Specification E2 Interface

Version 4.1

	Name	Date
Created:	Robert Mayr.	15.04.2011
Checked:	Haider A.	15.04.2011
Approved:		
Reason for change:	Text corrections	

TABLE OF CONTENTS

1 Introduction

1.1 Overview

The E2 interface has been specified by E+E Elektronik in 2004 and represents a subset of the E2 interface protocol. The E2 interface is based on similar principles to the I²C Bus[1] or the SMBus[1] introduced by Phillips Semiconductors in 1982. The most significant differences to the I²C Bus are the slower transmission rate, the slightly divergent addressing mechanism and the error detection (checksum). The E2 interface is used for the digital, bi-directional data transmission between a master module (e.g.: climate controller, microcontroller, etc.) and a slave module (often an E+E transmitter). The data transmission takes place in synchronous and serial modes, whereby the master is responsible for generating the clock pulse. The slave cannot send any data independently.

1.2 Agreements

The technical specifications in this document are to be regarded as recommendations and apply for all E+E transmitters with E2 interface, assuming nothing to the contrary is specified in their data sheets.

Terms:

Master: Module with E2 interface that initiates communication and can drive a clock signal.

Slave: Module with E2 interface that cannot drive a clock signal

2 Characteristics

In the following, the E2 interface will be specified in accordance with the ISO-OSI Layer Model up to Layer 3. In addition, it should also benoted that due to the significant similarity to the SM Bus or the I²C Bus, the E2 interface is hardware-compatible with popular microcontrollers with interfaces of that type. This means that E2-slaves can be directly connected to the corresponding pins of these controllers.

2.1 Layer 1: Physical Layer

The E2 interface consists of two active lines (Clock and Data) as well as an earthing line as reference potential. The Clock and Data lines are connected with the operating voltage via pull-up resistors. In the idle state, both lines are at High-Level (positive logic). The respective inputs and outputs of the modules are designed as Open Drain (or Open Collector), and thus can be connected directly as „wired AND“.

[1] *All brands, names, product names and logos listed are registered trademarks or brands of their respective owners that we hereby explicitly acknowledge.*

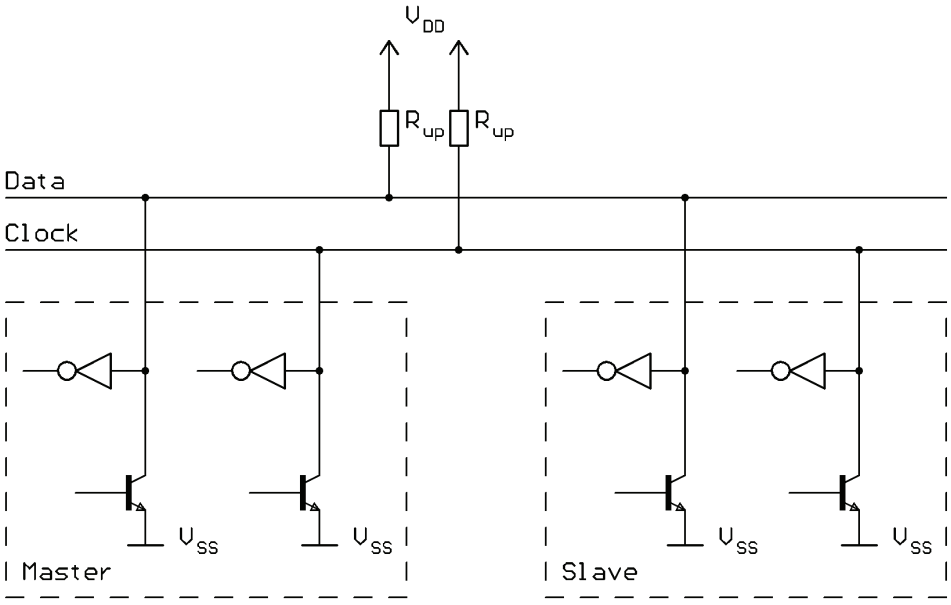


Fig. 1: Principle joining together of master- and slave Module

Parameter

Symbol	Parameter	Minimum	Maximum	Unit	Remark
V_{DD}	Operating Voltage			V	See module description
V_{IH}	Input High Level	0,8 V_{DD}	V_{DD} + 0,3	V	
V_{IL}	Input Low Level	V_{SS} – 0,3	0,2 * V_{DD} or 0,8	V	Smallest value is valid
V_{OL}	Output Low Level		0,7	V	I_{in} = 0,5 mA
C_{max}	Line capacity to ground		1	nF	@ R_{up} = 22 kΩ see Remark 1
f_{CLK}	Clock frequency (data rate)	500	5000	Hz	see Remark 1
t_{CLKH}	Clock-High time	100		µs	
t_{CLKL}	Clock-Low time	100		µs	
R_{up}	Pull-up resistor	1	100	kΩ	see Remark 1

For any other parameters not specified here, the specifications in the data sheets of the modules used apply.

