Communication protocol for DCX Data Loggers from KELLER V4.0 09.11.2020

1 Introduction

This document describes the communications protocol for the Data Logger DCX from KELLER Druckmesstechnik. In addition to these transmitters, other devices such as digital pressure transmitters or manometers are also offered. These products are distinguished by the designation CLASS. Within this device class, the individual device groups are differentiated by the designation GROUP.
All KELLER devices which have a record functionality work the same way as DCX-Loggers. Therefor this description is valid for devices like Leo-Record, ARC1, ADT1, …

2 Bit transfer layer (physical layer)

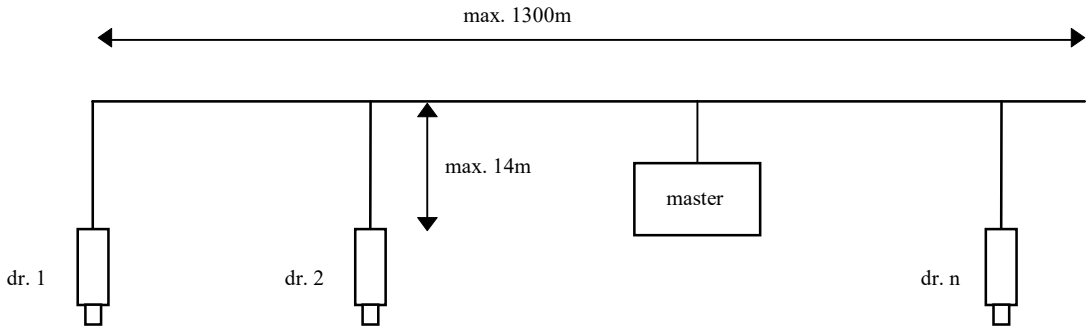
2.1 Introduction

The physical connection is provided by the RS485 serial interface. This guarantees good interference immunity and enables a flexible bus structure, i.e. several devices can be administrated as slaves by a single master.

In order to minimise the scope of cabling, the RS485 is used in half-duplex mode. This means that 2 wires are required for communications and 2 wires for power infeed.

2.2 Characteristic

In order to operate several devices at one serial interface, they are simply all connected in parallel (RS485A, RS485B, GND and +Vcc). Before incorporating the devices into the bus, each device must be programmed with a different address.
It is possible to configure a network up to a length of 1300 metres with a maximum of 128 devices. Each riser cable may be up to 14 m in length. The employed cable should correspond to specification EIA RS485. This means that the wires must be installed in twisted pairs, for example.



The following RS485 drivers are used in the devices:
Devices of *Class.Group* = 5.5: MAX3471

These drivers are slew rate-limited and do not require any bias resistors. Most applications do not require terminating resistors, either.
Voltage protection is installed on both lines (RS485A and RS485B) in the devices. The common mode voltage relative to GND is –7V ... + 12 V. This voltage must not be exceeded in any circumstances.

Further information on RS485: http://www.maxim-ic.com/MaximProducts/Interface/rs-485.htm
Information on wiring: http://www.maxim-ic.com/appnotes.cfm/appnote_number/763

3 Data-link layer

This section describes how data interchange is affected on this bus. The data and their check and control structures are grouped together to form messages. These constitute the smallest communication unit, i.e. only messages can be exchanged between the devices. As a half-duplex protocol is in use here, only one device can use the bus as a transmitter at any one time. All other devices are then in receiver mode. The master takes the form of a PC or microcontroller, for example, and the devices are the slaves. Each message exchange takes place under the control of the master. The message contains the address for the receiving slave.

This results in the following 2 options for data interchange:

a) Broadcasting	This mode of communication enables the master to transmit a message to all slaves simultaneously. The master does not receive a reply, however, and is thus unable to check whether the message has been correctly received by every slave.
b) Data interchange	This mode of communication enables the master to communicate with a single slave. This normally involves the transmission of two messages: the master transmits a request and the slave responds to this request. Only the master is permitted to request a response. The request is received by every slave, but only the selected slave responds. The response must be received within a stipulated time, otherwise the master will assess the attempt as failed and must transmit the request again.

3.1 Transmission format for the serial interface

The data are transmitted serially via the bus. The following format applies:

- 1 start bit
- 8 data bits (the least significant bit first)
- 1 stop bit
- no parity bit

This results in 10 bits per transmission byte.

3.2 Format of a message

3.2.1 Format of the message sent by the master

Note on the presentation of messages: Each box presents 1 data byte consisting of 8 bits, unless otherwise stated.

Each message sent by the master possesses the following format:

DevAddr	0	Function code	n byte parameters (optional)	CRC16_H	CRC16_L

- `DevAddr`: Address of the device.
  Address 0 is reserved for broadcasting.
  Addresses 1...249 can be used for bus mode.
  Address 250 is transparent and reserved for non-bus mode. Every device can be contacted with this address.
  Address 251 is used for modem function. Dataloggers addressed with this address will allow time gaps between the transferred bytes and enable a communication through a modem.
  Addresses 252...255 are reserved for subsequent developments.
- `Function code`: Function number
  A function is selected and executed by the device via the function number. The function number is encoded in 7 bits. Bit 7 is always 0. The functions are described further below.
- `Parameters`:
  The parameters required by the function (n = 0 .. 6, according to function)
- `CRC16`: 16-bit checksum
  These two check bytes serve to verify the integrity of the received data. If an error is established, the entire message will be discarded. The principle employed for CRC16 calculation is described in the appendix. The CRC16 standard is applied here.

Note: The length of a message from the master is at least **4** bytes.

3.2.2 Format of the message sent by the slave

A message transmitted by the slave possesses the following format:

DevAddr	X	Function code	n byte data (optional)	CRC16_H	CRC16_L

- `DevAddr`: Address of the device.
  This address corresponds to the address of the responding device.
- `Function code`:
  The function number is identical to the function number sent by the master. If the most significant bit is X = 0, this indicates that the function has been executed correctly. If bit X = 1, an exception error has occurred.
- `Data`:
  Any data requested via the function follow here.
- `CRC16`:
  See above.

