EXDUL-384E

EDP-No.: A-381940

EXDUL-384S

EDP-No.: A-381920

8 A/D inputs 16-bit (single ended) or
4 A/D inputs16-bit (differential)
8 D/A outputs16-bit
1 optocoupler isolated digital input
1 optocoupler isolated digital output
counter 32 bit
LCD display (EXDUL-384E only)

wasco®

user‘s guide

Copyright© 2018 by Messcomp Datentechnik GmbH

This documentation is copyright by Messcomp Datentechnik GmbH. All rights are reserved.
Messcomp Datentechnik GmbH reserves the right to modify the products described in this manual at any time and without preannouncement.
No parts of this manual are allowed to be reproduced, copied, translated or transmitted in any way without prior written consent of Messcomp Datentechnik GmbH.

Registered Trademarks

Windows®, Visual Basic®, Visual C++®, Visual C#® are registered trademarks of Microsoft.
wasco® is registered trademark.
EXDUL® is registered trademark.
LabVIEW® is registered trademark.
Other product and company names mentioned may be trademarks of their respective owners

Disclaimer

The information in this manual is intended to be accurate and reliable. The company Messcomp Datentechnik GmbH does not assume any liability for any damages arising out of the use of the A/D converter module EXDUL-384 and this documentation, neither for direct nor indirect damages.

Important Information:

This manual was made up for the modules EXDUL-384E and EXDUL-384S. EXDUL-384E additionally provides an LCD display, all other functions are identical. For the EXDUL-384S all commands and functions concerning the LCD display are not applicable.

wasco®

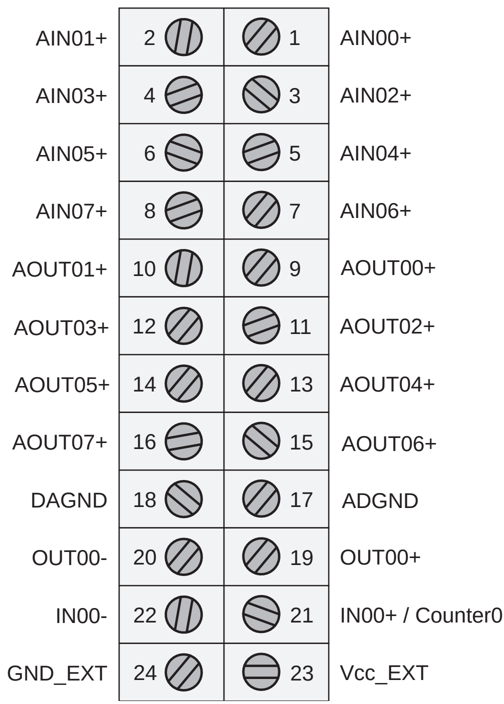
Table of Contents

1. Introduction

EXDUL-384 provides either eight ground referenced or four differential 16bit A/D input channels. You can adjust several bipolar input voltage ranges (+/-0.63 V, +/-1.27 V, +/-2.55 V, +/-5.1 V, +/-10.2 V). The conversion process including the associated configuration of the A/D components (selection of range and channel) is triggered by software commands. The output voltage ranges (+/-2.55 V, +/-5.1V, +/-10.2 V) of all of the eight 16bit D/A outputs are software-selectable as well.
Additionally the module provides one digital input and one digital output galvanically opto-isolated by high-quality optocouplers and additional protection diodes.
Special high power output optocouplers cope with a switching current up to 150 mA. If necessary, the optocoupler input can be programmed and used as a counter input. The programmable LCD display of the EXDUL-384E shows either digital I/O status information or programmable user-specific data.
The module is powered with the necessary operating voltage by USB or by an external power supply. The module provides a 24pin screw terminal block for connecting the external power supply as well as the input and output optocoupler.
The compact casing enables the module to be used as a portable device with a notebook. For mechanical or control engineering it can also be easily wall mounted or attached to DIN mounting rail.

2. Connection Terminals

2.1 Terminal Assignments of CN1



Vcc_EXT:
Connector for external voltage source
GND_EXT:
Ground connection when using external voltage source

3. System Components

3.1 Block Diagram EXDUL-384E

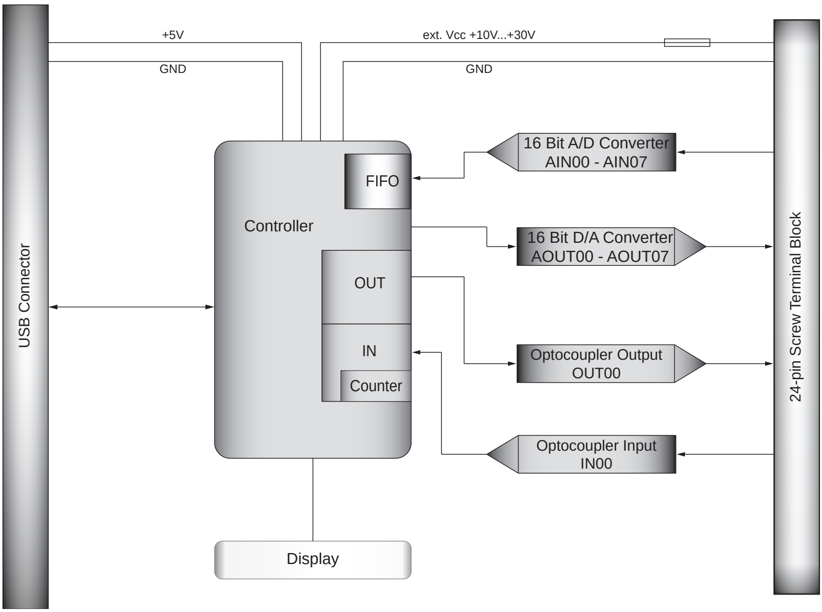


Fig. 3.1 Block diagram EXDUL-384E

3.2 Block Diagram EXDUL-384S

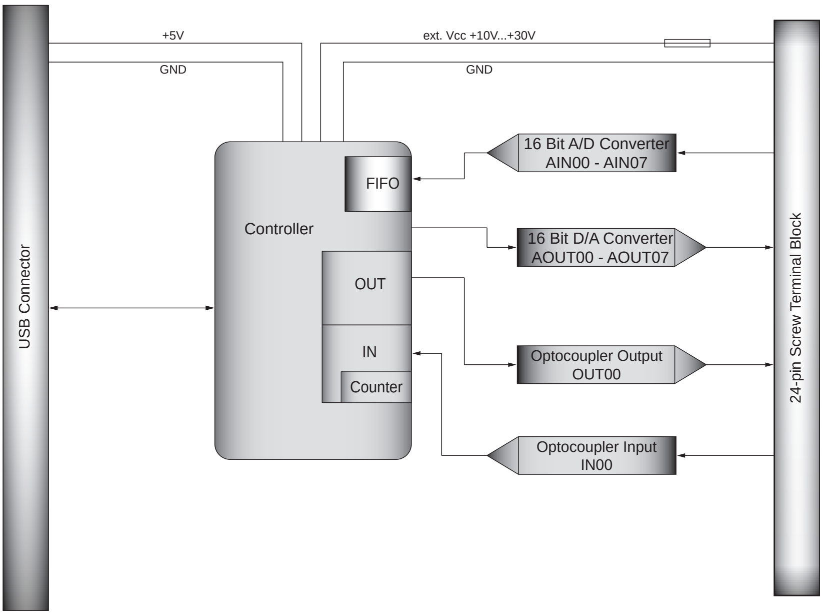


Fig. 3.2 Block diagram EXDUL-384S

3.3 A/D Inputs
8 inputs single-ended (se)
or 4 inputs differential (diff)
or combined se/diff software-selectable
Resolution: 16 bit
Input voltage ranges bipolar:
+/-0.63 V, +/-1.27 V, +/-2.55 V, +/-5.1 V, +/-10.2 V,
+/-20.4 V (differential inputs only)
FIFO: 10,000 measuring values
Input resistor: > 500 MΩ
Over voltage protection: +/- 50V
Measuring cycle: max. 10 µs
Sampling rate: max 100 kS/s

3.4 D/A Outputs
8 outputs
Resolution: 16 bit
Output voltage ranges
bipolar: +/-2.55 Volt, +/-5.1 Volt, +/-10.2 Volt
Output current: max +/-5 mA

3.5 Optocoupler Input
1 channel, galvanically isolated
Over voltage protection diodes
Input voltage range
high = 10..30 Volt
low = 0..3 Volt
Input frequency: max. 10 kHz

3.6 Optocoupler Output
1 channel, galvanically isolated
High capacity optocoupler
Reverse polarity protection
Output current: max. 150 mA
Switching voltage: max. 50 V

3.7 Counter
1 programmable counter 32 bit
Counting frequency: max. 5 kHz

3.8 LCD Display (only EXDUL-384E)
Matrix display with 2 lines and 16 columns displaying 16 characters each line
Programmable to display user specific data or I/O state

4. Commissioning

Connecting to a computer is made simple and uncomplicated in a Plug-and-Play manner via USB interface. The module is powered with the required operating voltage via USB port or via an external voltage source.

4.1 Connecting to a USB Port

The EXDUL-384E / EXDUL-384S provides a USB 2.0 interface and can be connected directly to the computer or to a USB hub using the enclosed USB connecting cable. The connection supports hot-plug function, that means, it is possible to connect the module even during the system is operating.

4.2 Power Supply via USB Port

If required, it is possible to power the module EXDUL-384 via the USB port exclusively without limitations. For this, it must be ensured that the PC is able to supply 500mA via the USB interface.

4.3 External Power Supply

EXDUL-384E / EXDUL-384S firmware automatically detects when an external voltage source is connected. Applying a voltage between +10V and +30 V across Vcc_EXT and GND_EXT (see figure terminal assignments) immediately causes the device to switch to „external“ source. The power supply from the USB port is automatically interrupted.
Attention: You must not change the mode of power supply during operation!

4.4 LCD Display during Commissioning (EXDUL-384E only)

During commissioning or at start of the module, the display shows an information representing the module name. After 5 seconds, the module name is replaced by either by I/O status display or UserLCD display, depending on the LCD display configuration.

4.5 LCD display during operating (EXDUL-384E only)

Starting the module the display switches from the info display to the digital I/O status display or the UserLCD display after approx. five seconds depending on the LCD display configuration. When I/O status display is selected, line1 shows the current input states, line2 the output states. If the UserLCD modus was activated by calling the intended command before the last shutdown of the system, the values from the memory areas UserLCD1m and UserLCD2m are shown instead of the I/O status display. Data from both of the registers are displayed until new user data is written to the display UserLCD line1 and UserLCD line2. To avoid „screen-burn“ while in operation the display switches from I/O status or UserLCD display to info display for approximately five seconds every minute.

5. 8 A/D Inputs 16 Bit

The EXDUL-384 provides 8 multiplexed single ended or 4 16bit A/D input channels with programmable input voltage range. When the conversion is triggered, the computer will transfer configuration data for conversion (channel, range) in the form of two bytes. After error corrections (such as offset errors) the module transmits the measured value transformed in a voltage value in µV as a response or stores it in a FIFO.

