0	0	1	1

ReadCount Response									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (R)	A7	A6	A5	A4	A3	A2	A1	1
1	Slave Address	Slave address of the selected device							
2	Count	Count of received bytes							
3	Status	x	x	x	x	x	x	x	**S**

Status		
S	0	Data already read
	1	New data packet
x	Do not care	

6.2.4 SetSlvAdr

The default slave address is 0xFE (1111 1110). The "SetSlvAdr" command allows to change the slave address of the adapter.

SetSlvAdr command (0x41)

Byte	content	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	SetSlvAdr	0	1	0	0	0	0	0	1
2	Slave address	Slave Address(7:1)							0

The new slave address is written to a temporary register and still not active. In order to read/verify this new slave address, issue the command "ReadSlvAdr". If the command "SaveInitSlvAdr" is issued, then the new slave address is saved into the EEPROM and the I2C-Interface is initialised.

6.2.5 ReadSlvAdr

read / verify the new slave address that was written to the temporary register.

ReadSlvAdr command (0x42)

Byte	content	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Slave Address (W)	A7	A6	A5	A4	A3	A2	A1	0
1	ReadSlvAdr	0	1	0	0	0	0	1	0

ReadSlvAdr Response

Byte	Content	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Slave Address (R)	A7	A6	A5	A4	A3	A2	A1	1
1	Slave Address	Slave address of the selected device							
2	New Slave Address	New Slave Address (temporary register)							

6.2.6 SaveInitSlvAdr

Save and initialize the new slave address.

SaveInitSlvAdr command (0x43)

Byte	Content	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	SaveInitSlvAdr	0	1	0	0	0	0	1	1

4.2.7 SetBaud

sets the baud rate of the UART.

SetBaud command (0x44)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	SetBaud	0	1	0	0	0	1	0	0
2	Baud Rate	x	x	x	x	x	Baud (2:0)		

Baud			Baud Rate
Bit2	Bit1	Bit0	
0	0	0	2400
0	0	1	4800
0	1	0	9600
0	1	1	19200
1	0	0	28800
1	0	1	38400
1	1	0	57600
1	1	1	115200
X (7:3)			Do not care

4.2.8 ReadBaud

reads the currently adjusted baud rate.

ReadBaud command (0x45)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (W)	A7	A6	A5	A4	A3	A2	A1	0
1	GetStatus	0	1	0	0	0	1	0	1

ReadBaud Response									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave address (R)	A7	A6	A5	A4	A3	A2	A1	1
1	Slave address	Slave address of the selected device							
2	Baud	X	X	X	X	X	Baud (2:0)		

4.2.9 ResetBuffers

deletes UART and I2C temporary RAM data.

SaveInitSlvAdr command (0x66)									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	ResetBuffers	0	1	1	0	0	1	1	0

4.2.10 ReadVersion

reads the version of the adapter.

ReadVersion command (0x76)									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (W)	A7	A6	A5	A4	A3	A2	A1	0
1	ReadVersion	0	1	1	1	0	1	1	0

ReadVersion Response									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave address (R)	A7	A6	A5	A4	A3	A2	A1	1
1	Slave address	Slave address of the selected device							
2	FM_Version HIGH	version HIGH Byte							
3	FM_Version LOW	version LOW Byte							

e.g. 0x0102 => HiByte = 0x01 ; LoByte = 0x02
The version is read as a BCD code:
=> Version 01.02

4.2.11 SetUARTMode

sets the UART mode.

SetUARTMode command (0x46)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	SetUARTMode	0	1	0	0	0	1	1	0
2	UART Mode	x	x	x	x	x	x	x	**M**

UART Mode		
M	0	***TimeOut*** Data received from the UART is forwarded to the I2C Output after a defined time (milliseconds) is elapsed.
	1	**ETX Character** Data received from the UART is forwarded to the I2C Output after the ETX character is detected.

4.2.12 GetUARTMode

reads UART operation mode.