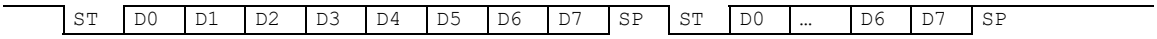
Note: A message from the slave has a minimum length of **5** bytes, and a maximum length of **10** bytes.

3.3 Principle of message interchange

3.3.1 General rules

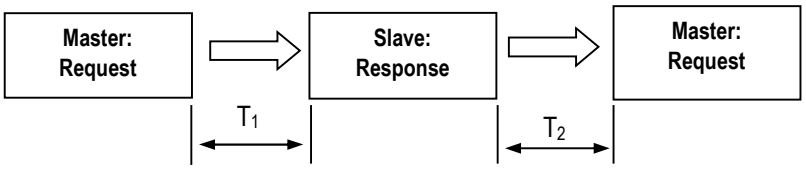
- An address may only be allocated to **one** device connected to the bus. If two devices on the bus have the same address, both will respond, leading to a conflict.
- Every data interchange is initiated by the master. This means that a device may only transmit data if requested to do so by the master.
- A message consists of several bytes. These bytes are transmitted **without any interruption**.
- The addressed device must respond within time T_1, otherwise the message will be invalid.
- After the device has respond, the master has to wait at least 1ms before he sends the next message.

Bit frame:



Whereby: ST: start bit, SP: stop bit, D0 .. D7: 8 data bits

Message frame:



Response times:

- T_1: Time between receipt of inquiry and beginning of response.
  Min. 1ms to max. 500ms for all functions and devices.
- T_2: Time to ready-to-receive state for the slave: 500 µs. Therefore it is recommended to wait at least 1ms until sending the next request.

3.3.2 Treatment of errors

2 types of errors may occur during the interchange of messages between master and slave: transmission errors and exception errors.

3.3.2.1 Transmission errors

These errors are primarily accountable to line faults. The message format is incorrect. The following problems are possible :

- A received message is too short.
- A message is longer than the internal transmission buffer permits.
- The word length is not correct and can therefore not be interpreted correctly.
- The CRC16 checksum is incorrect.

In response to a transmission error, all received data are ignored. The slave remains in receive mode while the master is required to initiate a new data interchange.

3.3.2.2 Exception errors

The message has been received correctly (no transmission error has occurred), but the transmitted function number and/or the parameters are invalid. The slave responds with an exception error, unless the message has been received in broadcasting mode.
The message transmitted as a response by the slave has the following format:

DevAddr	1	Function code	Exception code	CRC16_H	CRC16_L

4 types of exception errors are defined :

- non-implemented function 1
- incorrect parameters 2
- erroneous data 3
- initialisation 32

Exception error 32 occurs when the device is started up anew and initialisation has not been carried out. This happens every time the device is connected anew after a break in the power supply.

3.3.3 Activate the interface to the DCX

For reason of battery saving the interface of the DCX is disabled during the normal function.
It gets enabled as soon as there is a data request over the interface. This data packed however gets lost and the device will not answer. Therefore the data packed has to be sent once again.

The interface will be disabled again after ten seconds without communication on the interface.
On active interface all available channels are measured continuously (every second). This way the requested channel with function 73 returns always the actual value without delay.

Example:
Interface of DCX is disabled.

- PC sends initialization command (function 48):
  → DCX doesn't answer (timeout error). Interface gets enabled.
- PC sends again the initialization command (function 48):
  → DCX answers.

4 Description of functions

This section describes the functions of the bus protocol for data loggers DCX (device *Class.Group* 5.5).

Overview:

F30: Read out scaling values
F31: Write scaling values
F48: Initialise devices, whereby the device ID is returned
F66: Program bus address
F67: Read out record ROM
F68: Read out record ROM (not bus compatible)
F69: Read out serial number
F73: Read out current pressure and temperature values in floating-point format
F92: Read out record configuration
F93: Write record configuration
F95: Zeroing functions
F100: Read out configurations

Functions for DCX Dataloggers with CTD module (Conductivity measurement)

F0: Read out configuration from CTD
F170: Write configuration to CTD

4.1 Function 30: Read coefficient

Request:

DevAddr	30	Nr.	CRC16_H	CRC16_L

Response:

DevAddr	30	B3	B2	B1	B0	CRC16_H	CRC16_L

Exception errors:

2 if no. > 111
3 if message length incorrect
32 if device is not yet initialised

Note:
Every coefficient can be read in IEEE754 format (floating-point format 4-byte B0 .. B3) via this function.
→ Information on IEEE754: see appendix.
Unused coefficients contain undefined values (NaN)

4.1.1 Calibration values

No.	Description of coefficient	Unit
64	Offset of pressure sensor P1	bar
65	Gain factor of pressure sensor P1	
66	Offset of pressure sensor P2	bar
67	Gain factor of pressure sensor P2	

The calibration values can be read and written.

4.1.2 Information values

No.	Description of the coefficient	Unit
80	Minimum pressure of sensor P1	bar
81	Maximum pressure of sensor P1	bar
82	Minimum pressure of sensor P2	bar
83	Maximum pressure of sensor P2	bar
84	Minimum temperature of temperature sensor	°C
85	Maximum temperature of temperature sensor	°C
86	Minimum temperature of sensor P1	°C
87	Maximum temperature of sensor P1	°C
88	Minimum temperature of sensor P2	°C
89	Maximum temperature of sensor P2	°C
96	RC_ModusVal1	
97	RC_ModusVal2	
100	Reserved for custom applications	
…		
111		

No. 96 & 97: Values for event recording: just in RAM
For Firmware versions higher then 07.06, the settings are stored in the EEPROM and therefore they will not be lost after a battery change

4.1.3 Scaling of channels P1 and P2

P1 and P2 are linearly scalable with zero point and gain factor: **Value = gain factor * value + offset**
Standard values: Offset = 0.0, gain factor = 1.0
It is also possible to influence the offset values via function 95 (see function 95).
The gain factor should be used **for calibration purposes only**, and not to alter pressure units. The latter operation should always be carried out by the master!