5.1 Single ended Operation

In single ended operation mode, a maximum of 8 input channels are available. All input voltage ranges are measured against the ground (ADGND) of the A/D components (see figure 5.1). Find a more detailed description of the circuitry in chapter 10.4.

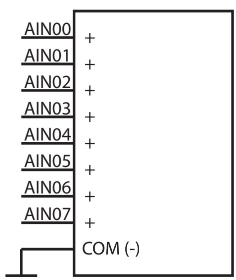


Figure 5.1 A/D converter single ended

As mentioned before, one byte for channel selection will be added to the command for measuring the voltage.
Please see table 5.1 to choose the appropriate channel for each value when single ended measuring is employed.

Channel Byte	Channel selection single ended								
	1	2	3	4	5	6	7	8	ADGND
0_{dez}	+								-
1_{dez}		+							-
2_{dez}			+						-
3_{dez}				+					-
4_{dez}					+				-
5_{dez}						+			-
6_{dez}							+		-
7_{dez}								+	-

Table 5.1 A/D Converter Single-ended Measurement

For example, for a single ended measurement of channel 3, the positive pole of the voltage source has to be connected to AIN02 and the negative pole to ADGND. The channel byte of the command then is 2_{dez}.

5.2 Differential Operation

In differential operation mode, a maximum of 4 input channels are available. In differential mode each channel provides one positive and one negative input (see figure 5.2-1). Please note, all channels must be referenced to ground (ADGND) as well. Find a more detailed description of circuitry in chapter 12.5.
The differential measurement can reduce generally occurring interference voltages on both of the signal lines and the analog ground.

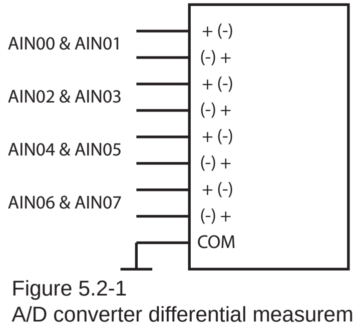


Figure 5.2-1
A/D converter differential measurement

Here too, the channel is selected via the channel byte added to the command for measuring the voltage. You can find the corresponding values in following table:

Channel Byte	Differential channel selection								
	1	2	3	4	5	6	7	8	AGND
8_{dez}	+	-							
9_{dez}	-	+							
10_{dez}			+	-					
11_{dez}			-	+					
12_{dez}					+	-			
13_{dez}					-	+			
14_{dez}							+	-	
15_{dez}							-	+	

Table 5.2 AD converter differential measurement

Serving as an example now the difference between two voltages shall be measured at the inputs AIN05 and AIN06. For this you have to connect the first voltage to AIN05 and the second one to AIN06 (see figure 5.2-2).
Now either the value 12_{dez} (AIN05+ / AIN06-) or 13_{dez} (AIN05- / AIN06+, the result is a negative differential voltage) can be used as the channel byte.

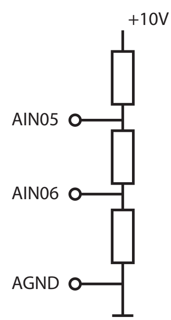


Fig. 5.2-2

Attention:
Please take particular care to ensure, that the difference between the inputs must be whithin the input voltage range.
An input voltage of +10V at AIN05 and an input voltage of -10V at AIN06 would result in a difference of +20V, i.e. an input voltage range of +/- 20.4V has to be chosen (see chapter 5.4)

5.3 Combination of single ended and differential Measurement

If required, the measurement methods can also be varied channel by channel as in fig. 5.3 or even changed „on the fly“ between the individual measurements.

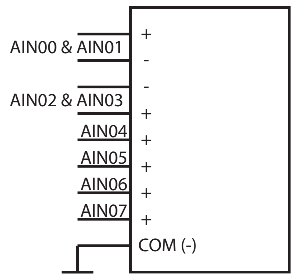


Fig. 5.3

5.4 Input Voltage Range

To measure a voltage several input voltage ranges are available (+/-0.63 V, +/-1.27 V, +/-2.55 V, +/-5.1 V, +/-10.2 V). This permits the range to be adjusted to the input signal, thus optimizing the measuring accuracy.

Along with the measuring command, the computer sends a range byte to the module to select the required voltage range.

Following the individual ranges and the corresponding byte values are listed:

Input Voltage Range	
Byte Value	Voltage
0	+/- 20.4V (differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Table 5.4 A/D converter input voltage ranges

a) Single-Ended Measurement

As shown in Fig. 5.4.1, when measuring single-ended the input signal is referenced to ground. The maximum or minimum voltage to be measured at a voltage range of +/- 10.2V is +10.2V and -10.2V respectively.

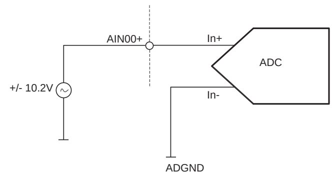


Fig. 5.4.1

Attention: since the maximum voltage to be measured at the analog input (e.g. AIN00+) is 10.2V, a voltage range of +/- 20.4V is not available for a single-ended measurement!

b) Differential measurement

For differential measurements, the input voltage range used corresponds to the maximum difference between the selected inputs. For this, as shown in Fig. 5.4.2 , an input voltage range of +/- 0.63V can be chosen, although at the inputs a voltage of up to +/- 10.2V is applied.

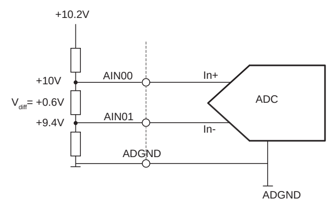


Fig. 5.4.2

When using differential measurement, in contrast to the single-ended measurement there is also an input voltage range of +/- 20.4V.

Attention: For an input voltage range of +/-20.4V the maximum or minimum input voltage of +10.2V resp. -10.2V is true. Only the difference between two inputs may be +20.4V or -20.4V (e.g. AIN00 = +10.2V and AIN01 = -10.2V, V_{diff} = 20.4V)

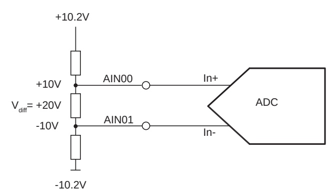


Fig. 5.4.3

5.5 Modes of Measurements

To facilitate the application, the EXDUL-384 provides several modes of measurement.

5.5.1 Single voltage measurement

In the single measurement, upon receiving the appropriate command, the module runs a measurement on the selected input, calibrates it and provides the value in µV in response to the user.

5.5.2 Single voltage measurement with averaging

In this measurement mode, the module runs 32 measurements at the user-selected input at intervals of 10 µs each, forms an average, calibrates the measurement and provides the result in µV to the user.
This measurement mode is particularly suitable for smaller input voltage ranges in order to suppress interferences such as noise.

5.5.3 Block measurement with averaging

This measurement mode is intended for applications, in which voltages at several inputs are to be measured as precisely as possible and in a timely manner. When transferring the command to the module, the selected channels (up to 8) with the respective voltage range are transferred. Upon receipt of the command, the module starts sampling each selected channel successively 32 times in 10µs increments.

Duration = Number of channels*32*10µs

After completion, the values are calibrated and returned to the user in µV.

Example:

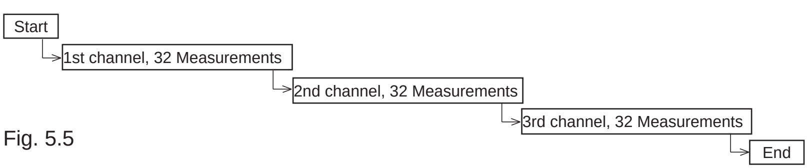


Fig. 5.5

In this example, three channels are to be sampled (e.g. AIN01+, AIN03+, AIN05+). These channels are transferred along with the command, and the module starts running 32 measurements of the first channel (here AIN01+). As soon as the measurements of the first channel have been completed, the sampling of the second channel is started. Once all of the channels have been sampled (duration here 960µs = number of channels*32*10µs), offset and gain errors are calibrated and the voltages in µV are transferred.

5.5.4 Multiple measurement

In the multiple measurement mode, up to 8 channels can be sampled several times (up to 65,535 times). Along with the command, the desired sampling rate (1 - 100kS/s) and the desired channels with the respective voltage range are transmitted. After receiving the command the module runs the measurements and stores the calibrated values in µV into the FIFO. These values can be retrieved from the FIFO at any time. It is important to ensure that the FIFO does not overflow. Additionally, you must not write to any EXDUL information register during this period.

5.5.5 Continuous measurement

In the continuous measurement mode, up to 8 channels with any measuring range and up to 100kS/s can be sampled in continuous operation. For this purpose, there is a start and a stop command. The calibrated measured values in µV are written to the FIFO and can be retrieved from there at any time. It is important to ensure that the FIFO does not overflow. Additionally, you must not write to any EXDUL information register during this period.

5.6 Adjustment of the A/D Inputs

The module is adjusted at an ambient temperature of approx. 20°C at the final test of our production. If there are larger temperature deviations at the end application, the A/D component can be adapted to the environment by subsequent adjustment. The required software is provided on the enclosed CD or on the Internet.

6. 8 D/A Outputs 16 Bit

The EXDUL-384 features a total of eight digital-to-analog converter (DAC) outputs. These may operate with different output voltage ranges each.

6.1 Output Voltage Range

The DAC outputs each provide a variable output voltage range, which can be configured via a range byte in a special intended command.
This selection can be changed „on-the-fly“, that is, for one voltage output (e.g. -7V) you can employ the range bipolar +/-10.2V and for the next voltage output (e.g. -3V) the range bipolar +/-5.1V to achieve a higher resolution.
Following table shows the assignment of the range byte value and the output voltage range:

Output Voltage Range	
Range Byte	bipolar
0	+/-10.2V
1	+/-5.1V
2	+/-2.55V (default)

Table 6.1 D/A converter output voltage ranges

6.2 Adjustment of the D/A Outputs

The module is adjusted at an ambient temperature of approx. 20°C at the final test of our production. If there should be larger temperature deviations at the end application, the D/A component can be adapted to the environment by subsequent adjustment. The required software is provided on the CD or on the Internet.

7. One Optocoupler Input

The EXDUL-384 provides one input channel, optically isolated by opto-coupler. The isolation voltage between the ground of the computer ad the and input is 500 volts.

