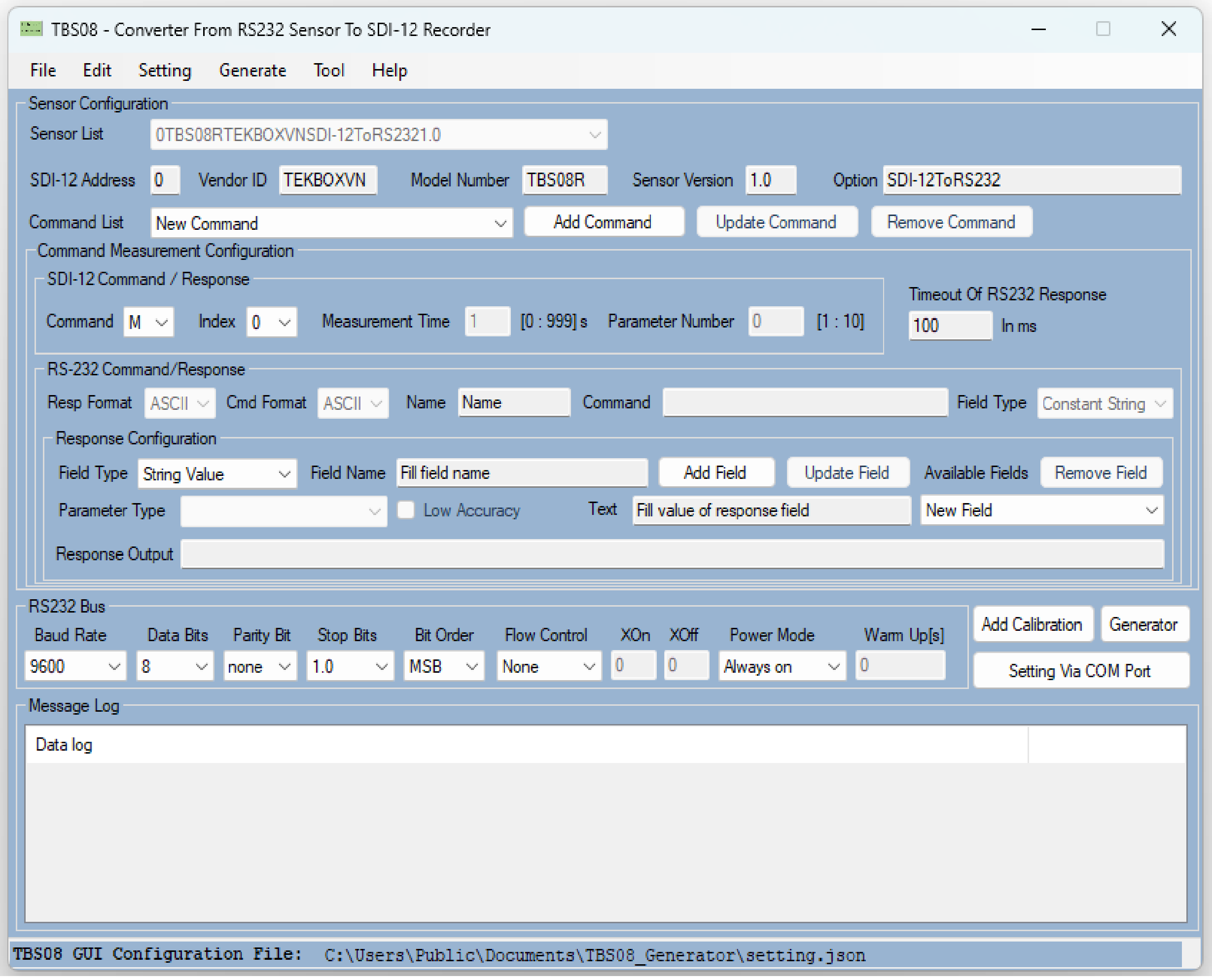
V1.0

TBS08

SDI-12 Configuration commands generator



Sensor List: 0TBS08RTEKBOXVNSDI-12ToRS2321.0

SDI-12 Address: 0 | Vendor ID: TEKBOXVN | Model Number: TBS08R | Sensor Version: 1.0 | Option: SDI-12ToRS232

Command List: New Command

Command: M | Index: 0 | Measurement Time: 1 [0 : 999] s | Parameter Number: 0 [1 : 10]

Timeout Of RS232 Response: 100 In ms

Resp Format: ASCII | Cmd Format: ASCII | Name: Name | Command: | Field Type: Constant String

Field Type: String Value | Field Name: Fill field name | Available Fields

Parameter Type: | ☐ Low Accuracy | Text: Fill value of response field | New Field

Response Output:

Baud Rate	Data Bits	Parity Bit	Stop Bits	Bit Order	Flow Control	XOn	XOff	Power Mode	Warm Up[s]
9600	8	none	1.0	MSB	None	0	0	Always on	0

Data log

```
TBS08 GUI Configuration File:  C:\Users\Public\Documents\TBS08_Generator\setting.json
```

Contents

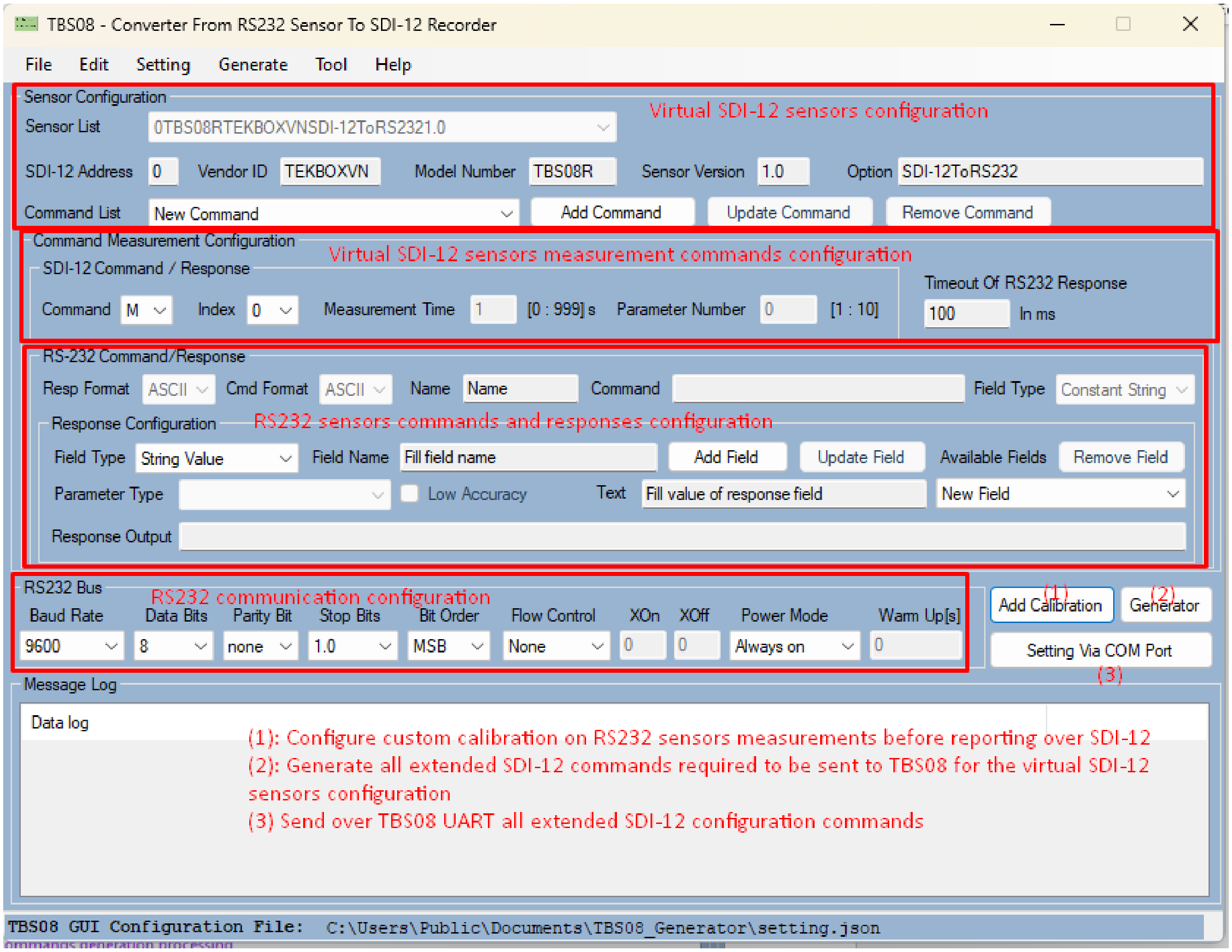
TBS08
SDI-12 Configuration commands generator

