I²C
Communication Protocol

X-Line Pressure Transmitters with I²C Interface

Version 06/2025

KELLER Druckmesstechnik AG
St. Gallerstrasse 119
CH-8404 Winterthur
+41 52 235 25 25
info@keller-pressure.com
www.keller-pressure.com

1 Introduction

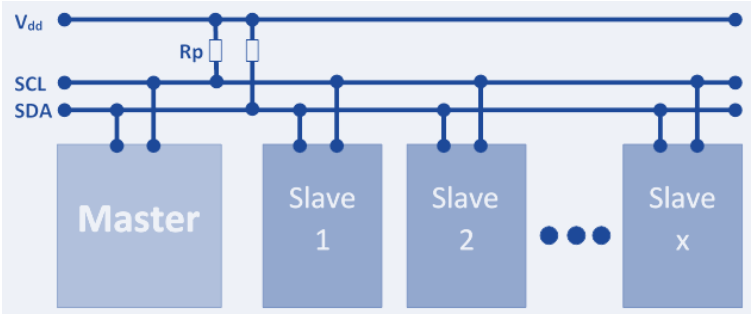
This document describes the I2C communications protocol for the X-Line digital pressure transmitters from KELLER Druckmesstechnik.

2 Installation / Electrical Connection

Several devices can be operated on one serial interface by connecting them all in parallel (SCL, SDA, GND and +VDD). Before incorporating the devices into the bus, each device must be programmed with a different address.

It is possible to configure a network, with a maximum of 128 devices in 7-bit address mode, but the I2C standard defines that only 112 nodes are possible because of the reserved addresses. Only the 7-bit address mode is supported by the KELLER transmitter.

The cable length shall be a maximum of few centimetres because I2C communication uses two bidirectional open–drain lines with pull-up resistors. The pull-up resistors need to be placed on the master. The transmitters don't have the pull-up resistors built in internally.



3 Bit transfer layer (physical layer)

Clock rate

A maximum clock rate of 400kHz is possible.

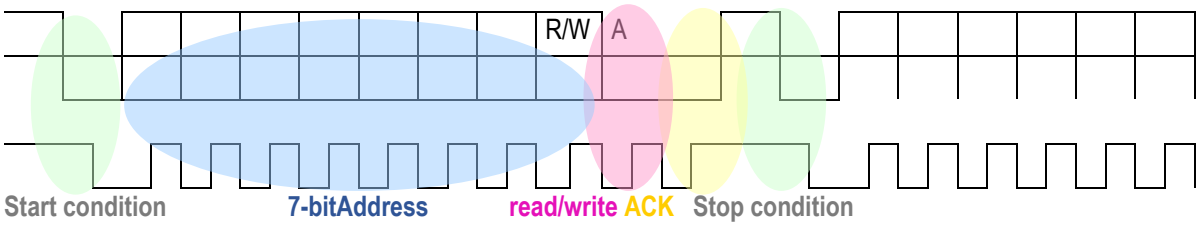
Clock stretching

In normal mode, the transmitter uses clock stretching only for short periods. This is a maximum period of 100us to prepare to receive or transmit data from a certain address. It is not possible to switch clock stretching off. Additionally, there is a continuous read mode, where clock stretching is used to denote that data is ready to read.

4 Data-link layer

This section describes how data exchange is affected on the bus. The data and their check and control structures are grouped together to form messages/frames. 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 one time. All other devices are then in receiving mode. Each message exchange takes place under the control of the master. The message contains the address for the receiving slave.

4.1 Transmission format for the serial interface



Read/write:	If this bit is set to high, the transmitter will receive data from the master. (R = 1) If this bit is set to low, the transmitter has to send data to the master. (W = 0)
ACK	The ACK bit has to be reset to low by the slave when the master is sending data to acknowledge that the byte has been received correctly and when the slave is transmitting, the master has to reset the ACK bit, except the master wants to stop the slave sending data, then it does not generate an ACK.
Start / Stop	These are conditions which do not appear during reception and transmission of data (level changes of SDA during SCL low).

