



METER

BARO INTEGRATOR GUIDE

SENSOR DESCRIPTION

The BARO Module is a precise barometer to compensate for matric potential measurements of TEROS 31 and TEROS 32 tensiometers. The BARO Module can be used as a standalone sensor to compensate one or more tensiometers at a measuring site, or as a digital/analog converter to compensate a connected TEROS 31 or TEROS 32 value and convert the SDI-12 signal into an analog voltage output (only 8-pin version). The BARO Module and TEROS 32 combination can be used as a T8 tensiometer replacement.

For a more detailed description of how this sensor makes measurements, refer to the [BARO Module User Manual](#).

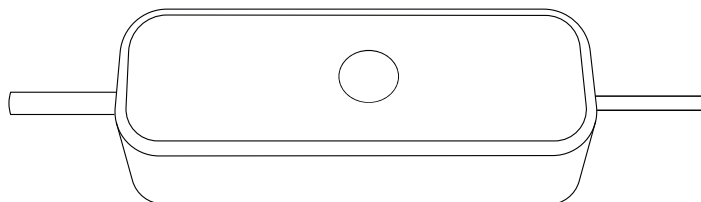


Figure 1 BARO Module

APPLICATIONS

- Barometric pressure measurement
- Barometric compensation of matric potential measurements
- Digital/analog converter for directly connected TEROS 31 and TEROS 32 tensiometers
- Appropriate for non-METER data loggers to connect TEROS 31 and TEROS 32

ADVANTAGES

- Digital sensor communicates multiple measurements over a serial interface
- Low-input voltage requirements
- Low-power design supports battery-operated data loggers
- SDI-12, Modbus RTU or tensioLINK serial communications protocol supported
- Analog output supported (only 8-pin version)

SPECIFICATIONS

MEASUREMENT SPECIFICATIONS

Barometric Pressure	
Range	+ 65 kPa to +105 kPa
Resolution	± 0.0012 kPa
Accuracy	± 0.05kPa
Temperature	
Range	-30 to + 60 °C
Resolution	± 0.01 °C
Accuracy	± 0.5 °C

COMMUNICATION SPECIFICATIONS

Output	
Analog Output (8-pin connector only)	
0 to 2,000 mV (default)	
0 to 1,000 mV (configurable with tensioVIEW)	
Digital Output	
SDI-12 communications protocol	
tensioLINK communication protocol	
Modbus RTU communication protocol	
Data Logger Compatibility	
Analog Output	
Any data acquisition system capable of switched 3.6- to 28-VDC excitation and single-ended or differential voltage measurement at a greater than or equal to 12-bit resolution.	
Digital Output	
Any data acquisition system capable of 3.6- to 28-VDC excitation and RS-485 Modbus or SDI-12 communication.	

PHYSICAL SPECIFICATIONS

Dimensions	
Length	80 mm (3.15 in)
Width	29 mm (1.14 in)
Height	30 mm (1.18 in)
Cable Length	
1.5 m (standard)	
NOTE: Contact Customer Support if a nonstandard cable length is needed.	
Connector Types	
4-pin and 8-pin M12 plug connector or stripped and tinned wires	

COMPLIANCE

EM ISO/IEC 17050:2010 (CE Mark)

EQUIVALENT CIRCUIT AND CONNECTION TYPES

Refer to [Figure 2](#) to connect the BARO Module to a data logger. [Figure 2](#) provides a low-impedance variant of the recommended SDI-12 specification.

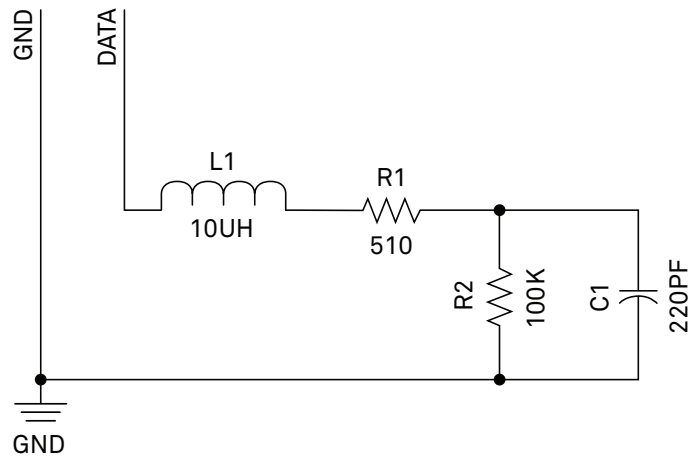


Figure 2 Equivalent circuit diagram

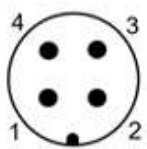
Wiring connections M12 4-pin				
Signal	Wire	Pin	Function	 4-pin male plug
V_{in}	brown	1	Supply +3.6...+28.0 V _{DC}	
RS485-A/ SDI-12	white	2	RS485-A 2-wire or SDI-12	
GND	blue	3	Supply minus	
RS485-B	black	4	RS485-B 2-wire	