Remark 1: The maximum data rate that can be achieved depends on the combination of the line capacity and the pull-up resistors. The R_{UP} value is the value of all pull-up resistors connected in parallel.

2.2 Layer 2: Data Link Layer

2.2.1 Bit-transfer

The data transmission always takes place serially by bit and synchronously. The clock line is used as a synchronisation line, which is always controlled by the master. With the exception of the start and stop conditions, a change of the level on the data line is only permissible during a LOW-phase of the clock line. At the beginning of every communication there is a start condition. This is realised by a negative edge on the data line, with simultaneous sustained high-level on the clock line. After a delay of at least 4us the clock line is drawn to low-level, and the first data bit (MSB) can be placed on the data line. The data transfer takes place during the high-phase of the clock line. After all data bits (incl. ACK/ NACK) have been sent, the communication is terminated with a stop condition. A stop condition is defined by a positive edge on the data line with simultaneous high-level on the clock line.

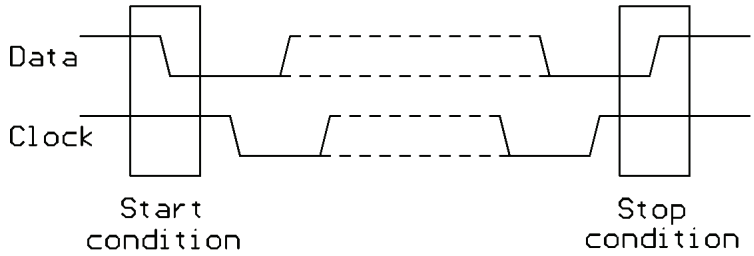


Fig. 2: Bit Transfer

Clock Low Extension

The slave has the option of holding the clock line at low level for up to 25ms after every transmitted data bit to gain a little time for internal operations. The total transmission time for a complete byte may not exceed 35ms, however.

2.2.2 Data Transfer Formats

The data is transmitted from the transmitter in bytes, and confirmed by the receiver individually (for each byte) with ACK (data line = Low) or NACK (data line = High) as ninth bit. The first data byte (control byte) is always sent from the master to the slave. The first four bits represent the main command and the next three bits represent the „address“ of the slave module. The eighth bit (R/W) specifies the direction of the data transfer (R/W=0 data from master to slave; R/W=’1’ data from slave to master).

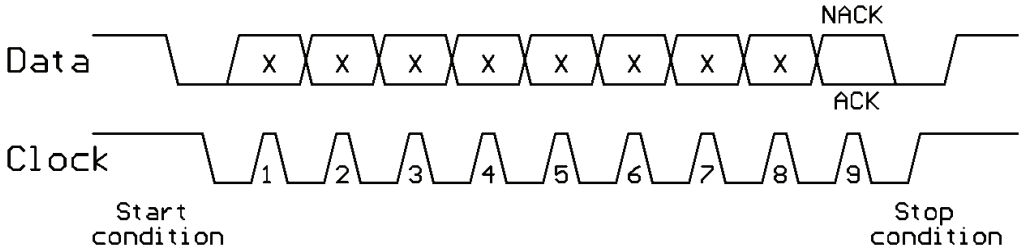


Fig. 3: Byte Transfer

2.2.3 Control Byte

The control byte of the E2 interface is used only for the differentiation of various command modes (which are defined in what is referred to as the main command) and the data flow direction (R/W). The control byte is defined as follows:

Control Byte							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Main Command				Device Address			R/W

Fig. 4: Control Byte Structure

Bit	Meaning
0 (LSB)	R/W
1	Device address low bit
2	Device address
3	Device address high bit
4	Main command low bit
5	Main command
6	Main command
7 (MSB)	Main command high bit

2.2.4 PEC Packet Error Code (Checksum)

For the detection of transmission errors a checksum byte is transmitted as the last data byte of every transmission. The checksum byte corresponds to the low byte of the sum (unsigned char) of all transmitted bytes.

Read: Checksum byte = (Control byte + Data byte) MOD 0x100

Write: Checksum byte = (Control byte + Address byte + Data byte) MOD 0x100

2.3 Layer 3: Network Layer (Protocol)

All defined commands of the E2 interface are described in this section.Refer to the data sheets of the modules used for the data format of the transmitted data bytes.

2.3.1 Read Byte from Slave

This command is supported by all modules with E2 interface and is used to read individual data bytes.