4.2 Format of a message

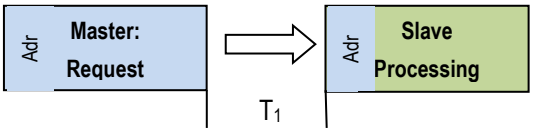
In I2C communication, it is only possible to receive or transmit data in one frame. This is why the master always has to first send a write command, to tell the slave from which address it wants to read data. After the write command the master can send a read command and the slave knows which data is requested and puts it on the bus. It is possible to read or write multiple addresses in one command.

4.3 Principle of message exchange

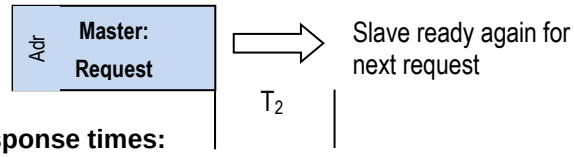
4.3.1 General rules

- 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 exchange is initiated by the master. This means that a device may only transmit data if requested to do so by the master.
- Every byte has to be confirmed by an ACK except the last byte of a message sent by the slave (here a NACK is set by the master to show the slave the end of its time to send).
- Bytes are always transmitted in the order MSB first.
- The transmitter expects a start condition at the beginning of each frame and a stop condition at the end of each frame. The transmitter takes the last byte of the frame as CRC and starts processing the request only after detecting a stop condition!

Message frames for read operation:



Message frames for write operation:



Response times:

T_1: Time between receipt of inquiry and beginning of response.

T_2: Time to ready-to-receive state for the slave.

4.3.2 Treatment of errors

Two types of errors may occur during the interchange of messages between master and transmitter: transmission errors and state errors.

4.3.3 Transmission errors

After a write operation of the master (writing data to the transmitter), the transmitter is not able to respond for several ms (T_2), because it is writing the received data to its internal EEPROM. During this time, a master request (address) is not acknowledged.

If a request is not completely sent or has no correct CRC8 at the end, the transmitter will not prepare any data.

Byte **State** (communication → SET if true)

Bit	Name	Description
7	Reserved	
6	I2CDataReceived	is set after the stop condition of the I2C, and if the transmitter had been able to receive the data. It is reset as soon, as the transmitter starts processing the request.
5	I2CProcessingData	is set as soon as the transmitter starts processing the request and it is reset as soon as data is ready to collect (I2CDataReady set) or an error has been detected and is set (I2CCRCError, I2CAddressError, I2CAmountError, I2CWriteError).
4	I2CDataReady	is set after a read request, when the data is ready to be collected by the master.
3	I2CCRCError	Is set after the detection of a CRC error (wrong CRC). It is reset after the reception of a new request by the master (Device Address W)
2	I2CRegisterError	Is set after the detection of a register address error (unused Address, address not available for the data format…). It is reset after the reception of a new request by the master (Device Address W)
1	I2CAmountError	Is set after the detection of an amount error (amount not available for the data format…). It is reset after the reception of a new request by the master (Device Address W)
0	I2CWriteError	It is set if the transmitter is not able to write the data to its internal EEPROM and save it. Please check the data value, which was requested to be written before. This error bit is reset after the reception of a new request by the master (Device Address W)

4.3.4 PTState errors

The state of the communication, as well as the state of each sensor is displayed in two state bytes; 1) general and for communication, 2) state of the sensors (StatePT).

Byte **StatePT** (sensor errors → SET if error occurred)

Bit	Name	Description
7	Startup	Startup ready (bit7 = 0) *
6	Reserved	
5	TOB2	Temperature of Pressure Sensor 2 (TOB = Top Of Bridge)
4	TOB1	Temperature of Pressure Sensor 1 (TOB = Top Of Bridge)
3	T	Temperature of Temperature Sensor (i.e. PT1000)
2	P2	Pressure Sensor 2
1	P1	Pressure Sensor 1
0	CH0	CH0 (for mathematical computations)

*State of Pressure and Temperature at the time of the request. If Startup bit is set, the process data values are NaN other values are available.