4.2 Function 31: Write coefficient

Request:

DevAddr	31	Nr.	B3	B2	B1	B0	CRC16_H	CRC16_L

Response:

DevAddr	31	0	CRC16_H	CRC16_L

Exception errors:

2 If writing is not allowed
3 If message length is incorrect
32 If device has not yet been initialised

Note:
Information on scaling of the channels: See functions 73 and 95. Information on which channels are active: See function 100.

4.3 Function 36 : Write to RECORD-ROM

Anforderung:

DevAddr	36	Page_H	Page_L	Position	N	DATA 0	DATA 1	CRC16_H	CRC16_L

Antwort:

DevAddr	36	0	CRC16_H	CRC16_L

Exception errors:

1 Write not allowed Page < (RecRomLastPagePhysik_H/L (Funktion 100) - RecRomAmountTextPage(F100))
2 Write not allowed Page > RecRomLastPagePhysik_H/L (Funktion 100) or Position + N > 64
3 If message length is incorrect
32 If device has not yet been initialised

Note:
The Record-Rom is used to store record data. This record data could not be changed (write is not allowed), nevertheless there is a small section where write to the Record-Rom is allowed. This section is defined as RecRomAmountTextPage. The address of this range could be found in the description of Function 92/93. This section could be used to write any user information to the device.

Page : The address/page where write should be done.
Range reserved for Record data (write not allowed): (RecRomFirstPagePhysik_H/L(F100) ... RecRomLastPagePhysik_H/L (F100) - RecRomAnzTextPage(F100))
Range for user text (write allowed): (RecRomLastPagePhysik_H/L (F100) - RecRomAnzTextPage(F100) ... RecRomLastPagePhysik_H/L (F100))

Position: The first byte within a Page where write is executed. Position is a value between 0 and 63.

N : Amount of bytes which are transfer and written to the Record-Rom. The data is transferred in (DATA 0, DATA 1). Due to the limitation of the data buffer (10 bytes), the number of bytes that could be transferred (N) is limited by max 2.

4.4 Function 48 : Initialise and release

Request:

DevAddr	48	CRC16_H	CRC16_L

Response:

DevAddr	48	Class	Group	Year	Week	BUF	STAT	CRC16_H	CRC16_L

Exception error:

3 If message length incorrect

Note:
Each time the device is switched on by applying the supply voltage or after a break in the power supply, the device must be initialised via this function. Calling a different function will lead to **exception error 32**.

The following information is returned:

Class — Device ID code
5: digital pressure transmitter

Group — Subdivision within a device class
5: data logger DCX

Year, Week — Firmware version

BUF — Length of the internal receive buffer

STAT — Status information
0: Device addressed for first time after switching on.
1: Device was already initialised

4.5 Function 66 : Write and read new device address

Request:

DevAddr	66	NewAddr	CRC16_H	CRC16_L

Response:

DevAddr	66	ActAddr	CRC16_H	CRC16_L

Exception error:

3 If message length is incorrect
32 If device is not yet initialised

Note:
This function programmes the device addresses to NewAddr. The address is returned in ActAddr as confirmation. It is to be ensured that the new address NewAddr is not already in use by another bus user.

Permissible addresses: 1 .. 249. Address 250 is transparent. This means that every device, irrespective of the set address, will respond to address 250. Consequently, *transparent* DevAddr = 250 may only be used in stand-alone operating mode! Address 251 is used for modem function and can also just be used for stand-alone operating mode. For further information see appendix.

For the purpose of **reading the device address** when the address is not known, for example, the value 250 is transferred as DevAddr and the value 0 is transferred as NewAddr. The current address is then returned in response.

4.6 Function 67 : Read out record ROM

Request:

DevAddr	67	Page_H	Page_L	Position	N	CRC16_H	CRC16_L

Response:

DevAddr	67	DATA 0	…	DATA N-1	CRC16_H	CRC16_L

Exception error:

2 If Page > RecRomLastPagePhysic **or** Position + N > 64
3 If message length is incorrect **or** N > int.buffer - 4
32 If device is not yet initialised

Note:
Page: Allocation and size see function 93! (0… RecRomLastPagePhysic)
Position: 0… 63
N: Amount of bytes which are read on the requested page and position. This is limited by the protocol (max. int. buffer – 4)

4.7 Function 68 : Read out record ROM (not bus compatible)

Request:

DevAddr	68	Page_H	Page_L	Index	CRC16_H	CRC16_L

Response:

DevAddr	68	DATA 0	…	DATA 7/63	CRC16_H	CRC16_L

Exception error:

2 If Page > RecRomLastPagePhysic
3 If message length is incorrect
32 If device is not yet initialised

Note:
Page: Allocation and size see function 93! (0… RecRomLastPagePhysic)
Index: If index = 0, just the first 8 bytes of the page (header) will be transmitted (DATA 0…7)
If index = 1, the hole page will be transmitted (DATA 0…63)

This function is not conform to the protocol declaration (10 transmission bytes max.) and hence can not be used if more than one slave is connected to the bus.

4.8 Function 69 : Read serial number

Request:

DevAddr	69	CRC16_H	CRC16_L

Response:

DevAddr	69	SN3	SN2	SN1	SN0	CRC16_H	CRC16_L

Exception errors:

3 If message length is incorrect
32 If device is not yet initialised.

Note:
The serial number is allocated at the factory. It consists of 4 bytes and is calculated as follows :

$$SN = 256^3 * SN3 + 256^2 * SN2 + 256 * SN1 + SN0$$

4.9 Function 73 : Read value of a channel (floating point)

Request:

DevAddr	73	CH	CRC16_H	CRC16_L

Response:

DevAddr	73	B3	B2	B1	B0	STAT	CRC16_H	CRC16_L

Exception errors:

2 If CH > 5
3 If message length is incorrect
32 If device is not yet initialised

Note:
A device can measure up to five signals (channels):
Two independent pressure sensors, P1 and P2. Plus the temperatures of pressure sensors TOB1 and TOB2 respectively. The temperatures of the pressure sensors (TOB1, TOB2) are required for temperature compensation of the pressure signal. A temperature sensor (T) can also be measured.
P1-P2 is a calculated channel.
On a standard pressure transmitter, only channels P1 and TOB1 are available. You can read out which channels are active via function 100.

