pyroscience
sensor technology

Pico-T

OEM Fiber-Optic Temperature Meter

MANUAL

T

Pico-T
OEM Fiber-Optic Temperature Meter

Document Version 1.04

The Pico-T is released by:
PyroScience GmbH
Hubertusstrasse 35
52064 Aachen
Germany

Phone +49 (0)241 5183 2210
Fax +49 (0)241 5183 2299
Email info@pyroscience.com
Web www.pyroscience.com

Registered: Aachen HRB 17329, Germany

TABLE OF CONTENT

1 INTRODUCTION

The **Pico-T** (item no. PICO-T) is a fiber-optic OEM meter for read-out of optical temperature sensors from PyroScience. The **Pico-T** is characterized by its small size, durability and low power consumption. This OEM module is easy to integrate and is controlled with a simple serial communication protocol.

To control the Pico-T, there are several options depending on the users´ level of experience with optical sensors:

Option 1: For **initial evaluation purposes**, **Pico-T** can be operated with the simple and customer-friendly logger software **Pyro Workbench**, which is typically used by end-users. This software offers comfortable settings and calibration wizards, as well as advanced logging features. Several modules can be operated in parallel within a single window. This software requires an encoded USB interface cable (item no. **PICO-USB**) for connecting the module to a Windows PC (see chapter 3).

Option 2: For advanced evaluation purposes, the module can be operated with the software **Pyro Developer Tool**. It offers simple settings and calibration procedures, as well as basic logging features. Furthermore, additional advanced settings offer full control on all features of the module. This software requires an encoded USB-interface cable (item no. **PICO-USB**) for connecting the module to a Windows PC (see chapter 4).

Option 3: A **simplified custom integration** of the module can be realized by adjusting the settings and performing sensor calibrations using the PyroScience software **Pyro Workbench** or **Pyro Developer** Tool (requires the encoded USB interface cable **PICO-USB**). After closing the software, the configuration is automatically saved within the internal flash memory of the module. The module can then be integrated into a specific setup, and your custom software can perform measurements using a proprietary USB/UART communication protocol (see chapter 5).

Option 4: For **advanced custom integration** the full USB/UART communication protocol is available on request, allowing custom software full control on all settings, calibration and measurement features of the module (see chapter 6).

2 OVERVIEW

Figure 1 provides an overview of the **Pico-T**. The front provides the port for connecting an optical fiber used for read-out of optical temperature sensors, as well as solder points for an external temperature sensor enabling calibration of the optical temperature sensor. The backside of the module provides the connector for the power supply and the digital communication interface, as well as a red status LED.

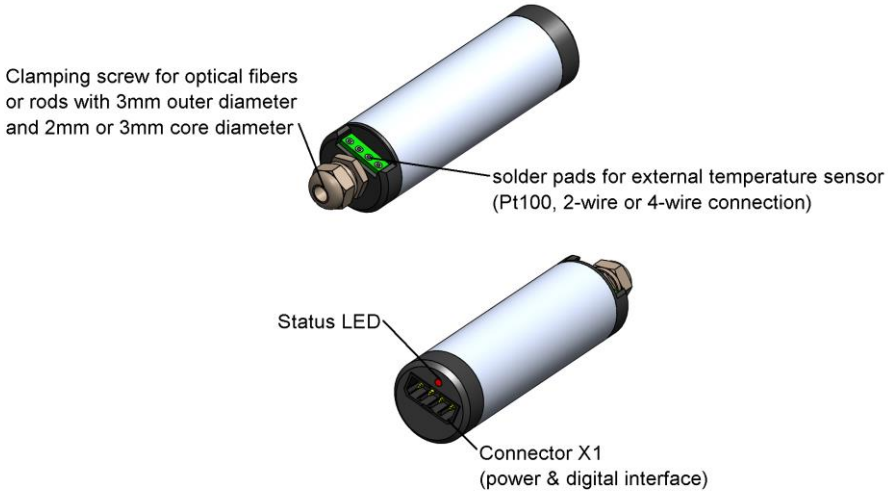


Figure 1: Overview of Pico-T

2.1 Optical port for temperature sensors

The **Pico-T** enables measurements with optical temperature sensor spots (**TSP5**). These spots can be mounted in a closed transparent vessel or placed directly in front of an optical fiber and can be used for temperature compensation of optical pH or optical oxygen measurements.