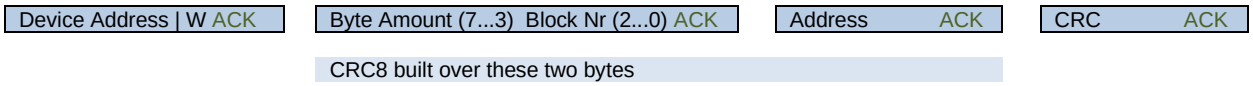
5 Description of Read and Write functions

In the following illustrations, a blue color is used for Data sent by Master, a green color for data sent by the device.

5.1 Read command

Frame

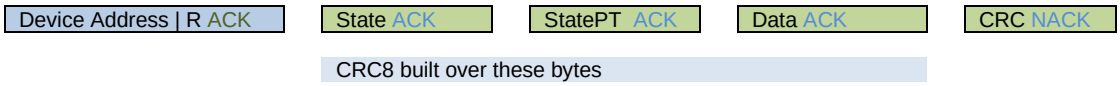
A read command always starts with a write command to tell the transmitter what kind of data to provide. This request always includes the transmitter address and the amount of bytes to read, the block number (from which to read and the address of the data in the block) as well as the CRC8 of the whole request without Device Address and R/W Bit.



Byte Amount means the number of data bytes to be read from the transmitter.

The requested data can be collected as follow:
The master has to send the Device Address with a set R/W bit (-> read). Afterwards it is continuous clocking and the transmitter puts its data on the bus. If the Transmitter has the requested data not yet ready, it sends the state byte with a cleared I2CDataReady bit. If the master now keeps on clocking, the transmitter repeats the state byte until the I2CDataReady bit is set. Like this, it is possible to do polling till the requested data is ready to collect. As long as I2CDataReady bit is RESET, no CRC is calculated and sent. It doesn't matter, if the Master appends a Stop-Condition after receiving the Status-Byte or if the data is clocked out continuously.

As soon as the I2CDataReady bit is set, the transmitter sends the requested data when the master keeps on clocking. The first byte is always the state of the I2C bus, the second byte holds the state of the sensor channels (StatePT) and the last byte to be transmitted is the CRC8 checksum over all sent bytes by the transmitter and the bytes in between hold the requested data.



Data holds as many bytes as requested by the master in “Byte Amount”

If the master keeps on clocking after the CRC, the transmitter will not drive the SDA-Line which leads to a 0xFF.

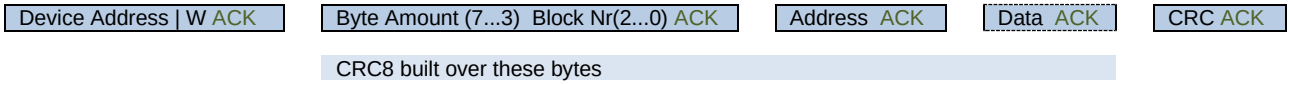
Timing

the transmitter needs up to 300us (T_1) to get the response ready (after this time, the state bit TI2CDataReady is set and data is transmitted from transmitter to master, if the master is clocking SCL).

5.2 Write command

Frame

A write command always starts with the I2C Device Address, followed by the number of data bytes to be written (Address and CRC not counted), the block number and the address in the block. Afterwards all the data to be written follow and in the end the CRC8 over all bytes (except I2CDeviceAddress) and R/W bit. The writing of the data can take up to 30ms, so the I2CProcessingData bit is set in the state byte. During this time the internal EEPROM of the transmitter is written and the transmitter will not acknowledge any request from the master.