GetUARTMode command (0x47)									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (W)	A7	A6	A5	A4	A3	A2	A1	0
1	GetUARTMode	0	1	0	0	0	1	1	1

GetUARTMode Response									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave address (R)	A7	A6	A5	A4	A3	A2	A1	1
1	Slave address	Slave address of the selected device							
2	UART Mode	x	x	x	x	x	x	x	**M**

6.2.13 SetETXChar

Sets a new End Of Text – Character, which triggers forwarding the received data to the I2C output.

SetETXChar command (0x48)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	SetSlvAdr	0	1	0	0	1	0	0	0
2	ETX Character	ETX Character(7:0)							

6.2.14 GetETXChar

Reads End Of Text - Character.

GetETXChar command (0x49)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (W)	A7	A6	A5	A4	A3	A2	A1	0
1	GetETXChar	0	1	0	0	1	0	0	1

GetETXChar Response									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (R)	A7	A6	A5	A4	A3	A2	A1	1
1	Slave Address	Slave address of the selected device							
2	ETX Character	ETX Character(7:0)							

6.2.15 SetUARTTimeOut

Sets UART's timeout interval. TimeOut is the wait time in milliseconds after receiving the last character on the UART before the received data is forwarded to the I2C output.

SetUARTTimeOut command (0x4A)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address	A7	A6	A5	A4	A3	A2	A1	0
1	SetSlvAdr	0	1	0	0	1	0	1	0
2	TimeOut	TimeOut Interval [ms] (7:0)							

6.2.16 GetUARTTimeOut

Reads UART's timeout interval.

GetUARTTimeOut command (0x4B)									
Byte	**content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (W)	A7	A6	A5	A4	A3	A2	A1	0
1	GetUARTTimeOut	0	1	0	0	1	0	1	1

GetUARTTimeOut Response									
Byte	**Content**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	Slave Address (R)	A7	A6	A5	A4	A3	A2	A1	1
1	Slave Address	Slave address of the selected device							
2	TimeOut	Timeout interval [ms] (7:0)							

7. ASCII Commands (Only RS232-controlled devices / #020101)

Command	Code			Description		
	Hex	Char	Parameter	CMD + Data (->RS232)	Response (<- RS232)	
GetSlaveAddress*	0x61	a		**’a’**+<CR>	**’61’**+**’XX’+**<CR>	Read Slave address
ChangeSlaveAdress *	0x63	c	d_1	**’c’**+**’XX’+**<CR>	**’63’**+<CR>	Change Slave Address
SaveSlaveAddress	0x73	s		**’s’+**<CR>	**’73’+**<CR>	Save Slave address into EEPROM
WriteData **	0x77	w	$d_1,d_2,...d_{128}$	**’w’+’XXYYZZ....**’+<CR>	**’77’+**<CR>	Write Data into RAM without Interrupt
WriteDataInt1	0x78	x	$d_1,d_2,...d_{128}$	**’x’+’ XXYYZZ....**’+<CR>	**’78’+**<CR>	Write Data into RAM with Interrupt output 1
WriteDataInt2	0x79	y	$d_1,d_2,...d_{128}$	**’y’+’ XXYYZZ....**’+<CR>	**’79’+**<CR>	Write Data into RAM with Interrupt output 2

The data of an I2C master are temporarily stored and then transmitted through the serial interface. A data packet may be maximal 128 bytes.

I2CdataAvail	0x69	i	$d_1,d_2,...d_{128}$		**’69’+’XXYYZZ....**’+<CR>	I2C-Data received from I2C-Master

<CR> : CarriageReturn (0x0D). Commands and data are always terminated with a Carriage Return.

*) ChangeSlaveAdress / GetSlaveAdress:
XX the Slave address.
e.g. in order to change the address to 0x40, we would send the following string terminated with a carriage return: **’c40’+<CR>**
**) WriteData:
XXYYZZ... data to send to the adapter. Packet size is at least one byte and maximum 128 bytes.
Example: in order to write 5 Bytes 0xA1, 0x1F, 0x22, 0x5C, 0xB0 into RAM of the adapter, then the following string (terminated with a carriage return) should be sent over the serial interface: ’**wA11F225CB0**’+**<CR>**

RS232 – Settings:

Baudrate:	19200
DataBits:	8
StopBits:	1
Parity:	None