1 Introduction

TBS08 requires extensive use of external SDI-12 commands to proceed with the SDI-12 to RS232 conversion configuration.
In order to ease this process, TBS08 generator tool can be used to generate all required extended commands by setting the desired configuration in the GUI.
These extended SDI-12 commands will have then to be sent from the SDI-12 data recorder to TBS08 or they can be directly sent from the GUI to TBS08 through its UART COM port.

2 Configuration flow

The tool main interface is divided into 4 configuration sections:



Virtual SDI-12 sensors configuration

Sensor List: 0TBS08RTEKBOXVNSDI-12ToRS2321.0
SDI-12 Address: 0 | Vendor ID: TEKBOXVN | Model Number: TBS08R | Sensor Version: 1.0 | Option: SDI-12ToRS232
Command List: New Command

Virtual SDI-12 sensors measurement commands configuration

Command: M | Index: 0 | Measurement Time: 1 [0 : 999] s | Parameter Number: 0 [1 : 10]
Timeout Of RS232 Response: 100 In ms

RS232 sensors commands and responses configuration

Resp Format: ASCII | Cmd Format: ASCII | Name: Name | Command: | Field Type: Constant String
Field Type: String Value | Field Name: Fill field name
Parameter Type: | □ Low Accuracy | Text: Fill value of response field | New Field
Response Output:

RS232 communication configuration

Baud Rate	Data Bits	Parity Bit	Stop Bits	Bit Order	Flow Control	XOn	XOff	Power Mode	Warm Up[s]
9600	8	none	1.0	MSB	None	0	0	Always on	0

(1) (2) (3)

Data log

(1): Configure custom calibration on RS232 sensors measurements before reporting over SDI-12
(2): Generate all extended SDI-12 commands required to be sent to TBS08 for the virtual SDI-12 sensors configuration
(3) Send over TBS08 UART all extended SDI-12 configuration commands

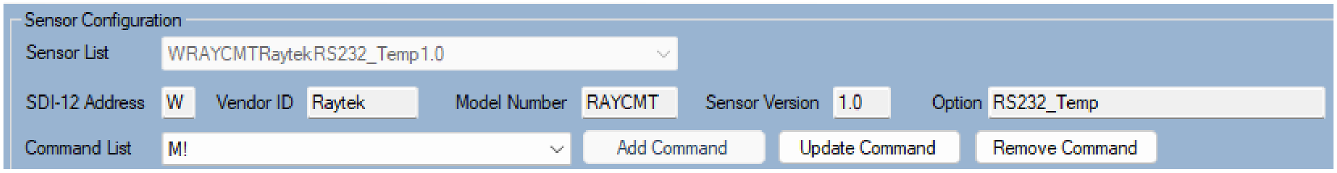
```
TBS08 GUI Configuration File:  C:\Users\Public\Documents\TBS08_Generator\setting.json
```

The configuration must be done in the following order:

- *Virtual SDI-12 Sensor Configuration*:
  - Creation of a virtual SDI-12 sensor and address that will allow taking measurements from the connected RS232 sensor over SDI-12.
  - Add/Update/Remove SDI-12 commands to/from virtual SDI-12 sensors
  - Customize virtual SDI-12 sensor identification
- *Command Configuration*:
  - Map SDI-12 commands to RS232 commands and define how to parse RS232 sensors measurement responses.
- *Add Calibration*:
  - This step is optional
  - It allows applying various kind of calibration methods to RS232 sensors measurements that are returned over SDI-12.
- *RS232 bus*:
  - Configured RS232 communication parameters
  - Select TBS08 power mode and warm-up time

3 RS232 sensor mapping to virtual SDI-12 sensor

3.1 Virtual SDI-12 sensor definition



Sensor Configuration

Sensor List: WRAYCMTRaytekRS232_Temp1.0

SDI-12 Address: W
Vendor ID: Raytek
Model Number: RAYCMT
Sensor Version: 1.0
Option: RS232_Temp

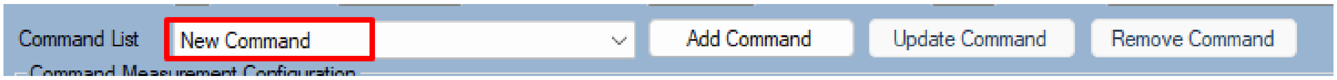
Command List: M!

This allows defining the fields related to the virtual SDI-12 sensor response to SDI-12 identification command I!. Any change to these parameters automatically updates the sensor identification in *Sensor List*.

3.2 Assign SDI-12 command

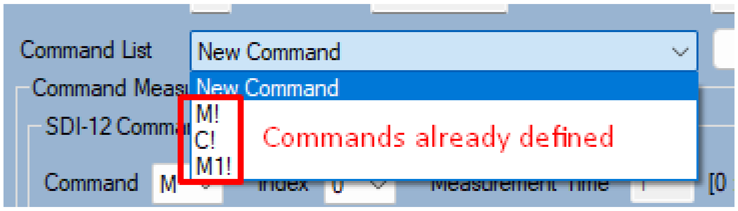
Once a virtual SDI-12 sensor has been created, SDI-12 commands need to be assigned.

Choose *New Command* from *Command List* menu:

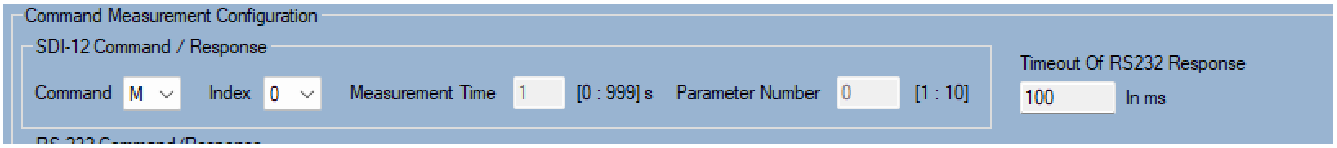


Command List: New Command

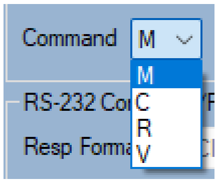
This shall be done to add any new SDI-12 command:



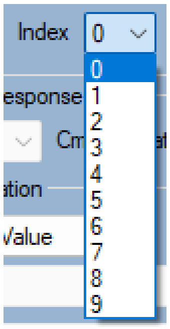
The configuration of the SDI-12 command is done in the sub-panel:


Command Measurement Configuration
SDI-12 Command / Response
Command M
Index 0
Measurement Time 1 [0 : 999] s
Parameter Number 0 [1 : 10]
Timeout Of RS232 Response
100 In ms

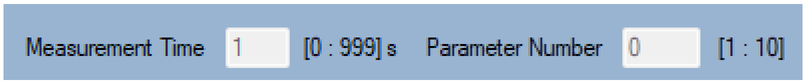
Select the desired SDI-12 command to configure:



Then choose the applicable index:

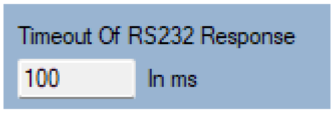


Both SDI-12 measurement time and number of returned measurements will be automatically calculated and updated by the tool at the end of the configuration phase:

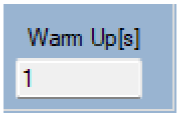


It shall be noted that the measurement time is made of three variables:

- RS232 response timeout, it can be configured here:



- Warm up time, it can be configured here:



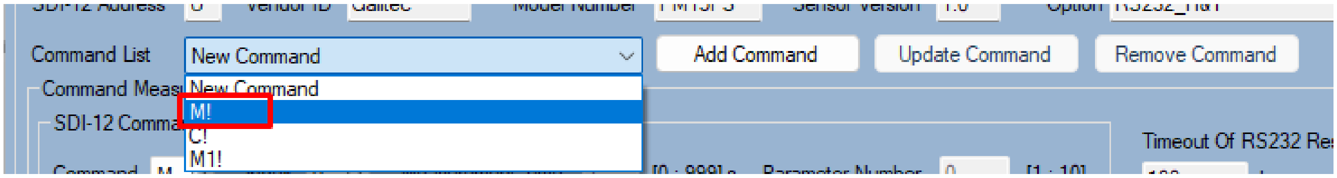
Warm Up[s]
1

- Additional 1s to compensate for TBS08 internal processing.