For connecting an optical fiber, **Pico-T** provides a clamping screw for fibers with an outer diameter of 3mm. This enables connecting fibers with 1m length (item no. **PICFIB2**) or short fiber rods (item no**. PICROD2 / PICROD3**). To insert the optical fiber/rod, slightly loosen the nut at the sensor port of the **Pico-T**. Remove the protective cap from the optical fiber/rod and insert it carefully into the sensor port of the **Pico-T** until there is resistance. Fasten the nut with your fingers for fixing the fiber/rod.

2.2 External temperature sensor

The optical temperature sensors need to be calibrated against an electrical Pt100 temperature sensor. For this, **Pico-T** offers a high-precision sensor interface, which can be directly connected to a Pt100 temperature sensor (not included, item no. **TSUB21-NC**). The temperature sensor has to be soldered to the 4 solder pads at the front of the module.

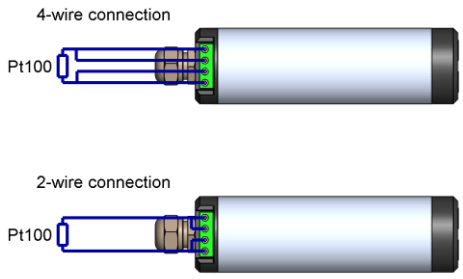


Figure 2: Connecting a resistive temperature sensor to the module

The Pt100 temperature sensor has to be soldered to the 4 solder pads at the front of the module (Figure 2). For short distances (e.g. 10 cm) a simple 2-wire connection might be sufficient. For this, it is important to shortcut the outer with the inner solder pads as indicated in Figure 2. For longer distances and/or for high precision measurements a 4-wire connection should be preferred.

> In order to minimize potential electrical noise coupling into the external temperature sensor, the cables should be twisted and kept as short as possible.

2.3 Status LED

The behavior of the status LED is given in Table 1.

Table 1: Status LED

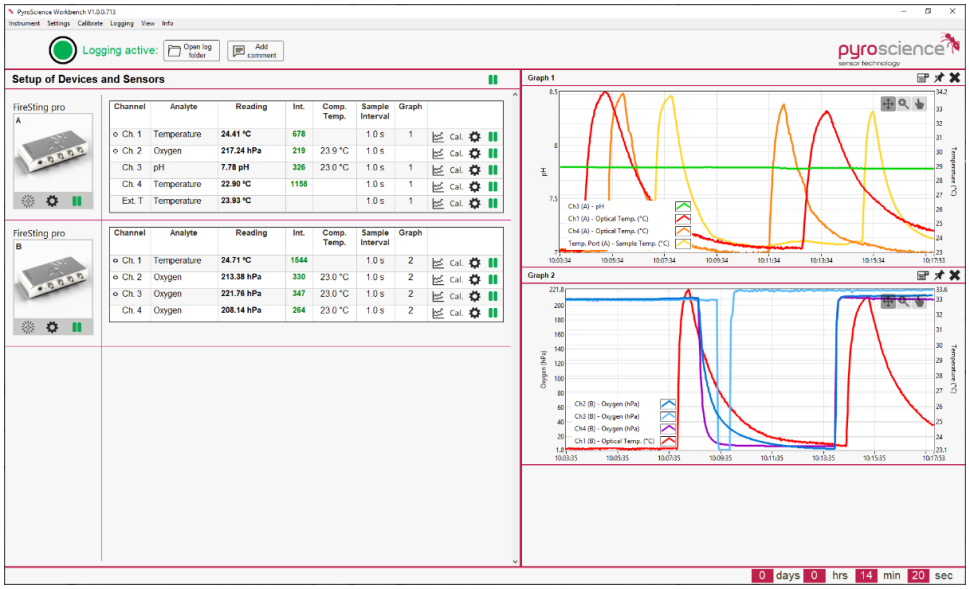
Status	Description	Behavior of status LED
Power-Up	The power supply is switched on.	A correct startup of the module is indicated by 4 flashes within 1-2 seconds.
Active	The module is either in idle mode waiting for a new command, or it is executing a command.	The LED flashes periodically with 1s interval.
Deep sleep	While the power supply is still enabled, the module can be put into deep sleep mode by the #STOP command.	The LED is switched off.
#LOGO-command	The #LOGO-command is sent to the module.	The LED flashes 4 times within 1-2 seconds.

2.4 USB interface cable

For the operation of **Pico-T** with a Windows PC, a coded USB interface cable (item no. **PICO-USB**) is available from PyroScience. It includes a license for the comfortable logger software **Pyro Workbench** and the software **Pyro Developer Tool**. Especially for initial testing purposes this software packages can speed up OEM-developments significantly. Additionally, the USB interface cable **PICO-USB** provides a virtual COM-port. Custom software can use this virtual COM-port for communicating directly with the module based on the communication protocol (see chapter Fehler! Verweisquelle konnte nicht gefunden werden. and Fehler! Verweisquelle konnte nicht gefunden werden.).

