GeoPrecision GmbH
Am Dickhäuterplatz 8
D-76275 Ettlingen
Germany

T +49(0)7243-924112-0
F +49(0)7243-924112-9
E info@geoprecision.de

www.geoprecision.de
www.geo-precision.com

Two-Wire to SDI12 Converter

Manual

Table of Contents

1. Version History 3
2. Introduction 4
3. Specification 5
4. Connection 6
5. Communication 7
   5.1 Basic SDI12 commands 7
   5.2 Example commands for measure 9
   5.3 Example Configuration of a Datalogger 10
6. Extended SDI12 X-Commands 11
   6.1 Overview 12
   6.2 “Basic” sensor to channel assignment 13
   6.3 “Position related” sensor to channel assignment for Thermistor-strings 14
   6.4 Example of positioning a Thermistorstring 16
   6.5 Table of sensor-types 17
   6.6 Offset and Factor 17
   6.7 Enable and Disable a channel 18
   6.8 Select I2C sensor type 18
7. I2C Sensor 20
   7.1 Keller-LD sensor 20
   7.2 Pewatron sensor 20
8. Known Issues 21

1. Version History

21.09.18: Initial

17.12.19: Complete rework for new firmware V1.1

15.07.20: Updated chapter “5 Communication” for Firmware 1v2.

29.10.20: Added chapter “6.8 Select I2C sensor type” and “7.2 Pewatron sensor”.

2. Introduction

The “Two-Wire to SDI12 Converter” is a configurable universal interface to connect up to 48 “Two-Wire sensors” to your SDI12 infrastructure. The converter acts as a SDI12-Sensor (slave) and can be connected to every compatible SDI12-recorder or host-interface.

This document guides through the required steps of

- Connecting the sensors.
- Connecting the SDI12-cable to a host.
- Configuration and communication via SDI12-commands.

➔ Delivery of the converter in combination with a **Thermistorstring**:
In this case the device **is already configured** for correct operation. Please refer directly to the chapters “Connection” and the measurement-command (and examples) at “List of SDI-commands”.

A special version of this device allows to interface one I2C-based sensor. A special factory-programmed firmware is required!

3. Specification

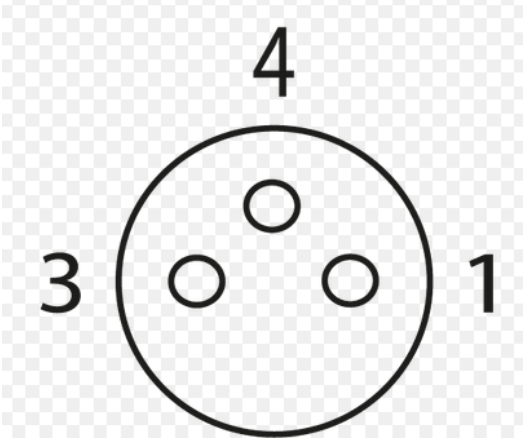
- SDI-12 Standard (see http://www.sdi-12.org for more information)
- 6V … 14V supply voltage
- Power consumption during measurement: >60 mA
- Power consumption idle: 2 mA
- Power up wait time for first command: 500 ms
- Overvoltage protection by TVS-Surge Absorber 400 Watt
- Up to 48 Two-Wire Sensors can be connected to one Interface
- Optional: Customer-defined interface for I2C-based sensor
- IP69 enclosure
- Operating Temperature Range: - 40°C … + 85°C

4. Connection

Two-Wire sensors:

Electrical interface: M8 industrial 3 pole connector.

The following schematic shows the pin-usage at sensor-side (cable with male plug).



Brown (1): Data

Black (4): Ground (GND)

Blue (3): not connected

The connection “Two-Wire-Sensors” follows the “BUS-topology”. Each sensor is electrically connected to the same Data- and GND-line!

SDI12-connection:

Brown: Supply (8 to 14 Volt, >60 mA cont.)

Black: GND

Blue: Data

I2C sensors:

Defined by customer.

5. Communication

In case of a preconfigured converter (in combination with a Thermistorstring) directly use the M- and D-commands to start and read a measurement.
Also have a look at the example-code for measurement.