Timing

Writing data to the transmitter is used to adjust gain, offset or to make configurations. All the data is stored in the internal EEPROM of the transmitter. During a write operation to the internal EEPROM, the transmitter is not able to do any communication in parallel! This is why the transmitter is not able to answer or even acknowledge any request of the master during T_2 after a master write request. The time T_2 depends on the amount of data to be written to the transmitter. A write to the block 1 takes usually 1ms, but it could take up to 30ms.

5.3 Continuous read command

Frame

This command is only available for block 0 address 0x80 …0xB4. It offers the possibility to read the pressure and temperature values as fast as possible.
At the beginning, the amount of data bytes, block number and address need to be transmitted to the transmitter:

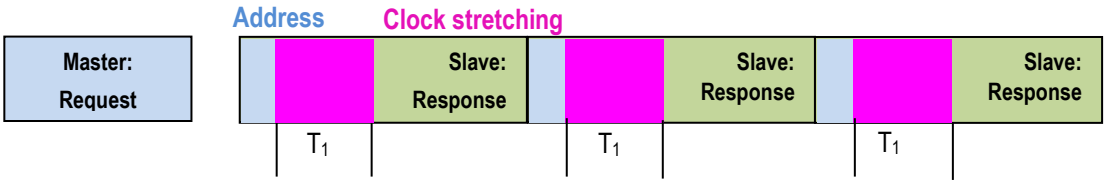
Device Address | W ACK	Byte Amount (7...3) Block Nr (2...0) ACK	Address ACK	CRC ACK
	CRC8 built over these two bytes		

Next, the data is requested with the I2C Device Address and a set read bit. The transmitter stretches the clock SCL after reception of the address until it has all the data ready to send. After it releases the clock line, the master can keep on clocking and get the data from the transmitter. For the next request, the master only has to do a read request (I2C Device Address (with a set read bit)) and the transmitter provides the data of the last Write-Request (i.e. Pressure and Temperature).

Device Address | R ACK	State ACK	StatePT ACK	Data ACK	CRC NACK
	CRC8 built over these bytes			

Timing

In this special case, the master only sends once the amount of data to collect and the address of the requested data. Afterwards it only puts the transmitter address on the bus and clocks the SCL for the requested data. As the transmitter needs to get the newest data ready, it takes up to 130µs (T1) until the transmitter prepared the response. During this time, the transmitter stretches the clock.



6 Address range

6.1 Address table

	Block 0 (read float / INT32)					Block 1 (read / write byte)				Block 2 (read /write/float)				Block 3 (read 4byte/float)			
Address	3	2	1	0	I2C Mode	3	2	1	0	3	2	1	0	3	2	1	0
0x00	P1 float				Polling	0	0	0	DeviceAddress	P1 offset				P1 Min			
0x04	TOB1 float				Polling	0	0	0	FILT_CTRL	P1 gain				P1 Max			
0x08	P2 float				Polling	0	0	0	LP_FILT	TOB1 offset				TOB1 Min			
0x0C	TOB2 float				Polling					P2 offset				TOB1 Max			
0x10	T float				Polling					P2 gain				P2 Min			
0x14	CH0 float				Polling					TOB2 offset				P2 Max			
0x18										T offset				TOB2 Min			
0x1C														TOB2 Max			
0x20	P1 INT32				Polling									T Min			
0x24	TOB1 INT32				Polling									T Max			
0x28	P2 INT32				Polling												
0x2C	TOB2 INT32				Polling												
0x30	T INT32				Polling									Serial_Number			
0x34	CH0 INT32				Polling									FW_Version			
0x38														Calibration_Date			
0x3C														Sensor_Type			
0x40														I2CInt_Vers			
0x44														CFG_Channel			
0x48																	
0x4C																	
0x80	P1 float cont				Stretch												
0x84	TOB1 float cont				Stretch												
0x88	P2 float cont				Stretch												
0x8C	TOB2 float cont				Stretch												
0x90	T float cont				Stretch												
0x94	CH0 float cont				Stretch												
0x98																	
0x9C																	
0xA0	P1 INT32 cont				Stretch												
0xA4	TOB1 INT32 cont				Stretch												
0xA8	P2 INT32 cont				Stretch												
0xAC	TOB2 INT32 cont				Stretch												
0xB0	T INT32 cont				Stretch												
0xB4	CH0 INT32				Stretch												
0xC0	P1 float *sleep*				Polling	Auto_SleepMode (U8)											
0xC4	TOB1 float *sleep*				Polling	Sleep_FallbackTime (U16)											
0xC8	P2 float *sleep*				Polling	Wakeup_SettleTime (U16)											
0xCC	TOB2 float *sleep*				Polling	SMAFilter_Depth(U16)											
0xD0	T float *sleep*				Polling												
0xD4	CH0 float *sleep*				Polling												
0xD8																	
0xDC																	
0xE0	P1 INT32 *sleep*				Polling												
0xE4	TOB1 INT32 *sleep*				Polling												
0xE8	P2 INT32 *sleep*				Polling												
0xEC	TOB2 INT32 *sleep*				Polling												
0xF0	T INT32 *sleep*				Polling												
0xF4	CH0 INT32 *sleep*				Polling												