3 OPTION 1: OPERATING THE MODULE WITH PYRO WORKBENCH

For **initial evaluation purposes** the module can be operated with the simple and customer-friendly software **Pyro Workbench,** which is typically used by end-users. This software offers comfortable settings and calibration wizards, as well as advanced logging features. Several modules can be operated in parallel within a single window. This software requires an encoded USB interface cable **PICO-USB** for connecting the module to a Windows PC.



3.1 Installing the software Pyro Workbench

System requirements: PC with Windows 7/8/10 and min. 1000 MB free disk space.

> Do not connect the USB-interface cable to your PC before the **Pyro Workbench** software has been installed. The software will automatically install the appropriate USB-drivers.

Installation steps:

- Download the **Pyro Workbench** from the downloads tab on www.pyroscience.com
- unzip and start the installer and follow the instructions

- connect the interface plug of the USB interface cable to the connector X1 of the **Pico-T**
- connect the USB plug to an USB port of the PC. The status LED of the **Pico-T** should flash shortly indicating the correct startup of the module.
- Start the **Pyro Workbench** software.

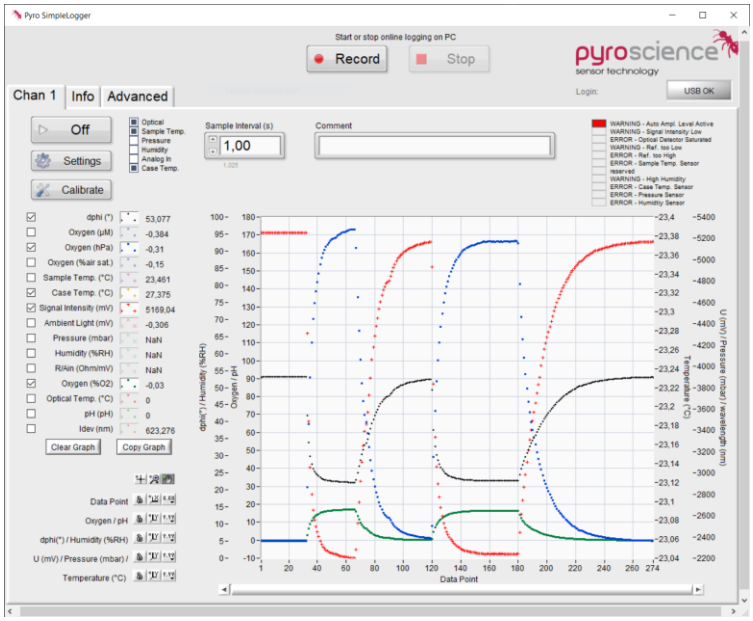
3.2 Using the software Pyro Workbench

Please refer to the **Pyro Workbench manual** for general operation instructions for the software (available on our website).

Please refer to the **Optical Temperature Sensor Manual** for general information on handling and calibration of the optical temperature sensors (available on our website).

4 OPTION 2: OPERATING THE MODULE WITH PYRO DEVELOPER TOOL

For **advanced evaluation purposes** the module can be operated with the software **Pyro Developer Tool**. It offers simple settings and calibration procedures, as well as basic logging features. Furthermore, additional advanced settings offer full control on all features of the module. This software requires the encoded USB interface cable **PICO-USB** for connecting the module to a Windows PC.



4.1 Installing the software Pyro Developer Tool

System requirements: PC with Windows 7/8/10 and min. 1000 MB free disk space.

> Do not connect the USB-interface cable to your PC before the **Pyro Developer Tool** has been installed. The software will install automatically the appropriate USB-drivers.

Installation steps:

- Download the **Pyro Developer Tool** from the downloads tab on www.pyroscience.com

- unzip and start the installer and follow the instructions
- connect the interface plug of the USB interface cable the connector X1 of the **Pico-T**
- connect the USB plug to an USB port of the PC. The status LED of the **Pico-T** should flash shortly indicating the correct startup of the module.
- Start the **Pyro Developer Tool** software.

4.2 Using the software Pyro Developer Tool

Please refer to the **Pyro Developer Tool manual** for general operation instructions for the software (available on our website).

Please refer to the **optical temperature sensor manual** for general information on handling and calibration of the optical temperature sensors (available on our website).

