

SDI 12 Master to Modbus Slave Converter

The TBS09DR is a converter to connect Modbus sensors to a SDI-12 master device such as a data logger or telemetry unit. It can connect multiple Modbus sensors in parallel by setting the corresponding address of the individual Modbus sensors using an extended SDI-12 command, upfront to issuing the measurement commands.

The TBS09DR provides a rich command set to offer maximum flexibility for configuring the device to a sensor with Modbus interface.



TBS09DR SDI 12 Master to Modbus Slave Converter

Features

- SDI-12 Master to Modbus Slave Converter
- Multiple sensors can be connected
- SDI-12 Standard V1.3
- Highly configurable
- Switched sensor supply voltage output
- 6 - 16V supply voltage
- 7mA current consumption when active

- Less than 100µA idle current
- Operating Temperature Range:
- 40°C ... + 80°C

Target Applications

- SDI-12 sensor networks

SDI 12 Master to Modbus Slave Converter

Contents

1	INTRODUCTION	3
2	CONNECTIONS	3
3	MEASUREMENT	5
3.1	GENERAL CONFIGURATION COMMANDS	5
3.2	MODBUS DEVICE ADDRESS MAPPING	7
3.3	MEASUREMENT COMMAND MAPPING	8
3.4	CONFIGURING MODBUS DATA FORMAT	9
3.5	CONFIGURING A FUNCTION CODE	10
3.6	INITIALIZE MODBUS DEVICE MEASUREMENT MODE	10
3.7	READ/WRITE ANY MODBUS INPUT/HOLDING REGISTERS	11
3.8	DEFAULT CONFIGURATION	11
4	SDI-12	12
5	SUPPORTED SDI-12 COMMANDS	12
6	EXAMPLE – CONTROLLING THE TQS3 TEMPERATURE SENSOR	16
7	ORDERING INFORMATION	17
8	HISTORY	17

Tables

Table 1 – Standard SDI-12 commands	13
Table 2 – Extended SDI-12 Commands	15
Table 3 – Ordering Information	17
Table 4 – History	17

Figures

Figure 1 – TBS09DR adding a Modbus interface to a telemetry system with SDI-12 interface	3
Figure 2 – TBS09DR terminals	3
Figure 4 – TBS09DR jumpers	4
Figure 5 – timing diagram; switched sensor supply mode	7

SDI 12 Master to Modbus Slave Converter

1 Introduction

The TBS09DR is a converter to connect one or multiple Modbus sensors to a SDI-12 master device such as a data logger or telemetry unit.

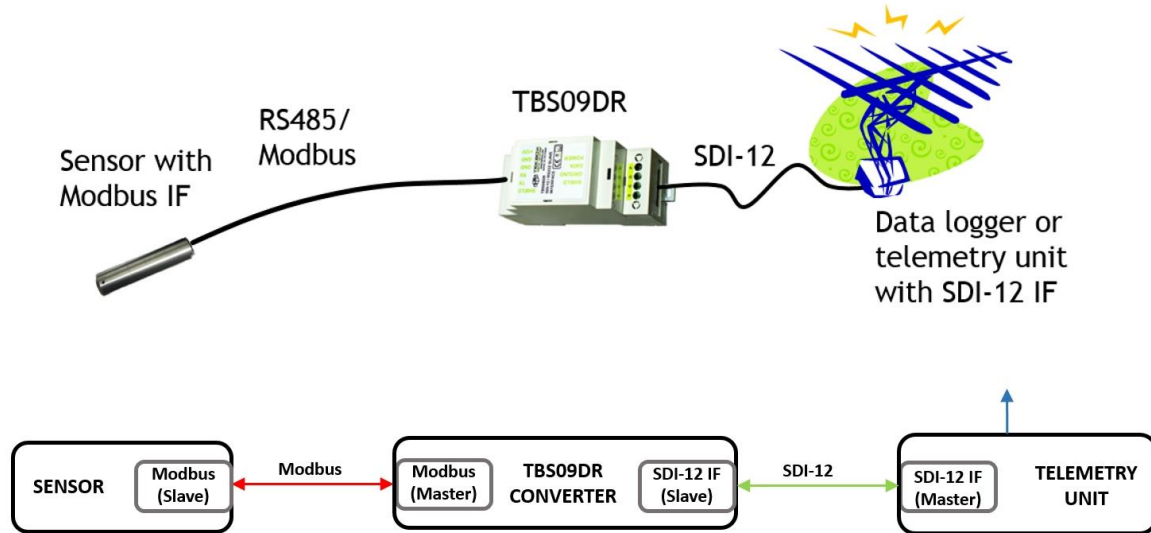


Figure 1 – TBS09DR adding a Modbus interface to a telemetry system with SDI-12 interface