Command structure:

Start	Control Byte									Data Byte									Checksum									Stop
	7	6	5	4	3	2	1	0	ACK	7	6	5	4	3	2	1	0	ACK	7	6	5	4	3	2	1	0	ACK	
Start								1	A	x	x	x	x	x	x	x	x	A	x	x	x	x	x	x	x	x	NA	Stop

From Master to Slave

From Slave to Master

Attention: For the Read command, the bit 0 of the control byte is always ‘1’ (High). The Read command is terminated by a **NACK** and a stop condition of the master.

For Read Byte from Slave, the Main Commands are organised so that important data and information bytes (all measured values, status byte, sensor type, etc.) can be read out with one single command (at the expense of I²C address space).

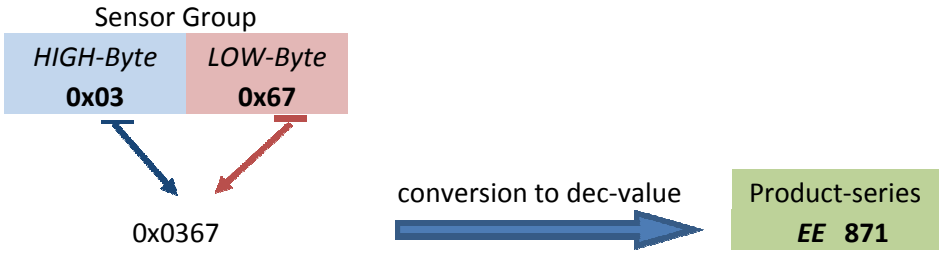
Non-implemented main commands will be answered as „0x55“or “0xFF”.

The following Main Commands are defined:

Control byte	Bit number 7	6	5	4	Type of command
0x11	0	0	0	1	Sensor type (group L-Byte)
0x21	0	0	1	0	Sensor type (subgroup)
0x31	0	0	1	1	Available physical measurements
0x41	0	1	0	0	Sensor type (group H-Byte)
0x51	0	1	0	1	Read from internal custom address
0x71	0	1	1	1	Status byte
0x81	1	0	0	0	Measurement value 1 low byte
0x91	1	0	0	1	Measurement value 1 high byte
0xA1	1	0	1	0	Measurement value 2 low byte
0xB1	1	0	1	1	Measurement value 2 high byte
0xC1	1	1	0	0	Measurement value 3 low byte
0xD1	1	1	0	1	Measurement value 3 high byte
0xE1	1	1	1	0	Measurement value 4 low byte
0xF1	1	1	1	1	Measurement value 4 high byte

2.3.1.2 Sensor type (group L-Byte & H-Byte, control byte 0x11 & 0x41)

Identifies the current sensor type (16 bits)



2.3.1.3 Sensor type (sub-group, control byte 0x21)

Indicate the slave sub-group (upper 4 bits) and the output type (lower 4 bits)

Example: sub-group = 19 hex for EExxx-1 in FT9 implementation (E2 interface).
sub-group = 46 hex for EExxx-4 in FT6 implementation (4-20mA).

For detailed information see slave product description. Available physical measurements (control byte 0x31)

This byte explains the supported active physical measurements of the slave (which is physical measured from the Slave), independent from the (analog) output. Every bit represents a physical measurement:

Available physical measurements							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserved	reserved	reserved	reserved	CO_2	air velocity	temperature	humidity

1...supported
0...**not** supported (unsupported)

2.3.1.5 Read from internal custom address (control byte 0x51)

When you read out the slave with the “read from internal custom address” command (Control byte = 0x51) you will get the data (one Byte) from the actual custom address (internal address pointer). After powering up the slave, this internal address pointer is always ‘0x00’ and increments after every reading (Control byte 0x51). A direct write operation to this address pointer is only possible by a write command. See 2.3.2 Write Byte to Slave. If the internal custom address is greater than 1Byte the High-Byte is ignored (Internal address = 0xFF ➔ read from internal address ➔ increment internal address ➔ new internal address = 0x00)

2.3.1.7 Read Status byte (control byte 0x71)

Reading the Status byte starts a new measurement (within the slave). After the slave specific measurement time the measurement values are ready for read out.
The status byte provides information about the validity of the last measurement.
Every bit represents a physical quantity (equal to the "Available physical measurements"):

Status byte							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserved	reserved	reserved	reserved	CO_2	air velocity	temperature	humidity

Bit value:

1...error during measurement (see slave description)
0...measurement is OK

2.3.1.8 Read measurements (control byte 0x81 to 0xF1)

When reading out a 16 bit measured variable, it is necessary to read out the low byte first and then the associated high byte. This ensures, that two associated bytes are always read out (when reading the low byte together, the high byte is „captured" in the slave). For the data format of the measured values **refer to the data sheets of the corresponding modules**.

Often used but not compulsory is the following assignment (similar to the bit order):

Measurement value 1 = humidity

Measurement value 2 = temperature

Measurement value 3 = air velocity

Measurement value 4= carbon dioxide (CO_2)

2.3.2 Write Byte to Slave

The “write Byte to Slave” is used to configure the slave.