5 OPTION 3: SIMPLIFIED CUSTOM INTEGRATION

A **simplified custom integration** of the module can be realized by adjusting the settings and performing sensor calibrations using the PyroScience software **Pyro Workbench** or the more advanced software **Pyro Developer Tool** (both requiring the encoded USB interface cable **PICO-USB**). After closing the software, the configuration is automatically saved within the internal flash memory of the module. The module can then be integrated into a specific setup, and your custom software can perform measurements using a proprietary **USB/UART communication protocol**.

5.1 Configuring the Module using PyroScience Software

Please install either the **Pyro Workbench** or the **Pyro Developer Tool**. Follow chapter 3 or chapter 4, respectively, how to operate the module with the PyroScience software. Adjust the settings and perform the required calibrations of the sensor.

After the module has been configured, close the PyroScience software. The configuration is automatically saved within the internal flash memory. This means that the adjusted settings and the last sensor calibration are persistent even after a power cycle of the module. Now the module can be integrated into a customer specific setup via its UART interface (or via the USB interface cable with its virtual COM port).

5.2 Electrical Connector for Custom Integration

The electrical interface of the **Pico-T** consists of the connector X1 (Figure 3). The package includes the fitting connector plug S1 (manufacturer: **Phoenix Contact**, type: **PTSM0,5/4-P-2,5**, Item no.: **1778858**). Stripped cable ends can be connected to S1 without any soldering or crimping. When inserting or removing a stripped cable end (stripping length 6 mm, max. core diameter 0.5 mm^2) into one of the connector holes of the connector S1, an internal spring mechanism has to be unlocked. This can be achieved by pushing relatively strongly with a small screw-driver (flat-bladed 2 mm in width) into the adjacent rectangular hole (Figure 3). The same manufacturer offers also fitting connector plugs for PCB mounting (details on request).

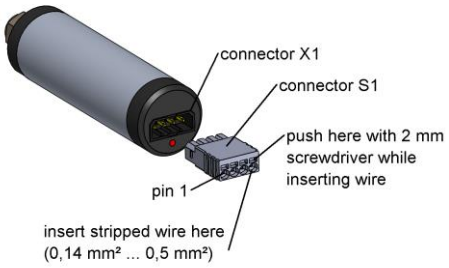


*Figure 3: Electrical connectors of **Pico**-T*

The pin configuration of the connector X1 is given in Table 2.

Table 2: Pin configuration of the connector X1

Pin	Name	Function	Description
1	VCC	Power	Power supply min. 3.3 VDC max. 5.0 VDC
2	RXD	Digital input 3.0 V levels (3.3 V & 5 V tolerant)	Data receive line of the UART interface
3	TXD	Digital output 3.0 V levels	Data transmission line of the UART interface
4	GND	Power	Ground

5.3 Configuration of the Serial Interface

Pico-T is operated via a serial interface, which is realized as a UART interface at 3.0 V levels (3.3 V and 5 V tolerant) consisting of a receive and a transmit line. The configuration of the UART-interface is as follows:

19200 baud, 8 data bit, 1 stop bit, no parity, no handshake

Such an UART interface is very common for microcontrollers or microcontroller boards (e.g. Arduino or Raspberry Pi). The module can be directly connected to such UART interfaces without any further interface electronics.

> **Note:** The serial interface of this module is **not** an RS232 interface. However, the UART interface can be made compatible to RS232 by integrating an appropriate "level shifter electronics".

5.4 Communication Protocol

5.4.1 General Definitions

A command always starts with a specific command header (e.g. MEA, #VERS, #LOGO) optionally followed by several input parameters. Input parameters are given as human readable decimal numbers, separated by spaces from each other. Each command must be terminated by a carriage return. If the command could be successfully interpreted by the module, the response is sent back to the master **after completion of the requested task**. The first part of response consists always of a copy of the original command, optionally appended with output parameters, and again terminated by a carriage return. After a response has been received by the master, the module is immediately ready for receiving the next command. If the internal processing of the received command causes any error within the module, the response will be the error header #ERRO followed by a space and an error code (see below).

Syntax Definitions	
MEA **#VERS** **#LOGO**	Examples for a command header
C S R	Examples for place holder for signed integer values transmitted as human readable ASCII strings of decimal numbers. The absolute maximum range of all values transmitted in the communication protocol is from -2147483648 to +2147483647 (signed 32bit integer), if not otherwise indicated.
␣	Space (ASCII code 0x20)
↵	Carriage return (ASCII code 0x0D)

5.4.2 MEA – Trigger Measurement

This command triggers a measurement and returns the results.

Command: `MEA␣C␣S↵`
Response: `MEA␣C␣S␣R0␣R1…R17↵`

Input Parameters:

C Optical channel number. Set C=1.