2 Connections

RS485 Side, from left to right:

- TX+ output (or half duplex)
- TX- output (or half duplex)
- RX+ input
- RX- input
- Ground
- 12V, switched sensor supply voltage

SDI-12 Side, from left to right:

- Cable shield
- Ground
- SDI-12 Data line
- SDI-12 Supply voltage



Figure 2 – TBS09DR terminals

4 Pin terminal block:

SDI 12 Master to Modbus Slave Converter

CON1 – SDI-12 Interface

Shield: connect to the shield of the SDI-12 cable or leave it unconnected; shield and ground are internally connected together

Ground: connect to the GND wire of the SDI-12 cable

SDI-12 data: connect to the data wire of the SDI-12 cable

SDI-12 Power: connect to the positive supply voltage wire of the SDI-12 cable;

6 Pin terminal block:

CON2 – Power supply & RS485 (Modbus) interface

TX+, connect to RX+ of the Modbus Sensor; half duplex

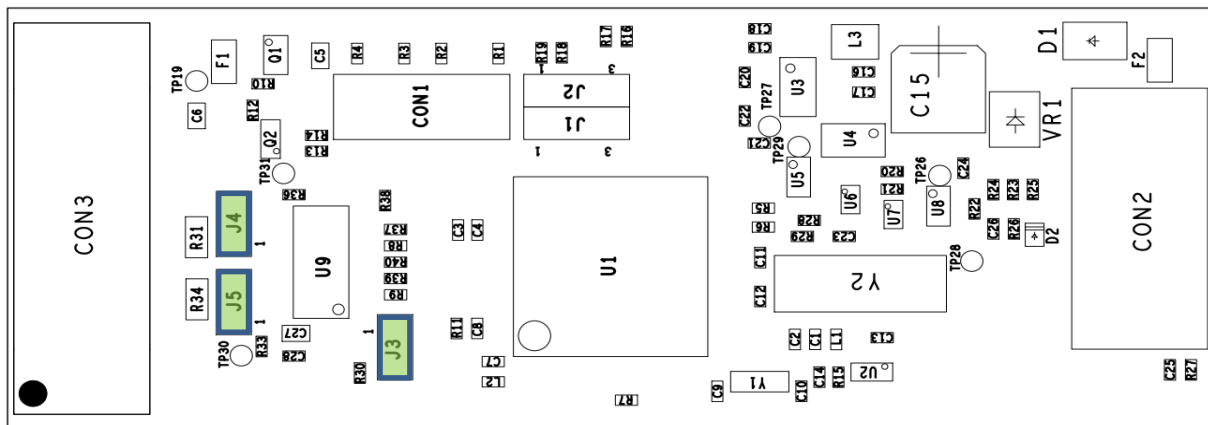
TX-, connect to RX- of the Modbus Sensor; half duplex

RX+, connect to TX+ of the Modbus Sensor

RX-, connect to RX- of the Modbus Sensor

Ground, connected to the GND of the SDI-12 supply for TBS09 DR

+12V, supply output for the Modbus sensor. It is connected to the SDI-12 supply line, with a high side FET switch and a 700mA fuse in between. The switch can either be controlled by SDI-12 (ON-time determined by the warm-up time setting) or it can be configured to be permanently on.



Default J4 and J5 jumpered - Modbus terminated with 120Ω
 Default J3 jumpered – half duplex

Figure 3 – TBS09DR jumpers

SDI 12 Master to Modbus Slave Converter

3 Measurement

3.1 General configuration commands

Modbus communication settings

In order to match Modbus devices with the TBS09, a few extended SDI-12 commands need to be issued.

Configuration of the Modbus data rate: **aXSB,BR,parity!**

Where the parameter **a** represents the SDI-12 address

BR max = 115200;

Parity: None = 0, Even =1, Odd = 2

Example: TBS09 SDI-12 address = 0; set Baud rate to 19200 and parity to None

Issue following string: **0XSB,19200,0!**

The settings are stored in none volatile memory until they are overwritten by another configuration.

Use following command to query the Modbus communication settings: **aXGB!**

Where the parameter **a** represents the SDI-12 address. The TBS09 will respond with the Modbus baud rate and parity settings then.