7.1 Pin assignment of the input optocoupler

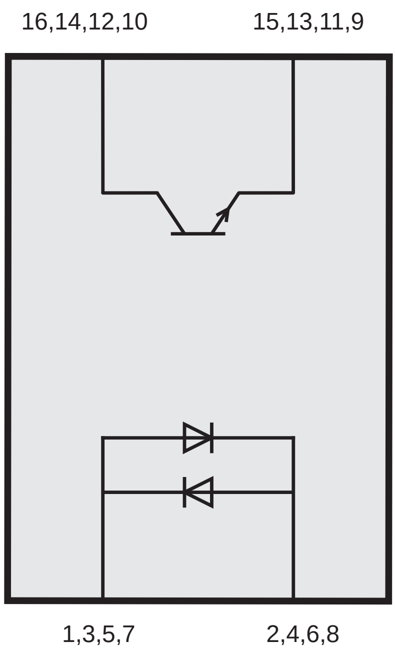


Fig. 7.1

7.2 Input Circuitry

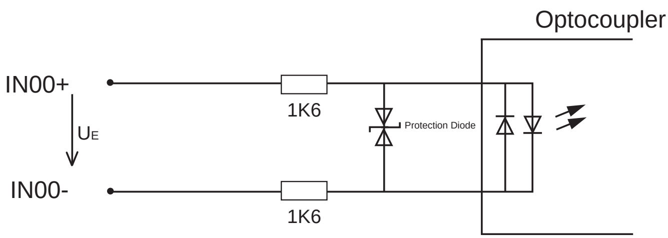


Fig. 7.2

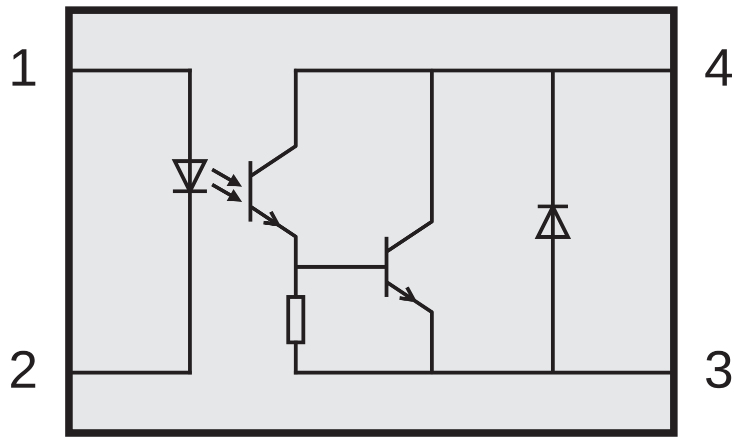
7.3 Input Current

$$I_E \approx \frac{U_E - 1{,}1V}{3200\Omega}$$

8. One Optically Isolated Output

The EXDUL module provides one output channel, which is optically isolated by optocoupler. The isolation voltage between the ground of the module and the output is 500 volts.

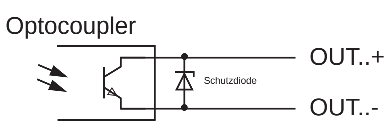
8.1 Pin assignment of the output optocoupler



8.2 Optocoupler specifications

Voltage collector-emitter:	max. 50V
Voltage emitter-collector:	0,1V
Current collector-emitter:	150 mA

8.3 Output circuitry



9. Information, LCD and User Register

9.1 Register HW Identification and Serial Number

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
HW Identification	E	X	D	U	L	-	3	8	4			V	1	.	0	1
	45_{hex}	58_{hex}	44_{hex}	55_{hex}	$4C_{hex}$	$2D_{hex}$	33_{hex}	38_{hex}	34_{hex}	20_{hex}	20_{hex}	56_{hex}	31_{hex}	$3E_{hex}$	30_{hex}	31_{hex}
S/N	1	0	4	4	0	2	6									
	31_{hex}	30_{hex}	34_{hex}	34_{hex}	30_{hex}	32_{hex}	36_{hex}									

Table 9.1 Register HW identification and serial number

The module name as well as the firmware version are stored in the HW identification register and can be used to verify the product identity by the user. The table above serves as an example as for module EXDUL-384 with firmware version 1.01. The line HW identification shows each Hex value and the corresponding ASCII character.

The register Serial Number is a read-only register. The serial number in the table above serves as a format example. The line S/N shows each Hex value and the corresponding ASCII character as for the serial number 1044026.

9.2 Memory Spaces UserA, UserB, UserLCD1m* and UserLCD2m*

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UserA																
	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}
UserB																
	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}
UserLCD1m*																
	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}
UserLCD2m*																
	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}	20_{hex}

Each of the registers UserA and UserB hold 16 digits (16 byte) for customizing. The data is retained when you switch off, registers can be set back to their factory settings (delivery state) by a default reset. In delivery state in all of the user memory areas each digit is set to the Hex value 20 corresponding to a blank in ASCII code.
The table above shows each Hex value and the corresponding ASCII character.

9.3 Display Register UserLCD-line1*, UserLCD-line2* and LCD Contrast*

If UserLCD mode is activated you can write to both of the UserLCD-line1 and UserLCD-line2 registers any 16 characters. Once entered this will be displayed instead of the data from UserLCD1m* and UserLCD2m*. The data in the registers UserLCD-line1 and UserLCD-line2 are **not** stored when switching off.
You can adjust the LCD display contrast in register LCD contrast. This adjustment is retained when switching off.

*: EXDUL-384E only, not applicable with EXDUL-384S!

10. Driver Installation

10.1 Windows Driver

Attention: as of Windows10 no special driver needs to be installed for the module!

When you connect the USB module EXDUL-384E / EXDUL-384S to your PC for the first time, Windows® automatically detects a new hardware and searches for a suitable driver.

To install the driver type the folder or the directory and the name of the setup file „wascoxmfe_v0**x**.inf" to the Windows Hardware Wizard (type the number of the INF file instead of x, for example wascoxmfe_v0**6**.inf)

Having updated the driver database the hardware wizard will inform you of the successful driver installation.

The Windows® Device Manager will now show your USB module EXDUL-384E / EXDUL-384S as a "Wasco-USB-Kommunikationsport COMx" in its directory connections tree (COM/LTP). All Windows® software can access the virtual interface as if it were a real COM port.

10.2 Linux Driver

The EXDUL-384 is using a default COM Port Driver, which is already installed in the most common Linux distributions.
When the module is connected to the USB interface the module will be listed in the folder dev (e.g. as ttyACM0 under Ubuntu).

11. Programming under Windows®

11.1 Introduction

After successful installation the USB module EXDUL-384E / EXDUL-384S is listed as “Wasco Communications Port COMx “ in the Windows® Device Manager. This is a CDC device (Communications Device Class), that is addressed via a virtual COM port.
This virtual COM port operates like a normal COM interface and can be accessed by default Windows® drivers, there is no need to install any additional drivers.

11.2 Modes of Programming

There are several ways to access to the EXDUL module. So the library EXDUL.dll can be used for programming under Windosw and .NET. This allows a quick and easy start to program the access to the module. Furthermore, you can use Com Port libraries which are available in many programming languages such as C or Delphi. They often allow a wide range of interface settings and in parts also an event programming (read buffer does not need to be polled).
LabVIEW user also easily can access to the module using the EXDUL.dll or VISA functions blocks (Serial Port).

11.3 Programming under Windows using the .NET EXDUL.dll Library

If you use a .NET programming language (C#, C++, .NET or VB.NET) to access to the module, you can use the EXDUL.dll Library. It is structured object-oriented, so each EXDUL module is represented by an object with its methods.
Developing the library, special regard was paid to an API between the different EXDUL modules as uniform as possible. This facilitates the user, if necessary, to switch from e.g. a USB EXDUL module to an Ethernet EXDUL module (for example EXDUL-384 -> EXDUL-584) without extensive programming efforts.

Open:
bool Open()
Returned values: true if successful / false at error
Description: Establishes the connection to the module

Close
void Close()
Description: Closes the connection to the module

Write to Info register:
void SetModulInfo (byte type, string info)
Parameter: type: Info Type (see manual)
info: Info string with up to 16 characters
Description: writes to the information registers

Info Area	Info Byte
UserA	0
UserB	1

Read Info Register:
string GetModulInfo(byte type)
Parameter: type: Info-Typ (see manual)
Returned values: Returns the register "type" as a string
Description: Reads the Module Information Register

Info Area	Info Byte
UserA	0
UserB	1
Hardware Identifier	3
Serial Number	4

Write to LCD Register UserLCD:
void SetUserLCD(byte *line*, string *text*)
Parameter: *line*: 0 = 1st line / 1 = 2nd line
text: LCD text up to 16 characters long
Description: Writes to the UserLCD registers. The parameter *line* determines the line (0 or 1) and *text* the text of 16 characters.

Write to LCD-Register UserLCDm:
void SetUserLCDm(byte *line*, string *text*)
Parameter: *line*: 0 = 1st line / 1 = 2nd line
text: LCD text up to 16 characters long
Description: Writes to the UserLCDm registers. The parameter *line* determines the line (0 or 1) and *text* the text of 16 characters.

Write the LCD-Mode:
void SetLCDMode(byte *mode*)
Parameter: *mode*: LCD mode
Description: sets the LCD mode

LCD Mode	LCD Mode Byte
IO Mode	0
User Mode	1

Read the LCD Mode:
byte GetLCDMode()
Returned values: LCD-Mode
Description: Reads the LCD mode

LCD Mode	LCD Mode Byte
IO Mode	0
User Mode	1

Write the LCD Contrast Value:
void SetLCDContrast(ushort *contrast*)
Parameter: contrast:Values between 0 and 4095 (recommended 800 to 1800)
Description: Sets the LCD contrast

Read the LCD Contrast Value:
ushort GetLCDContrast()
Returned values: LCD contrast
Description: Reads the LCD contrast

Read the Optocoupler outputs:

uint GetOptoOut()

Returned values:	State of the optocoupler outputs
Description:	Reads the state of the optocoupler outputs

- -

Write the Optocoupler outputs:

void SetOptoOut(uint *value*)

Parameter:	*value*: state of the outputs
Description:	Sets the optocoupler outputs

- -

Read the Optocoupler inputs:

uint GetOptoIn()

Returned values:	current state of the optocoupler inputs
Description:	Reads the current state of the optocoupler inputs

- -

Start Counter:

void StartCounter(byte *index*)

Parameter:	*index*: Counter index
Description:	Starts the counter with the number index

- -

Stop Counter:

void StopCounter(byte *index*)

Parameter:	*index*: Counter index
Description:	Stopps the counter with the number index

- -

Reset Counter:

void ResetCounter(byte *index*)

Parameter: *index*: Counter index

Description: Sets the counter reading of the counter with the number index back to 0

Read Counter:

uint ReadCounter(byte *index*)

Parameter: *index*: Counter index

Returned values: Counter reading

Description: Reads the counter reading of the counter with the number index

---

Read Overflow Flag:

bool ReadOverflowFlagCounter(byte *index*)

Parameter: *index*: Counter index

Returned values: Overflow flag false = no Overflow
true = Overflow

Description: Reads the Overflow Flag of the counter with the number index

---

Reset Overflow Flag:

void ResetOverflowFlagCounter(byte *index*)

Parameter: *index*: Counter index

Description: Resets the Overflow Flag of the counter with the number index

---

AD Single Measurement:

int GetADC(byte channel, byte range)

Parameter: *channel*: Channel

range: Measurement range

Returned values: Measured value in µV

Description: Performs an ADC measurement.

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

AD Single measurement averaging 32 measurements:
public int GetADC_Mean(byte channel, byte range)
Element of EXDUL.EXDUL384

Parameter:	*channel*: Channel *range*: Measurement range
Returned values:	Measured value in µV
Description:	Performs an ADC measurement averaging 32 single measurements.