S If in doubt, then set S to 47!
This parameter defines the enabled sensor types,
given as decimal representation of the following bit field:

Bit 0 (add 1): optical channel
Bit 1 (add 2): sample temperature (typ. the external Pt100-sensor)
Bit 2 (add 4): ambient air pressure
Bit 3 (add 8): relative humidity within the module
Bit 4 (add 16): reserved
Bit 5 (add 32): case temperature (temperature within the module)

Example: S = 1 + 2 + 4 + 8 + 32 = 47 means, that the command will trigger the following measurements: optical channel (optical temperature sensor), sample temperature, case temperature, ambient air pressure, and relative humidity within the module housing.

Output Parameters:

R_0 Returns errors and/or warnings of the last measurement as a decimal representation of the following bit field. The user has to distinguish between warnings and errors. A warning indicates, that the measurement results are in principle still valid, but their precision and/or accuracy might be deteriorated. An error means, that the respective measurement result is not at all valid.

Bit 0 (add 1): WARNING - automatic amplification level active
Bit 1 (add 2): WARNING - sensor signal intensity low
Bit 2 (add 4): ERROR - optical detector saturated
Bit 3 (add 8): WARNING - reference signal intensity too low
Bit 4 (add 16): ERROR - reference signal too high
Bit 5 (add 32): ERROR - failure of sample temperature sensor (e.g. Pt100)
Bit 6 (add 64): reserved
Bit 7 (add 128): WARNING high humidity (>90%RH) within the module
Bit 8 (add 256): ERROR - failure of case temperature sensor
Bit 9 (add 512): ERROR - failure of pressure sensor
Bit 10 (add 1024): ERROR - failure of humidity sensor

Example: R_0 = 34 = 2 + 32 means, that there is a warning about low signal intensity of the optical sensor, and that the external temperature sensor (Pt100) had a failure.

If R_0 = 0 then no error or warning appeared.

$R_{1...}R_{17}$ The results of the measurement given as 17 values. The most important result values are highlighted.

	Name	Unit	Description
R1	*dphi*	m°	Phase shift of optical measurement (raw data)
R2-R4	*-reserved-*		
R5	*tempSample*	0.001 °C	Pt_{100} temperature (typ. external Pt100 sensor), can be used for calibrating the optical temperature sensor
R6	*tempCase*	0.001 °C	Case temperature (internal T-sensor within module)
R7	*signalIntensity*	0.001 mV	Signal intensity of the optical measurement
R8	*ambientLight*	0.001 mV	Ambient light entering the sensor
R9	*pressure*	0.001 mbar	Ambient air pressure
R10	*humidity*	0.001 %RH	Relative humidity within the module housing
R11	*resistorTemp*	0.001 Ohm	Resistance of the temperature sensor (raw data)
R12	*-reserved-*		
R13	*tempOptical*	0.001 °C	Temperature measured with optical temperature sensor
R14-R17	*-reserved-*		

This command is the essential command for triggering measurements. The input parameter S defines which sensor types should be measured.

IMPORTANT: If automatic temperature compensation is enabled for the optical sensor, it is mandatory to enable Bit1 of the input parameter S!

The output parameters *tempOptical* and *tempSample* give the results of the optical temperature measurement and of the temperature measurement with the external Pt100. The latter can be used for calibrating the optical sensor.

The output parameter *signalIntensity* is a measure of the signal quality ("signal intensity") of the connected optical temperature sensor. As a rule of thumb, typical values will be in the range of 20-500 mV. Low signal intensities (<20 mV) might lead to noisy optical temperature measurements. A low signal intensity might be an indicator that the sensor is not configured optimally and/or that the sensor is "worn out"/depleted and has to be replaced.

The output parameter *ambientLight* is a measure how much ambient infrared light is entering the optical temperature sensor. In principle, such ambient light is not influencing the optical temperature measurement. However, excess ambient light might lead to a saturation of the optical detector (indicated by an enabled ERROR Bit2 in R_0), which will lead to an invalid optical temperature measurement. As a rule of thumb, the sum of *signalIntensity* and *ambientLight* should be kept below ca. 2000 mV (the optical detector saturates around 2500 mV).

Example Communication:

Command	MEA␣1␣3↵
Response	MEA␣1␣3␣0␣30120␣0␣0␣0␣27135␣0␣87016␣11788␣0␣0␣123022␣0␣27105␣0␣0␣0␣0↵

This example command triggers the measurement of the Pt100 temperature and of the optical temperature sensor. The highlighted output parameters of the shown example response are interpreted as follows:

R_0 = 0 → No error or warning occurred; the measurement is valid!
tempSample = 27.135 °C
signalIntensity = 87.016 mV
ambientLight = 11.788 mV
tempOptical = 27.105

5.4.3 COT – Calibrate the Optical Temperature Sensor

This command performs a 1-point calibration of the optical temperature sensor.