Note: data bits = 8, stopbits = 1

Modbus device warm-up time

One of the characteristics of SDI-12 sensor networks is its low power capability. Whenever idle, the TBS09 will be in sleep mode. The TBS09 will only wakeup upon a measurement command, then power on the Modbus device, wait for a configurable warm up time, initiate a measurement, wait for the measurement response time, transmit the measurement result to the SDI-12 master and go into sleep mode again.

The necessary warmup time depends on the connected Modbus device. It must be configured long enough to ensure that the Modbus device got sufficient time for booting and settling its measurement circuitry to deliver a an accurate result.

Extended SDI-12 command to set warm-up time: **aXSMBW,t!**

Where the parameter **a** represents the SDI-12 address and **t** represents the warm-up time in seconds.

t_{min} = 1 second

t_{max} = 100 seconds

The measurement takes place after the warm-up time passed. This means that after a measurement command, the sensor will be powered for the warm-up time and the measurement response time, before the measurement parameter is read from the Modbus slave.

Example: set the warm-up time to 10 seconds.

Issue following string: **0XSMBW,10!**

The sensor will be powered upon a measurement command and the measurement value will be queried from the sensor after 10 seconds + response time.

The settings are stored in none volatile memory until they are overwritten by another configuration.

SDI 12 Master to Modbus Slave Converter

Use following command to query the warm-up time setting: **aXGMBW!**

Where the parameter **a** represents the SDI-12 address. The TBS09 will respond with the warm-up time then.

Modbus device measurement response time

The response time is the time between initiating a measurement of the Modbus device time and reading the measurement parameter from the register.

Extended SDI-12 command to set the sensor response time: **aXSMBT,n,t!**

Where the parameter **a** represents the SDI-12 address, **n** represents the measurement command and **t** represents the response time of the corresponding SDI-12 command in seconds

n = 0 to 9 corresponds to the measurement commands aM! to aM9!

t corresponds to the response time

t_{min} = 1 second

t_{max} = 999 seconds

Extended SDI-12 command to set the sensor response time: **aXGMBT,n!**

Where the parameter **a** represents the SDI-12 address and **n** represents the measurement command

n = 0 to 9 corresponds to the measurement commands aM0 to aM9!

SDI-12 measurement response time

The SDI-12 response time cannot be set with a dedicated command. It is the sum of Modbus device warm-up time and Modbus device measurement time.

Example: if the Modbus device warm-up time is set to 10 seconds and the Modbus device measurement response time is set to 1 second, the resulting SDI-12 response time will be 11 seconds.

Permanent supply / switched supply

The TBS09 provides a supply output for the Modbus sensor which can be switched ON/OFF according to the warm-up time settings or the output can be switched permanently on.

When first setting up or testing a Modbus sensor, it is recommended to keep the Modbus supply permanently on, or in case that the SDI-12 supply voltage is already switched by the SDI-12 data logger or telemetry unit, it is recommended to supply the sensor separately.

Once the configuration is validated and the sensor is delivering data, you can change to switched supply and figure out a suitable setting for the warm-up time.