→The converter must stay powered until all operations are finished. In case of a power loss (e.g. between the M- and the D-command) you have to repeat the whole procedure.
→A **power up wait time** of at least 800 ms is required!

5.1 Basic SDI12 commands

The command set is based on extended SDI12 (V1.1) command set.
→'**a**' represents the SDI-address, this might also be '?' (as wild card). Default SDI-address: '**1**'.

aAn!
Change address from 'a' to 'n'.
aI!
Identify Node.
aM! / aM0!
Start measure of **all** configured sensors. All values are stored at the internal cache. This must be always the first "initial" M-command!
Reply: **atttn**. 'ttt' seconds to wait till measurement done (or service reply will be send first) for 'n' values (up to 9).
aDn!
This will read the values from the previous M-command. With n = 0 to 9 to get the number of values announced by the previous M-command.

➔ Before requesting the data you have to wait for the service-request send by the converter or n seconds, replied by the previous M-command.

aMn!

Prepare the next set of values. ‘n’ must be between 1 and 9.

➔ Note: The required number of M-commands depends on the connected number of sensors. So keep in mind: Each M-command responds max 9 values. E.g. a thermistor-string containing 13 sensors requires the M0 and M1 command.

Error codes:

-98.00 : communication error or sensor missing

-99.00 : channel not activated.

-353: Could not read sensor.

5.2 Example commands for measure

The following commands and responds will demonstrate how to measure and read values from the converter. The output depends on the connected and configured Two-Wire sensors.

→At the following example the converter is configured to measure a thermistor-string containing **13** sensors (values).

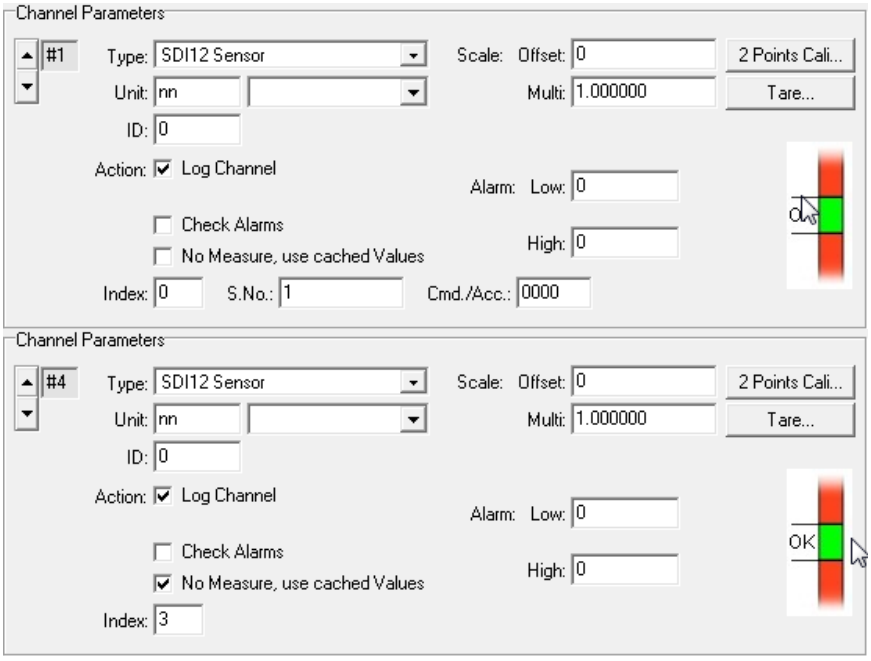
“>>” marks the command to the sensor, “<<” is the response. Each Command and response ends with <CR><LF>. Default address “1” is used.

```
>>   "1M0!"
<<   "10089"
<<   "1"
>>   "1D0!"
<<   “1[Value 0-2]”
>>   "1D1!"
<<   “1[Value 3-5]”
>>   "1D2!"
<<   “1[Value 6-8]”
>>   "1M1!"
<<   "10014"
<<   "1"
>>   "1D0!"
<<   “1[Value 9-11]”
>>   "1D1!"
<<   “1[Value 12]”
```