Channel:

Channel	**Channel byte**
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	**Voltage**
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

AD block measurement with averaging:

int[] GetADC_BlockMean(EXDUL.ADC_CHANNEL_CONFIG_1[] *config*)

Parameter:	*config*:
Returned values:	Measured value in µV
Description:	Performs an ADC block measurement over several channels (see manual)

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Reset ADC-FIFO:
void ResetFIFO()

Description	This command performs a reset of the FIFO. This shoud be done after an overflow.

Read ADC-FIFO Overflow flag:
bool ReadOverflowFlagFIFO()

Returned values:	Overflowflag false = no Overflow / true = Overflow
Description:	Reads the overflow flag of the ADC FIFO. Along with the readout, the overflow flag is reset automatically.

Readout ADC-FIFO:
int[] ReadFIFO()

Returned values:	Returns an array with the measured values. The size of the array depends on the number of measurements
Description:	Reads the ADC-FIFO

AD Multiple Measurement

int[] GetADC_Multi(ushort *counts*, uint *samplerate*, EXDUL.ADC_CHANNEL_CONFIG_1[] *config*)

Parameter: *counts*: number of measurements
samplerate: sampling rate
config: channel configurations

Returned values: Measured value in µV

Description: performs an ADC multiple measurement over one or more channels. The measured values can be retrieved by the function ReadFIFO.

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Start AD Continuous sampling
void StartADC(uint *samplerate*, EXDUL.ADC_CHANNEL_CONFIG_1[]*config*)

Parameter: *samplerate*: sampling rate
config: channel configurations

Description: Starts an ADC continuous sampling over one or more channels. The measured values can be retrieved by the function ReadFIFO. The function StopADC is needed to stop the continuous sampling.

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Stop AD Continuous Sampling
void StopADC()
Description: Ends an ADC continuous sampling

- -

Set DAC Output Voltage:
void SetDAC(byte *channel*, int *voltage*)
Parameter: *channel*: Output channel 0 to 7
voltage: Output voltage

Description: Applies a voltage „voltage“ to the DAC channel "channel". The voltage has to be within the set range.

- -

Determinate the DAC Output Voltage Range:
void SetDACRange(byte *channel*, byte *range*)
Parameter: *channel*: Output channel 0 to 7
range: Output voltage range

Description: Sets the range on the DAC channel „channel“

- -

Factory Reset:
void DefaultReset()
Description: Resets the module to the factory settings. After this command the module has to be shut down and restarted again.

11.4 Programming with Serial COM Port Libraries
Due to the access to the module via standard COM Port libraries, you can program your application across a variety of languages. So under Windows, you can use Delphi or Java besides the .Net Framework. Also on Linux based operation systems applications can be designed (in case a virtual COM Port driver is available).

11.4.1 Communicating with the EXDUL-381
Data is exchanged by transmitting or receiving byte arrays of variable length via the virtual COM interface.

Each permitted transmission string is replied by a defined result or confirmation string.

The last result or confirmation string has to be read before transmitting a new string.

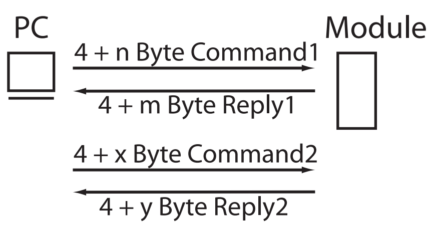


Fig. 11.4 Communication model

11.4.2 Windows® Functions for Programming

You can program EXDUL-384E / EXDUL-384S either via WIN32 API functions or very conveniently via an already existing serial port object in a programming language. You can find example programs in your installation directory on your computer after having installed the software.

Windows® functions for programming:
– CreateFile
– GetCommState
– SetCommState
– WriteFile
– ReadFile
– DCB structure (describes the control parameter of the device)

10.4.3 Command and Data Format

Data is exchanged by transmitting and receiving byte arrays. Each byte array to be transmitted or received consists of at least 4 bytes. The first three bytes perform the command and the fourth byte indicates the number of the following 4 byte blocks.

Command Byte 0	Command Byte 1	Command Byte 2	Length Byte

The number of the 4 byte blocks varies from command to command and depends in part on the volume of data to be transmitted. More detailed information can be found in the individual command descriptions

11.4.4 Command Overview

Hexcode	Representing
0C 00 00	Read and write info register
0C 00 03	Read and write LCD register
0C 00 08	Read and write network configuration
0C 00 0C	Read and write security configuration
0C 00 0D	Change password
08 00 00	Read and write optocoupler outputs
08 00 01	Edit optocoupler inputs
0A 00 00	AD Single measurement
0A 00 01	AD Single measurement with averaging
0A 00 02	AD block measurement with averaging
0A 00 06	ADC FIFO Reset
0A 00 07	ADC FIFO read out overflow flag
0A 00 08	Read out ADC FIFO
0A 00 09	AD Multiple measurement
0A 00 0A	Start AD continuous sampling
0A 00 0B	Stop AD continuous sampling
0A 80 00	Configurate DA input voltage range
0A 80 01	Output DA voltage
09 00 00	Counter0

11.4.5 Structure of Commands

Write to information register

The EXDUL module provides several writable information registers. UserA/B are two 16-byte spaces for the user to store information into a non-volatile memory (FLASH). The registers are writable only as a complete 16-byte block.

Info Space	Info Byte
UserA	0
UserB	1

Example: enter character string EXDUL-384 into register UserA and UserB

Byte	Transmit	Receive	Representing
0	0C	0C	Command code 1st Byte
1	00	00	Command code 2nd Byte
2	00	00	Command code 3rd Byte
3	05	00	Lenght prefix byte => 20 Byte
4	00 (UserA) 01 (UserB)		Info byte
5	00		reserved
6	00		reserved
7	00		Info space of write operation
8	45		Data 1st character E_{asci}
9	58		Data 2nd character X_{asci}
10	44		Data 3rd character D_{asci}
11	55		Data 4th character U_{asci}
12	4C		Data 5th character L_{asci}
13	2D		Data 6th character $-_{asci}$
14	33		Data 7th character 3_{asci}
15	38		Data 8th character 8_{asci}
16	34		Data 9th character 4_{asci}
17	20		Data 10th character $[blank]_{asci}$
18	20		Data 11th character $[blank]_{asci}$
19	20		Data 12th character $[blank]_{asci}$
20	20		Data 13th character $[blank]_{asci}$
21	20		Data 14th character $[blank]_{asci}$
22	20		Data 15th character $[blank]_{asci}$
23	20		Data 16th character $[blank]_{asci}$

Read from information register

The EXDUL module provides several 16-byte wide information spaces which contain module information such as serial number or Hardware identifier. Additionally, the user can read out the writable user registers.

Info Space	Info Byte
UserA	0
UserB	1
Hardware Identifier	3
Serial Number	4

Information: All of the information spaces can only be read as a complete 16-byte block.

Example: Read Information space UserA (User string = „EXDUL-384“)
An 8-byte block is transmitted and a 20-byte block is received with content from UserA or UserB

Byte	Transmit	Representing	Receive	Representing
0	0C	Command code 1st Byte	0C	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01	Length byte → 4Byte	04	Length byte → 16 Byte
4	00 (UserA) 01 (UserB)	Information byte	45	Data 1st character E_{asci}
5	00	reserved	58	Data 2nd character X_{asci}
6	00	reserved	44	Data 3rd character D_{asci}
7	01	Read function information space	55	Data 4th character U_{asci}
8			4C	Data 5th character L_{asci}
9			2D	Data 6th character $-_{asci}$
10			33	Data 7th character 3_{asci}
11			38	Data 8th character 8_{asci}
12			34	Data 9th character 4_{asci}
13			20	Data 10th character [blank]$_{asci}$
14			20	Data 11th character [blank]$_{asci}$
15			20	Data 12th character [blank]$_{asci}$
16			20	Data 13th character [blank]$_{asci}$
17			20	Data 14th character [blank]$_{asci}$
18			20	Data 15th character [blank]$_{asci}$
19			20	Data 16th character [blank]$_{asci}$

Example: Read information space hardware identifier
An 8-byte block is transmitted and a 20-byte block is received with hardware identifier

Byte	Transmit	Representing	Receive	Representing
0	0C	Command code 1st Byte	0C	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01	Length byte → 4Byte	04	Length byte → 16 Byte
4	04	Information byte	45	Data 1st character E$_{asci}$
5	00	reserved	58	Data 2nd character X$_{asci}$
6	00	reserved	44	Data 3rd character D$_{asci}$
7	01	Read function information space	55	Data 4th character U$_{asci}$
8			4C	Data 5th character L$_{asci}$
9			2D	Data 6th character -$_{asci}$
10			33	Data 7th character 3$_{asci}$
11			38	Data 8th character 8$_{asci}$
12			34	Data 9th character 4$_{asci}$
13			20	Data 10th character [blank]$_{asci}$
14			20	Data 11th character [blank]$_{asci}$
15			20	Data 12th character [blank]$_{asci}$
16			20	Data 13th character [blank]$_{asci}$
17			20	Data 14th character [blank]$_{asci}$
18			20	Data 15th character [blank]$_{asci}$
19			20	Data 16th character [blank]$_{asci}$

Example: Read information space serial number
An 8-byte block is transmitted and a 20-byte block is received with serial number

Byte	Transmit	Representing	Receive	Representing
0	0C	Command code 1st Byte	0C	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01	Length byte → 4Byte	03	Length byte → 16 Byte
4	04	Information byte	31	Data 1st character 1_{dez}
5	00	reserved	30	Data 2nd character 0_{dez}
6	00	reserved	34	Data 3rd character 4_{dez}
7	01	Read function information space	34	Data 4th character 4_{dez}
8			30	Data 5th character 0_{dez}
9			32	Data 6th character 2_{dez}
10			36	Data 7th character 6_{dez}
11				reserved
12				reserved
13				reserved
14				reserved
15				reserved
16				reserved
17				reserved
18				reserved
19				reserved

Writing to LCD register

The EXDUL module provides several writable LCD registers. UserLCD1 and UserLCD2 correspond to the two lines when using UserMode-LCD display. UserLCD1m and UserLCD2m are two 16-Byte spaces, which are stored directly in a non-volatile memory (FLASH) and are loaded into the registers UserLCD1m or UserLCD2m at module start. All of the registers are writable as a complete 16-byte block only.