Permanent supply ON/OFF: **aXSMVS,p!**

Where the parameter **a** represents the SDI-12 address

p = 0 ON/OFF according to warm-up time

p = 1 for 12V supply output permanently switched to ON

SDI 12 Master to Modbus Slave Converter

Timing diagram

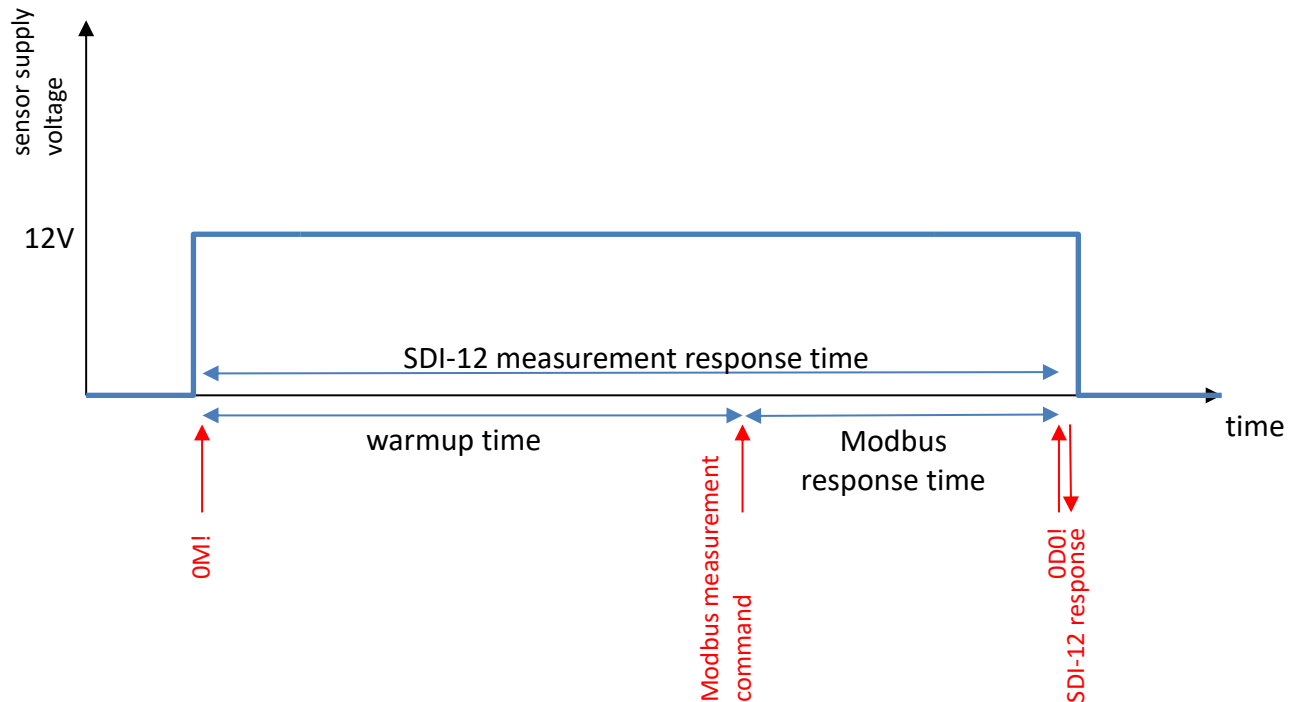


Figure 4 – timing diagram; switched sensor supply mode

The Modbus measurement command is triggered internally upon issuing a SDI-12 measurement command such as for example **OM!**. After the sensor is warmed up, the TBS09 issues a Modbus measurement command. After the SDI-12 measurement response time passed, the SDI-12 master needs to retrieve the data by issuing a **ODO!** command.

3.2 Modbus device address mapping

Both SDI-12 and Modbus are networks where sensors share the same physical bus. Every sensor needs to be configured to an individual address. The SDI-12 to Modbus converter appears like a single sensor connected to the SDI-12 bus and consequently can only have a single address to be compliant with the SDI-12 standard. Direct mapping between a SDI-12 sensor address and a Modbus sensor address would restrict the Modbus interface of the TBS09 to a single Modbus sensor address. Consequently a workaround was implemented to allow addressing of any Modbus sensor using only a single SDI-12 address.

Assume we configure the TBS09 to SDI-12 address "**0**". Assume, that we connect 3 sensors - **A**, **B**, **C** – to the Modbus interface. The Modbus device addresses are from within the range 1 to 247. However most devices are shipped with a default address of "**1**". In order to connect the three devices to a single physical Modbus, we have to assign individual addresses first. This may be done by setting DIP switches on the device hardware or by using software tools provided by the Modbus device manufacturers. Refer to the respective Modbus device manuals.

Let's assume that each connected Modbus device got its individual address, e.g. **A**: 110, **B**: 120, **C**: 130. In order to address this devices through a single SDI-12 address, we simply make use of extended SDI-12 commands.

SDI 12 Master to Modbus Slave Converter

Command string to map a Modbus device address: **aXSMBA,b!**

Where the parameter **a** represents the SDI-12 address and **b** represents the Modbus address.

Range for **b**: 1 to 247

Example: assume the SDI-12 address is “0” and the Modbus device address is “110”

Issue extended SDI-12 command **0XSMBA,110!**

Any consecutive Modbus data traffic with the Modbus sensor will use device address “110”

If we next want to communicate with Modbus sensor “B” with device address “120”, we simply need to issue the extended command to set the Modbus address to 120: **0XSMBA,120!**

Any consecutive Modbus data traffic with the Modbus sensor will then use device address “120”

We proceed similarly with any other Modbus device connected to the TBS09. After setting the Modbus address, it will be stored in non-volatile memory and remain valid until a new address is set.