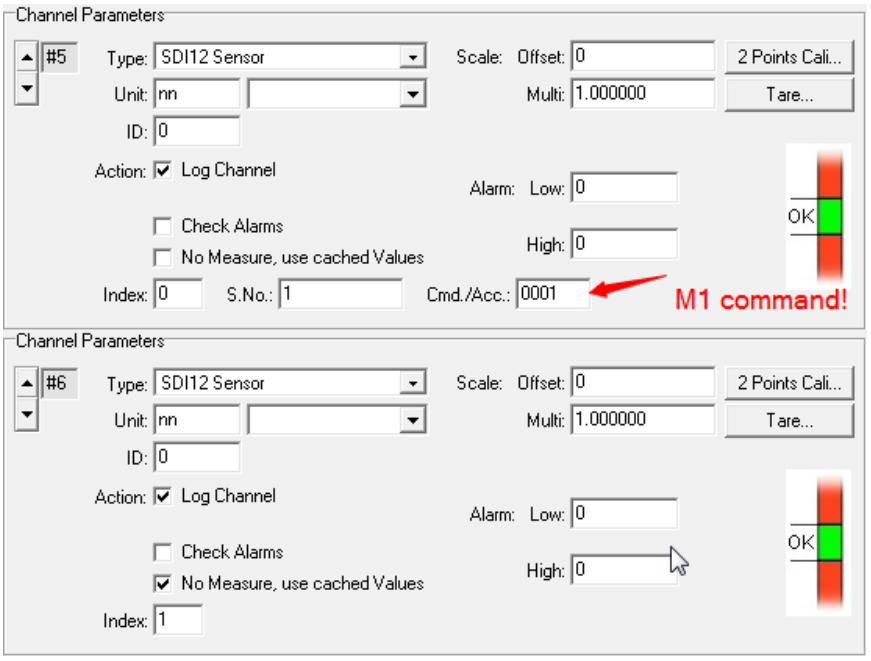
5.3 Example Configuration of a Datalogger

These steps will show you how to configure a GeoPrecision SDI12-Datalogger for the measurement of a thermistor-string with more than one “M-command”.
The M1, M2, M3, ... command are necessary for a converter configured for more than **four** 2W-sensors.
The screenshots are taken from the “FG2-Shell”, showing the configuration of a datalogger for 6 2W-Sensors connected to the converter.

Channel 1 to 4



Channel 5+6



6. Extended SDI12 X-Commands

To configure the converter for correct measurement and output of the connected Two-Wire sensors, a sequence of special not-SDI-conform X-command is used. The configuration is stored in non-volatile memory, so it will resist a power loss of the device. Changing sensors requires a reconfiguration!

→Please note: A converter delivered with a connected sensor/thermistorstring **is already configured** for correct operation

To apply SDI12-Commands to the converter you have to connect it to a power-supply and a PC-Interface for your preferred SDI12-Terminal (e.g. **SDI Win** or **SDI Term**). Or your data-recorder provides a command line to send SDI12 commands directly to its interface.
→It must be continuously powered during execution of all steps!

The extended commands are not compatible to the SDI12-specification. The detailed input- and output-format of each command is described within the examples below.

→For the following examples “>>” marks the command to the sensor, “<<” is the response. Each Command and response ends with <CR><LF>.
Default address “1” is used.

6.1 Overview

aXS!

Scan Two-Wire bus for the number connected sensors.

aXUn!

Read detailed information from the specific sensor ‘**n**’.

aXCk,n,t!

Allocate channel ‘**k**’ with sensor ‘**n**’ of type ‘**t**’.

aXKk=o,f!

Set a specific offset ‘**o**’ and factor ‘**f**’ for channel ‘**k**’.

aXAk=n!

Activate or deactivate channel ‘**k**’, setting ‘**n**’ 1 to activate, 0 to deactivate.

aXR! For factory-updates only!

aXP=t,z,e! Select I2C sensor type. For I2C firmware only!

6.2 “Basic” sensor to channel assignment

The following shows the “**basic**” assignment of a sensor to a channel. For **Thermistorstrings** a special, position related assignment is shown afterwards.

Step 1)

Scan Two-Wire bus for connected sensors.

>>**1XS!** <CR><LF>

<<**1nn** <CR><LF>

The response is the number of connected sensors (nn).

Step 2)

Assign a specific sensor to a channel of the converter.