Command: **COT**␣C␣T↵
Response: **COT**␣C␣T↵

Input Parameters:

C	Optical channel number. Set C=1
T	Temperature of the calibration standard in units of 10^{-3} °C (e.g. 20000 means 20°C)

This command performs 16 repeated optical measurements, and uses the average for the calibration. The total duration for this procedure varies between ca. 3s and ca. 6s depending on the configuration of the module. **In order to keep the calibration permanently even after a power cycle, the command SVS must be executed afterwards.**

5.4.4 SVS – Save Configuration Permanently in Flash Memory

This command is used for storing the current configuration in the flash memory:

Command: **SVS**␣C↵
Response: **SVS**␣C↵

Input Parameters:

C	Optical channel number. Set C=1

Saves the actual settings and calibration as the new default values into the internal flash memory. These default values are automatically loaded after a power cycle.

Example Communication:

Command: SVS_1↵
Response: SVS_1↵

5.4.5 #VERS – Get Device Information

This command returns general information about the device.

Command: **#VERS**↵
Response: **#VERS**_D_N_R_S_B_F↵

Output Parameters:

D Device ID, identifies the specific device type. For the **Pico-T** the device ID is always 4.

N Number of optical channels. For the **Pico-T** this value is 1.

R Firmware version, e.g. *R*=403 designates firmware version 4.03

S Bit field about available sensor types and supported optical analytes as follows:

Bit 0-7: Available Sensor Types	Bit 8-15: Supported Optical Analytes
Bit 0: optical channel(s)	Bit 8: oxygen
Bit 1: sample temperature (typ. Pt100)	Bit 9: optical temperature
Bit 2: pressure	Bit 10: pH
Bit 3: humidity	Bit 11: CO2
Bit 4: analog in	Bit 12: reserved
Bit 5: case temperature	Bit 13: reserved
Bit 6: reserved	Bit 14: reserved
Bit 7: reserved	Bit 15: reserved

Example: *S* = 1 + 2 + 4 + 8 + 32 + 512 = 559 means, that the device provides an optical channel as well as sample and case temperature, pressure, and humidity sensors, and the optical channel supports the analytes temperature.

B Firmware build number starting at 1 for each firmware version (reflects minor firmware revisions which normally do not require a software or firmware update for the user)

F Bit field about available features as follows:

Bit 0: analog out 1	Bit 5: battery
Bit 1: analog out 2	Bit 6: stand-alone logging
Bit 2: analog out 3	Bit 7: sequence commands
Bit 3: analog out 4	Bit 8: user memory
Bit 4: user interface (display, buttons)	Bit 9-31: reserved

Example: F = 1 + 2 + 4 + 8 + 256 = 271 means that 4 analog outputs are supported and the module possesses a user memory. Note, the optional analog outputs require additional hardware (more information on request).

Example Communication:

Command: #VERS_1↵
Response: #VERS_1_4_403_1071_2_271↵

5.4.6 #IDNR – Get Unique ID Number

This command returns the unique identification number of the respective device.

Command: **#IDNR**↵
Response: **#IDNR**␣N↵

Output Parameters:

N	Unique ID number. Note, this parameter is given as an unsigned 64 bit integer!

Returns the unique identification number of the device (does NOT correspond to the serial number of the device).

Example Communication:

Command: #IDNR↵
Response: #IDNR␣2296536137892833272↵

5.4.7 #LOGO – Flash Status LED

This command lets the status LED flash for 4 times within ca. 1 s.

Command: **#LOGO**↵
Response: **#LOGO**↵

This command can be used to check proper communication with the device. Or it might be helpful in setups with more than one device, in order to identify which COM port is connected to which device.

5.4.8 #PDWN – Power Down Sensor Circuits

This command switches off the power supply of the sensor circuits.

Command: **#PDWN**↵
Response: **#PDWN**↵

This command can be used for some power saving during idle operation periods. Note, that the sensor circuits are automatically powered up again, if the module receives any command (e.g. MEA) requiring a sensor measurement. This is also the case if a broadcast measurement takes place.

5.4.9 #PWUP – Power Up Sensor Circuits

This command switches on the power supply of the sensor circuits.

Command: **#PWUP**↵
Response: **#PWUP**↵

The wake-up duration is up to 250 ms.