Figure 3 M12 4-pin output connector

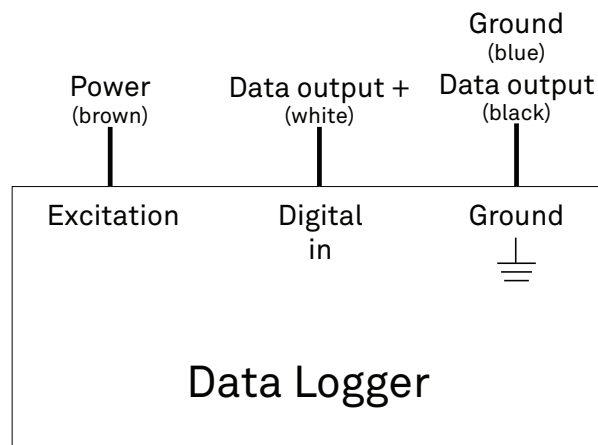


Figure 4 Wiring diagram SDI-12 4-pin connector

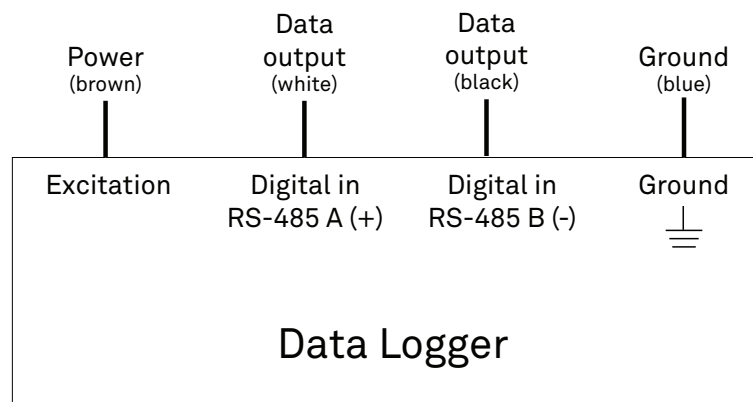


Figure 5 Wiring diagram RS-485 4-pin connector

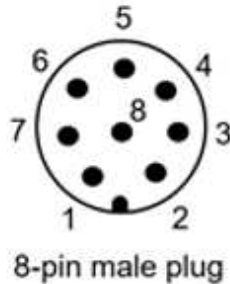
Wiring connections M12 8-pin				
Signal	Wire	Pin	Function	 8-pin male plug
V_{in}	white	1	Supply +3.6...+28.0 V _{DC}	
GND	brown	2	Supply minus	
A-OUT+1	green	3	Analog output 1 (matric potential)	
A-OUT-	yellow	4	Analog minus	
digital OUT	grey	5	Digital switching channel	
RS485-A / SDI-12	pink	6	RS485- <u>A</u> 2-wire or SDI12	
RS485-B	blue	7	RS485- <u>B</u> 2-wire	