Once the SDI-12 command has been selected, the next step consists in defining the matching RS232 command and response. Only then will it be possible to add the created command.

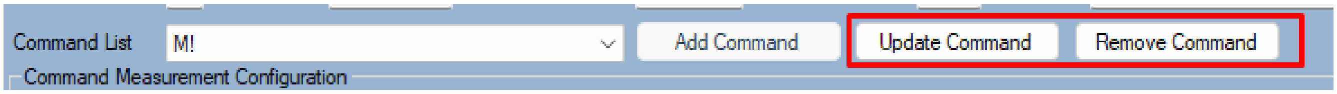
3.3 Updating or removing an existing SDI-12 command

This is achieved by selecting the desired SDI-12 command from the list



Command List New Command
New Command
M!
C!
M1!

and then either clicking on *Update Command* or *Remove Command*



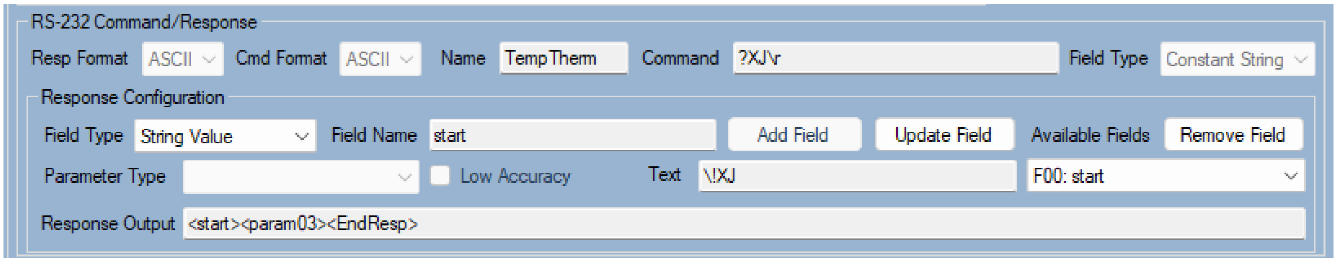
Command List M!

3.4 Configuring RS232 command

Once an SDI-12 command has been defined for the virtual SDI-12 sensor, it is needed then to configure the matching RS232 command.

Following example is based on Raytek CML RS232 infrared temperature sensor.

The RS232 sensor configuration panel is made of 2 parts, one to input the RS232 command and the other to define how to parse the RS232 response:



RS-232 Command/Response
Resp Format ASCII
Cmd Format ASCII
Name TempTherm
Command ?XJ\r
Field Type Constant String

Response Configuration
Field Type String Value
Field Name start
Parameter Type
☐ Low Accuracy
Text \!XJ
Available Fields F00: start
Response Output <start><param03><EndResp>

NOTE:

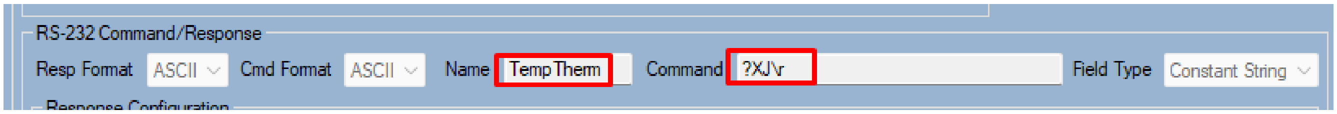
To avoid any misconfiguration with SDI-12 commands, the following special characters need to be input as follows:

- **!** **=> input as \! to avoid any conflict with SDI-12 command end character (!)**
- **Carriage return** **=> input as \r**
- **Line feed** **=> input as \n**

3.4.1 Define RS232 command

Only 2 fields need to be defined: one for the command's description (aka *Name*) and one for the RS232 command itself (aka *Command*)

For instance, Raytek RS232 command to read the temperature is defined as:



RS-232 Command/Response
Resp Format: ASCII
Cmd Format: ASCII
Name: TempTherm
Command: ?XJ\r
Field Type: Constant String
Response Configuration

Note that the mandatory carriage return character (as instructed in Raytek CM user manual) is encoded as \r.

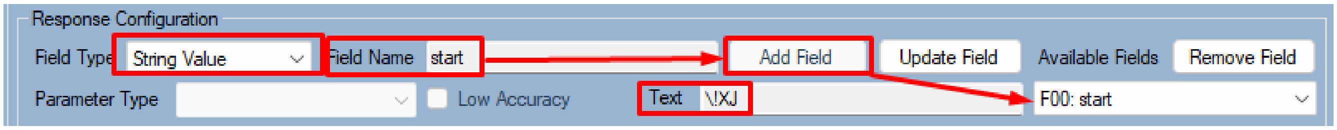
3.4.2 Define RS232 response

The next step consists in describing how the RS232 sensor's response must be parsed.

Raytek response to above command is made of 3 fields:

- A header
  - => represented as an ASCII string
- The temperature value
  - => represented as a float
- End characters
  - => represented as an ASCII string

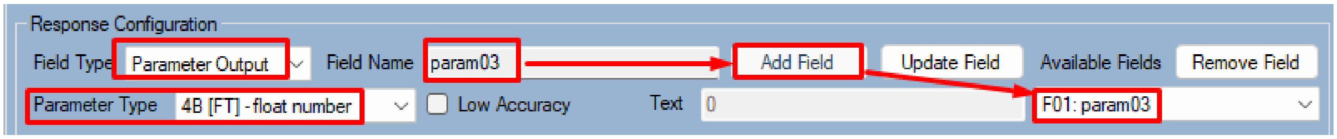
Header:



Response Configuration
Field Type: String Value
Field Name: start
Parameter Type:
☐ Low Accuracy
Text: \!XJ
Available Fields: F00: start

The *Field Type* shall be selected as *String Value*, then input the field's description and the string corresponding to this header (\!*XJ* – Note: ! is one of the 3 special characters mentioned above and shall be entered as \!).
Finally add the field, it will create field tag F00.

Temperature:



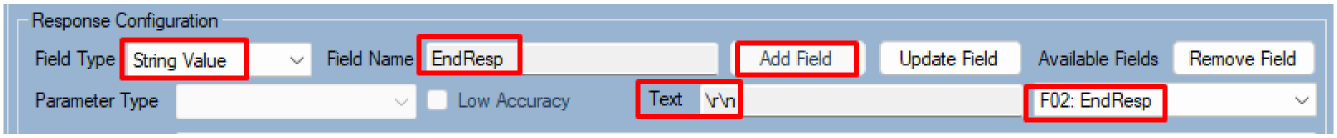
Response Configuration

Field Type: Parameter Output
Field Name: param03
Parameter Type: 4B [FT] - float number
☐ Low Accuracy
Text: 0
Available Fields: F01: param03

The *Field Type* shall be selected as *Parameter Output* and the *Parameter Type* set to *float number.*

Finally add the field, it will create field tag F01.