The measured value is returned in IEEE754 format (4-byte B0 ... B3).

CH	Name	Description	Unit
0	P1-P2	Difference of both pressure sensors	bar
1	P1	Pressure from pressure sensor 1	bar
2	P2	Pressure from pressure sensor 2	bar
3	T	Additional temperature sensor	°C
4	TOB1	Temperature of pressure sensor 1	°C
5	TOB2	Temperature of pressure sensor 2	°C

The **STAT** byte contains the current status.

Bit position	.7	.6	.5	.4	.3	.2	.1	.0
Name	/STD	---	TOB2	TOB1	T	P2	P1	---

A set **/STD** bit indicates whether the transmitter is in Power-up mode, otherwise it is in Standard mode.

A set **P1**, **P2**, **T**, **TOB1**, **TOB2** bit indicates that a measuring or computation error has occurred in the channel concerned.

4.10 Function 73 : Read value of a channel (floating point) (Additional Function for DCX with CTD Module)

Request:

DevAddr	73	CH	CRC16_H	CRC16_L

Response:

DevAddr	73	B3	B2	B1	B0	STAT	CRC16_H	CRC16_L

Exception errors:

2 f CH not valid
3 If message length is incorrect
32 If device is not yet initialised

Note:
A device with CTD Module can measure 3 additional signals (channels):
Conductivity uncompensated, conductivity compensated to 25°C and the temperature measured with a PT1'000 sensor.
Settings and factors for the correction/calculation of this measuring values are accessible with Function 0 and Function 170.

The measured value is returned in IEEE754 format (4-byte B0 ... B3).

CH	Description	Unit
10	Conductivity compensated to 25°C	mS/cm
11	Conductivity uncompensated	mS/cm
3	Temperature of PT1'000 Conductivity sensor	°C

The **STAT** byte contains the current status.

Bit position	.7	.6	.5	.4	.3	.2	.1	.0
Name	/STD	CTD	TOB2	TOB1	T	P2	P1	---

A set **/STD** bit indicates whether the transmitter is in Power-up mode, otherwise it is in Standard mode.

A set **P1**, **P2**, **T**, **TOB1**, **TOB2** bit indicates that a measuring or computation error has occurred in the channel concerned.

A set **CTD** bit indicates whether the CTD Module is in Power-up mode which takes around 1,5 seconds or a measuring or computation error has occurred. If the CTD bit is set, the values are not valid or not actual.

4.11 Function 92 : Read out record configuration

Request:

DevAddr	92	Index	CRC16_H	CRC16_L

Response:

DevAddr	92	PARA0	PARA1	PARA2	PARA3	PARA4	CRC16_H	CRC16_L

Exception errors:

2 If Index > 8
3 If message length incorrect
32 If device is not yet initialised

Note:
Index table see function 93.

4.12 Function 93 : Write record configuration

Request:

DevAddr	93	Index	PARA0	PARA1	PARA2	PARA3	PARA4	CRC16_H	CRC16_L

Response:

DevAddr	93	0	CRC16_H	CRC16_L

Exception errors:

2 If Index > 9
3 If message length incorrect
32 If device is not yet initialised

Note:
All variables are readable.
Marked variables are writable, they are saved in the RAM. That means: On a battery change they are lost.

For Firmware versions higher then 07.06, the settings are stored in the EEPROM
and therefore they will not be lost after a battery change

Index	Para0	Para1	Para2	Para3	Para4	Note
0	FUNC	---	---	---	---	---
1	CFG	REC_CTRL	EE_CTRL	PAGE_H	PAGE_L	---
2	RecRomFirstPagePhysik_H	RecRomFirstPagePhysik_L	RecRomLastPagePhysik_H	RecRomLastPagePhysik_L	RecRomAmountTextPage	EEPROM Info
3	Time_Byte3	Time_Byte2	Time_Byte1	Time_Byte0	---	Actual time
4	Start_Time_Byte3	Start_Time_Byte2	Start_Time_Byte1	Start_Time_Byte0		Record_StartTime
5	LoadFixCounter_LH	LoadFixCounter_LL	---	---	---	* until 03.10
6	LoadModusMessCounter_H	LoadModusMessCounter_L	LoadFastMessCounter_H	LoadFastMessCounter_H	LoadSaveCounter	---
7	ModusSelect_H	ModusSelect_L	ModusChannel	---	---	---
8	RecChannels _H	RecChannels _L	Available RecordCH_H	Available RecordCH_L	Bat_Capacity	---
9	LoadFixCounter_LH	LoadFixCounter_LL	LoadFixCounter_HH*	LoadFixCounter_HL*	---	* from 03.10 on

** from software version 03.10 on: Totally 4 bytes LoadFixCounter. Until 03.10 use Index 5, after 03.10 use Index 9 instead*

Index 0:

FUNC

Bit	7	6	5	4	3	2	1	0
description						f_RecStart	f_RecStop	f_RecFree

f_RecFree = 1, complete memory is write enabled.
0 = no ‚old record' can be over written and not the hole memory space is available.

f_RecStop = 1, a active record will be stopped.

f_RecStart = 1, starts a record (if start condition is met)

Index 1:

CFG

Bit	7	6	5	4	3	2	1	0
description								fRE_CFG_Continue

Set **fRE_CFG_Continue** for endless recording. (just writing)

REC_CTRL

Bit	7	6	5	4	3	2	1	0
description	fRE_Error	fRE_Active	fRE_Start				fRE_Memoryfull	fRE_Continue

fRE_Continue =1, endless recording. 0 = record until memory is full.
fRE_Memoryfull = 1, no more memory is available. Enable writing with f_RecFree
fRE_Start = 1, start conditions are set, wait for start (trigger)
fRE_Active = 1, record configuration written, record active or waits for start (prepared)
fRE_Error = 1, error occurred, record stopped.