Figure 6 M12 8-pin output connector

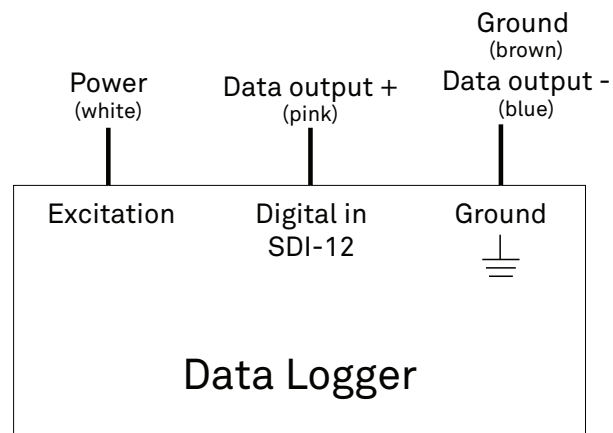


Figure 7 Wiring daigram SDI-12 8-pin connector

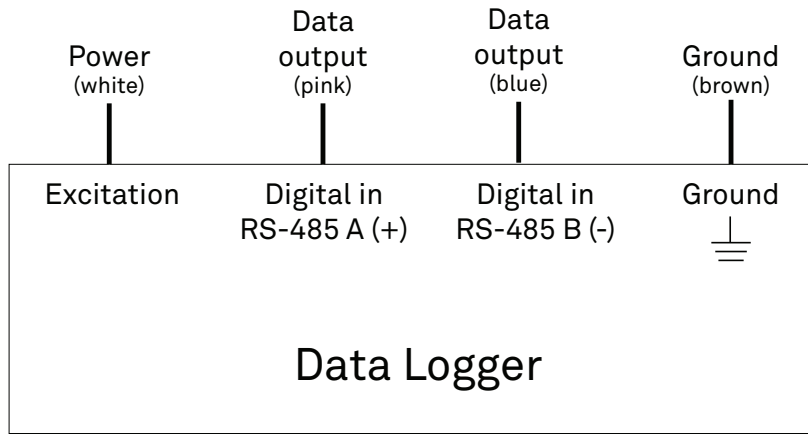


Figure 8 Wiring diagram RS-485 8-pin connector

⚠ PRECAUTIONS

METER sensors are built to the highest standards, but misuse, improper protection, or improper installation may damage the sensor and possibly void the warranty. Before integrating sensors into a sensor network, follow the recommended installation instructions and implement safeguards to protect the sensor from damaging interference.

SENSOR COMMUNICATIONS

METER digital sensors feature a serial interface with shared receive and transmit signals for communicating sensor measurements on the data wire. The sensor supports SDI-12, tensioLINK, and Modbus over RS-485 two-wire. The sensor automatically detects the interface and protocol which is being used. Each protocol has implementation advantages and challenges. Please contact METER [Customer Support](#) if the protocol choice for the desired application is not obvious.

SDI-12 INTRODUCTION

SDI-12 is a standards-based protocol for interfacing sensors to data loggers and data acquisition equipment. Multiple sensors with unique addresses can share a common 3-wire bus (power, ground, and data). Two-way communication between the sensor and logger is possible by sharing the data line for transmit and receive as defined by the standard. Sensor measurements are triggered by protocol command. The SDI-12 protocol requires a unique alphanumeric sensor address for each sensor on the bus so that a data logger can send commands to and receive readings from specific sensors.

Download the [SDI-12 Specification v1.3](#) to learn more about the SDI-12 protocol.

RS-485 INTRODUCTION

RS-485 is a robust physical bus connection to connect multiple devices to one bus. It is capable of using very long cable distances under harsh environments. Instead of SDI-12, RS-485 uses two dedicated wires for the data signal. This allows the use of longer cables and is more insensitive to interference from outside sources, since the signal is related to the different wires, and supply currents do not influence the data signal. See [Wikipedia](#) for more details on RS-485.

TENSIOLINK RS-485 INTRODUCTION

tensioLINK is a fast, reliable, proprietary serial communications protocol that communicates over the RS-485 interface. This protocol is used to read out data and configure features of the device. METER provides a tensioLINK PC USB converter and software to communicate directly with the sensor, read out data, and update the firmware. Please contact [Customer Support](#) for more information about tensioLINK.

MODBUS RTU RS-485 INTRODUCTION

Modbus RTU is a common serial communications protocol used by Programmable Logic Controllers (PLCs) or data loggers to communicate with all kinds of digital devices. The communication works over the physical RS-485 connection. The combination of RS-485 for the physical connection and Modbus as serial communications protocol allows fast and reliable data transfer for a high number of sensors connected to one serial bus wire. Use the following links for more Modbus information: [Wikipedia](#) and [modbus.org](#).

INTERFACING THE SENSOR TO A COMPUTER

The serial signals and protocols supported by the sensor require some type of interface hardware to be compatible with the serial port found on most computers (or USB-to-serial adapters). There are several SDI-12 interface adapters available in the marketplace; however, METER has not tested any of these interfaces and cannot make a recommendation as to which adapters work with METER sensors. METER data loggers and the ZSC handheld device can operate as a computer-to-sensor interface for making on-demand sensor measurements.

The BARO Module can also be configured and measured via tensioLINK using METER software tensioVIEW, available to download at meter.ly/software. To connect a BARO Module to a computer a tensioLINK USB converter and a suitable adapter cable is necessary.

METER SDI-12 IMPLEMENTATION

If a BARO Module is connected between a TEROS 31 or 32 tensiometer, both the barometric air pressure and the absolute pressure of the TEROS tensiometer can be read out via Modbus. The compensated matrix potential can be read out via Modbus as well.

METER sensors use a low-impedance variant of the SDI-12 standard sensor circuit ([Figure 2](#)). During the power-up time, sensors output some sensor diagnostic information and should not be communicated with until the power-up time has passed. After the power up time, the sensors are fully compatible with all commands listed in the [SDI-12 Specification v1.3](#) except for the continuous measurement commands (aR0 – aR9 and aRC0 – aRC9). M, R, and C command implementations are found on pages 8–9.

Out of the factory, all METER sensors start with SDI-12 address 0.

SENSOR BUS CONSIDERATIONS

SDI-12 sensor buses require regular checking, sensor upkeep, and sensor troubleshooting. If one sensor goes down, that may take down the whole bus even if the remaining sensors are functioning normally. Power cycling the SDI-12 bus when a sensor is failing is acceptable. METER SDI-12 sensors can be power-cycled and read on the desired measurement interval or powered continuously and commands sent when a measurement is desired based on specified communication timing. Many factors influence the effectiveness of the bus configuration. Visit metergroup.com for articles and virtual seminars containing more information.