In order to query the Modbus device address mapping, issue following string: **aXGMBA!**

Where the parameter **a** represents the SDI-12 address. The TBS09 will respond with the Modbus address then.

3.3 Measurement command mapping

The SDI-12 protocol specifies a set of measurement commands **aM!**, **aM0!**, **aM1!** ... **aM9!**

Each command can be used to read a different parameter, or a different ensemble of parameters. Upon issuing the measurement command by a SDI-12 master, the TBS09 will respond with the time required to carry out the measurement (equivalent to the warm-up time specified in chapter 3.1) and the number of parameters measured (always a single parameter in case of the TBS09).

The SDI-12 master then needs to wait for the warmup time and then issue the string **aD0!** to trigger the TBS09 to transmit the measured parameter.

On the Modbus device side, carrying out a measurement is equivalent to reading a register which contains the measurement result. Consequently the SDI-12 measurement commands need to be mapped with the corresponding Modbus device register addresses which hold the measurement parameters.

Use following extended SDI-12 command string to map SDI-12 measurement commands with the corresponding Modbus device input register: **aXSMBM,n,b!**

Where the parameter **a** represents the SDI-12 address, **n** represents the index of the SDI-12 measurement command and **b** represents the input register address of the Modbus device.

n: 0 to 9 for **aM!**, **aM1!** ... **aM9!**

b: 0 to 65535

Example: **0XSMBM,0,1!**

Issuing the string **aM!** will initialize reading the data content from Modbus input register address 1; after sending **aM!**, the SDI-12 master needs to wait for the warm-up time and then send **aD0!** to trigger the TBS09 to transmit the measured parameter.

Example: **0XSMBM,9,12345!**

aM9! Is configured to read data from Modbus device register address 12345

After configuring the measurement command mapping, it will be stored in non-volatile memory and remain valid until a new configuration is set. A new configuration may be necessary when addressing another Modbus device connected to the TBS09. As different Modbus devices may have their measurement parameter stored at

SDI 12 Master to Modbus Slave Converter

different register addresses, not only the address mapping needs to be updated, but also the measurement command mapping procedure needs to be carried out accordingly.

In order to query the Modbus measurement command mapping, issue following string: **aXGMBM,n!**
Where the parameter **a** represents the SDI-12 address and **n** = 0 ...9 for **aM!**, **aM1!** ... **aM9!**. The TBS09 will respond with the corresponding Modbus register address then.

3.4 Configuring Modbus data format

Data type

Modbus parameters may be stored in various data formats. In order to decode it correctly, the Modbus data type needs to be configured upfront to any measurement.

Set Modbus data type: **aXSMBD,t!**

Where the parameter **a** represents the SDI-12 address and **t**: **0** = floating, **1** = unsigned integer, **2** = signed integer

When t is set to floating, the TBS09 will read the contents of two registers with consecutive address. The address mapping command will refer to the lower address.

Example: set data type to floating: **0XSMBD,0!**
 set register address to 0: **0XSMBM,0,0!**

Upon issuing the **0M!** / **0D0!** commands, the TBS09 will read 4 bytes of sensor data; two bytes from register address 0 and another two bytes from register address 1

E.g. Modbus device parameter data content is: 3fe0,0000. The TBS09 will then translate it to the value of 1.75

Example: set data type to unsigned integer: **0XSMBD,1!**
 set register address to 5: **0XSMBM,1,5!**

Upon issuing the **0M1!** / **0D0!** commands, the TBS09 will read 2 bytes of sensor data from register address 5

E.g. Modbus device parameter data content is: 000a. The TBS09 will then translate it to the value of 10

Example: set data type to signed integer: **0XSMBD,2!**
 set register address to 9: **0XSMBM,2,9!**

Upon issuing the **0M2!** / **0D0!** commands, the TBS09 will read 2 bytes of sensor data from register address 9

E.g. Modbus device parameter data content is: ff9c. The TBS09 will then translate it to the value of -100

Query Modbus data type: **aXGMBD!**

Where the parameter **a** represents the SDI-12 address.

SDI 12 Master to Modbus Slave Converter

Precision

According to the SDI-12 standard, the maximum length of an SDI-12 response string is 9 digits, with the decimal point anywhere. Default setting, when converting Modbus parameters into SDI-12 measurement result strings, is 3 digits. In case of integer results, the length can be reduced to avoid unnecessary zeros.