>>**1XCk,n,t!**<CR><LF>

Allocate channel ‘**k**’ with sensor ‘**n**’ of type ‘**t**’. Select the type from “Table of sensor types” below.

→‘**nn**’ represents a specific sensor, counting from ‘1’ to the number of connected sensors (‘**nn’,** see Step 1).

<<**1kk,nn,ttt**<CR><LF>

Example:

Sensor 2 (e.g. a “TNode HD”, type 147) should be assigned to Channel 1 (first value of data output). The complete command for this operation:

>>	“1XC1,2,147!”
<<	“101,02,147”

Step 3)

Repeat Step 2 for all connected sensors by adjusting channel (k) and sensor number (n).

6.3 “Position related” sensor to channel assignment for Thermistor-strings

A Thermistorstring can contain up to 48 separate temperature-sensors. In most cases it is useful to assign the **first sensor** in the string to **channel 1** of the converter, **second sensor** to **channel 2** and so on.
The position of each sensor within the string is factory-programmed to the sensor itself.

A complete sample-code for a 3 sensor Thermistorstring is attached.

Step 1)
Scan Two-Wire bus for connected sensors.
>>**1XS!** <CR><LF>
<<**1nn** <CR><LF>
The response is the number of connected sensors (nn).

Step 2)
Read positioning information from one specific sensor.
>>**1XUn!**<CR><LF>
‘**n**’ represents a specific sensor, counting from ‘1’ to the number of connected sensors (see Step 1: ‘**nn’**).
<<1**I**nn,**A$**aaaa,**T**tttt,**P**pp<CR><LF>
‘**I**’: Followed by the sensor number (nn).
‘**A$**’: 2Wire address of the sensors (aaaa).
‘**T**’: Type of the sensor (ttt).
‘**P**’: Sensor position (pp). ‘-1’ is shown if the sensor has no position information.

Step 3)

Repeat Step 2 by incrementing ‘n’ at the request-command till the number of connected sensors is reached. Create a table with the sensor-number (n) and the related sensor-position (pp) and type e.g.:

Sensor Number (n)	Sensor Position (pp)	Type (t)
1	3	147
2	1	147
3	2	147

This example shows the sensor with the first position (pp) in the string is sensor-number 2 (n) of type (t) 147. The second position in the string is sensor-number 3, the third position is sensor-number 1.

Step 4)

Allocate channel ‘**k**’ with sensor ‘**n**’ of type ‘**t**’. For the type you have to enter the exact value shown as output ‘**T**’ at step 2!

Use the information from table above (step 3) to assign a channel to its corresponding sensor.

>>**1XCk,n,t!**<CR><LF>

<<**1kk,nn,ttt**<CR><LF>

Step 5)

Repeat Step 4 for each sensor and channel, like channel 1 (k=1) to sensor number 2 (n=2), channel 2 to sensor number 3 and channel 3 to sensor number 1.

6.4 Example of positioning a Thermistorstring

A Thermistorstring containing 3 sensors of type 140 (TNode EX) should be assigned to the corresponding channel numbers of the converter. This is the full code for all 3 sensors. Default address “1” is used.

```
>>   "1XS!"
<<   "103"
>>   “1XU1!”
<<   “1I01,A$3DA1,T147,P03”
>>   “1XU2!”
<<   “1I02,A$3D41, T147,P01”
>>   “1XU3!”
<<   “1I03,A$3AE1, T147,P02”
>>   “1XC1,2, 147!”
<<   “101,02, 147”
>>   “1XC2,3, 147!”
<<   “102,03, 147”
>>   “1XC3,1, 147!”
<<   “103,01, 147”
```

6.5 Table of sensor-types

Sensor-Type	Description
140	TNode EX V1.0
141	Counter 32Bit
142	Keller-LDS (Pres./Temp.)
143	Baro (Pres./Temp.)
144	SHT (%rh/Temp.)
145	TNode/EX V3.0
146	U1 Node
147	TNode EX/HD V2.0
148	TNode EX V1.1
149	Multi-Sense Wind speed/direction
150	Multi 0.7 sec
151	Counter 16Bit

6.6 Offset and Factor

Adjust the offset '**o**' and factor '**f**' for channel '**k**'.