Termination characters:



Response Configuration

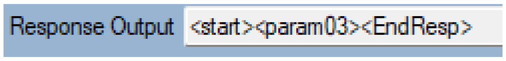
Field Type: String Value
Field Name: EndResp
Parameter Type:
☐ Low Accuracy
Text: \r\n
Available Fields: F02: EndResp

Raytek sensor adds CR/EOL as termination characters therefore they will be input as \r\n string.

The *Field Type* shall be selected as *String Value*, then input the field's description and the string corresponding to this termination (*\r\n*).

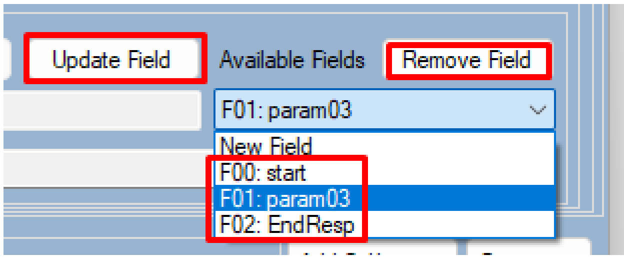
Finally add the field, it will create field tag F02.

The sensor's response will therefore be parsed as:



Response Output: <start><param03><EndResp>

All field can be selected from this drop-down list and updated/removed if needed:

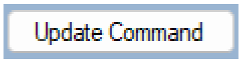


Available Fields: F01: param03

New Field
F00: start
F01: param03
F02: EndResp

If a field needs to be updated, change the desired parameters and then:

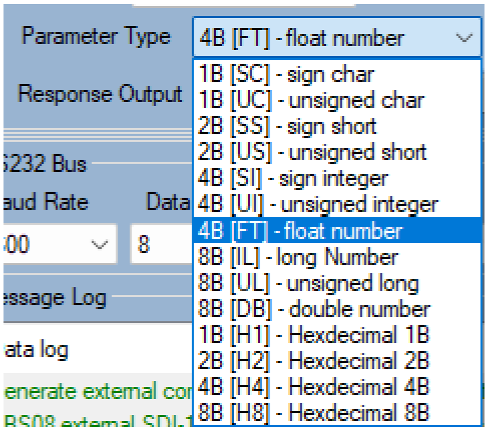
Update the field



Update the command

3.4.3 Parameter output format

TBS08 supports the following parameters format:



Parameter Type: 4B [FT] - float number

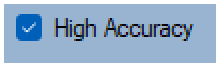
- 1B [SC] - sign char
- 1B [UC] - unsigned char
- 2B [SS] - sign short
- 2B [US] - unsigned short
- 4B [SI] - sign integer
- 4B [UI] - unsigned integer
- 4B [FT] - float number
- 8B [IL] - long Number
- 8B [UL] - unsigned long
- 8B [DB] - double number
- 1B [H1] - Hexdecimal 1B
- 2B [H2] - Hexdecimal 2B
- 4B [H4] - Hexdecimal 4B
- 8B [H8] - Hexdecimal 8B

Due to the limitations of SDI-12 protocol regarding the maximum number of digits that can be used to return measurements (7 digits + sign + optional decimal point), there is an accuracy loss while using float numbers and double/long numbers can't be correctly reported.

In this case, TBS08 uses a special encoding to allow reporting these values over SDI-12: to leverage this option it is required to switch from *Low Accuracy* to *High Accuracy* mode:

☐ Low Accuracy

➔



☑ High Accuracy

Therefore, TBS08 allows defining 2 different accuracy modes for the returned measurement values:

- Low accuracy mode: the measurement value is represented as a float coded over 7 digits with decimal point as per SDI-12 standard (so the measurement value will be rounded if exceeded 7 digits).
- High accuracy mode: the MODBUS sensor measurement value is converted to hexadecimal floating point (4 bytes – B3 B2 B1 B0) and each 2 bytes pair (B3B2 and B1B0) is reported as an SDI-12 measurement. In this case, after having been collected by the SDI-12 data logger the measurement value must be reconstructed by the backend application.

Below example shows how the RS232 sensor measurement value +45236.36818 is reported over SDI-12 in both accuracy modes.

Low accuracy mode	
RS232 sensor measurement (decimal float value)	+45236.36718
aD0! Response	**a+45236.37** Value rounded to be represented over 7 digits as per SDI-12 standard
High accuracy mode	
RS232 sensor measurement (decimal float format)	+45236.36718

RS232 sensor measurement (hexadecimal floating-point format)	0x4730B45E			
	Byte3	Byte2	Byte1	Byte0
	47	30	B4	5E
	Hexadecimal: *4730* Decimal: *18224*		Hexadecimal: *B45E* Decimal: 46174	
aD0! Response	**a+18224+46174**			

The user shall therefore be cautious when selecting the accuracy mode, keeping in mind that the low accuracy mode is only suitable for values ranging from -9999999/+9999999 to comply with SDI-12 standard.

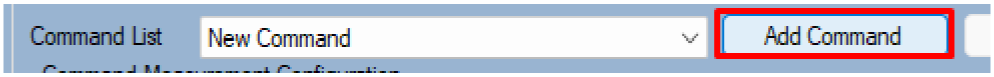
Would larger numbers be required or having decimal numbers whose total number of digits is more than 7 then the high accuracy mode shall be selected.

The principle is the same regarding long and double numbers which are encoded over 64 bits instead of 32 bits for float numbers. Their hexadecimal representation is then split over 4 measurement values (8 bytes: B7B6 | B5B4 | B3B2 | B1B0) over SDI-12 instead of 2 for float numbers.

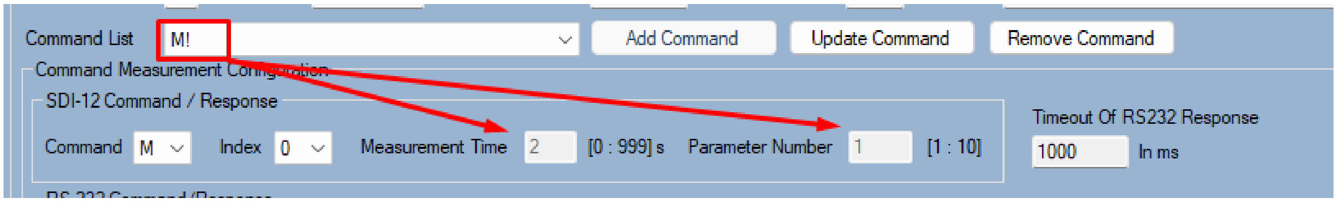
It is then up to the SDI-12 data logger or the backend service to reconstruct the right decimal values.

3.5 Creation of the virtual SDI-12 sensor mapped to RS232 command

Once the RS232 sensor command and response fields have been defined, the virtual SDI-12 sensor can be finally created by clicking on *Add Command*:



The created SDI-12 command becomes available in the Command List and both SDI-12 measurement time and number of measurement values are automatically calculated by the tool:


Command List: M!

Command Measurement Configuration

SDI-12 Command / Response

Command: M   Index: 0   Measurement Time: 2 [0 : 999] s   Parameter Number: 1 [1 : 10]

Timeout Of RS232 Response: 1000 In ms

With this example, a virtual SDI-12 sensor has been created so the SDI-12 data recorder can send SDI-12 command 0M! and then retrieve the RS232 sensor measurement value (the measured temperature) after 2s through TBS08.

Subsequent SDI-12 commands can be defined likewise depending on the need.

3.6 About SDI-12 measurement commands with CRC

The SDI-12 standard defines commands additional measurement commands with CRC:

- MC!, MC1!, … MC9!
- CC!, CC1!, … CC9!