Set the length (number of digits) of the SDI-12 result: **aXSDP,l!**

Where the parameter **a** represents the SDI-12 address and **l** = 0 ... 9 the number of digits

Query SDI-12 result digits setting: **aXGDP!**

Scaling

In some cases it may be useful to scale the measurement result. A scaling factor can be multiplied with the Modbus parameter, before being delivered as measurement result over SDI-12. The default scaling factor is 1.

Set data scaling factor: **aXSMBS,f,f!**

Where the parameter **a** represents the SDI-12 address and **f.f** is the scaling factor. It is multiplied with the data from the Modbus device

Range for **f.f**:

The scaling factor is not retained in permanent memory so whenever TBS09DR is restarted, it is restored to its default value (1).

Query the data scaling factor: **aXGMBS!**

Where the parameter **a** represents the SDI-12 address

3.5 Configuring a function code

Function codes basically tell the addressed Modbus slave device what kind of action to perform. Refer to the Modbus device manual for details.

The TBS09 offers maximum versatility by being able to configure function codes via extended SDI-12 commands.

Set Modbus device function code: **aXSMBF,f!**

Where the parameter **a** represents the SDI-12 address and **f** represents the function code (0x03: read holding registers, 0x04: read input registers...); refer to the manual of your Modbus sensor

Query the function code: **aXGMBF!**

3.6 Initialize Modbus device measurement mode

Whereas some Modbus sensors automatically start measuring after power on, other Modbus sensors may simply remain idle after power On. Means the TBS09 only is able to read data from mapped registers, if the sensor automatically measures after powering ON (such as the TQS3 temperature sensor...). Other sensors need to be initialized before issuing measurement commands over SDI12. Refer to the corresponding Modbus device manual.

SDI 12 Master to Modbus Slave Converter

This command starts the enable sequence to put the Modbus sensor into measurement mode: **aXSMSF,t!**

Where the parameter **a** represents the SDI-12 address and **t: 0** = disable, **1** = enable

This command completes the enable sequence to put the Modbus sensor into measurement mode:
aXSMSM,add,data!

Where the parameter **a** represents the SDI-12 address and **add:** register address and **data:** data need to write

After configuration, TBS09 will write **data** to register at **add** every measurement to start the sensor.

3.7 Read/write any Modbus input/holding registers

Read any input register or holding register: **aXGMBR,addr!**

Where the parameter **a** represents the SDI-12 address and **addr** is the input or holding register address which shall be read out. The extended command to read input/holding registers will read out a single register.

Use the extended command for function codes to switch between input and holding registers (0x04 <-> 0x03)

after issuing a read input/holding register command **aXGMBR,addr!**, wait 1 minute and transmit following extended SDI-12 command to retrieve the register content over SDI-12: **aXGMBRD!**

Example: read the content of the input register with address 130

0XSMBF,4!	configure the function code to read input registers (0x04)
0XGMBR,130!	read input data register, address 130
0XGMBRD!	retrieve the register content over SDI-12

3.8 Default configuration

The TBS09 is factory configured with all parameters set to a default value. See Table 2 – *Extended SDI-12 commands*. The default settings can be restored by issuing the command **aXSDF!**

Where the parameter **a** represents the SDI-12 address

SDI 12 Master to Modbus Slave Converter

4 SDI-12

SDI-12 is a standard for interfacing data recorders with microprocessor-based sensors. SDI-12 stands for serial/digital interface at 1200 baud. It can connect multiple sensors with a single data recorder on one cable. It supports up to 60 meter cable between a sensor and a data logger.

The SDI-12 standard is prepared by

**SDI-12 Support Group
(Technical Committee)
165 East 500 South
River Heights, Utah
435-752-4200
435-752-1691 (FAX)
<http://www.sdi-12.org>**

The latest standard is version V1.3 which dates from July 18th, 2005. The standard is available on the website of the SDI-12 Support Group.

More information on SDI-12 is presented in chapter 3.

5 Supported SDI-12 Commands

Following standard SDI-12 commands are supported:

Command	Description	Response
a!	Acknowledge Active	a<CR><LF>
a!	Send Identification	013TEKBOXVNTBS9MB0.1xxxxxx<CR><LF> With xxxxxx representing the serial number
aAb!	Change Address	b<CR><LF> Changing the sensor address from a to b
?!	Address Query	a<CR><LF>
aM!	Start Measurement Reads Modbus register mapped to aM!	att1<CR><LF> Delay (ttt = 010) in seconds and number of values (1)
aM1!	Additional Measurement Reads Modbus register mapped to aM1!	att1<CR><LF> Delay (ttt = 010) in seconds and number of values (1)
aM2! ... aM9!	Additional Measurement Reads Modbus register mapped to aM2! ... aM9!	att1<CR><LF> Delay (ttt = 001) in seconds and number of values (1)