After a “write byte to slave” command it is required to verify that the data is at the right address. The slave ACK only indicates only that the last command was transferred correctly. The checksum is verified afterwards, meaning it is possible for the previous command to be invalid. Command structure:

Start	Control byte									Address byte									Data byte									Checksum									Stop
	7	6	5	4	3	2	1	0	ACK	7	6	5	4	3	2	1	0	ACK	7	6	5	4	3	2	1	0	ACK	7	6	5	4	3	2	1	0	ACK	
Start	x	x	x	x	x	x	x	0	A	x	x	x	x	x	x	x	x	A	x	x	x	x	x	x	x	x	A	x	x	x	x	x	x	x	x	A	Stop

From Master to Slave
From Slave to Master

Attention: The Bit 0 of the Control byte during the write command is always ‘0’ (low).

For “Write Byte to Slave”, the main commands are organized in such a form, that important bytes can be written with one single command. (at the expense of I²C address space)

The main command is not implemented on all components using the E2 interface. Therefore please check the written variables by an additional read command.

Defined main commands:

Control byte	Bit number 7	6	5	4	Command description
0x10	0	0	0	1	Direct write to custom area
0x50	0	1	0	1	Set internal custom pointer

2.3.2.2 Direct write to custom area (0x10)

The main command 0x10 can be used to write one byte directly into the custom area. The address in the custom area and the data is stated directly in the command structure (see command structure above).

2.3.2.5 Set internal custom (address) pointer (0x50)

When there is a custom area in a slave, there will also be a custom (address) pointer.

Internal custom (address) pointer low Byte = Data byte
Internal custom (address) pointer high Byte = Address byte

This pointer increments automatically after every read from an internal address.

This custom pointer can be used to read out data from the custom area:

1.) Set the custom (address) pointer to the desired value with the **write** main command (Control Byte = 0x50)

2.) Read the custom area with the main command "**read** from internal custom address" (Control Byte = 0x51, auto increment after reading)

2.4 Memory areas

2.4.1 Custom memory

This page is also a functional overview. If in “supported functions” (bytes 0x03…0x3F) a bit is set, the function could be read (and written). See bytes 0x40 and following

Adr.	R /W	Group description	Description	Detail								Comment
0x00	R	**Firmware-Version**	**Main version**	1 = Version 1.xx								FW-version = 0x55.0x55 means, that no command is supported
0x01	R	**Firmware-Version**	**Sub-Version**	12 = Version x.12								FW-version = 0x55.0x55 means, that no command is supported
0x02	R	**E2-Spec**	Version of E2-specification	4 = Version 4								Version of the E2 specification used during product development
0x03	R	**Supported functions**	**Custom adjustment**	7	6	5	4	3	2	1	0	OG = Offset and Gain (each physical quantity) 0= not supported
				reserved	reserved	reserved	reserved	**OG- CO_2**	**OG-v**	**OG-T**	**OG-RH**	
				MSB							LSB	
0x04	R		**Custom adjustment**	7	6	5	4	3	2	1	0	Availability of “save adjustment point value at custom Calibration” (each physical quantity) 0= not supported
				reserved	reserved	reserved	reserved	**APt- CO_2**	**APt -v**	**APt -T**	**APt -RH**	
				MSB							LSB	
0x05	R		**Custom adjustment**	7	6	5	4	3	2	1	0	Availability of the timestamp (custom adjustment, general) 0= not supported
				reserved	reserved	reserved	reserved	reserved	reserved	reserved	**ATime-gen**	
				MSB							LSB	
0x06	R		**Custom adjustment**	7	6	5	4	3	2	1	0	Availability of the timestamp (custom Adjustment each physical quantity) 0= not supported
				reserved	reserved	reserved	reserved	**ATime- CO_2**	**ATime-v**	**ATime-T**	**ATime-RH**	
				MSB							LSB	
0x07	R		**Operating functions**	7	6	5	4	3	2	1	0	Availability of several functions 0= not supported
				Error Code	**Measurement Value Filter**	**Specific Measuring Interval**	**Global Measuring Interval**	reserved	**E2 Bus address**	**Custom name**	**E+E serial number**	
				MSB							LSB	
0x08	R		**Operating mode**	7	6	5	4	3	2	1	0	Availability of several functions 0= not supported
				reserved	reserved	reserved	reserved	reserved	reserved	**E2 priority**	**Measure mode**	
				MSB							LSB	
0x09	R		**Special features**	7	6	5	4	3	2	1	0	Availability of several functions 0= not supported
				reserved	reserved	reserved	reserved	reserved	reserved	reserved	**Auto adjustment**	
				MSB							LSB	
0x0A to 0x3F	R		54 reserved									