SDI-12 CONFIGURATION

[Table 1](#) lists the SDI-12 communication configuration.

Table 1 SDI-12 communication configuration

Baud Rate	1,200
Start Bits	1
Data Bits	7 (LSB first)
Parity Bits	1 (even)
Stop Bits	1
Logic	Inverted (active low)

SDI-12 TIMING

All SDI-12 commands and responses must adhere to the format in [Figure 9](#) on the data line. Both the command and response are preceded by an address and terminated by a carriage return and line feed combination (<CR><LF>) and follow the timing shown in [Figure 10](#).

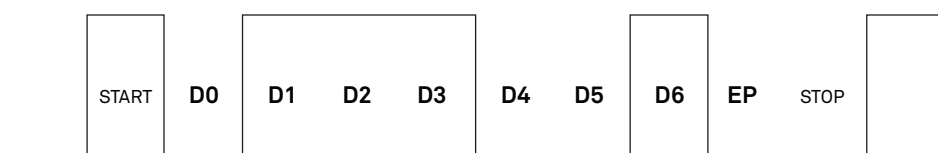


Figure 9 Example SDI-12 transmission of the character 1 (0x31)

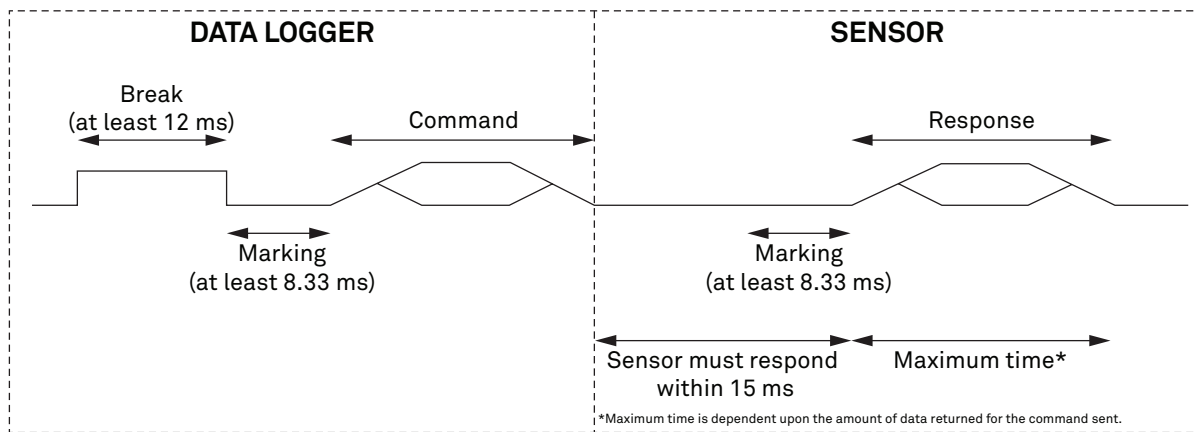


Figure 10 Example data logger and sensor communication

COMMON SDI-12 COMMANDS

This section includes tables of common SDI-12 commands that are often used in an SDI-12 system and the corresponding responses from METER sensors.

IDENTIFICATION COMMAND (**aI!**)

The Identification command can be used to obtain a variety of detailed information about the connected sensor. An example of the command and response is shown in [Example 1](#), where the command is in **bold** and the response follows the command.

Example 1 **1I!**113METER_ _ _BARO_

Parameter	Fixed Character Length	Description
1I!	3	Data logger command. Request to the sensor for information from sensor address 1 .
1	1	Sensor address. Prepended on all responses, this indicates which sensor on the bus is returning the following information.
13	2	Indicates that the target sensor supports SDI-12 Specification v1.3 .
METER_ _ _	8	Vendor identification string. (METER and three spaces _ _ _ for all METER sensors)
BARO_	6	Sensor model string. This string is specific to the sensor type. For the BARO, the string is BARO.
100	3	Sensor version. This number divided by 100 is the METER sensor version (e.g., 100 is version 1.00).
BARO-00001	≤13, variable	Sensor serial number. This is a variable length field. It may be omitted for older sensors.

CHANGE ADDRESS COMMAND (**aAB!**)

The Change Address command is used to change the sensor address to a new address. All other commands support the wildcard character as the target sensor address except for this command. All METER sensors have a default address of **0** (zero) out of the factory. Supported addresses are alphanumeric (i.e., **A**–**Z**, and **0**–**9**). An example output from a METER sensor is shown in [Example 2](#), where the command is in **bold** and the response follows the command.