SDI 12 Master to Modbus Slave Converter

aMC!	Start Measurement and request CRC Reads Modbus register mapped to aM! and calculates CRC	att1<CR><LF> Delay (ttt = 001) in seconds and number of values (1)
aMC1!	Additional Measurement and request CRC Reads Modbus register mapped to aM1! and calculates CRC	att1<CR><LF> Delay (ttt = 001) in seconds and number of values (1)
aMC2! ... aMC9!	Additional Measurement and request CRC Reads Modbus register mapped to aM2! ...aM9! and calculates CRC	att1<CR><LF> Delay (ttt = 001) in seconds and number of values (1)
aC!	Start Concurrent Measurement Reads Modbus register mapped to aM1!	att1<CR><LF> Delay (ttt) in seconds and number of values (4)
aC1!	Start Concurrent Measurement Reads Modbus register mapped to aM1!	att1<CR><LF> Delay (ttt) in seconds and number of values (4)
aC2! ... aC9!	Start Concurrent Measurement Reads Modbus register mapped to aM2! ...aM9!	att1<CR><LF> Delay (ttt) in seconds and number of values (4)
aCC!	Start Concurrent Measurement and request CRC Reads Modbus register mapped to aM! and calculates CRC	att1<CR><LF> Delay (ttt) in seconds and number of values (4)
aCC1!	Start Concurrent Measurement and request CRC Reads Modbus register mapped to aM1! and calculates CRC	att1<CR><LF> Delay (ttt) in seconds and number of values (4)
aCC2! ... aCC9!	Start Concurrent Measurement and request CRC Reads Modbus register mapped to aM2! ...aM9! and calculates CRC	att1<CR><LF> Delay (ttt) in seconds and number of values (4)
aD0!	Get Measurement Result(s)	Upon issuing the aD0! Command, the TBS09 will send the measurement results. The response format depends on the measurement command and device settings issued before.
aV!	Start Verification	a0000<CR><LF> Not supported
aRn! aRCn!	Continuous Measurement Continuous Measurement + CRC	a<CR><LF> Not supported

Table 1 – Standard SDI-12 commands

Following extended SDI-12 commands are supported by the TBS09:

Command	Description	Response
---------	-------------	----------

SDI 12 Master to Modbus Slave Converter

aXSB,BR,Parity!	Configuration of the Modbus data rate a represents the SDI-12 address BR max = 115200; Parity: None = 0, Even =1, Odd = 2	aX_ok<CR><LF>
aXGB!	Query the Modbus communication settings	a,BR,parity<CR><LF>
aXSMBW,t!	Configuration of the warm-up time a represents the SDI-12 address t represents the warm-up time in seconds t_{min} = 1 second t_{max} = 100 seconds	aX_ok<CR><LF>
aXGMBW!	query the warm-up time	a,t<CR><LF>
aXSMBT,n,t!	Extended SDI-12 command to set the sensor response time a represents the SDI-12 address n = 0 to 9 corresponds to the measurement commands aM! to aM9! t corresponds to the response time t_{min} = 1 second t_{max} = 999 seconds	aX_ok<CR><LF>
aXGMBT,n!	Extended SDI-12 command to set the sensor response time a represents the SDI-12 address n = 0 to 9 corresponds to the measurement commands aM! to aM9!	a,t<CR><LF>
aXSMVS,p!	Permanent supply ON/OFF a represents the SDI-12 address p = 0 ON/OFF according to warm-up time; p = 1 for 12V supply output always ON	a,t<CR><LF>
aXSMBa,b!	Command string to map a Modbus device address a represents the SDI-12 address b represents the Modbus address. range for b : 1 to 247	aX_ok<CR><LF>
aXGMBa!	query the Modbus device address mapping	a,b<CR><LF>
aXSMBM,n,b!	Command string to map SDI-12 measurement commands with the corresponding Modbus device input register a represents the SDI-12 address n represents the index of the SDI-12 measurement command b represents the input register address of the Modbus device n = 0 ...9 for aM! , aM1! ... aM9! range of b : 0 to 65535	aX_ok<CR><LF>
aXGMBM,n!	Command to query the Modbus measurement command mapping a represents the SDI-12 address n represents the index of the SDI-12 measurement command b represents the input register address of the Modbus device n = 0 ...9 for aM! , aM1! ... aM9! range of b : 0 to 65535	a,b<CR><LF>
aXSMBD,t!	Set Modbus data type a represents the SDI-12 address t : 0 = floating, 1 = unsigned integer, 2 = signed integer	aX_ok<CR><LF>
aXGMBD!	Query modbus data type a represents the SDI-12 address	a,b<CR><LF>
aXSDP,l!	Set the length of the SDI-12 result (number of digits) a represents the SDI-12 address l = 0 to 9, length (number of digits)	aX_ok<CR><LF>
aXGDP!	Query SDI-12 result length	a,l<CR><LF>
aXSMBS,f,f!	Set data scaling factor (multiplier) a represents the SDI-12 address	aX_ok<CR><LF>