EE_CTRL

Bit	7	6	5	4	3	2	1	0
description					f_AckError	f_LoBatError		

PAGE_H/L current page in which data will be saved

Index 2: Available record memory
Index 3: **Time_Byte0…3:** Actual time in the DCX
Index 4: **Start_Time_Byte0…3:** If actual time is later then start time, then record is possible.
Index 5: **LoadFixCounter:** Interval for fixed record saving interval in seconds.
Index 6:

LoadModusMessCounter_H/L: Interval for event check in seconds.
LoadFastMessCounter_H/L: Interval for event check if a event has occurred (fast event check).
LoadSaveCounter: After *LoadSaveCounter* measurements the average of these measurements is saved. (Average in 1 second interval)

Index 7:

ModusSelect_H/L: Record type according to the set bit.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
description											CH< Val1	CH> Val1	Delta CH	On Off	Intervall	Fix Intervall

ModusChannel: Event check for selected channel.

Index 8:

Available RecordCH_H/L & RecordChannels_H/L: channels which are available and which are recorded.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
description	---	CH 14	CH 13	CH 12	CH 11	CH 10	CH9	CH8	CH7	CH6	TOB2	TOB1	T	P2	P1	P1-P2

Bat_Capacity Battery state (of capacity) 0...100%

4.13 Function 95 : Commands for setting the zero point

Requests:

Request a:

DevAddr	95	CMD	CRC16_H	CRC16_L

Request b with setpoint:

DevAddr	95	CMD	*B3*	*B2*	*B1*	*B0*	CRC16_H	CRC16_L

where B3:B0: Floating-point number IEEE754 format (4-byte B0 ... B3) for the setpoint.

Response:

DevAddr	95	0	CRC16_H	CRC16_L

Exception errors:

- **1** If in Power-up mode
- **2** If CMD > 3
- **3** If message length incorrect
- **32** If device is not yet initialised

Note:
Fixed amount of bytes on modem communication (address 251): 5

The following actions can be carried out with this function:

CMD	Meaning
0	Set zero point of P1
1	Reset zero point of P1 to standard value
2	Set zero point of P2
3	Reset zero point of P2 to standard value

CMD 0, 2:
Zero point values for pressure channels P1 and P2. These values can also be read via function 30 and written via function 31.

Request a: The zero point is calculated such that the current measured value = 0.0
Request b: The zero point is calculated such that the current measured value equals the setpoint (B3:B0)

CMD 1, 3: Reset zero point to factory setting
The zero point values are reset to 0.

4.14 Function 100 : Read configuration

Request:

DevAddr	100	Index	CRC16_H	CRC16_L

Response:

DevAddr	100	PARA0	PARA1	PARA2	PARA3	PARA4	CRC16_H	CRC16_L

Exception errors:

2 If index > 8
3 If message length is incorrect
32 If device is not yet initialised

Note:
CFG_P & CFG_T : available channels

This function supplies the information as to how the device has been configured. This configuration is carried out at the factory and cannot be altered by the customer.
A pressure transmitter can read two independent pressure sensors (P1 and P2), plus the temperatures of the respective pressure sensors (TOB1 and TOB2) and an independent temperature (T).

Index	**Para0**	**Para1**	**Para2**	**Para3**	**Para4**
2	CFG_P	CFG_T	---	---	CNT_T

CFG_P: Channels which are measured max. every second or in the record interval
CFG_T: Channels which are measured if the channel has been selected for recording or if a temperature compensation (correction of pressure signal) is made.

CFG_P and **CFG_P** are encoded as follows:

Bit position:	.7	.6	.5	.4	.3	.2	.1	.0
Description:	CH7	CH6	TOB2	TOB1	T	P2	P1	P1-P2

4.15 Functions for DCX with CTD Module

4.15.1 Function 0: CTD – read CTD configuration (Function only for DCX with CTD Module)

Anforderung:

DevAddr	0	Index	PARA0	PARA1	PARA2	PARA3	CRC16_H	CRC16_L

Antwort:

DevAddr	0	STAT	PARA0	PARA1	PARA2	PARA3	CRC16_H	CRC16_L

Note:
A device with CTD Module can measure 3 additional signals (channels):
Conductivity uncompensated, conductivity compensated to 25°C and the temperature measured with a PT1'000 sensor.
Settings and factors for the correction/calculation of this measuring values are accessible with Function 0 and Function 170.
The measuring values are accessible with function F73.

STAT:
A set **STAT** bit indicates (STAT = 1) whether the CTD Module is in Power-up mode which takes around 1,5 seconds or a measuring or computation error has occurred. If the CTD bit is set, the values are not valid or not actual.

Index:
Table below at Function 170.

4.15.2 Funktion 170: CTD – write CTD configuration (Function only for DCX mit CTD Module)

Anforderung:

DevAddr	170	Index	PARA0	PARA1	PARA2	PARA3	CRC16_H	CRC16_L

Antwort:

DevAddr	170	STAT	0	0	0	0	CRC16_H	CRC16_L

STAT: wie Funktion 0

Index:

Index	Bezeichnung	Describtion	Byte Position / Datentyp	Einheit
25	Measuring range selector	1 = Measuring range 1 (0...0.2mS/cm) 2 = Measuring range 2 (0...2mS/cm) 3 = Measuring range 3 (0...20mS/cm) 4 = Measuring range 4 (0...200mS/cm)	Para0 / Byte	
28	Serialnumber CTD Module	Serialnumber = Para1 + Para2*2^8 + Para3*2^16	Para1 Para2 Para3 / Integer	
30	Device type / Firmware	Device type (Para0.Para1) / Firmware (Para2.Para3)	Para0 Para1 Para2 Para3 / Byte	
31	Factor Temperature (T)	Factor for Temperature adjustment (Factory setting = 1,000)		
32	Offset Temperature (T)	Offset für Temperature adjustment (Factory setting = 0,000)		°C
33	Factor cond. for range 1	Faktor für Leitfähigkeit Bereich 1 (Factory setting = 1,000)		
34	Factor cond. for range 2	Faktor für Leitfähigkeit Bereich 2 (Factory setting = 1,000)		
35	Factor cond. for range 3	Faktor für Leitfähigkeit Bereich 3 (Factory setting = 1,000)		
36	Factor cond. for range 4	Faktor für Leitfähigkeit Bereich 4 (Factory setting = 1,000)		
38	Media temperature factor	Factor for temperature dependency of the media. The conductivity will be calculated to ist value at 25°C. (Factory setting = 0,0225 corresponds to 2,25%/°C ~standard for fresh water)		%/°C
39	Cell constant	Zellkonstante für Leitfähigkeit (Werkseinstellung = 1,000) (wirksam auf alle Bereiche)		

Interpretation of data type Float (Function F0 and F170):

Float: The Value is in the normed format IEEE754-Format (4-Byte Para0 ... Para3).

PARA0			PARA1	PARA2	PARA3
s	e[30:23]		f[22:0]		
31	30	24	23		0

5 Appendix

5.1 Interface converter

The serial RS232 interface or the USB interface can be used for connection to a PC. KELLER offers converters for this purpose. Various other products are commercially available, however. The following requirements apply when working with KELLER software:

- The converter must control transmit / receive switch-over automatically.
- KELLER converters feature a hardware echo, i.e. the transmitted message is received again immediately as an echo. This echo is required by some KELLER software programmes.

5.2 Communication with modem (or Palmtop)

5.2.1 Problem

Devices with standard bus protocol detect the end of a message through a timeout counter which starts after reception of each byte. Follows another byte within ~200us the timeout counter stops and starts again after the end of the new byte. Reaches the timeout counter zero, then the message gets interpreted. If the message is correct the device gives a answer.

For communications with modem the time between each byte can take up to 300ms (small send buffer). The device detects such a timeout as the end of a message, interprets it as not complete and rejects it: The device will not answer!

5.2.2 Solution

A interpretation of a message is only made after the reception of a certain amount of byte. The detection of such a specific message is made by the **bus address 251**. (That means further, that just one device can be connected to the bus)

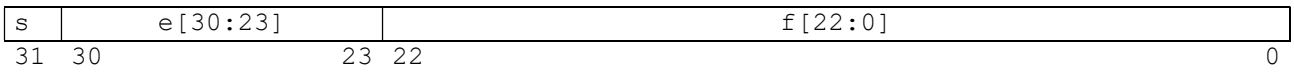
Now the time between each byte can take up to 400ms.

If the message is 400ms after the reception of the last byte still not complete (amount of bytes smaller than expected) then the message will be rejected: The device does not answer and goes back in receive mode again. The amount of byte for each message is defined by the function number.

5.3 Floating-point format IEEE754

As data transmission is effected byte-wise (8-bit data), the floating-point values are represented as follows :
B0: Bit 0..7; B1: Bit 8..15, B2: Bit 16..23, B3: Bit 24..31

Representation in accordance with IEEE754:



If you use the DLL which is available from KELLER, you do not need to carry out conversion, as this is encapsulated in the DLL. If you wish to address the devices directly, however, you must convert the individual bytes into a floating-point value.
To obtain a floating-point value from the individual bytes, proceed as follows:

1. Define data structure in which an array of 4 bytes and a 32-bit floating-point value is defined at the same memory location.
2. Write the bytes into the byte array.
3. Read out the floating-point value.

You do not need to carry out any actions, therefore, as the computer attends to interpretation. Some microcontrollers have a different data structure for floating-point values. In such cases, adaptation is necessary.

Further information is to be found at:
http://cch.loria.fr/documentation/IEEE754/numerical_comp_guide/ncg_math.doc.html - 556

5.4 Memory map

The memory of the DCX is divided into Pages. The amount of pages depend on the device type. On YEAR.WEEK 02.35 there are 2048 pages, the 03.15 contains 4096 pages.

Each page contains 64 bytes. The first 8 bytes represent the header of the page. The following bytes are the data. A dataset consists of 4 bytes. These can contain the measure channel, time and value.

Memory: 2048 or 4096 Pages

Page	Description
0	Data memory
1	
2	
...	
...	
...	
X-1	End of data memory
X	Record text
...	
...	
...	
2047	End of record text

1 Page with 64 bytes :

Byte	Description
0	Header
1	
2	
3	
4	
5	
6	
7	
8	Data 1
9	
10	
11	
12	Data 2
13	
14	
15	
...	...
...	...
...	...
60	Data 14
61	
62	
63	

Header

The header consists of 8 bytes.

Byte	Description of the header
0	Start detection / overflow counter / start pointer
1	Remaining 8 bit of the start pointer
2	Absolute time: 1st byte (byte1)
3	Absolute time 2nd byte (byte2)
4	Absolute time 3rd byte (byte3)
5	Absolute time 4th byte (byte4)
6	Reserved
7	Reserved

Absolute time in seconds since 01.01.2000
(2^32*Byte1+2^16*Byte2+2^8*Byte3+Byte4)

Byte 0:

7	6	5	4	3	2	1	0

Bit 7: **Start detection** – Set bit indicates page as start of record.
Bit 5 + 6: **Overflow counter** – Is incremented if last page has reached and a new one is started.
Bit 0 .. 4: Together with the 2nd byte it represents the start pointer. It points to the page on which the record starts. 5 Bit + 8 Bit = 13 Bit

Data x
A dataset consists of 4 bytes. There are four types of them:

a) Dataset with measured channel, value and time gap

Byte	Description	
0	Bit 7..4: Measured channel ‚0000' to ‚1110'	Bit 3..0: Time gap to last dataset (0..15s)
1	Measured value 1st byte	
2	Measured value 2nd byte	
3	Measured value 3rd byte	