Adr.	R /W	Group description	Description	Detail	L/H-Byte	Comment
0x40	R/W	**Custom adjustment**	Humidity	Offset	L-Byte	Offset = signed Int [1/100 %RH]
0x41	R/W			Offset	H-Byte	
0x42	R/W			Gain	L-Byte	Gain = GainValue/32768 (Example: GainValue = 32768 ➔Gain of 1,0000000)
0x43	R/W			Gain	H-Byte	
0x44	R/W			Fpoint_L	L-Byte	[1/100 %RH] Level of last “lower” adjustment point
0x45	R/W			Fpoint_L	H-Byte	
0x46	R/W			Fpoint_U	L-Byte	[1/100 %RH] Level of last “upper” adjustment point
0x47	R/W			Fpoint_U	H-Byte	
0x48	R/W		Temp.	Offset	L-Byte	Offset = signed Int [1/100 K]
0x49	R/W			Offset	H-Byte	
0x4A	R/W			Gain	L-Byte	Gain = GainValue/32768 (Example: GainValue = 32768 ➔Gain of 1,0000000)
0x4B	R/W			Gain	H-Byte	
0x4C	R/W			Tpoint_L	L-Byte	[1/100 K] Level of last “lower” adjustment point
0x4D	R/W			Tpoint_L	H-Byte	
0x4E	R/W			Tpoint_U	L-Byte	[1/100 K] Level of last “upper” adjustment point
0x4F	R/W			Tpoint_U	H-Byte	
0x50	R/W		Air velocity	Offset	L-Byte	Offset = signed Int [1/100 m/s]
0x51	R/W			Offset	H-Byte	
0x52	R/W			Gain	L-Byte	Gain = GainValue/32768 (Example: GainValue = 32768 ➔Gain of 1,0000000)
0x53	R/W			Gain	H-Byte	
0x54	R/W			Vpoint_L	L-Byte	[1/100 m/s] Level of last “lower” adjustment point
0x55	R/W			Vpoint_L	H-Byte	
0x56	R/W			Vpoint_U	L-Byte	[1/100 m/s] Level of last “upper” adjustment point
0x57	R/W			Vpoint_U	H-Byte	
0x58	R/W		CO_2	Offset	L-Byte	Offset = signed Int [ppm]
0x59	R/W			Offset	H-Byte	
0x5A	R/W			Gain	L-Byte	Gain = GainValue/32768 (Example: GainValue = 32768 ➔Gain of 1,0000000)
0x5B	R/W			Gain	H-Byte	
0x5C	R/W			CO2point_L	L-Byte	[ppm] Level of last “lower” adjustment point
0x5D	R/W			CO2point_L	H-Byte	
0x5E	R/W			CO2point_U	L-Byte	[ppm] Level of last “upper” adjustment point
0x5F	R/W			CO2point_U	H-Byte	
0x60 to 0x7F	R/W		32 reserved	Reserved for other physical quantities		

Adr.	R /W	Group description	Description	Detail	L/H-Byte	Comment
0x80	R/W	Custom adjustment	CA - global	Year		Last custom adjustment (date); Year = 6 ➔ 2006
0x81	R/W		CA - global	Month		1 to 12
0x82	R/W		CA - global	Day		
0x83	R/W		CA- Humidity	Year		Last custom adjustment (date) RH; Year = 6 ➔ 2006
0x84	R/W		CA- Humidity	Month		1 to 12
0x85	R/W		CA- Humidity	Day		
0x86	R/W		CA- Temperature	Year		Last custom adjustment (date)Temp.; Year = 6 ➔ 2006
0x87	R/W		CA- Temperature	Month		1 to 12
0x88	R/W		CA- Temperature	Day		
0x89	R/W		CA-Velocity	Year		Last custom adjustment (date) V; Year = 6 ➔ 2006
0x8A	R/W		CA-Velocity	Month		1 to 12
0x8B	R/W		CA-Velocity	Day		
0x8C	R/W		CA-CO_2	Year		Last custom adjustment (date) CO2; Year = 6 ➔ 2006
0x8D	R/W		CA-CO_2	Month		1 to 12
0x8E	R/W		CA- CO_2	Day		
0x8F to 0x9A	R/W		**12 reserved**	**Reserved for other physical quantities**		
0x9B to 0x9F	R/W		**5 reserved**			
0xA0 to 0xAF	R	Configuration	Serial number	Unique E+E serial number		
0xB0 to 0xBF	R/W		Part name	Free usable part name		On delivery it is filled with the E+E sensor type. eg.EE871
0xC0	R/W		Bus-address	Configurable bus-address (0…7)		On delivery the Bus-Address = 0
0xC1	R/W	Error handling	**Error code**	Relevant if status byte marks an error		Gives information about failure
0xC2 to 0xC5	R/W		**4 reserved**			
0xC6	R/W	Time interval	measurement interval	Global measurement interval	L-Byte	unsigned int // unit = 1/10 s
0xC7	R/W		measurement interval	Global measurement interval	H-Byte	unsigned int // unit = 1/10 s
0xC8	R/W		measurement interval	Specific interval moisture		Positive = global Interval multiplier // Negative = global Interval divider
0xC9	R/W		measurement interval	Specific interval temperature		Positive = global Interval multiplier // Negative = global Interval divider
0xCA	R/W		measurement interval	Specific interval velocity		Positive = global Interval multiplier // Negative = global Interval divider
0xCB	R/W		measurement interval	Specific interval CO_2		Positive = global Interval multiplier // Negative = global Interval divider
0xCC to 0xCF	R/W		**4 reserved**	**Reserved for other physical quantities**		
0xD0	R/W	Measurement filter	Filter Humidity	Details see product datasheet		
0xD1	R/W		Filter temperature	Details see product datasheet		
0xD2	R/W		Filter velocity	Details see product datasheet		
0xD3	R/W		Filter CO_2	Details see product datasheet		
0xD4 to 0xD7	R/W		**4 reserved**	**Reserved for other physical quantities**		
0xD8	R/W		Operating mode	See chapter operating mode		
0xD9	R/W		Special features	See chapter special features		
0xDA to 0xDF	R/W		**6 reserved**	**Reserved for other configurations**		
0xE0 to 0xFD	R/W		30 reserved			
0xFE	R	**Addresspointer**	Custom area		L-Byte	
0xFF	R		Custom area		H-Byte	