LCD Command	LCD Command Byte
UserLCD1	0
UserLCD2	1
UserLCD1m	2
UserLCD2m	3

Example: enter the character string EXDUL-384 to register

Byte	Transmit	Receive	Meaning
0	0C	0C	Command code 1st Byte
1	00	00	Command code 2nd Byte
2	03	03	Command code 3rd Byte
3	05	00	Lenght prefix byte => 20 Byte
4	00 (UserLCD1) 01 (UserLCD2) 02 (UserLCD1m) 03 (UserLCD2m)		LCD command
5	00		reserved
6	00		reserved
7	00		Info space of write operation
8	45		Data 1st character E_{asci}
9	58		Data 2nd character X_{asci}
10	44		Data 3rd character D_{asci}
11	55		Data 4th character U_{asci}
12	4C		Data 5th character L_{asci}
13	2D		Data 6th character $-_{asci}$
14	33		Data 7th character 3_{asci}
15	38		Data 8th character 8_{asci}
16	34		Data 9th character 4_{asci}
17	20		Data 10th character [blank]$_{asci}$
18	20		Data 11th character [blank]$_{asci}$
19	20		Data 12th character [blank]$_{asci}$
20	20		Data 13th character [blank]$_{asci}$
21	20		Data 14th character [blank]$_{asci}$
22	20		Data 15th character [blank]$_{asci}$
23	20		Data 16th character [blank]$_{asci}$

Reading from LCD registers

The EXDUL module provides several writable and readable LCD registers. UserLCD1 and UserLCD2 correspond to the two lines when using UserMode LCD display. UserLCD1m and UserLCD2m are two 16-byte areas, which are stored directly in a non-volatile memory (FLASH) and are loaded into the registers UserLCD1m or UserLCD2m at module start. All of the registers are readable as whole 16-byte blocks only.

LCD Command	LCD Command Byte
UserLCD1 & UserLCD2	0
UserLCD1m & UserLCD2m	2

Example: read the character string EXDUL-384 from register

Byte	Transmit	Representing	Receive	Representing
0	0C	Command code 1st Byte	0C	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	03	Command code 3rd Byte	03	Command code 3rd Byte
3	01	Length byte → 20 Byte	08	Length byte → 20 Byte
4	00 (UserLCD1&2) 02 (UserLCD1m&2m)	LCD Command	45	Data Line1 1st character E$_{asci}$
5	00	reserved	58	Data Line1 2nd character X$_{asci}$
6	00	reserved	44	Data Line1 3rd character D$_{asci}$
7	01	Read function LCD registers	55	Data Line1 4th character U$_{asci}$
8			4C	Data Line1 5th character L$_{asci}$
9			2D	Data Line1 6th character -$_{asci}$
10			33	Data Line1 7th character 3$_{asci}$
11			38	Data Line1 8th character 8$_{asci}$
12			34	Data Line1 9th character 4$_{asci}$
13			20	Data Line1 10th character [blank]$_{asci}$
14			20	Data Line1 11th character [blank]$_{asci}$
15			20	Data Line1 12th character [blank]$_{asci}$
16			20	Data Line1 13th character [blank]$_{asci}$
17			20	Data Line1 14th character [blank]$_{asci}$
18			20	Data Line1 15th character [blank]$_{asci}$
19			20	Data Line1 16th character [blank]$_{asci}$
20			45	Data Line2 1st character E$_{asci}$
21			58	Data Line2 2nd character X$_{asci}$
22			44	Data Line2 3rd character D$_{asci}$
23			55	Data Line2 4th character U$_{asci}$
24			4C	Data Line2 5th character L$_{asci}$
25			2D	Data Line2 6th character -$_{asci}$
26			33	Data Line2 7th character 3$_{asci}$
27			38	Data Line2 8th character 8$_{asci}$
28			34	Data Line2 9th character 4$_{asci}$
29			20	Data Line2 10th character [blank]$_{asci}$
30			20	Data Line2 11th character [blank]$_{asci}$
31			20	Data Line2 12th character [blank]$_{asci}$
32			20	Data Line2 13th character [blank]$_{asci}$
33			20	Data Line2 14th character [blank]$_{asci}$
34			20	Data Line2 15th character [blank]$_{asci}$
35			20	Data Line2 16th character [blank]$_{asci}$

Writing the LCD Mode

The module's LCD display provides several display modes. These can be set by the following command. The LCD mode is stored in a non-volatile memory and is also employed after a restart of the module.

LCD Mode	LCD Mode Byte
I/O Mode	0
User Mode	1

Example: writing the LCD Mode

Byte	Transmit	Representing	Receive	Representing
0	0C	Command code 1st Byte	0C	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	03	Command code 3rd Byte	03	Command code 3rd Byte
3	02	Length byte → 8 Byte	00	Length byte → 0 Byte
4	04	LCD Command LCD mode		
5	00	reserved		
6	00	reserved		
7	00	Write function		
8	00 (IO Mode) 01 (User Mode)	LCD mode		
9	00	reserved		
10	00	reserved		
11	00	reserved		

Reading the LCD Mode

The module‘s LCD display provides several modes of display. The set LCD mode can be read out by following command.

LCD Mode	LCD Mode Byte
I/O Mode	0
User Mode	1

Example: reading the LCD Mode

Byte	Transmit	Representing	Receive	Representing
0	0C	Command code 1st Byte	0C	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	03	Command code 3rd Byte	03	Command code 3rd Byte
3	01	Length byte → 4 Byte	01	Length byte → 4 Byte
4	04	LCD command LCD mode	00 (IO-Mode) 01 (User-Mode)	LCD mode
5	00	reserved	00	reserved
6	00	reserved	00	reserved
7	01	read function	00	reserved

Writing the LCD contrast value

This command is used to adjust the display contrast. Values between 0 and 4095 are accepted. The display contrast decreases with increasing values. A comfortable display is achieved in the range 800 to 1800.

Example: writing display contrast value 800

Byte	Transmit	Representing	Receive	Representing
0	0C	Command code 1st Byte	0C	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	03	Command code 3rd Byte	03	Command code 3rd Byte
3	02	Length byte → 8 Byte	00	Length byte → 0 Byte
4	0B	LCD command LCD contrast		
5	00	reserved		
6	00	reserved		
7	00	write function		
8	50	Contrast value (Lowbyte - 00...FF)		
9	03	Contrast value (Highbyte - 00...0F)		
10	00	reserved		
11	00	reserved		

Reading the LCD contrast value

This command is used to read out the display contrast. The value can be between 0 and 4095. The display contrast decreases with increasing values. A comfortable display is achieved in the range 800 to 1800.

Example: reading display contrast value 800

Byte	Transmit	Representing	Receive	Representing
0	0C	Command code 1st Byte	0C	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	03	Command code 3rd Byte	03	Command code 3rd Byte
3	01	Length byte → 4 Byte	01	Length byte → 4 Byte
4	0B	LCD command LCD contrast	50	Contrast value (Lowbyte - 00...FF)
5	00	reserved	03	Contrast value (Highbyte - 00...0F)
6	00	reserved	00	reserved
7	01	read function	00	reserved

Reading optocoupler output

This command permits to read the current state of the optocoupler output.

Example: Reading the optocoupler output state
An 8-byte block is transmitted and a 8-byte block is received with the optocoupler output state

Byte	Transmit	Representing	Receive	Representing
0	08	Command code 1st Byte	08	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01 (→ 4Byte)	Length byte	01 (→ 4Byte)	Length byte
4	01	r/w Byte (1→ read)	0w 00 (LOW an DIN00) 01 (HIGH an DIN00)	State of the optocoupler output
5	00	reserved	00	reserved
6	00	reserved	00	reserved
7	00	reserved	00	reserved

Writing optocoupler output

This command permits the user to disable or enable the optocoupler output

Example: output of a state at the optocoupler output
An 8-byte block is transmitted and a 4-byte block is received as confirmation

Byte	Transmit	Receive	Representing
0	08	08	Command code 1st Byte
1	00	0	Command code 2nd Byte
2	00	00	Command code 3rd Byte
3	01 (→ 4Byte)	00	Length byte
4	00		r/w byte
5	0w 00 (disabled) 01 (enabled)		Optocoupler state
6	00		reserved
7	00		reserved

Reading optocoupler input

This command permits to read the current state of the optocoupler input.

Example: Read optocoupler input state
A 4-byte block is transmitted and an 8-byte block is received with optocoupler input state

Byte	Transmit	Representing	Receive	Representing
0	08	Command code 1st Byte	08	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	01	Command code 3rd Byte	00	Command code 3rd Byte
3	00	Length byte	01 (→ 4Byte)	Length byte
4			0w	State of optocoupler input
5			00	reserved
6			00	reserved
7			00	reserved

Counter0

This command permits access to the Counter0. Thus you can start, stop, reset and read the counter. Additionally, you can read and reset the overflow flag.

Code	Counter command code
00	Start Counter
01	Stop Counter
02	Reset Counter
03	Read counter value
04	reserved
05	Read Overflow Flag
06	Reset Overflow Flag

Counter Start / Stop / Reset

Byte	Transmit	Representing	Receive	Representing
0	09	Command code 1st Byte	09	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01	Length byte	01	Length byte
4	bb 00 01 02	Counter command code Start Counter0 Stop Counter0 Reset Counter0	bb	Counter command code
5	00	reserved	00	reserved
6	00	reserved	00	reserved
7	00	reserved	00	reserved

Read Counter

Byte	Transmit	Representing	Receive	Representing
0	09	Command code 1st Byte	09	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01	Length byte	02 (→ 8Byte)	Length byte
4	03	Counter command code	03	Counter command code
5	00	reserved	00	reserved
6	00	reserved	00	reserved
7	00	reserved	00	reserved
8			ww	Counter reading Byte0
9			ww	Counter reading Byte1
10			ww	Counter reading Byte2
11			ww	Counter reading Byte3

Counter reading = Counter reading Byte3 * 0x1000000 + Counter reading Byte2 * 0x10000 + Counter reading Byte1 * 0x100 + Counter reading Byte0

Read overflow flag

Byte	Transmit	Representing	Receive	Representing
0	09	Command code 1st Byte	09	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01	Length byte	02 (→ 8Byte)	Length byte
4	05	Counter command code Read overflow flag	05	Counter command code Read overflow flag
5	00	reserved	00	reserved
6	00	reserved	00	reserved
7	00	reserved	0f	Overflow flag

Reset overflow flag

Byte	Transmit	Representing	Receive	Representing
0	09	Command code 1st Byte	09	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01	Length byte	01 (→ 4Byte)	Length byte
4	06	Counter command code Reset overflow flag	06	Counter command code Reset overflow flag
5	00	reserved	00	reserved
6	00	reserved	00	reserved
7	00	reserved	00	reserved

AD Single measurement

The command AD single measurement performs a voltage measurement on a desired analog input channel and returns the value calibrated as an integer in µV to the PC. The command has to contain the desired channel as well as the measuring range.

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Example of measuring a voltage at an input signal

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01 (→ 4Byte)		01 (→ 4Byte)	Length byte
4	cc	Channel byte	ww	Measured value Byte0
5	bb	Range byte	ww	Measured value Byte1
6	00		ww	Measured value Byte2
7	00		ww	Measured value Byte3

Voltage = (integer) (Byte3 * 0x1000000 + Byte2 * 0x10000 + Byte1 * 0x100 + Byte0) [µV]

AD Single measurement with averaging

The command AD single measurement with averaging performs 32 voltage measurements at a rate of 100kS/s on a desired analog input channel, averages it and returns the value calibrated as an integer in µV to the computer. The desired channel as well as the measuring range has to be transmitted to the command.