The measured value is a 4 byte float, where the last byte is omitted.

b) Dataset with time space

Byte	Description	
0	Bit 7..4: ‚1111'	Bit 3..0: ‚0000'
1	Time space to last data set 1st byte	
2	Time space to last data set 2nd byte	
3	‚0000 0000'	

A dataset with time space is used when the time gap to the last data set is more than 15 seconds.
Time space = 256* 1st byte + 2nd byte = Range (0.. 65'535 s)

c) Dataset with text

Byte	Description	
0	Bit 7..4: ‚1111'	Bit 3..0: ‚0100'
1	8 bit ASCII character	
2	8 bit ASCII character	
3	8 bit ASCII character	

Allows saving of a text packet with 3 characters (3 bytes).

d) Empty dataset

Byte	Description	
0	Bit 7 .. 4: ‚1111'	Bit 3 .. 0: ‚1111'
1	‚xxxx xxxx'	
2	‚xxxx xxxx'	
3	‚xxxx xxxx'	

A empty dataset can mark the end of a record.

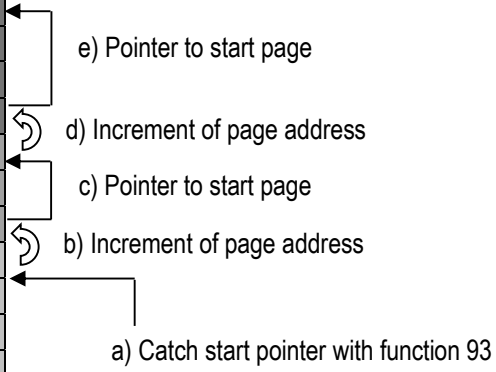
Procedure for reading the directory of a record

1. Reading of the active page
Read the address of the actual page with function 92 Index 1.

2. Reading of the last pages
Increment the address of the actual page and check the address of the start pointer. It's found in the header of the page which can be read with function 76. The difference of actual page and start page shows the size of the record. The start time of the start page is also found in the header of the start page.

With further increments of the page addresses and the same procedure as described above also the next records can be found. Take care of overflows.

Page	Description
0	start page of record 1
1	record 1
2	record 1
3	record 1
4	start page of record 2
5	record 2
6	record 2
7	start page of record 3
8	record 3
9	record 3
10	record 3
...	



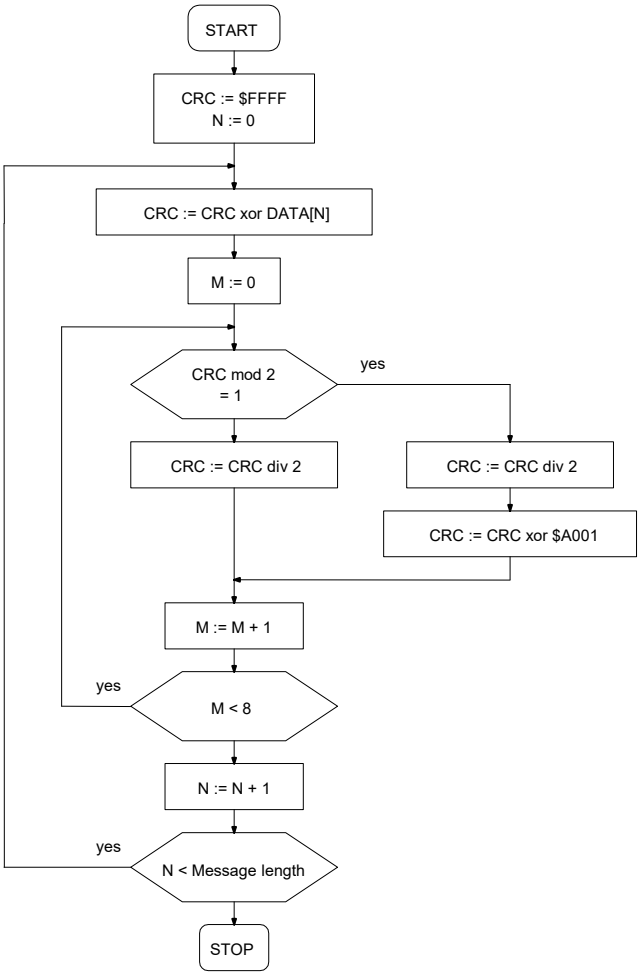
5.5 Calculation of the CRC16 checksum

The checksum can either be calculated or derived from a table.
Here is an example of CRC16 calculation in C:

```
////////////////////////////////////////////////////////////////////////
// CRC-16 calculation in C
//
// Calculation of CRC-16 checksum over an amount of bytes in the serial buffer.
// The calculation is done without the 2byte from crc16 (receive-mode).
// SC_Buffer[]: Byte-Buffer for the serial interface. Type: unsigned char (8bit)
// SC_Amount : Amount of Bytes which should be transmitted or are received (without CRC16)
//
////////////////////////////////////////////////////////////////////////
void CalcCRC16(unsigned char* CRC_H, unsigned char* CRC_L)
{
    // locals
    unsigned int Crc;
    unsigned char  n, m, x;

    // initialisation
    Crc= 0xFFFF;
    m= SC_Amount;
    x= 0;

    // loop over all bits
    while(m>0)
    {
        Crc^= SC_Buffer[x];
        for(n=0; n<8; n++)
        {
           if(Crc&1)
           {
               Crc>>= 1;
               Crc^= 0xA001;
           }
           else
               Crc>>= 1;
        }
        m--;
        x++;
    }
    // result
    *CRC_H= (Crc>>8)&0xFF;
    *CRC_L= Crc&0xFF;
}// end CalcCRC16
```



This results in the following calculation for function 48 with device address 250: CRC16_H= 4, CRC16_L= 67.