SDI 12 Master to Modbus Slave Converter

	f.f is the scaling factor. It is multiplied with the data from the Modbus device range of f.f :	
aXGMBS!	Query the data scaling factor a represents the SDI-12 address	a,f<CR><LF>
aXSMBF,f!	Set Modbus device function code a represents the SDI-12 address f represents the function code (0x03: read holding registers, 0x04: read input registers...); refer to the manual of your Modbus sensor	aX_ok<CR><LF>
aXGMBF!	Query the Modbus function code	a,f<CR><LF>
aXGMBR,addr!	Read any input register or holding register a represents the SDI-12 address and addr is the input or holding register address	aX_ok<CR><LF>
aXGMBRD!	wait 1 minute after issuing the input register or holding register command and transmit this extended SDI-12 command to retrieve the register content over SDI-12:	a,register data<CR><LF>
aXSMSF,f!	Step1 for measurement initialisation a represents the SDI-12 address f = 0: disable, 1 enable	a,register data<CR><LF>
aXSMSM,addr,data!	Step2 for measurement initialisation a represents the SDI-12 address addr is register address data is start value	a,register data<CR><LF>
aXSDF!	Restore TBS09 default settings default TBS09 address: 0 default TBS09 serial number : 000001 default Modbus Baud rate: 19200, no parity default Modbus address: 1 default Modbus measurement command mapping: aM! - 0x0000 aM1! - 0x0002 aM9! - 0x0012 default Measurement response time: aM! ...aM9! - 001 second default warm-up time: 1 second default data format: floating default data length: 3 digits default 12V supply: permanently ON	a,register data<CR><LF>

Table 2 – Extended SDI-12 Commands

SDI 12 Master to Modbus Slave Converter

6 Example – controlling the TQS3 Temperature Sensor

The TBS09 controls a TQS3 temperature sensor.

The TQS3 default baud rate is **38400, Odd parity** and its Modbus address is 5.
In order to communicate with the TQS3 sensor, the TBS09 must be configured to Modbus address 5.

Set Baud rate to 38400, Odd parity

0XSB,38400,2! *Baud rate is 38400, 2 means odd parity*
0X_ok *SDI-12 response: settings OK*

Read data from TQS3:

0XSMB,5! *Set the Modbus address to 5, which is the default address of the TQS3 sensor*
0X_ok *SDI-12 response: settings OK*

0XSMBM,0,1! *Map aM! with input register 1 (the temperature is stored at address 1)*
0X_ok *SDI-12 response: settings OK*

The TQS3 measures only parameter, so the TBS09 just needs to configure the aM! measurement.

0XSMBD,2! *Temperature is stored in signed integer format*
0X_ok *SDI-12 response: settings OK*
0M! *start temperature measurement*
00021 *SDI-12 response: wait 2 seconds to get 1 parameter value*
0D0! *get data*
0+27.245 *temperature = 27.245*

SDI 12 Master to Modbus Slave Converter

7 Ordering Information

Part Number	Description
TBS09DR	SDI 12 Master to Modbus Slave Interface, DIN Rail housing

Table 3 – Ordering Information

8 History

Version	Date	Author	Changes
V1.0	4.11.2014	Mayerhofer	Creation of the document
V1.1	4.12.2015	Mayerhofer	formatting
V1.2	2.11.2019	PhuThinh	Updated baudrate
V1.3	24.06.2020	Hoa Hoang	Updated document name from SDI 12 to MODBUS converter to SDI 12 Master to Modbus Slave Converter Update ordering information
V1.4	2.04.2021	Mayerhofer	Updated figure 1
V1.5	08.03.2022	Philippe Hervieu	Updates related to default values and XSMVS parameters.

Table 4 – History