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Example of measuring the voltage on an input signal

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	01	Command code 3rd Byte	01	Command code 3rd Byte
3	01 (→ 4Byte)	Length byte	01 (→ 4Byte)	Length byte
4	cc	Channel byte	ww	Measured value Byte0
5	bb	Range byte	ww	Measured value Byte1
6	00	reserved	ww	Measured value Byte2
7	00	reserved	ww	Measured value Byte3

Voltage = (integer) (Byte3 * 0x1000000 + Byte2 * 0x10000 + Byte1 * 0x100 + Byte0) [µV]

AD block measurement with averaging

This command performs sampling of up to 8 channels in quick succession. Each channel to be measured is sampled 32 times, each averaged (see chapter 5.2) and the value returned as an integer in µV to the PC.

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Command structure n = 1 8

Byte	Transmit	Representing
0	0A	Command code 1st Byte
1	00	Command code 2nd Byte
2	02	Command code 3rd Byte
3	(n*4)	Length byte (n = number of channels)
4	00	reserved
5	00	reserved
6	c_0c_0	Channel byte
7	b_0b_0	Range byte
	:	
	:	
3 + n*4	$c_{n-1}c_{n-1}$	Channel byte
4 + n*4	$b_{n-1}b_{n-1}$	Range byte

Byte	Recieve	Representing
0	0A	Command code 1st Byte
1	00	Command code 2nd Byte
2	02	Command code 3rd Byte
3	(n*4)	Length byte (n = number of channels)
4	w_1w_1	Measured value$_1$ Byte0$_1$
5	w_1w_1	Measured value$_1$ Byte1$_1$
6	w_1w_1	Measured value$_1$ Byte2$_1$
7	w_1w_1	Measured value$_1$ Byte3$_1$
	:	
	:	
3 + n*4	w_nw_n	Measured value$_n$ Byte0$_n$
4 + n*4 +1	w_nw_n	Measured value$_n$ Byte1$_n$
4 + n*4 +2	w_nw_n	Measured value$_n$ Byte2$_n$
4 + n*4 +3	w_nw_n	Measured value$_n$ Byte3$_n$

Example:
In the following example, AIN01, AIN02 and AIN04 are to be sampled. The measuring range may be +/- 10.2V for all values.

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	02	Command code 3rd Byte	02	Command code 3rd Byte
3	03 (→ 12Byte)	Length byte	03 (→ 12Byte)	Length byte
4	00	reserved	w_1w_1	Measured value AIN01 Byte0_1
5	00	reserved	w_1w_1	Measured value AIN01 Byte1_1
6	01	Channel byte AIN01	w_1w_1	Measured value AIN01 Byte2_1
7	01	Range byte +/- 10.2V	w_1w_1	Measured value AIN01 Byte3_1
8	00	reserved	w_2w_2	Measured value AIN02 Byte0_2
9	00	reserved	w_2w_2	Measured value AIN02 Byte1_2
10	02	Channel byte AIN02	w_2w_2	Measured value AIN02 Byte2_2
11	01	Range byte +/- 10.2V	w_2w_2	Measured value AIN02 Byte3_2
12	00	reserved	w_3w_3	Measured value AIN04 Byte0_3
13	00	reserved	w_3w_3	Measured value AIN04 Byte1_3
14	04	Channel byte AIN04	w_3w_3	Measured value AIN04 Byte2_3
15	01	Range byte +/- 10.2V	w_3w_3	Measured value AIN04 Byte3_3

Measured value AIN01
= (integer) (Byte3_1 * 0x1000000 + Byte2_1 * 0x10000 + Byte1_1 * 0x100 + Byte0_1) [µV]
Measured value AIN02
= (integer) (Byte3_2 * 0x1000000 + Byte2_2 * 0x10000 + Byte1_2 * 0x100 + Byte0_2) [µV]
Measured value AIN04
= (integer) (Byte3_3 * 0x1000000 + Byte2_3 * 0x10000 + Byte1_3 * 0x100 + Byte0_3) [µV]

Reset ADC FIFO

The following command performs a reset of the ADC FIFO.
This should be done after an overflow.

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	06	Command code 3rd Byte	06	Command code 3rd Byte
3	00	Length byte	00	Length byte → 0 Bytes

Read ADC FIFO overflow flag

The following command reads the overflow flag of the ADC-FIFO. Along with the read, the overflow flag is reset.

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	07	Command code 3rd Byte	07	Command code 3rd Byte
3	00	Length byte	01	Length byte → 4 Bytes
4			0w	Overflow flag 00 FIFO no overflow 01 FIFO overflow
5			00	reserved
6			00	reserved
7			00	reserved

Read out ADC FIFO

Some commands do not return the measurement results directly along with the response command, but store the measured values in a FIFO. As a command example, multiple measurement or continuous measurement can be mentioned. The FIFO can be read out with the ADC FIFO readout command. The values hold in the FIFO are appended directly to the response of the command (up to 255 readings). If the FIFO does not contain any data, only a 4-byte response is returned to the computer.

Command structure

4 bytes are to be transmitted, 4 + n*4 bytes are to be received depending upon the amount of data n in the FIFO.

n = 1 8

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	08	Command code 3rd Byte	08	Command code 3rd Byte
3	00	Length byte	nn	Length byte → n*4 Bytes
4			ww_1	Measured value$_1$ Byte0$_1$
5			ww_1	Measured value$_1$ Byte1$_1$
6			ww_1	Measured value$_1$ Byte2$_1$
7			ww_1	Measured value$_1$ Byte3$_1$
			:	
			:	
n*4			ww_n	Measured value$_n$ Byte0$_n$
n*4 + 1			ww_n	Measured value$_n$ Byte1$_n$
n*4 + 2			ww_n	Measured value$_n$ Byte2$_n$
n*4 + 3			ww_n	Measured value$_n$ Byte3$_n$

Example 1:
FIFO is empty:

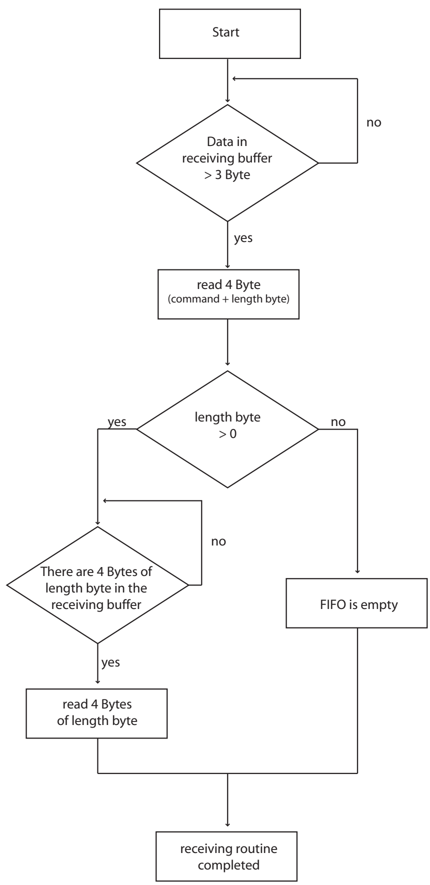
Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	08	Command code 3rd Byte	08	Command code 3rd Byte
3	00	Length byte	00	Length byte

Example 2:
The FIFO holds two measured values

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	08	Command code 3rd Byte	08	Command code 3rd Byte
3	00	Length byte	2	Length byte → 8 bytes
4			ww_1	Measured value$_1$ Byte0_1
5			ww_1	Measured value$_1$ Byte1_1
6			ww_1	Measured value$_1$ Byte2_1
7			ww_1	Measured value$_1$ Byte3_1
8			ww_2	Measured value$_2$ Byte0_2
9			ww_2	Measured value$_2$ Byte1_2
10			ww_2	Measured value$_2$ Byte2_2
11			ww_2	Measured value$_2$ Byte3_2

Programming:

- Transmitting: to read data from the FIFO the 4 byte holding command has to be sent to the module
- Receiving the data: since the array length of the data to be received may vary, the receiving of the entire data block has to be partitioned.



AD Multiple Measurement

The A/D multiple measurement allows the user to sample one or more channels several times (up to 65,535 times) in an adjustable clock (1 - 100000kHz). The measured values are stored by the module in the internal FIFO and can be retrieved there during and after the sampling process. The values are buffered in the FIFO until they either have been fetched or a new sampling command has been called.

Attention: it must be ensured that the FIFO can be emptied fast enough, since the FIFO is limited to 10,000 readings. Furthermore, no EXDUL information register (e.g. UserA, UserB) may be written during the processing.

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Command structure

n = 1 8

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	09	Command code 3rd Byte	09	Command code 3rd Byte
3	n + 2	Length byte	00	Length byte
4	ff	Sampling rate Byte0		
5	ff	Sampling rate Byte1		
6	ff	Sampling rate Byte2		
7	00	reserved		
8	aa	Number of readings Byte0		
9	aa	Number of readings Byte1		
10	00	reserved		
11	00	reserved		
12	00	reserved		
13	00	reserved		
14	cc_n	Channel byte$_1$		
15	bb_n	Range byte$_1$		
	:			
	:			
n*4 + 8	00	reserved		
n*4 + 9	00	reserved		
n*4 + 10	cc_n	Channel byte$_1$		
n*4 + 11	bb_n	Range byte$_1$		

Sampling rate = Byte2 * 65536 + Byte1 * 256 + Byte0
Number of readings = Byte1 * 256 + Byte0

Start AD continuous sampling

The A/D continuous measurement allows the user to sample one or more channels at regular intervals (1s - 10µs). The measured values are stored by the module into the internal FIFO and can be retrieved there during and after the sampling process. The values are buffered in the FIFO until they either have been fetched or a new sampling command has been called. To stop the continuous measurement the command „stop continuous A/D sampling“ must be sent to the module.

Attention: it must be ensured that the FIFO can be emptied quickly enough since the FIFO is limited to 10,000 readings. Furthermore, no EXDUL information register (e.g. UserA, UserB) may be written during the processing.

Channel:

Channel	Channel byte
Single Ended	
AIN00	0
AIN01	1
AIN02	2
AIN03	3
AIN04	4
AIN05	5
AIN06	6
AIN07	7
Differential measuring	
AIN00+ / AIN01-	8
AIN00- / AIN01+	9
AIN02+ / AIN03-	10
AIN02- / AIN03+	11
AIN04+ / AIN05-	12
AIN04- / AIN05+	13
AIN06+ / AIN07-	14
AIN06- / AIN07+	15

Measuring range:

Range byte	Voltage
0	+/- 20.4V (Differential measuring only max +/- 10.2V → GND)
1	+/-10.2V
2	+/- 5.1V
3	+/-2,55V
4	+/-1.27V
5	+/- 0.63V

Command structure

n = 1 8

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	0A	Command code 3rd Byte	0A	Command code 3rd Byte
3	n + 1	Length byte	00	Length byte
4	ff	Sampling rate Byte0		
5	ff	Sampling rate Byte1		
6	ff	Sampling rate Byte2		
7	00	reserved		
8	aa	reserved		
9	aa	reserved		
10	cc_1	Channel $byte_1$		
11	bb_1	Range $byte_1$		
	:			
	:			
n*4 + 4	00	reserved		
n*4 + 5	00	reserved		
n*4 + 6	cc_n	Channel $byte_1$		
n*4 + 7	bb_n	Range $byte_1$		

Sampling rate = Byte2 * 65536 + Byte1 * 256 + Byte0

Stop AD continuous sampling

This command stops the AD continuous measurement.