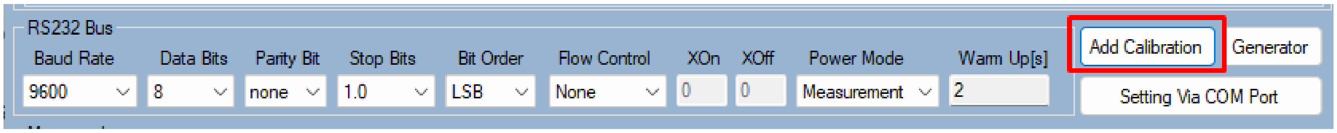
- RC0!, RC1!, … RC9!

TBS08 supports these measurement commands with CRC although they don’t appear in the available SDI-12 command list.
As long as M, C or R commands have been configured with the GUI, TBS08 will automatically handle any measurement command with CRC (MC, CC, RC) sent by the SDI-12 data logger.

So, there’s no need for further configuration for commands with CRC.

4 Calibration

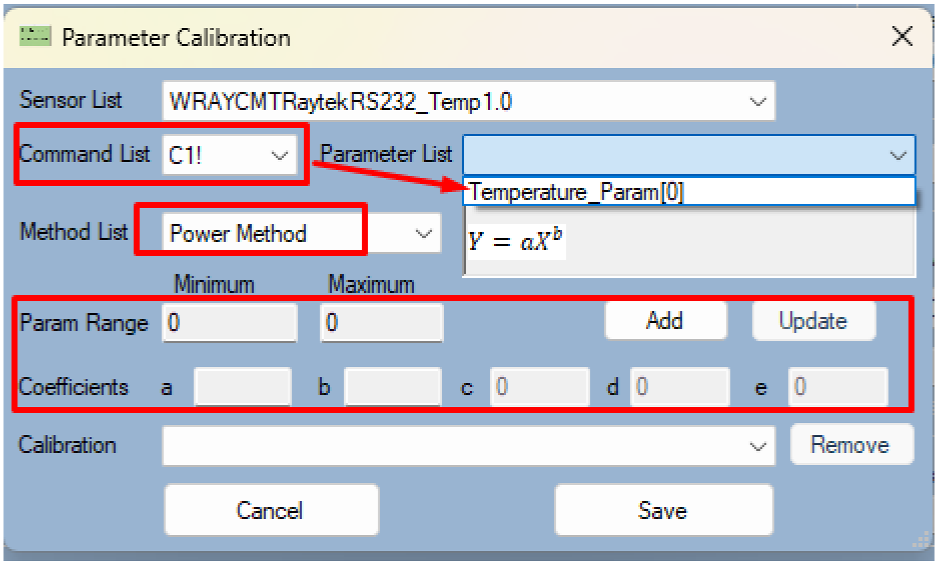
If required, TBS08 can be configured to apply calibration over RS232 sensor measurement values before reporting them over SDI-12. Click on the calibration button:



RS232 Bus

Baud Rate	Data Bits	Parity Bit	Stop Bits	Bit Order	Flow Control	XOn	XOff	Power Mode	Warm Up[s]
9600	8	none	1.0	LSB	None	0	0	Measurement	2

This will open a configuration window:



Parameter Calibration

Sensor List: WRAYCMTRaytekRS232_Temp1.0
Command List: C1!
Parameter List: Temperature_Param[0]
Method List: Power Method

$Y = aX^b$

Param Range: Minimum 0, Maximum 0
Coefficients: a, b, c 0, d 0, e 0
Calibration

- Select the SDI-12 command
- Select the measurement value which requires calibration
- Select the method to be applied (Power, Linear, Percentage, Polynomial, Logarithmic, Exponential)
- Enter required coefficients
- *Add* and *Save*

Any calibration can be updated if needed, in that case click on *Update* and *Save* after setting the new parameters.

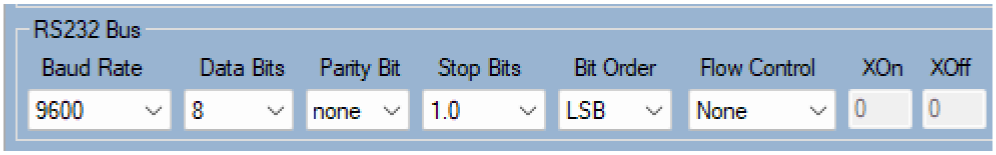
NOTE:

A maximum number of 5 calibration methods can be programmed to TBS08 (so if there are 10 measurement values, only up to 5 can be calibrated).

5 RS232 communication

5.1 RS232 bus

The RS232 communication parameters can be customized to tune the baud rate, the parity and the number of data and stop bits:

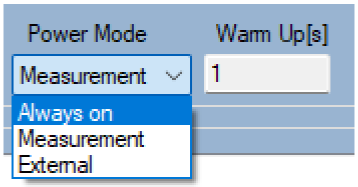


RS232 Bus

Baud Rate	Data Bits	Parity Bit	Stop Bits	Bit Order	Flow Control	XOn	XOff
9600	8	none	1.0	LSB	None	0	0

5.2 Power modes

RS232 sensor power source is configured via this combo box:



Power Mode: Measurement

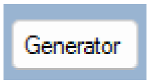
Warm Up[s]: 1

- *Always On*
  - In this mode, TBS08 constantly powers connected RS232 sensor
- *Measurement*
  - In this mode, TBS08 powers connected RS232 sensor only while taking measurements and after applying the configured *Warm Up Time*
- *External*
  - RS232 sensor is externally supplied so TBS08 does not deliver any power to the connected sensor.

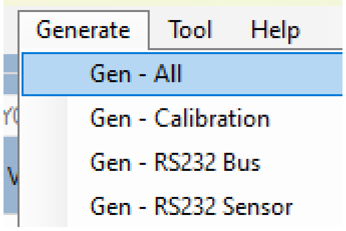
6 Generate TBS08 extended SDI-12 configuration commands

6.1 TBS08 configuration commands

List of extended commands to perform a global configuration of TBS08 can then be generated by clicking on:

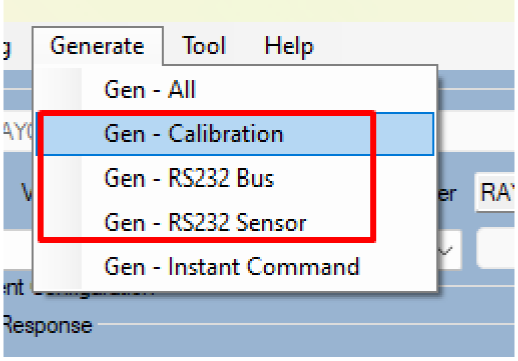


Or selecting:

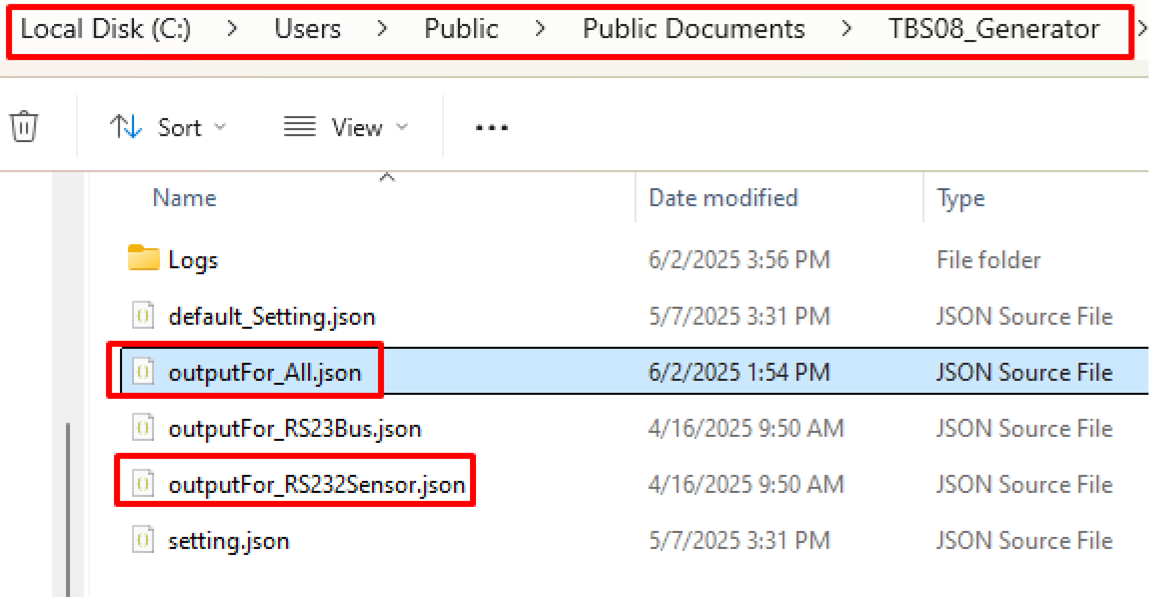


This generates all extended SDI-12 configuration commands.