5.4.10 #STOP – Enter Deep Sleep Mode

This command puts the device into a deep sleep mode with very low power consumption

Command: **#STOP**↵
Response: **#STOP**↵

During deep sleep mode the device has very low power consumption. No standard communication via USB/UART is possible. The deep sleep mode can be only exit by sending a <CR> (0x0D) to the device. The device will respond then also with a single <CR> indicating that it is ready to receive new commands. The wake-up duration can be up to 250 ms.

5.4.11 #RSET – Reset Device

This command triggers a reset of the device.

Command: **#RSET**↵
Response: **#RSET**↵

Triggers a reset of the device, as if the device experienced a power cycle.

5.4.12 #RDUM – Read User Memory

This command reads values from the user memory registers.

Command: **#RDUM**␣R␣N↵
Response: **#RDUM**␣R␣N␣Y_0…Y_N↵

Input Parameters:

R	Address of first register to be read from the user memory (0…63)
N	Total number of registers to be read (1…64)

Output Parameters:

Y_0…Y_N	Content of the requested user memory registers (signed 32bit integers).

The device offers a user memory of altogether 64 signed 32bit integer numbers (range -2147483648 to 2147483647) which is located in the flash memory and is therefore retained even after power cycles. This read command returns *N* (*N*=1…64) consecutive values Y_1 … Y_N from the user memory starting at the user memory address *R* (*R*=0…63). Note, that *N*+*R* must be <=64. The content of the user memory has no influence on the module itself. It can be used for any user specific purpose.

Example Communication:

Command: `#RDUM␣12␣4`↵
Response: `#RDUM␣12␣4␣-40323␣23421071␣0␣-555`↵

This example shows a command which requests the values of 4 consecutive beginning with the user memory address 12.

5.4.13 #WRUM – Write User Memory

This command writes values into the user memory registers.

Command: **#WRUM**␣R␣N␣Y_0…Y_N↵
Response: **#WRUM**␣R␣N␣Y_0…Y_N↵

Input Parameters:

R	Address of first user memory register to be written (0…63)
N	Total number of registers to be written (1…64)
Y_0…Y_N	Values to be written to the user memory registers (signed 32bit integers).

This command writes *N* (*N*=1…64) values Y_1 … Y_N consecutively starting at the user memory address *R* (*R*=0…63). Note, that *N*+*R* must be <=64. **This command must be used economically**, because the flash memory is designed for typ. max. 20000 flash cycles. Each time this command is executed, it will trigger a flash cycle.

Example Communication:

Command: #WRUM␣0␣2␣-16␣777↵
Response: #WRUM␣0␣2␣-16␣777↵

This example shows a command which writes the value -16 into the memory address 0, and the value 777 into the memory address 1.

5.4.14 #ERRO – Response if Error Occurred

If an error occurred, the device will give the following response:

Command: *any command*
Response: **#ERRO**␣C ↵

This error response is mostly given, if the master did not send the command with the correct communication syntax. The output parameter C represents the general PyroScience error types as given by the following table.

> Note: Warnings and errors directly related to the sensor measurements (e.g. a broken Pt100 temperature sensor, or a "worn out" optical temperature senor) will not result in such an #ERRO response. Instead, such warning and errors are given in the output parameter R_0 of the MEA command (see above).

C	Error Type	Description
-1	*General*	A non-specific error occurred.
-2	*Channel*	The requested optical channel does not exist.
-11	*Memory Access*	Memory access violation either caused by a not existing requested register, or by an out of range address of the requested value.
-12	*Memory Lock*	The requested memory is locked (system register) and a write access was requested.
-13	*Memory Flash*	An error occurred while saving the registers permanently. The SVS request should be repeated to ensure a correct permanent memory.
-14	*Memory Erase*	An error occurred while erasing the permanent memory region for the registers. The SVS request should be repeated.
-15	*Memory Inconsistent*	The registers in RAM are inconsistent with the permanently stored registers after processing SVS. The SVS request should be repeated.
-21	*UART Parse*	An error occurred while parsing the command string. The last command should be repeated.
-22	*UART Rx*	The command string was not received correctly (e.g. device was not ready, last request was not terminated correctly). Repeat the last command.
-23	*UART Header*	The command header could not be interpreted correctly (must contain only characters from A-Z). Repeat the last command.
-24	*UART Overflow*	The command string could not be processed fast enough to prevent an overflow of the internal receiving buffer
-25	*UART Baudrate*	The requested baudrate is not supported. No baudrate change took place.
-26	*UART Request*	The command header does not match any of the supported commands.
-27	*UART Start Rx*	The device was waiting for incoming data; however, the next event was not triggered by receiving a command.
-28	*UART Range*	One or more parameters of the command are out of range.
-30	*I2C Transfer*	There was an error transferring data on the I2C bus.
-40	*Temp Ext*	The communication with the sample temperature sensor was not successful.
-41	*Periphery No Power*	The power supply of the device periphery (sensors, SD card) is not switched on.