2.4.1.1 Supported functions

Group description “supported functions” (see table above, column “group description”):

At this addresses you can “ask the slave” if the described functions are supported

See the column “Comment” and “Detail” in the table above.

Appropriate Bit = 0 ➔ function not supported

Appropriate Bit = 1 ➔ function supported

2.4.1.2 Custom adjustment

With this feature you can adjust the physical values in offset and gain.

The values you can set are (if supported):

1.) Offset (signed int)

2.) Gain (unsigned int ➔ gain in “unsigned int” = gain in “float” *38)

3.) Lower calibration point (unsigned int)

4.) Upper calibration point (unsigned int)

5.) Last custom adjustment (Calibration time) global (year, month, day)

6.) Last custom adjustment (Calibration time) every physical quantity (year, month, day)

2.4.1.3 Operating functions

	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Adr. 0x07	Error code	M.Value Filter	Specific M.Interval	Global M.Interval	reserved	E2 bus address	Custom name	E+E serial number
	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

BIT0: E+E serial number
1: It is possible to read (only) the unique E+E serial number
0: Reading the E+E serial number isn’t possible.

BIT1: Custom namer
1: It is possible to read and write a free usable part name (16 Byte)
0: A free usable part name is unsupported

BIT2: E2 bus address
1: It is possible to change the device address. You can control up to 8 devices on the same E2-lines (CLK & DATA). That means one master and up to 8 slaves
0: Only a single master to slave connection is possible

BIT3: reserved
1:
0:

BIT4: Global measurement interval
1: Changing the global measurement interval is possible
0: A fixed measurement interval is implemented (see product datasheet)

BIT5: Specific measurement interval
1: Changing the specific measurement interval is possible (see product datasheet)
0: no specific measurement intervals. Only a global measurement interval is possible. (see product datasheet)

BIT6: Measurement value filter
1: Changing the measurement value filter is possible (see product datasheet)
0: There is a fixed measurement value filter implemented

BIT7: Error code
1: An error code can be requested if the status byte marks an error (see product datasheet)
0: No error code supported

2.4.1.4 Operating mode

Supported function

	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Adr. 0x08	reserved	reserved	reserved	reserved	reserved	reserved	E2 priority	Low power mode
	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

BIT0: Low power mode
0: going into Low power mode is not possible
1: going into Low power mode is supported

BIT1: E2-priority (slave measurement and E2 communication at the same time)
0: Changing the E2-priority is not possible. Slave answers with NACK during measurement
1: Changing the E2-priority is supported

BIT2-7: reserved

Documentation

	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Adr. 0xD8	reserved	reserved	reserved	reserved	reserved	reserved	E2 priority	Low power mode
	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

BIT0: Measure mode
0: Free running mode or trigger mode
1: Low power mode (measurement only after “read statusbyte”)

BIT1: E2-priority (slave measurement and E2 communication at the same time)
0: Priority to measurement. Slave answers with NACK during measurement
1: Priority to E2 communication

BIT2-7: reserved

2.4.1.5 Special features

Supported function

	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Adr. 0x09	reserved	reserved	reserved	reserved	reserved	reserved	reserved	Auto adjustment
	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