Extended commands generation for specific feature can be selected from the list:



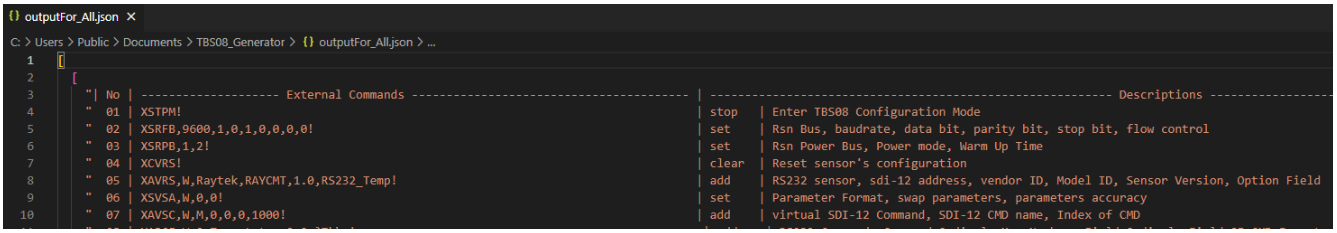
The list of generated commands is saved into a file *outputFor_<name>.json* where *<name>* is *All, Calibration, RS232Bus* or *RS232Sensor*:


Local Disk (C:) > Users > Public > Public Documents > TBS08_Generator >
Sort
View

Name	Date modified	Type
Logs	6/2/2025 3:56 PM	File folder
default_Setting.json	5/7/2025 3:31 PM	JSON Source File
outputFor_All.json	6/2/2025 1:54 PM	JSON Source File
outputFor_RS23Bus.json	4/16/2025 9:50 AM	JSON Source File
outputFor_RS232Sensor.json	4/16/2025 9:50 AM	JSON Source File
setting.json	5/7/2025 3:31 PM	JSON Source File

6.2 Sending extended configuration commands to TBS08 from an SDI-12 data recorder

All SDI-12 extended commands that shall be sent to TBS08 are found in *outputFor_All.json* file. All of them must be sent one by one from the SDI-12 data recorder:



{} outputFor_All.json ×
C: > Users > Public > Documents > TBS08_Generator > {} outputFor_All.json > ...

```
[
  [
    "| No | ------------------- External Commands ---------------------------------------- | ----------------------------------------------------- Descriptions -----------------
    "  01 | XSTPM!                                                                           | stop   | Enter TBS08 Configuration Mode
    "  02 | XSRFB,9600,1,0,1,0,0,0,0!                                                        | set    | Rsn Bus, baudrate, data bit, parity bit, stop bit, flow control
    "  03 | XSRPB,1,2!                                                                       | set    | Rsn Power Bus, Power mode, Warm Up Time
    "  04 | XCVRS!                                                                           | clear  | Reset sensor's configuration
    "  05 | XAVRS,W,Raytek,RAYCMT,1.0,RS232_Temp!                                            | add    | RS232 sensor, sdi-12 address, vendor ID, Model ID, Sensor Version, Option Field
    "  06 | XSVSA,W,0,0!                                                                     | set    | Parameter Format, swap parameters, parameters accuracy
    "  07 | XAVSC,W,M,0,0,0,1000!                                                            | add    | virtual SDI-12 Command, SDI-12 CMD name, Index of CMD
```

NOTE:

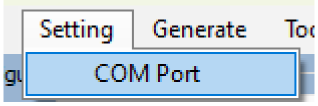
- **TBS08 SDI-12 address shall be appended to these commands (and not the address of the virtual SDI-12 sensor !)**
- **The delay between each extended command should be at least 1s**

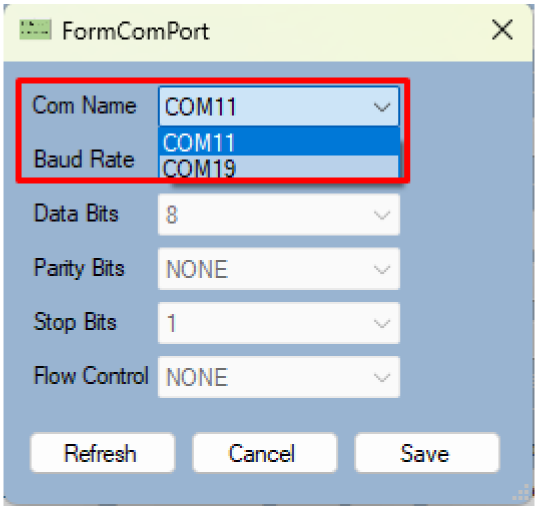
6.3 Sending extended configuration commands through TBS08 UART

It is also possible to configure TBS08 without any SDI-12 data logger.

Above configuration commands can be sent directly from the GUI to TBS08 through its UART port.

Set the PC serial USB COM port to be used:

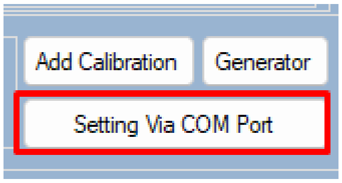


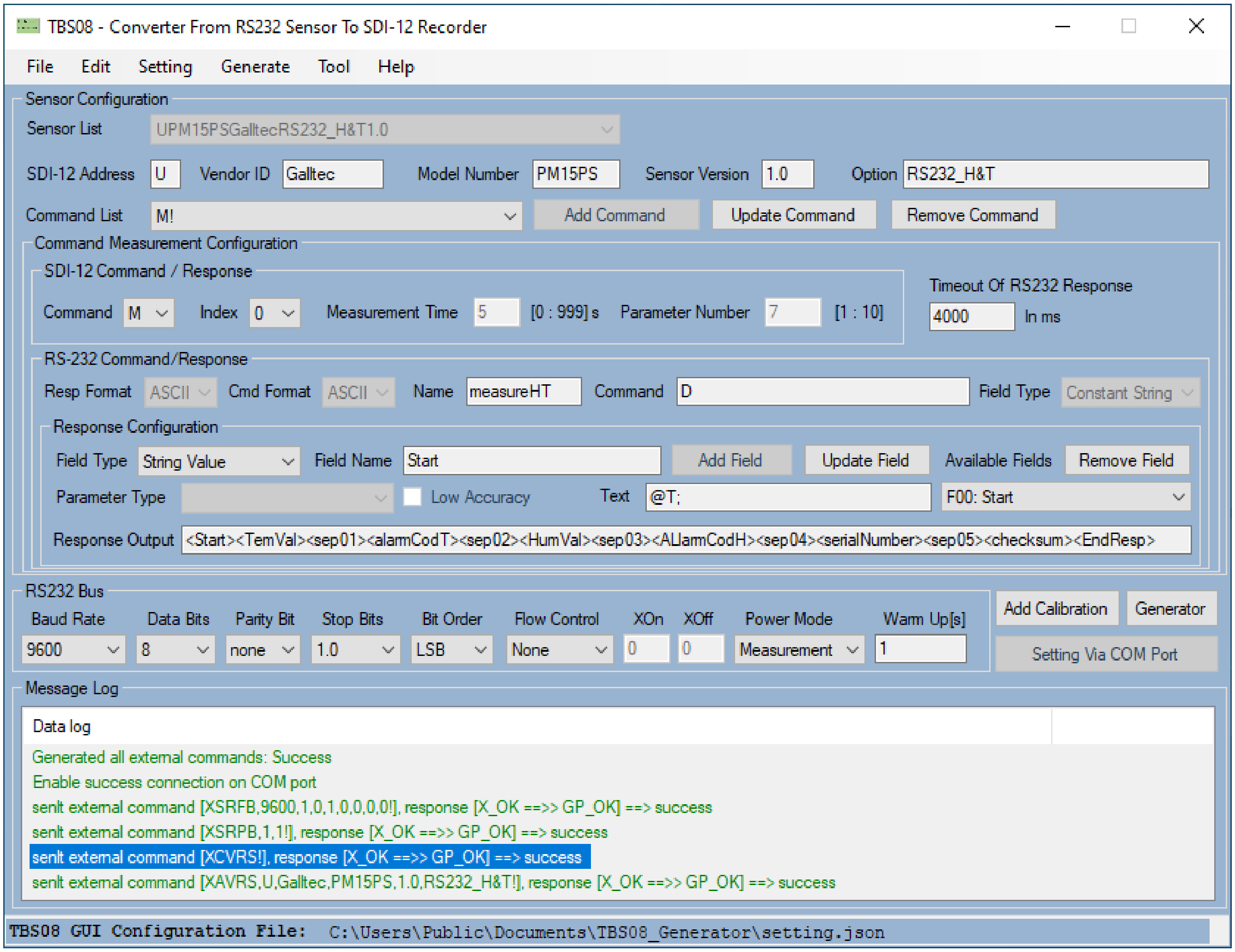


Com Name: COM11 (list: COM11, COM19)
Baud Rate
Data Bits: 8
Parity Bits: NONE
Stop Bits: 1
Flow Control: NONE

Other UART parameters can't be modified as they're matching with TBS08 UART configuration.

Once the right COM port has been set, TBS08 can then be configured by clicking:





Sensor Configuration

Sensor List: UPM15PSGalltecRS232_H&T1.0

SDI-12 Address: U   Vendor ID: Galltec   Model Number: PM15PS   Sensor Version: 1.0   Option: RS232_H&T

Command List: M!

Command Measurement Configuration

SDI-12 Command / Response

Command: M   Index: 0   Measurement Time: 5 [0 : 999] s   Parameter Number: 7 [1 : 10]

Timeout Of RS232 Response: 4000 In ms

RS-232 Command/Response

Resp Format: ASCII   Cmd Format: ASCII   Name: measureHT   Command: D   Field Type: Constant String

Response Configuration

Field Type: String Value   Field Name: Start   Available Fields

Parameter Type: □ Low Accuracy   Text: @T;   F00: Start

Response Output: <Start><TemVal><sep01><alarmCodT><sep02><HumVal><sep03><ALlarmCodH><sep04><serialNumber><sep05><checksum><EndResp>

RS232 Bus

Baud Rate	Data Bits	Parity Bit	Stop Bits	Bit Order	Flow Control	XOn	XOff	Power Mode	Warm Up[s]
9600	8	none	1.0	LSB	None	0	0	Measurement	1

Message Log

```
Data log
Generated all external commands: Success
Enable success connection on COM port
senlt external command [XSRFB,9600,1,0,1,0,0,0,0!], response [X_OK ==>> GP_OK] ==> success
senlt external command [XSRPB,1,1!], response [X_OK ==>> GP_OK] ==> success
senlt external command [XCVRS!], response [X_OK ==>> GP_OK] ==> success
senlt external command [XAVRS,U,Galltec,PM15PS,1.0,RS232_H&T!], response [X_OK ==>> GP_OK] ==> success
```

TBS08 GUI Configuration File: C:\Users\Public\Documents\TBS08_Generator\setting.json

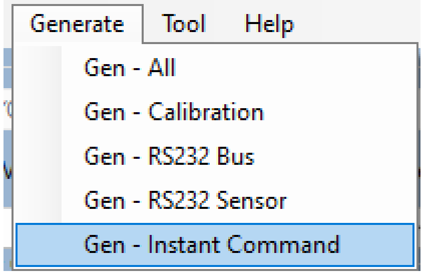
7 Other features

7.1 Direct access to RS232 sensor from an SDI-12 data logger

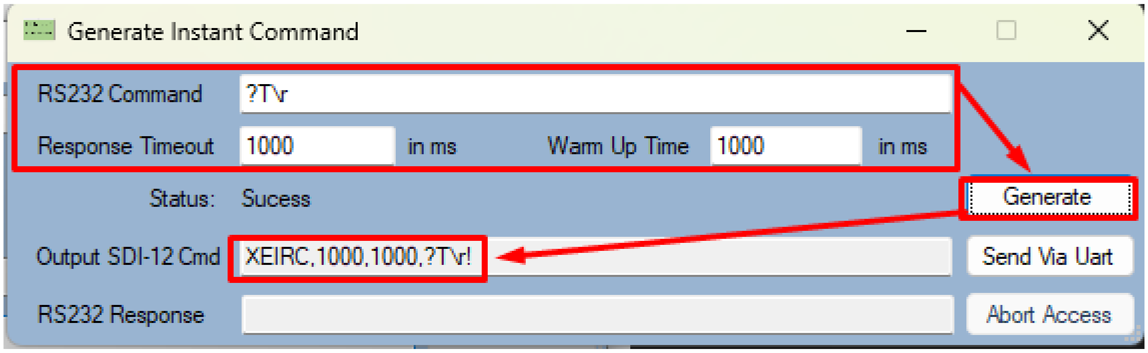
For testing purpose, it can be handy to directly access to the RS232 sensor without having to map virtual SDI-12 commands and define the RS232 command/response format.

Therefore, TBS08 provides a tool to build a specific extended SDI-12 command that embeds the RS232 command to be directly sent to RS232 sensor, either from an SDI-12 data recorder or directly from the GUI over TBS08 UART port.

Click on *Gen - Instant Command* to launch the generator for direct RS232 access:



Send RS232 command through TBS08 with an SDI-12 extended command from an SDI-12 data recorder



RS232 Command ?T\r

Response Timeout 1000 in ms Warm Up Time 1000 in ms

Status: Sucess

Output SDI-12 Cmd XEIRC,1000,1000,?T\r!

RS232 Response

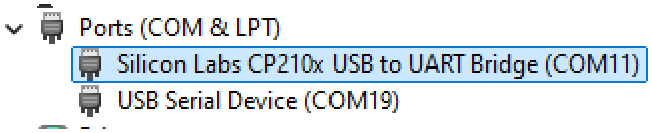
Proceed with the following steps:

- Input the desired RS232 command and set both RS232 response timeout and warm-up time
- Click on *Generate*
- The extended SDI-12 command is generated in *Output SDI-12 Cmd* box
- Send that command from a SDI-12 data recorder
- Wait for (*Response Timeout* + *Warm-up Time*) rounded to the upper second boundary
- Retrieve the RS232 sensor's response with **XGIRR!**

Send RS232 command directly from TBS08 GUI to the RS232 sensor over UART port

In this case, TBS08 must be connected to a PC via its UART port (CON1 – refer to TBS08 user manual) using a USB serial converter.