>>**1XKk=o,f!** <CR><LF>

<<**1** <CR><LF>

You always have to set both, offset and factor, separated by a comma '**,**'. The decimal separator is dot '**.**'! Example:

```
>>   "1XK1=-0.25,2.54!"
<<   "1"
```

To read the configured offset '**o**' and factor '**f**' for channel '**k**' use:

>>**1XKk?!** <CR><LF>

<<**1=o,f** <CR><LF>

Example:

```
>>   "1XK1?!"
<<   "1=-0.25,+2.54"
```

6.7 Enable and Disable a channel

Deactivate or activate channel '**k**'. Replace '**n**' by '**0**' to de- and '**1**' to activate.

>>**1XAk=n!** <CR><LF>

<<**1** <CR><LF>

Read the status of channel '**k**' use:

>>**1XAk?!** <CR><LF>

<<**1=0**<CR><LF>

6.8 Select I2C sensor type

➔ This belongs to the I2C firmware since Version 1.3 (or higher) only!

Set up the converter to the sensor-type '**t**' with "Zeropoint" '**z**' and "Endpoint" '**e**'.

>> **1XP=t,z,e!** <CR><LF>

<<**1** <CR><LF>

't': '0' for Keller-LD and '1' for Pewatron KKD18.

'z': '0' for Keller-LD, for KKD18 refer to sensor datasheet for the "Zeropoint".

'e': '0' for Keller-LD, for KKD18 refer to sensor datasheet for the "Endpoint".

You always have to set all values, separated by a comma '**,**'. The decimal separator is dot '**.**'!

The following example sets the sensor type to KKD18 with 0 bar "Zeropoint" and 2 bar "Endpoint":

```
>>   "1XP=1,0,2!"
<<   "1"
```

For the Keller-LD sensor you can use:

```
>>   "1XP=0,0,0!"
<<   "1"
```

To read the configured sensor-type '**t**' with "Zeropoint" '**z**' and "Endpoint" '**e**' use:

>>**1XP?!** <CR><LF>

<<**1=t,z,e**<CR><LF>

Example for KKD18 sensor with 0 bar "Zeropoint" and 2 bar "Endpoint":

>>	"1XP?!"
<<	"1=+1.0000,+0.0000,+2.0000"

7. I2C Sensor

By ordering the device with a special, factory-programmed firmware, it is possible to convert the output of one I2C-sensor to SDI12 output.

- Connection and data output depends on the sensor-type.
- A configuration is required if the connected sensor is not the “Keller-LD”.
- To communicate with the sensor follow the steps at **5 Communication**.
- All extended SDI12 X-commands, except **6.6 Offset and Factor**, **6.7 Enable and Disable a channel** and **6.8 Select I2C sensor type** are deactivated!

7.1 Keller-LD sensor

This I2C sensor type is automatically selected after updating the firmware with I2C capability!

- Output mBar on channel 1 and °C on channel 2.
- Automatic reading of factory calibration from sensor.
- Output values are based on the calculation of the measurement result and the factory-calibration.

Error-Codes: -100,-100: No Sensor

7.2 Pewatron sensor

The Pewatron KKD18 is supported with firmware version 1.3 or higher. To use this type of sensor you have to configure the coefficients first! Follow the instructions at **6.8 Select I2C sensor type**.

- Output mBar on channel 1 and °C on channel 2.
- Output values are based on the calculation of the measurement result and the coefficients applied to the converter at **6.8 Select I2C sensor type**.

Error-Codes: -100,-100: No Sensor

8. Known Issues

I get no reply from my sensor connected to a PC via USB-Serial-Converter.
→Sometimes the delay of a USB converter is too high or it is blocked somehow. Shut down your PC. Disconnect the USB converter and the sensor, disconnect the sensor from power-source. Start your PC and connect everything properly. Now it should work!

The configuration via X-command fails or I get no reply to the X-command.
→Try to send the command again. These commands are not SDI-conform and very complex. So it might happen that the sensor or your PC did not receive everything correctly!
→Be sure the device is powered continuously for the whole sequence of commands. A power loss between two, e.g. the “XS” and “XC”, command will clear all previous actions and the second command fails!

I cannot measure the connected sensor. The ‘M’ command always replies with "1000".
→The sensor is not configured or the channel is deactivated. You have to set up each sensor the first time it is connected to the SDI-Converter!