5.5 Available Implementations of Communication Protocol

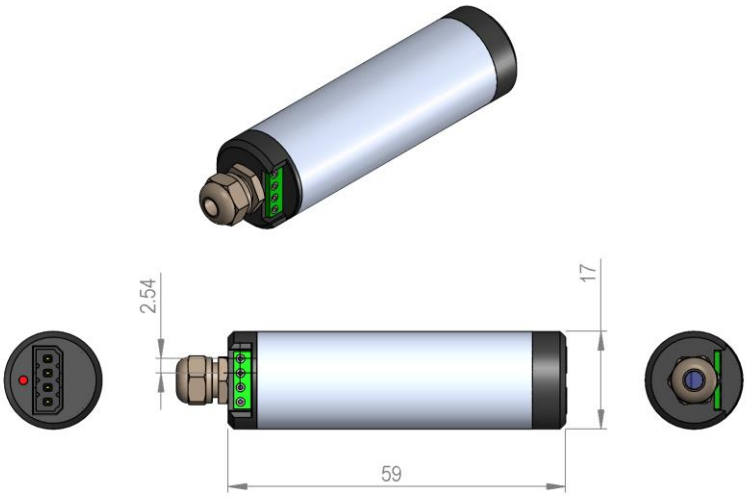
We offer libraries for controlling **Pico-T** using **LabView** programming language. The libraries and corresponding documentation are free for download from our website.

6 OPTION 4: ADVANCED CUSTOM INTEGRATION

For **advanced custom integration** the **full USB/UART communication protocol** is available on request, allowing custom software full control on all settings, calibration and measurement features of the module.

7 TECHNICAL DRAWING

The solder pads have 2.54mm pitch.



8 SPECIFICATIONS

General Specifications	
Dimensions	L=59 mm, Ø 17mm (without the optical port)
Weight	ca. 20 g
Power supply	min. 3.3 VDC max. 5.0 VDC
Connector plug	Phoenix Contact PTSM0,5/4-P-2,5
Power consumption	
-during operation	typ. 10 mA
-during deep sleep mode	typ. <100 µA (<10 µA on request)
Start-up time	
-from power off	1-2 s
-from deep sleep	ca. 200 ms
Interface	UART (3.0V levels, 5V tolerant), 19200 baud, 8 data bit, 1 stop bit, no parity, no handshake
Max. sample rate[1]	ca. 20 samples/s
Operating temperature	0 to 50 °C
Storage temperature	-20 to 70 °C
Max. relative humidity	Non-condensing conditions

Optical Temperature Sensor	Refer to the separately available specifications for the connected sensor
Port for External Temperature Sensors	
Compatible sensor types	Pt100
Measurement principle	2-wire or 4-wire resistance measurement via 24bit ADC
Resolution	<0.02 °C
Accuracy	<+-0.2 °C
Range	-30 to 150 °C

Internal Temperature Sensor	(located on internal PCB)
Resolution	0.02 °C
Accuracy	+-0.3 °C
Range	-40 to 125 °C

[1]Note: This max. sample rate refers only to the limits of the UART communication. It does not consider the actual response time of the connected optical temperature sensor or of the temperature sensor.

9 SAFETY GUIDELINES

Before using the **Pico-T** and its sensors, read carefully the instructions and user manuals.

In case of problems or damage, disconnect the instrument and mark it to prevent any further use. Consult PyroScience for advice. There are no serviceable parts inside the device. Please note that opening the housing will void the warranty.

The **Pico-T** is not watertight. The **Pico-T** should be kept under dry and clean conditions, avoiding moisture, dust, corrosive conditions and excessive heating of the instrument (e.g. direct sun light).

Calibration and application of the sensors is on the user's authority, as well as data acquisition, treatment and publication.

The sensors and the **Pico-T** are not intended for medical, aerospace or military purposes or any safety-critical applications.

The sensors should be used in the laboratory by qualified personnel only, following the user instructions and the safety guidelines of the manual, as well as the appropriate laws and guidelines for safety in the laboratory.

Keep the sensors and the **Pico-T** out of reach of children.

CONTACT

PyroScience GmbH
Hubertusstraße 35
52064 Aachen
Deutschland

Tel.: +49 (0)241 5183 2210
Fax: +49 (0)241 5183 2299
info@pyroscience.com
www.pyroscience.com