Example 2 1A0!0

Parameter	Fixed Character Length	Description
1A0!	4	Data logger command. Request to the sensor to change its address from 1 to a new address of 0.
0	1	New sensor address. For all subsequent commands, this new address will be used by the target sensor.

ADDRESS QUERY COMMAND (?!)

While disconnected from a bus, the Address Query command can be used to determine which sensors are currently being communicated with. Sending this command over a bus will cause a bus contention where all the sensors will respond simultaneously and corrupt the data line. This command is helpful when trying to isolate a failed sensor. [Example 3](#) shows an example of the command and response, where the command is in **bold** and the response follows the command. The question mark (?) is a wildcard character that can be used in place of the address with any command except the Change Address command.

Example 3 ?!0

Parameter	Fixed Character Length	Description
?!	2	Data logger command. Request for a response from any sensor listening on the data line.
0	1	Sensor address. Returns the sensor address to the currently connected sensor.

COMMAND IMPLEMENTATION

The following tables list the relevant Measurement (M), Continuous (R), and Concurrent (C) commands and subsequent Data (D) commands, when necessary.

MEASUREMENT COMMANDS IMPLEMENTATION

Measurement (M) commands are sent to a single sensor on the SDI-12 bus and require that subsequent Data (D) commands are sent to that sensor to retrieve the sensor output data before initiating communication with another sensor on the bus.

Please refer to [Table 2](#) and for an explanation of the command sequence and to [Table 5](#) for an explanation of response parameters.

Table 2 aM! command sequence

Command	Response
This command reports average, accumulated, or maximum values.	
aM!	atttn
aD0!	a±<Press>±<Temp>+<Status>
Comments	When a slave TEROS tensiometer is connected, <Press> hold the barometric compensated tensiometer output. If the BARO module is used in standalone <Press> returns the current barometric pressure.
NOTE: The measurement and corresponding data commands are intended to be used back to back. After a measurement command is processed by the sensor, a service request a <CR><LF> is sent from the sensor signaling the measurement is ready. Either wait until ttt seconds have passed or wait until the service request is received before sending the data commands. See the SDI-12 Specifications v1.3 document for more information.	

CONCURRENT MEASUREMENT COMMANDS IMPLEMENTATION

Concurrent Measurement (C) commands are typically used with sensors connected to a bus. C commands for this sensor deviate from the standard C command implementation. First, send the C command, wait the specified amount of time detailed in the C command response, and then use D commands to read its response prior to communicating with another sensor.

Please refer to [Table 3](#) for an explanation of the command sequence and to [Table 5](#) for an explanation of response parameters.

Table 3 aC! measurement command sequence

Command	Response
This command reports instantaneous values.	
aC!	atttnn
aD0!	a±<Press>±<Temp>+<Status>
NOTE: The measurement and corresponding data commands are intended to be used back to back. After a measurement command is processed by the sensor, a service request a<CR><LF> is sent from the sensor signaling the measurement is ready. Either wait until ttt seconds have passed or wait until the service request is received before sending the data commands. Please see the SDI-12 Specifications v1.3 document for more information.	

CONTINUOUS MEASUREMENT COMMANDS IMPLEMENTATION

Continuous Measurement (R) commands trigger a sensor measurement and return the data automatically after the readings are completed without needing to send a D command.

aR0! returns more characters in its response than the 75-character limitation called out in the [SDI-12 Specification v1.3](#). It is recommended to use a buffer that can store at least 116 characters.

Please refer to [Table 4](#) for an explanation of the command sequence and see [Table 5](#) for an explanation of response parameters.

Table 4 aR0! measurement command sequence

Command	Response
This command reports average, accumulated, or maximum values.	
aR0!	a±<Press>±<Temp>+<Status>
NOTE: This command does not adhere to the SDI-12 response timing. See METER SDI-12 Implementation for more information.	

PARAMETERS

[Table 5](#) lists the parameters, unit measurement, and a description of the parameters returned in command responses for the BARO Module.

Table 5 Parameter Descriptions

Parameter	Unit	Description
±	—	Positive or negative sign denoting sign of the next value
a	—	SDI-12 address
n	—	Number of measurements (fixed width of 1)
nn	—	Number of measurements with leading zero if necessary (fixed width of 2)
ttt	s	Maximum time measurement will take (fixed width of 3)
<TAB>	—	Tab character
<CR>	—	Carriage return character
<LF>	—	Line feed character
<sensorType>	—	ASCII character denoting the sensor type For BARO Module, the character is ;
<Checksum>	—	METER serial checksum
<CRC>	—	METER 6-bit CRC

METER MODBUS RTU SERIAL IMPLEMENTATION

Modbus over Serial Line is specified in two versions - ASCII and RTU. BARO Modules communicate using RTU mode exclusively. The following explanation is always related to RTU. [Table 6](#) lists the Modbus RTU communication and configuration.