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	00	Command code 2nd Byte	00	Command code 2nd Byte
2	0B	Command code 3rd Byte	0B	Command code 3rd Byte
3	00	Length byte	00	Length byte

Configuration of the D/A output voltage range

This command allows the user to configurate the output voltage ranges of the single DAC channels. The new voltage range of a channel is adopted as soon as a new voltage is output on the respective channel.
A further 4-byte block is added to the command holding a channel byte (0 up to 7) and a range byte (see table)

Output voltage range	
Range byte	bipolar
0	+/-10.2V
1	+/-5.1V
2	+/-2.55V

Command structure

Byte	Transmit	Representing	Receive	Representing
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	80	Command code 2nd Byte	80	Command code 2nd Byte
2	00	Command code 3rd Byte	00	Command code 3rd Byte
3	01	Length byte	00	Length byte
4	cc	Channel byte (0 ... 7)		
5	bb	Range byte		
6	00	reserved		
7	00	reserved		

DA Voltage output

This command can be used to output a desired voltage on one of the available channels. The command on one hand holds a 4-byte block for the channel to be changed and on the other hand the voltage in µV.

Command structure

Byte	Transmit	Meaning	Receive	Meaning
0	0A	Command code 1st Byte	0A	Command code 1st Byte
1	80	Command code 2nd Byte	80	Command code 2nd Byte
2	01	Command code 3rd Byte	01	Command code 3rd Byte
3	02	Length byte	00	Length byte
4	cc	Channel byte (0 ... 7)		
5	00	reserved		
6	00	reserved		
7	00	reserved		
8	ww	voltage Byte0		
9	ww	voltage Byte1		
10	ww	voltage Byte2		
11	ww	voltage Byte3		

Voltage = (integer) (Byte3 * 0x1000000 + Byte2 * 0x10000 + Byte1 * 0x100 + Byte0) [µV]

12. Programming under Linux®

12.1 Introduction

After your operating system successfully recognized the EXDUL-384E/ EXDUL-384S, the module is listed as a ttyACM* device in the folder /dev. This is a CDC device (Communications Device Class), that is addressed via a virtual COM port.
This virtual COM port operates like a normal COM interface and can be accessed by a default driver, there is no need to install any additional drivers.

12.2 Programming with serial COM Port Libraries

When the module has been recognized you can communicate via default libraries of serial interfaces. We refer to chapter 11.4. for more detailed information.

13. Specifications

A/D Inputs
8 inputs single-ended (se)
or 4 inputs differential (diff)
or combined se/diff software-sectable
Resolution: 16 Bit
Input voltage range bipolar:
+/-0.63V, +/-1.27V, +/-2.55V, +/-5.1V, +/-10.2V,
+/-20.4V (differential inputs only)
FIFO: 10000 readings
Input resistor: > 500 MΩ
Over voltage protection: +/- 50V
Measuring cycle: max. 10 µs
Sampling rate: max 100 kS

D/A Outputs
8 outputs
Resolution: 16 bit
Output voltage range
bipolar: +/-2.55 Volt, +/-5.1 Volt, +/-10.2 Volt
Output current: max +/-5 mA

Optocoupler input
1 channel galvanically isolated, progammable as a counter input
Over voltage protection diodes
Input voltage range
high = 10..30 Volt
low = 0..3 Volt
Input frequency: max. 10 kHz

Optocoupler output
1 channel, galvanically isolated
High capacity optocoupler
Reverse polarity protection
Output current: max. 150 mA
Switching voltage: max. 50 V

Counter
Channel: 1 programmable counter 32 bit
Counting frequency: max. 5 kHz

LCD Display
Matrix display with 2 lines and 16 columns displaying 16 characters each line
Programmable to display application-specific data or as I/O status display

Operating voltage
via USB port (PC able to supply 500mA) or
+10 V...+30 V (external power supply)

USB Interface
Compatible with USB 2.0
USB Connection Plug and Play (hot-swappable, also connectable during operation)

Connection Terminals
1 * 24pin screw terminal block
1 * USB socket Type B

USB connection lines
1 * USB plug Type A
1 * USB plug Type B

Dimensions
105 mm x 89 mm x 59 mm (l x b x h)

Casing
Insulating plastic housing with integrated snap-in technology for DIN EN top hat rail mounting.
Suitable for control and engineering technology mounted to control and distribution boxes, surface mounting or mobile use on a desk.

14. Circuitry Examples

14.1 Wiring of the Optocoupler Input

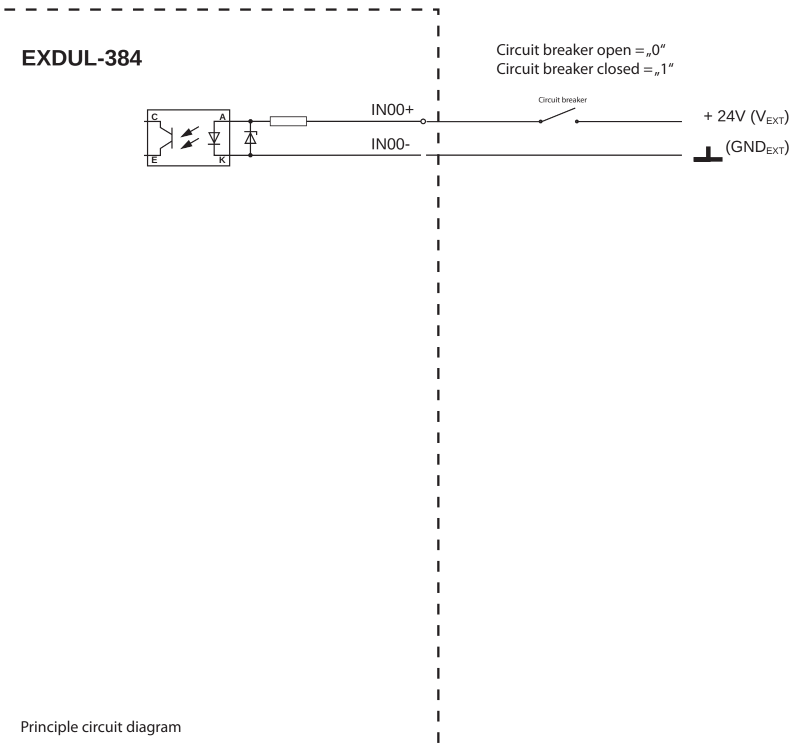


Figure 14.1 Optocoupler input wiring

14.2 Wiring of the Optocoupler Output

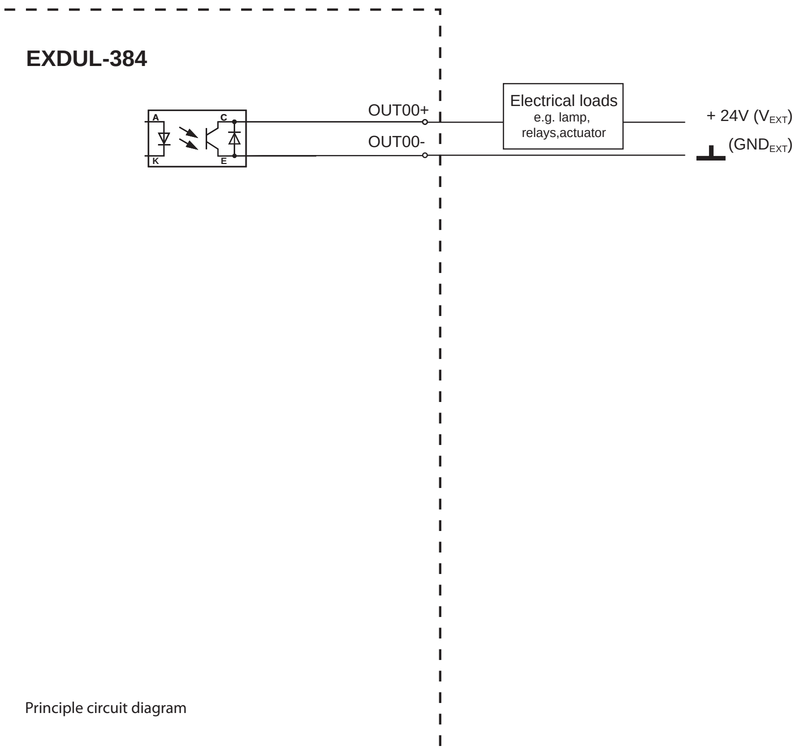


Figure 14.2 Optocoupler output wiring

14.3 Wiring of the D/A Outputs

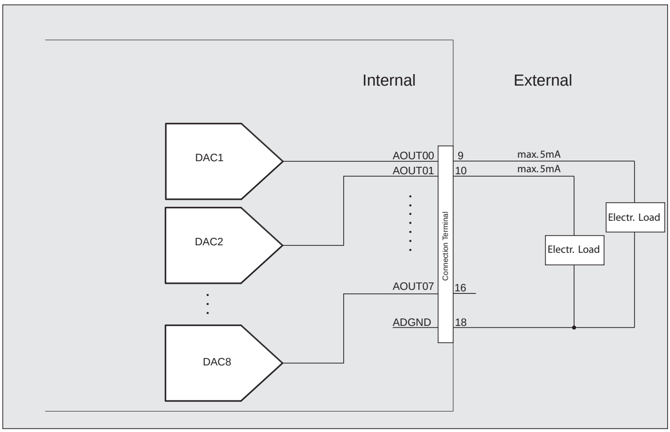


Figure 14.3 Wiring of the D/A outputs

14.4 Wiring of the A/D Inputs single ended

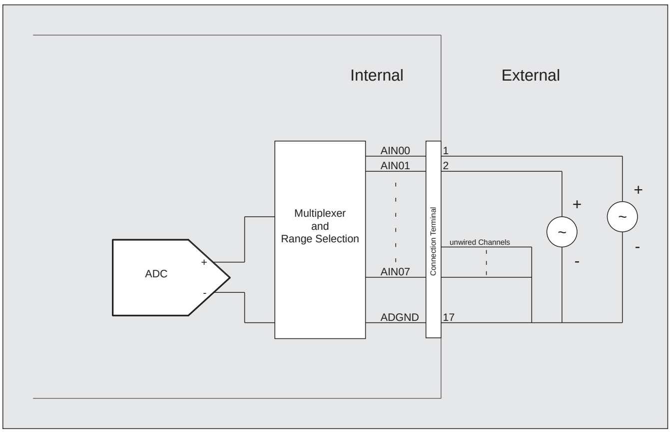


Figure 14.4 Wiring of the A/D inputs (single ended)