6.2 Block 0: Measurements

The pressure and temperature values can be read as single precision float with the pressure unit in bar and temperature degree in Celsius, or as 32bit signed integer fix point values.

The values displayed as 32bit signed integer are fix point values and are displayed as followed:

Unit:	P1 & P2	Pascal (1Pa = 10^{-5} bar)
	T, TOB1, TOB2	0.01°C

6.2.1 Sleep Mode

If the transmitter does support the sleep-mode, a read access to address 0xC0 to 0xF4 will put the transmitter to sleep-mode (sleep) directly after the process data is sent. The transmitter doesn't acknowledge new requests up to 500us after the process data is sent. The transmitter is using up to 4uA supply current during sleep mode.
The transmitter will wake up with the first address match. After waking up, the transmitter needs up to 9ms to provide the first measurement (from Start of Communication until the data is transmitter (with 400kHz Bus speed)

6.3 Block 1: Configurations

The I2C frame looks the same as the one of Block 0 but the most significant byte B3 and the second data byte B2 are always zero. The relevant data are for a data width U8 in B0 and for a width of U16 in B1 and B0. So, be aware to read always 4 data bytes, but only interpret the relevant ones.

Reading: interpretation of the data inside the frame:

Device Address		State		StateP T		Data B3		Data B2		Data B1		Data B0		CRC

Writing: interpretation of the data inside the frame:

Device Address		Amount Block Nr		Address		Data B3		Data B2		Data B1		Data B0		CRC

Address	Name	Datawidth	Description
0x00	DeviceAddress	U8	I2C Device address. The default address is 0x40
0x04	FILT_CTRL	U8	Filter setting for one conversion: Bit 0: Adaptive filter for P1 and P2 (on / off) Bit 1: Low pass filter for T, TOB1 and TOB2 (on / off)
0x08	LP_FILT	U8	LowpassFilter for P1 and P2 (High nibble, Bit 7 .. 4) (Filter is disabled if LowpassFilter = 0) The formula for the low pass filter is given as: $P_{n+1} = \frac{(2^{LowpassFilter} - 1) * P_{n-1} + P_n}{2^{LowpassFilter}}$ where: Pn+1: new filtered value, P_n: actual measured value, P_{n-1}: old filtered value
0xC0	Auto_SleepMode	U8	Value: 0x80: Normal mode, Sensor does not go to sleep after power-up. (default setting) Value: 0x00: Sensor will go directly to Sleep Mode after power-up (only active after transmitter has gone to Sleep Mode once).
0xC4	FallbackTime	U16	The time in milliseconds until sensor goes back to sleep mode if no valid request was sent (invalid command (i.e. wrong amount of bytes, unknown address, invalid CRC). (Minimum value is 20ms) (default value 500ms)
0xC8	Wakeup_SettleTime	U16	The time in milliseconds until a settled and filtered value will be available (process value fulfils the Tolerance (digital) in the Calibration Certificate). **The higher the Startup Settle Time, the better the Noise.** (default value 5)
0xCC	SMAFilter_Depth	U16	Max Value: 256 (recommended for sleep mode) This is directly the Window Length of the SMA (Simple moving average Filter). The filtering is depending on the Sampling Rate, which is around 2.5kSPS. Max. time of Filter Window: 1/2500*256 = 102.4ms. When the process value is requested before i.e. 100ms (with a filter depth of 256), the filtering was done with the measurements collected up to this point (less filtering) (default value 0)