BIT0: Auto adjustment
0: A manually triggered auto adjustment is not supported
1: A manually triggered auto adjustment is supported

BIT1-7: reserved

Documentation

	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Adr. 0xD9	reserved	reserved	reserved	reserved	reserved	reserved	reserved	Auto adjustment
	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

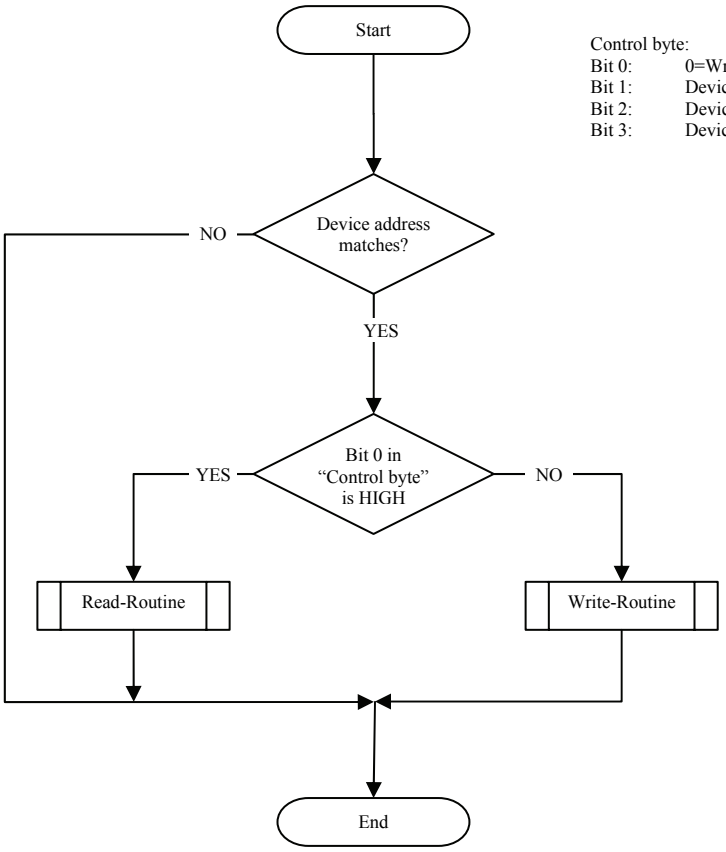
BIT0: Auto adjustment
read = 1 ➔ an auto adjustment is currently running
read = 0 ➔ currently normal operation (no auto adjustment)
set to 1 ➔ starts an auto adjustment. After the auto adjustment is finished, this bit will be cleared automatically.
set to 0 ➔ interrupting the auto adjustment is not allowed and not possible

During the auto adjustment the measurement values are held on the last measured value.