Table 6 Modbus communication characters

Baud Rate (bps)	9,600 bps
Start Bits	1
Data Bits	8 (LSB first)
Parity Bits	0 (none)
Stop Bits	1
Logic	Standard (active high)

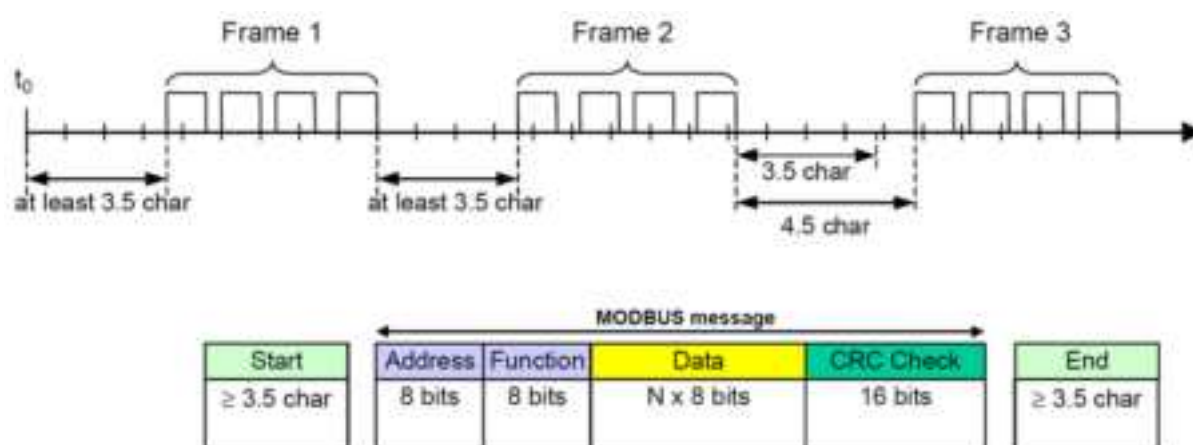
**Figure 11 RTU Message Frame**

Figure 11 shows a message in RTU format. The size of the data determines the length of the message. The format of each byte in the message has 10 bits, including Start and Stop Bit. Each byte is sent from left to right: Least Significant Bit (LSB) to Most Significant Bit (MBS). If no parity is implemented, an additional stop bit is transmitted to fill out the character frame to a full 11-bit asynchronous character.

The Modbus application layer implements a set of standard Function codes divided into three categories: Public, User-defined, and Reserved. Well-defined public function codes for BARO Modules are documented in the Modbus Organization, Inc. (modbus.org) community.

For a reliable interaction between the BARO Module and a Modbus Master, a **minimum 50ms** delay is required between every Modbus command sent on the RS-485 bus. An additional timeout is needed for every Modbus query; this timeout is device-specific and depends on the quantity of the polled registers. Generally, **100ms** will work fine for most of the BARO Module.

SUPPORTED MODBUS FUNCTIONS

Table 7 Function Definitions

Function Code	Action	Description
01	Read coil/port status	Reads the on/off status of discrete output(s) in the ModBusSlave
02	Read input status	Reads the on/off status of discrete input(s) in the ModBusSlave
03	Read holding registers	Reads the binary contents of holding register(s) in the ModBusSlave
04	Read input registers	Reads the binary contents of input register(s) in the ModBusSlave
05	Force single coil/port	Forces a single coil/port in the ModBusSlave to either on or off
06	Write single register	Writes a value into a holding register in the ModBusSlave
15	Force multiple coils/ports	Forces multiple coils/ports in the ModBusSlave to either on or off
16	Write multiple registers	Writes values into a series of holding registers in the ModBusSlave

DATA REPRESENTATION AND REGISTER TABLES

Data values (setpoint values, parameters, sensor-specific measurement values, etc.) sent to and from the BARO Module use 16-bit and 32-bit holding (or input) registers with a 4-digit address notation. The address spaces are virtually distributed in different blocks for each data type. This is an approach to the Modbus Enron implementation. [Table 8](#) shows the four main tables used by the BARO Module with their respective access rights. [Table 9](#) describes the sub-blocks for each different data type representation.

Please note that some Modbus dataloggers use addressing with a +1 offset. This sometimes causes confusion and is based on a Modbus specification void. If there are problems in implementing your Modbus program on the datalogger always try testing different register offsets and data types. Using a known value, like temperature, where it's known what value to expect is a good practice to start testing.

Table 8 Modbus Primary Tables

Register Number	Table Type	Access	Description
1xxx	Discrete Output Coils	Read/Write	on/off status or setup flags for the sensor
2xxx	Discrete Input Contacts	Read	sensor status flags
3xxx	Analog Input Registers	Read	numerical input variables from the sensor (actual sensor measurements)
4xxx	Analog Output Holding Registers	Read/Write	numerical output variables for the sensor (parameters, setpoint values, calibrations etc.)