6.4 Block 2: Gain / Offset

The default value for all offsets is 0 and for all gains the default is 1.0. All values are stored as single float.
The offset of P1 and P2 is in bar.
The offset of TOB1, TOB2 and T the offset is in degree Celsius.

6.5 Block 3: Information values

The minimum and maximum values are stored as single precision float with the pressure unit in bar and temperature degree in Celsius.
Integer values are transmitted highest byte(B3) first and lowest byte(B0) at the end.

Address	Name	Datawidth	Description
0x00	P1 Min	Float32	Minimum Pressure of P1 in [bar]
0x04	P1 Max	Float32	Maximum Pressure of P1 in [bar]
0x08	TOB1 Min	Float32	Minimum Temperature of P1 in [°C]
0x0C	TOB1 Max	Float32	Maximum Temperature of P1 in [°C]
0x10	P2 Min	Float32	Minimum Pressure of P2 in [bar]
0x14	P2 Max	Float32	Maximum Pressure of P2 in [bar]
0x18	TOB2 Min	Float32	Minimum Temperature of P2 in [°C]
0x1C	TOB2 Max	Float32	Maximum Temperature of P2 in [°C]
0x20	T Min	Float32	Minimum Temperature of PT1000 in [°C]
0x24	T Max	Float32	Maximum Temperature of PT1000 in [°C]
0x30	Serial_Number	U32	Stored as 32bit integer value
0x34	FW_Version	U32	Byte 3: class Byte 2: group Byte 1: year Byte 0: week
0x38	Calibration_Date	U32	Byte 3: day Byte 2: month Byte 1:0: year
0x3C	Sensor_Type	U8	Byte 0 low nibble P1 and high nibble P2 0x0: PR (relative) 0x1: PA (absolute with 1 bar as reference) 0x2: PAA (absolute with 0 bar as reference) 0xF: Channel not configured i.e.: only P1 PAA → 0xF2
0x40	I2CInt_Vers	Float32	I2C version as float (i.e. 1.0)
0x44	CFG_Channel	U16	Byte 1: CFG_P available pressure channels: Bit 1: P1 Bit 2: P2 Byte 0: CFG_T available temperature channels: Bit 3: T (Temperature sensor) Bit 4: TOB1 (Temperature of pressure sensor P1) Bit 5: TOB2 (Temperature of pressure sensor P2)

7 Appendix

7.1 Floating-point format IEEE754

Interpretation of the Data inside the frame:

State		StateP T		Data B3		Data B2		Data B1		Data B0		CRC

Representation in accordance with IEEE754:

B3 DATA H (Reg. 0)	**B2** DATA L (Reg. 0)	**B1** DATA H (Reg. 1)	**B0** DATA L (Reg. 1)	
b01000001 (0x41)	b00101001 (0x29)	b00000010 (0x02)	b11011110 (0xDE)	Valid Number
b01111111 (0x7F)	b10000000 (0x80)	b00000000 (0x00)	b00000000 (0x00)	∞ / Overflow
b11111111 (0xFF)	b10000000 (0x80)	b00000000 (0x00)	b00000000 (0x00)	-∞ / Underflow
bx1111111 (0xFF)	b11111111 (0xFF)	b11111111 (0xFF)	b11111111 (0xFF)	Not a Number