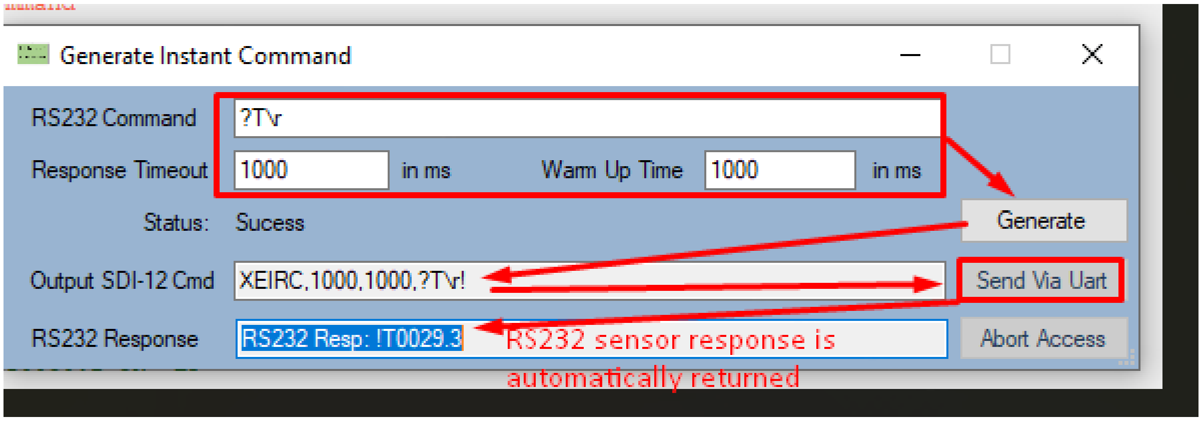
The mounted virtual COM port (using a USB serial converter) appears in Windows Device Manager



Ports (COM & LPT)
Silicon Labs CP210x USB to UART Bridge (COM11)
USB Serial Device (COM19)

and shall be correctly set in the GUI.

Then the process is similar to previous case except that by clicking *Send Via Uart*, the GUI will automatically send the RS232 to TBS08 which will return the response from the RS232 sensor:



Generate Instant Command

RS232 Command: ?T\r
Response Timeout: 1000 in ms
Warm Up Time: 1000 in ms
Status: Sucess
Output SDI-12 Cmd: XEIRC,1000,1000,?T\r!
RS232 Response: RS232 Resp: !T0029.3

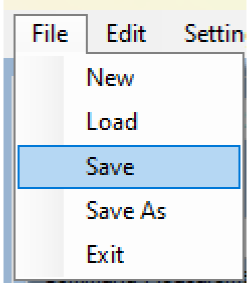
RS232 sensor response is automatically returned

Proceed with the following steps:

- Input the desired RS232 command and set both RS232 response timeout and warm-up time
- Click on *Generate*
- The extended SDI-12 command is generated in *Output SDI-12 Cmd* box
- Click on *Send Via Uart*
- The connected RS232 sensor's response will be automatically displayed in *RS232 Response* box after that (*Response Timeout* + *Warm-up time +1s)* have elapsed
  - Would the sensor not be responding or have a very large response time, it's possible to abort the measurement by clicking on *Abort Access*.

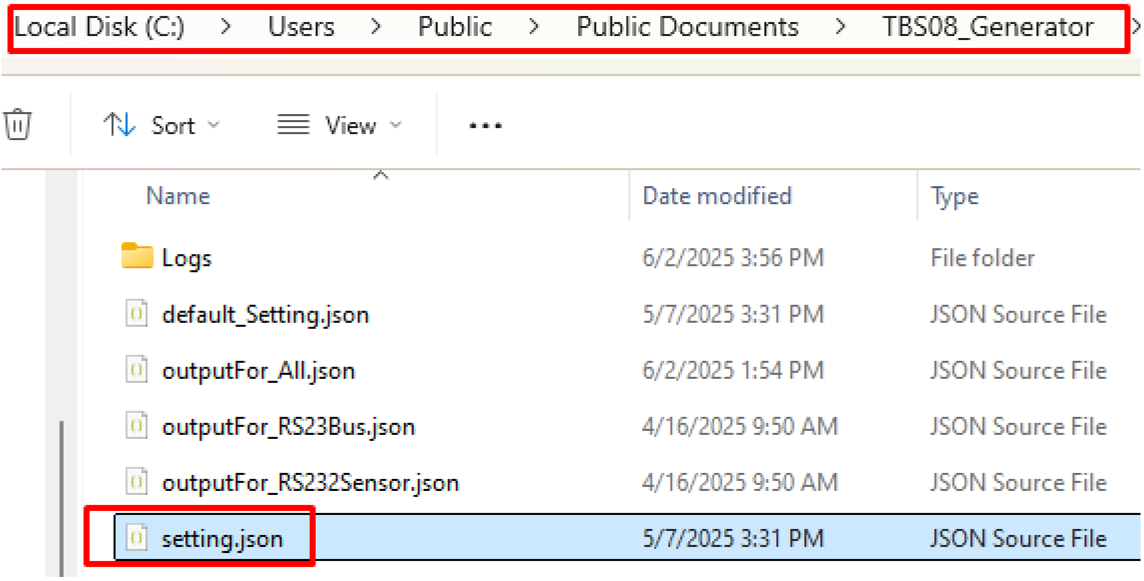
7.2 Saving/restoring GUI settings

While using the GUI to configure the extended commands, it's possible to save and restore the configuration:



This is useful to keep templates that can be later on customized.

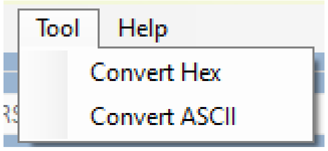
By default, the configuration is saved to:


Local Disk (C:) > Users > Public > Public Documents > TBS08_Generator >

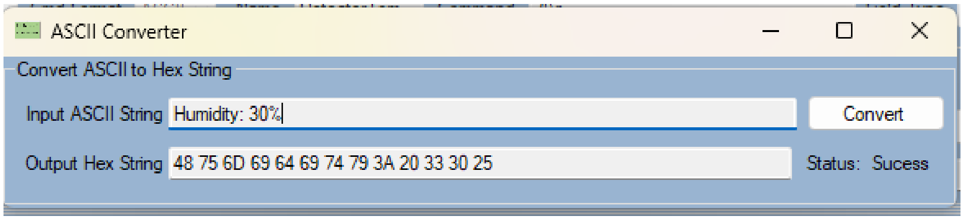
Name	Date modified	Type
Logs	6/2/2025 3:56 PM	File folder
default_Setting.json	5/7/2025 3:31 PM	JSON Source File
outputFor_All.json	6/2/2025 1:54 PM	JSON Source File
outputFor_RS23Bus.json	4/16/2025 9:50 AM	JSON Source File
outputFor_RS232Sensor.json	4/16/2025 9:50 AM	JSON Source File
setting.json	5/7/2025 3:31 PM	JSON Source File

7.3 ASCII-hexadecimal data converter

An ASCII-hexadecimal converter is embedded within the GUI to ease the decoding of hexadecimal strings to ASCII and vice versa:



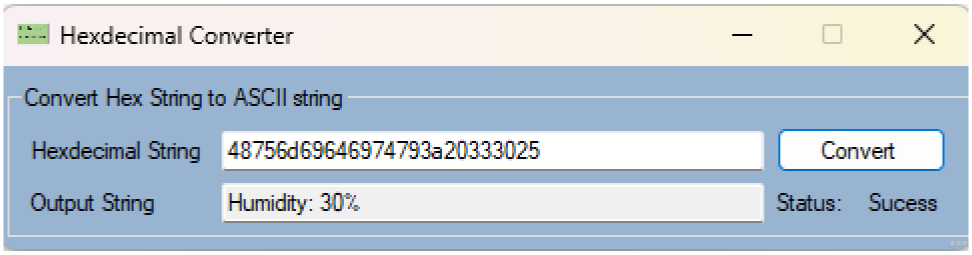
Simply enter the desired ASCII or hexadecimal string and click on *Convert*:



Input ASCII String: Humidity: 30%

Output Hex String: 48 75 6D 69 64 69 74 79 3A 20 33 30 25

Status: Sucess



Hexdecimal String: 48756d69646974793a20333025

Output String: Humidity: 30%

Status: Sucess

8 History

Version	Date	Author	Changes
V1.0	05.06.2025	Philippe Hervieu	Creation of the document