For example, register 3001 is the first analog input register (first data address for the input registers). The numeric value stored here would be a 16-bit unsigned integer-type variable that represents the first sensor measurement parameter (pressure value). The same measurement parameter (pressure value) could be read at register 3201, but this time as a 32-bit floating-point value with a Big-Endian format. If the Modbus Master (Datalogger or a PLC) supports only 32-bit float-values with a Little-Endian format, then one could read the same measurement parameter (same pressure value) at register 3301. The Virtual Sub-Blocks are meant to simplify the user's effort in programming the Modbus query of the sensors.

Table 9 Modbus Virtual Sub-Blocks

Register Number	Access	Size	Sub-Table Data Type
X001-X099	Read/Write	16 bit	signed integer
X101-X199	Read/Write	16 bit	unsigned integer
X201-X299	Read/Write	32 bit	float Big-Endian format
X301-X399	Read/Write	32 bit	float Little-Endian format

REGISTER MAPPING

Table 10 Holding Registers

41000 (41001*)	Modbus Slave Address
Detailed Description	Read or update the sensor's modbus address
Data Type	Unsigned integer
Allowed Range	1 - 247
Unit	-
Comments	Updated slave address will be stored in the sensor's nonvolatile memory

Table 11 BARO Module Input Registers

32000 (32001*)	Soil Water Potential
Detailed Description	Compensated tension value from tensiometer
Data Type	32 bit floating Big-Endian
Allowed Range	-200 to +200
Unit	kPa
Comments	Tensiometer needs to be connected as slave

32001 (32002*)	Soil Temperature
Detailed Description	High accuracy on board temperature measurement
Data Type	32 bit floating Big-Endian
Allowed Range	-30 to +60
Unit	degC
Comments	Tensiometer needs to be connected as slave

32002 (32003*)	Sensor Supply Voltage
Detailed Description	On board supply voltage measurement
Data Type	32 bit floating Big-Endian
Allowed Range	-10 to +60
Unit	Volts
Comments	-

32003 (32004*)	BARO Status
Detailed Description	Binary status
Data Type	32 bit floating Big-Endian
Allowed Range	0/1
Unit	-
Comments	-

32004 (32005*)	BARO Reference Pressure
Detailed Description	On board high accuracy barometric pressure measurement
Data Type	32 bit floating Big-Endian
Allowed Range	+70 to +120
Unit	kPa
Comments	-

Table 11 Baro Module Input Registers (continued)

32005 (32006*)	Tensiometer Pressure
Detailed Description	Absolute pressure value from tensiometer
Data Type	32 bit floating Big-Endian
Allowed Range	-200 to +200
Unit	kPa
Comments	Tensiometer needs to be connected as slave

32006 (32007*)	BARO Temperature
Detailed Description	On board temperature measurement
Data Type	32 bit floating Big-Endian
Allowed Range	-30 to +60
Unit	degC
Comments	-

*Some devices report Modbus register addresses with an offset of +1. This is true for Campbell Scientific Loggers and Dataloggers. In order to read the desired register use the number in the parenthesis.

EXAMPLE USING A CR6 DATALOGGER AND MODBUS RTU

The Campbell Scientific, Inc. CR6 Measurement and Control Datalogger supports Modbus master and Modbus slave communication to integrate Modbus SCADA networks. The Modbus communications protocol facilitates the exchange of information and data between a computer/HMI software, instruments (RTUs), and Modbus-compatible sensors. The CR6 datalogger communicates exclusively in RTU mode. In a Modbus network, each slave device has a unique address. Therefore, sensor devices must be configured correctly before connecting to a Modbus Network. Addresses range from 1 to 247. Address 0 is reserved for universal broadcasts.

PROGRAMMING A CR6 DATALOGGER

The Programs running on the CR6 (and CR1000) Loggers are written in CRBasic, a language developed by Campbell Scientific. It is a high-level language designed to provide an easy yet extremely flexible and powerful method of instructing the data logger how and when to take measurements, process data, and communicate. Programs can be created using either the ShortCut Software or edited using the CRBasic Editor, both of which are available for downloading as stand-alone applications on the official [Campbell Scientific website](http://www.campbellsci.com) (www.campbellsci.com).

[ShortCut Software](https://www.campbellsci.com/shortcut) (<https://www.campbellsci.com/shortcut>)

[CRBasic Editor](https://www.campbellsci.com/crbasiceditor) (<https://www.campbellsci.com/crbasiceditor>)

A typical CRBasic program for a Modbus application consists of the following:

- Variables and constants declarations (public or private)
- Units declarations
- Configuration parameters
- Data tables declarations
- Logger Initializations
- Scan (Main Loop) with all the sensors to be queried
- Function call to the Data Tables

CR6 LOGGER RS-485 CONNECTION INTERFACE

The universal (U) terminal of the CR6 offers 12 Channels that connect to nearly any sensor type. It gives the CR6 the ability to match more applications and eliminates the use of many external peripherals.