1 bit Sign + 8 bit Exponent + 23 bit Mantis = 32 bit

Calculation of the value transmitted (single precision float):

$$V = (-1)^S \cdot (1.0 + \frac{M}{2^{23}}) \cdot 2^{E-127}$$

0 = 0
10000010 = 130
01010010000001011011110 = 2687710

$-1^0 * (1.0 + 2687710/8388608) * 2^{130-127} = 10.5631999969482421875$

These values directly show the value in the requested unit [bar] or [°C] -> 10.5632 bar

7.2 Signed32 data format

Interpretation of the Data inside the frame:

State		StateP T		Data B3		Data B2		Data B1		Data B0		CRC

B3 DATA H (Reg. 0)	**B2** DATA L (Reg. 0)	**B1** DATA H (Reg. 1)	**B0** DATA L (Reg. 1)	
b01000001 (0x00)	(0x00)	(0x2a)	(0x76)	Valid Number
b01111111 (0x7F)	(0xFF)	(0xFF)	(0xFF)	∞ / Overflow
b11111111 (0x80)	(0x00)	(0x00)	(0x00)	-∞ / Underflow
bx1111111 (0x7F)	(0xFF)	(0xFF)	(0xFF)	Not a Number

1 Sign Bit

When Sign-Bit is not set (positive value)
Data-Value = 0x00002a76 -> 10870

When Sign-Bit would be set Data-Value **0xFFFFD58A**:

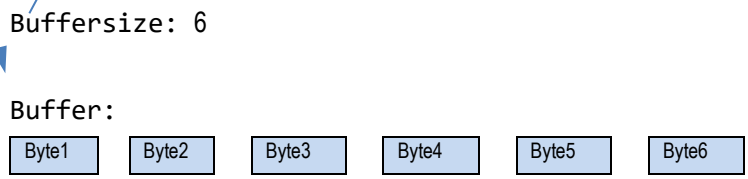
1. Invert Bitwise the hex-value -> NOT(0xFFFFD58A) = 0x 0000 2A75
2. Subtract 1 from the value -> 0x 0000 2A74 = 10870
3. Add Minus-Sign because the Sign-Bit was set: **-10870**

These values directly show the value in the requested unit [Pa] or [0.01°C] -> 10870Pa

7.3 CRC8 Calculation

An 8Bit CRC is used with the polynomial x^8 + x^2 + x + 1
The c-function on the transmitter looks like:

```
volatile uint8_t Crc8(uint8_t* Buffer, uint8_t Buffersize)
{
  uint16_t data;
  uint16_t crc = 0;
  uint8_t i, j;
  for (j = 0; j < Buffersize; j++)
  {
    data = (uint16_t)*(Buffer+j);
    crc ^= (data << 8);
    for(i = 0; i < 8; i++)
    {
      if (crc & 0x8000)
        crc ^= 0x8380;
      crc = crc << 1;
    }
  }
  return (uint8_t)(crc >> 8);
}
```



7.4 Example

Read pressure as Float-Value
Transmitter address: 0x40 (default)

Device Adr | W ACK	Byte Amount (7...3) Block Nr (2...0) ACK	Address ACK	CRC ACK
0x80	0x20	0x00	0xAE

Device Adr | R ACK	State ACK	StPT ACK	Data ACK	CRC NACK
0x81	0x10	0x00	0x3D 0xDE 0xE3 0x1D	0xE5