BIT1-7: reserved

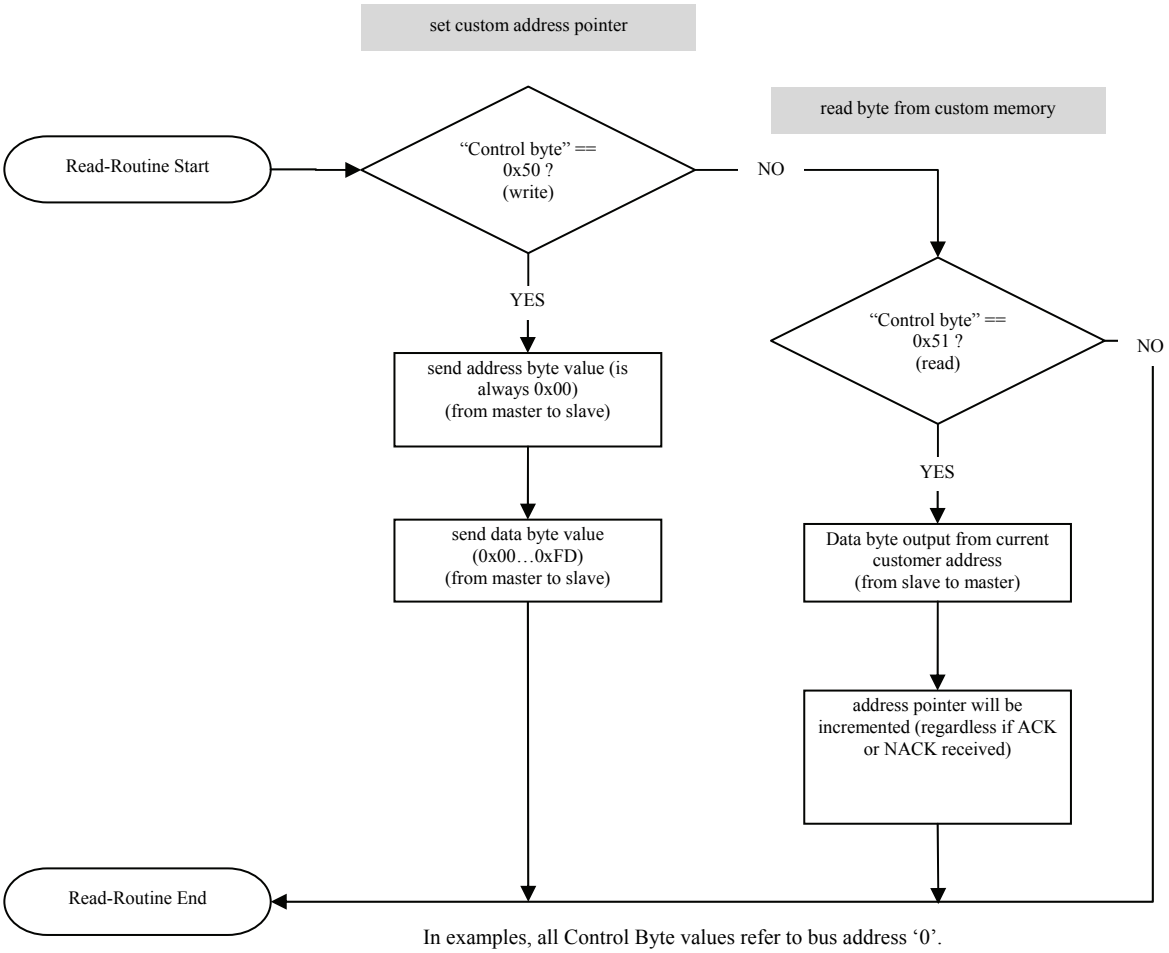
3 Appendix: Flow-charts

3.1 “Addressing + Read/Write distinction” flow chart



3.2 “Read from Memory” flow chart

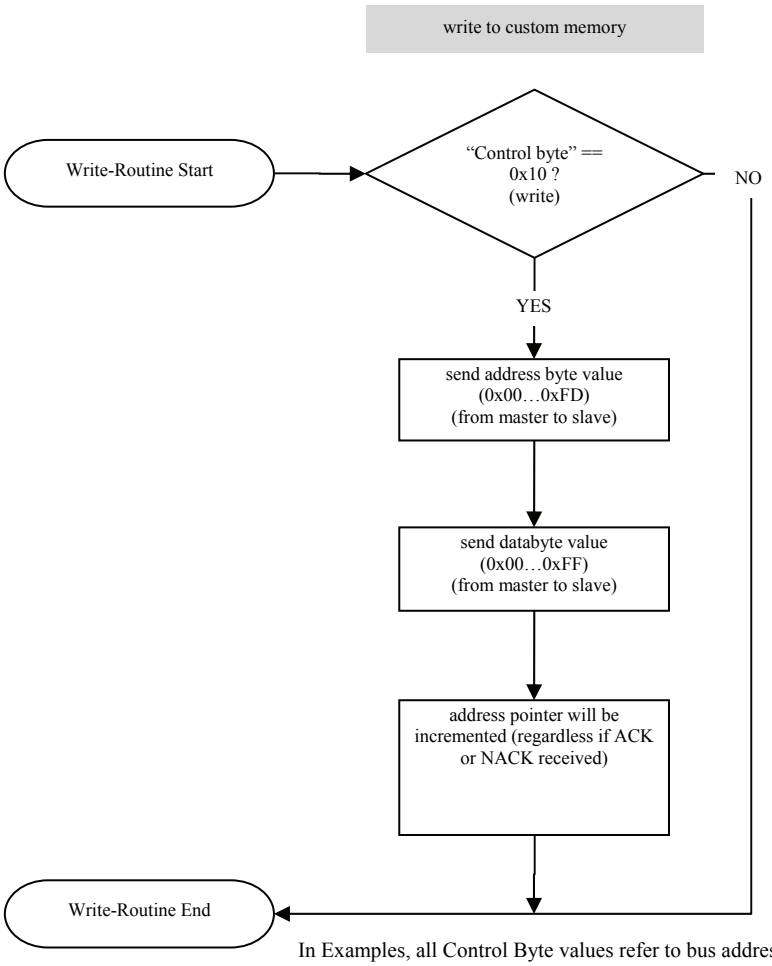
3.2.1 “Read from Custom Memory” flow chart



In examples, all Control Byte values refer to bus address ‘0’.
Unsupported commands or address values will be answered with 0x55 or 0xFF

3.3 “Write to Memory” flow chart

3.3.1 “Write to Custom Memory” flow chart



In Examples, all Control Byte values refer to bus address ‘0’.
Unsupported commands or address values will be answered with 0x55 or 0xFF