5.6 Description of the software driver (DLL)

5.6.1 General

The available DLL ***DCXc.dll*** has been tested on the Windows 95, 98, NT and 2000 operating systems.
Examples of the use of this DLL are available for the following programming languages:

- LabVIEW
- C++
- Delphi
- VB
- VBA

The call convention **stdcall** is used for assigning the parameters to the functions. This means that:

- all parameters are passed via the stack,
- the parameter furthest to the right is calculated and passed first, the parameter furthest to the left is calculated and passed last,
- the function itself deletes the parameters from the stack.

As the declarations for the functions presented below show, many variables are declared with the prefixed word *var*. This means that these variables are passed as pointers and not as values.
The types employed for declaration purposes are described below:

Type	**Range**	**Format**
Byte	0..255	8-bit without sign
Word	0..65535	16-bit without sign
Smallint	-32768..32767	16-bit with sign
Longint	-2147483648.. 2147483647	32-bit with sign
Pbyte		Pointer to byte
Single	+/- $1.5x10^{-45}$..$3.4x10^{38}$	32-bit

5.6.2 The functions of the DLL

Each function returns a value which indicates whether the desired function has been successfully executed or not. All the possible return values are specified below. The returned parameters are only valid and may only be processed if the function concerned has been successfully executed.

Return value		**Description**
RS_OK	0	Function successfully executed; return parameters are valid
RS_EX1	1	Function successfully executed; but exception error 1 has occurred
RS_EX2	2	Function successfully executed; but exception error 2 has occurred
RS_EX3	3	Function successfully executed; but exception error 3 has occurred
RS_EX32	32	Function successfully executed; but exception error 32 has occurred
RS_BROADCAST	100	Broadcast
RS_ERROR	-1	General error
RS_TXERROR	-2	Transmit error
RS_RXERROR	-3	Receive error in UART
RS_TIMEOUT	-4	No data or insufficient data received
RS_BADDATA	-5	Data erroneous (e.g. CRC16 erroneous)

5.6.2.1 Port functions

The devices are connected to the PC via a serial interface. The port functions serve to open and close this interface. Ports 1 to 9 (COM1..COM9) are valid. The standard setting should be used for the timeout time (Timeout = 0). When the desired port has been successfully opened, the **OpenComPort** function returns the value RS_OK, otherwise RS_ERROR.
An open port is closed automatically on ending the programme.
It is additionally possible to set the baud rate and the data format via the **OpenComExt** function. DCX devices only support 9600 baud.
As a standard setting, **no** parity is used (none). This results in a data format of 10 bits per byte. If parity is active, the data format is 11 bits per byte.

```
function OpenComPort( intPort, intTimeout: Smallint ): Smallint; stdcall; export;

function OpenComExt( intPort, intTimeout: Smallint; longBaud: Longint; intParity:Smallint
): Smallint; stdcall; export;
```

`intParity`: 0: no parity bit (sStandard), 1: odd parity bit, 2: even parity bit
`longBaud`: 9600 for 9600 baud

```
function CloseComPort : Smallint; stdcall; export;
```

5.6.2.2 Echo function

Interface converters from KELLER Druckmesstechnik always supply an echo of the message transmitted by the PC.
This function has the standard value 1 (Echo On), to enable operation with the converters supplied by KELLER. If other converters are used which do not supply a hardware echo, the function must be set to 0 = Echo Off .

```
function EchoOn(  bteEcho: Byte ): Smallint; stdcall; export;
```

5.6.2.3 Protocol functions

The following functions encapsulate the above-described bus functions. The parameter sequences are identical. The CRC16 checksum is not included here, as it is calculated and checked in the DLL. Some parameters consist of several bytes. These are grouped together for the sake of clarity. The different requests a and b pertaining to function 95 are split into two functions: F95 and F95val.

```
function F30(  bteDeviceAddr, bteCoeffNo: Byte; var sinCoeff: Single
): Smallint; stdcall; export;

function F31(  bteDeviceAddr, bteCoeffNo: Byte; sinCoeff: Single
): Smallint; stdcall; export;

function F48(
  bteDeviceAddr: Byte; var bteClass, bteGroup, bteYear, bteWeek, bteBuffer, bteState: Byte
): Smallint; stdcall; export;

function F66(  bteDeviceAddr, bteNewAddr: Byte; var bteActualAddr: Byte
): Smallint; stdcall; export;

function F67(
  bteDeviceAddr: Byte; wrdPage: Word; btePos: Byte; bteAmount: Byte; pbteData: Pbyte
): Smallint; stdcall; export;

function F68(  bteDeviceAddr: Byte; wrdPage: Word; bteIndex: Byte; pbteData: Pbyte
): Smallint; stdcall; export;

function F69(  bteDeviceAddr: Byte; var linSN: Longint
): Smallint; stdcall; export;

function F73(  bteDeviceAddr, bteChannel: Byte; var sinValue: Single; var bteStat: Byte
): Smallint; stdcall; export;

function F92(
  bteDeviceAddr, bteIndex: Byte; var btePara0, btePara1, btePara2, btePara3, btePara4: Byte
): Smallint; stdcall; export;

function F93(
  bteDeviceAddr, bteIndex: Byte; btePara0, btePara1, btePara2, btePara3, btePara4: Byte
): Smallint; stdcall; export;

function F95(  bteDeviceAddr, bteCmd: Byte
): Smallint; stdcall; export;

function F95val(  bteDeviceAddr, bteCmd: Byte; sinVal: Single
): Smallint; stdcall stdcall; export;

function F100(
  bteDeviceAddr, bteIndex: Byte; var btePara0, btePara1, btePara2, btePara3, btePara4: Byte
): Smallint; stdcall stdcall; export;


function F0 and F170
is not supported with the dll yet.
```

5.7 Support

We are pleased to offer you support in implementing the protocol.

For implementation under Windows, a DLL with diverse reference implementations is available.

The software READ30 or CCS30 is available free of charge for the configuration and read-out of the devices.

All information and software is available under:
http://www.keller-druck.com

KELLER AG für Druckmesstechnik
St. Gallerstrasse 119 • CH-8404 Winterthur
Tel: ++41 52 235 25 25
http://www.keller-druck.com