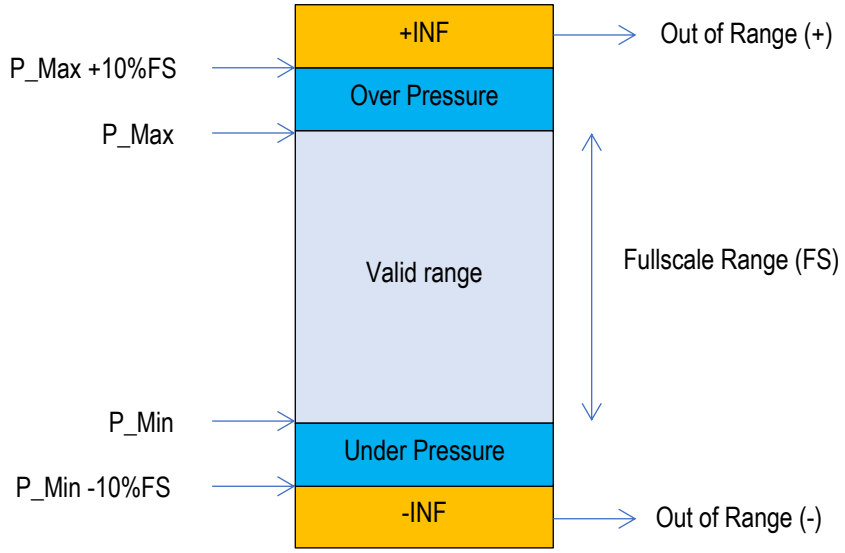
Data-Value = 0x3D 0xDE 0xE3 0x1D -> 108.83162mBar

7.5 Error handling and recognition

The electronic unit can read five physical measures: pressure values from two pressure sensors (**P1, P2**), temperature values from both pressure sensors (**TOB1, TOB2**) and temperature value from one additional temperature sensor (**T**). Additionally, out of these values, other values can be calculated (**CH0**). These values are described as **channels** in this documentation.
The active channels are shown in CFG_Channel.

Measuring range

The signals are being measured with an analogue to digital converter (ADC). The measuring range is limited upwards and downwards.
For the pressure signals P1 and P2 the limitation is depending on the compensated pressure range, which can be read out of block4: P Min / P Max.
Readable range: (pressure range minimum – 10%) up to (pressure range maximum + 10%)
Is the pressure below or above this range, then the bit in the Status-Byte will be set. The measured value itself is no longer valid when the Status-Bit is set.



Special Integer / float Values

	32bit signed Integer	float	Reason
NaN	2147483644 (0x7FFFFFFF)	(0xFFFFFFFF)	No measurement (i.e. HW-Failure or Powerup)
+INF	2147483640 (0x7FFFFFFF)	(0x7F800000)	More than 10%FS over the measuring range
-INF	-2147483640 (0x80000000)	(0xFF800000)	More than 10%FS under the measuring range

7.6 Firmware Versions

An overview of the released versions:

Revision Number (Index 0x17)	**Release Date**	**Major Changes**	**Sign**

5.24-23.25	June 2023	Initial Version, used for Prototypes only
		Known Bugs:
		• Fallback to Sleep not working after
		  ○ Power-Up
		  ○ Request but reading the response
		• Uint32 Data-Formats not working
		• Max. Response-Time is 12ms (will be 8ms later on)
			chg

5.24-23.44	November 2023	Bug fix	chg

7.7 Related Links

- Product page of X-series
  www.keller-pressure.com

- I2C Specifications from NXP
  https://www.nxp.com/docs/en/user-guide/UM10204.pdf

- Single precision float (IEEE 754) converter
  https://www.h-schmidt.net/FloatConverter/IEEE754.html

7.8 Contact Information

KELLER Druckmesstechnik AG
St. Gallerstrasse 119
8404 Winterthur
Switzerland
Phone: +41 52 235 25 25
Email: info@keller-pressure.com

7.9 Document History

Date	Version	Description	Sign
03.07.2023	1.2	Initial Version	S. Mazenauer / Ch. Gysel
16.11.2023	1.3	Changed write-time from 8ms to 30ms (worst case).	Ch. Gysel
05.02.2024	1.4	Corrected Pressure state Block3 Register 0x3C and 0x44 and PTState bit 7 startup	Ch. Gysel
22.04.2024	1.5	Display of Sensor Type changed from two byte to one byte (high and low nibble), corresponding to KELLER Modbus. Special integer values adapted to firmware.	S. Mazenauer
04.06.2025	1.6	-Clarification about byte amount (only the data bytes counted) -Explication about Block 1 reading and writing	S. Mazenauer