14.5 Wiring of the A/D Inputs differential

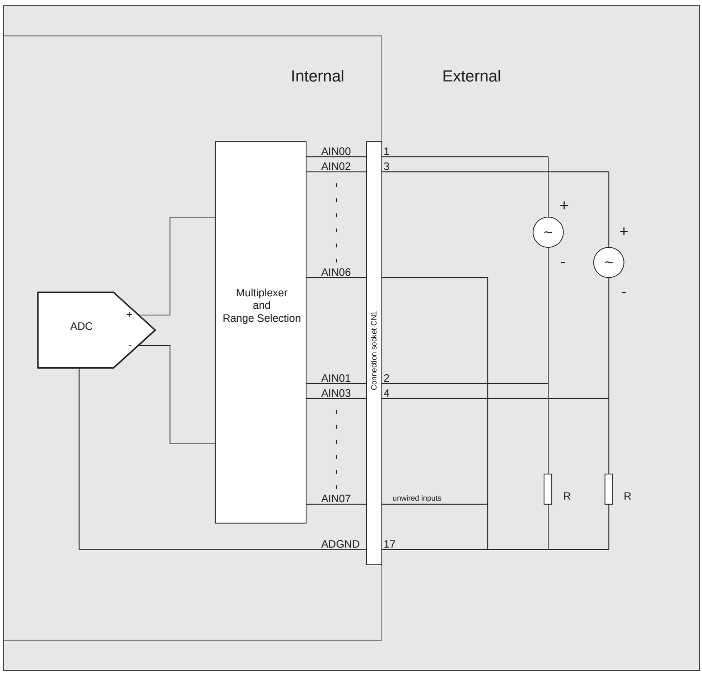


Figure 14.5 Wiring of the A/D inputs (differential)

15. ASCII Table

Hex	Dec	Binary	Character
00	0	00000000	
01	1	00000001	
02	2	00000010	
03	3	00000011	
04	4	00000100	
05	5	00000101	
06	6	00000110	
07	7	00000111	
08	8	00001000	
09	9	00001001	
0A	10	00001010	
0B	11	00001011	
0C	12	00001100	
0D	13	00001101	
0E	14	00001110	
0F	15	00001111	
10	16	00010000	
11	17	00010001	
12	18	00010010	
13	19	00010011	
14	20	00010100	
15	21	00010101	
16	22	00010110	
17	23	00010111	
18	24	00011000	
19	25	00011001	
1A	26	00011010	
1B	27	00011011	
1C	28	00011100	
1D	29	00011101	
1E	30	00011110	
1F	31	00011111	
20	32	00100000	[space]
21	33	00100001	!
22	34	00100010	”
23	35	00100011	#
24	36	00100100	$
25	37	00100101	%
26	38	00100110	&
27	39	00100111	’

Hex	Dec	Binary	Character
28	40	00101000	(
29	41	00101001	)
2A	42	00101010	*
2B	43	00101011	+
2C	44	00101100	,
2D	45	00101101	-
2E	46	00101110	.
2F	47	00101111	/
30	48	00110000	0
31	49	00110001	1
32	50	00110010	2
33	51	00110011	3
34	52	00110100	4
35	53	00110101	5
36	54	00110110	6
37	55	00110111	7
38	56	00111000	8
39	57	00111001	9
3A	58	00111010	:
3B	59	00111011	;
3C	60	00111100	<
3D	61	00111101	=
3E	62	00111110	>
3F	63	00111111	?
40	64	01000000	@
41	65	01000001	A
42	66	01000010	B
43	67	01000011	C
44	68	01000100	D
45	69	01000101	E
46	70	01000110	F
47	71	01000111	G
48	72	01001000	H
49	73	01001001	I
4A	74	01001010	J
4B	75	01001011	K
4C	76	01001100	L
4D	77	01001101	M
4E	78	01001110	N
4F	79	01001111	O

Hex	Dec	Binary	Character
50	80	01010000	P
51	81	01010001	Q
52	82	01010010	R
53	83	01010011	S
54	84	01010100	T
55	85	01010101	U
56	86	01010110	V
57	87	01010111	W
58	88	01011000	X
59	89	01011001	Y
5A	90	01011010	Z
5B	91	01011011	[
5C	92	01011100	
5D	93	01011101	]
5E	94	01011110	^
5F	95	01011111	_
60	96	01100000	`
61	97	01100001	a
62	98	01100010	b
63	99	01100011	c
64	100	01100100	d
65	101	01100101	e
66	102	01100110	f
67	103	01100111	g
68	104	01101000	h
69	105	01101001	i
6A	106	01101010	j
6B	107	01101011	k
6C	108	01101100	l
6D	109	01101101	m
6E	110	01101110	n
6F	111	01101111	o
70	112	01110000	p
71	113	01110001	q
72	114	01110010	r
73	115	01110011	s
74	116	01110100	t
75	117	01110101	u
76	118	01110110	v
77	119	01110111	w
78	120	01111000	x
79	121	01111001	y
7A	122	01111010	z
7B	123	01111011	{

Hex	Dec	Binary	Character
7C	124	01111100	|
7D	125	01111101	}
7E	126	01111110	
7F	127	01111111	
80	128	10000000	
81	129	10000001	
82	130	10000010	
83	131	10000011	
84	132	10000100	
85	133	10000101	
86	134	10000110	
87	135	10000111	
88	136	10001000	
89	137	10001001	
8A	138	10001010	
8B	139	10001011	
8C	140	10001100	
8D	141	10001101	
8E	142	10001110	
8F	143	10001111	
90	144	10010000	
91	145	10010001	
92	146	10010010	
93	147	10010011	
94	148	10010100	
95	149	10010101	
96	150	10010110	
97	151	10010111	
98	152	10011000	
99	153	10011001	
9A	154	10011010	
9B	155	10011011	
9C	156	10011100	
9D	157	10011101	
9E	158	10011110	
9F	159	10011111	
A0	160	10100000	
A1	161	10100001	
A2	162	10100010	
A3	163	10100011	
A4	164	10100100	
A5	165	10100101	
A6	166	10100110	
A7	167	10100111	

Hex	Dec	Binary	Character
A8	168	10101000	
A9	169	10101001	
AA	170	10101010	
AB	171	10101011	
AC	172	10101100	
AD	173	10101101	
AE	174	10101110	
AF	175	10101111	
B0	176	10110000	
B1	177	10110001	
B2	178	10110010	
B3	179	10110011	
B4	180	10110100	
B5	181	10110101	
B6	182	10110110	
B7	183	10110111	
B8	184	10111000	
B9	185	10111001	
BA	186	10111010	
BB	187	10111011	
BC	188	10111100	
BD	189	10111101	
BE	190	10111110	
BF	191	10111111	
C0	192	11000000	
C1	193	11000001	
C2	194	11000010	
C3	195	11000011	
C4	196	11000100	
C5	197	11000101	
C6	198	11000110	
C7	199	11000111	
C8	200	11001000	
C9	201	11001001	
CA	202	11001010	
CB	203	11001011	
CC	204	11001100	
CD	205	11001101	
CE	206	11001110	
CF	207	11001111	
D0	208	11010000	
D1	209	11010001	
D2	210	11010010	
D3	211	11010011	

Hex	Dec	Binary	Character
D4	212	11010100	
D5	213	11010101	
D6	214	11010110	
D7	215	11010111	
D8	216	11011000	
D9	217	11011001	
DA	218	11011010	
DB	219	11011011	
DC	220	11011100	
DD	221	11011101	
DE	222	11011110	
DF	223	11011111	
E0	224	11100000	
E1	225	11100001	
E2	226	11100010	
E3	227	11100011	
E4	228	11100100	
E5	229	11100101	
E6	230	11100110	
E7	231	11100111	
E8	232	11101000	
E9	233	11101001	
EA	234	11101010	
EB	235	11101011	
EC	236	11101100	
ED	237	11101101	
EE	238	11101110	
EF	239	11101111	
F0	240	11110000	
F1	241	11110001	
F2	242	11110010	
F3	243	11110011	
F4	244	11110100	
F5	245	11110101	
F6	246	11110110	
F7	247	11110111	
F8	248	11111000	
F9	249	11111001	
FA	250	11111010	
FB	251	11111011	
FC	252	11111100	
FD	253	11111101	
FE	254	11111110	
FF	255	11111111	

16. Product Liability Act

Information for Product Liability

The Product Liability Act (Act on Liability for Defective Products - ProdHaftG) in Germany regulates the manufacturer‘s liability for damages caused by defective products.

The obligation to pay compensation can already be given, if the product’s presentation could cause a misconception of safety to a non-commercial end-user and also if the end-user is expected not to observe the necessary safety instructions when handling this product.

It must therefore always be verifiable, that the end-user has been made familiar with the safety rules.

In the interest of safety, please always point out the following safety instructions to your non-commercial customers:

Safety instructions

The applicable VDE-instructions must be observed, when handling products that come into contact with electrical voltage.

Particular attention must be drawn to the following instructions:
VDE100; VDE0550/0551; VDE0700; VDE0711; VDE0860.
You can obtain the instructions from:
vde-Verlag GmbH
Bismarckstr. 33
10625 Berlin

* pull the mains plug before you open the unit or make sure, there is no current to/in the unit.

* You only may put into operation any components, boards or devices, if they have been installed inside a secure touch-protected casing before. During installation there must be no current to the equipment.

* Make sure that the device is disconnected from the power supply before using any tools on any components, boards or devices. Any electric charges saved in components in the device are to be discharged prior.

* Live cables or wires, which are connected to the unit, the components or the boards, must be inspected for insulation faults or breakages. In case of any defect in a line the device must be taken out of operation immediately until the defective line has been replaced.

* When using components or boards you must strictly adhere to the characteristic data for electrical parameters specified in the corresponding description.

* As a non-commercial end-user, if it is not clear whether the electrical parameters given in the description provided are applicable for a component, you must consult an expert.

Apart from that, compliance with consruction regulations and safety instructions of all kinds (VDE, TÜV, professional associations, industrial injuries corporation, etc.) is subject to the user/customer.

wasco®

17. CE Declaration of Conformity

This is to certify, that the products

EXDUL-384E   EDP Number A-381940
EXDUL-384S   EDP Number A-381920

comply with the requirements of the relevant EC directives. This declaration will lose its validity, if the instructions given in this manual for the intended use of the products are not fully complied with.

EN 5502 Class B
IEC 801-2
IEC 801-3
IEC 801-4
EN 50082-1
EN 60555-2
EN 60555-3

The following manufacturer is responsible for this declaration:

Messcomp Datentechnik GmbH
Neudecker Str. 11
83512 Wasserburg

issued by

Dipl.Ing.(FH) Hans Schnellhammer

Wasserburg, 31.01.2018 ______________________

Reference system for intended use

The multi functional modules EXDUL-384E and EXDUL-384S are not stand-alone devices. The CE-conformity only can be assessed when using additional computer components simultaneously. Thus the CE conformity only can be confirmed when using the following reference system for the intended use of the multi functional modules:

Control Cabinet:	Vero IMRAK 3400	804-530061C 802-563424J 802-561589J
19“ Casing:	Vero PC-Casing	145-010108L
19“ Casing:	Additional Electronic	519-112111C
Motherboard:	GA-586HX	PIV 1.55
Floppy-Controller:	on Motherboard	
Floppy:	TEAC	FD-235HF
Grafic Card:	Advantech	PCA-6443
Interface:	EXDUL-384E EXDUL-384S	A-381940 A-381920