The Modbus CR6 connection shown in [Figure 12](#) uses the RS-485 (A/B) interface mounted on terminals (C1-C2) and (C3-C4). These interfaces can operate in Half-Duplex and Full-Duplex. The serial interface of the BARO Module used for this example is connected to (C1-C2) terminals.

BARO Module to CR6 Datalogger Wiring Diagram

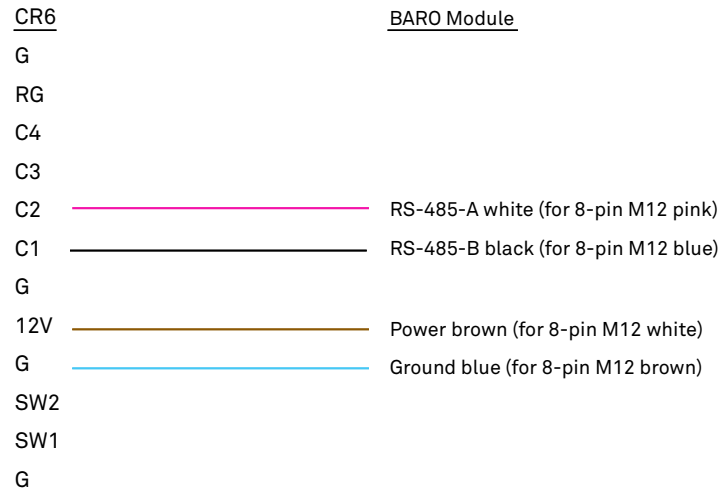


Figure 12 RS-485 interface

After assigning the BARO Module a unique Modbus Slave Address, it can be wired to the CR6 logger according to [Figure 12](#). Make sure to connect the white and black wires according to their signals, respectively, to the C1 and C2 ports—the brown wire to 12V (V+) and the blue to G (GND). To control the power supply through your program, connect the brown wire directly to one of the SW12 terminals (switched 12V outputs).

EXAMPLE PROGRAMS

```

'CR6 Datalogger
'This is an example program for reading out the BARO Module using a CR6
'datalogger and the MODBUS RTU protocol over a RS-485 Bus. The measurement values polled
'from the sensor will be: Water Potential, Temperature and the Sensor's Supply voltage.

'This program runs a scan every 1 Min and stores the data in a 1 Min table.

'Declare Constants
Const BARO_MB_ADDR=1      'BARO Module Modbus slave address
Const MB_TIMEOUT= 10      '100ms timeout (value * 0.01sec)
Const MB_RETRIES= 1

'Declare Public Variables
Public PTemp, batt_volt
Public mb_statu 'variable used for monitoring the modbus poll status

'Declare Private Variables
Dim Measurements (3)      'array for holding the measurements values read from the sensor

'Aliases used for the BARO Module
Alias Measurements (1)= Water Potential
Alias Measurements (2)= Temperature
Alias Measurements (3)= Sensor Supply

'Declare Units
Units Water Potential=kPa
Units Temperature=degC
Units Sensor supply=V

'Define Data Tables.
DataTable (BARO_Table,1,-1) 'Set table size to # of records, or -1 to auto allocate.
DataInterval (0,1,Min,10)   'Store new measurement every 1 Minute
Minimum (1,batt_volt,FP2,False,False)
Sample (1,PTemp,FP2)
Sample (3,Measurements(),IEEE4)

EndTable

'Main Program

BeginProg

SW12(2,1)      'Switch ON the SW12-2 terminal (if used for powering the BARO Module)

SerialOpen(ComC1,9600,3,2,50,4)      'open communication port, setup for RS-485
                                      '(BaudRate, Format, TXDelay, BufferSize, CommsMode)

Scan (1,Min,0,0)      'Scan Loop
PanelTemp (PTemp,15000)
Battery (batt_volt)

'Read multiple Input registers from the BARO Module using a 32 bit float, Big-Endian
'format
ModbusMaster(mb_status,ComC1,9600,BARO_MB_ADDR,4,Measurements(),32001,3,MB_RETRIES,MB_
TIMEOUT,2)

'Call Output Tables
CallTable BARO_Table

NextScan

EndProg

```

CUSTOMER SUPPORT

NORTH AMERICA

Customer service representatives are available for questions, problems, or feedback Monday through Friday, 7:00 am to 5:00 pm Pacific time.

Email: support.environment@metergroup.com
sales.environment@metergroup.com

Phone: +1.509.332.5600

Fax: +1.509.332.5158

Website: metergroup.com

EUROPE

Customer service representatives are available for questions, problems, or feedback Monday through Friday, 8:00 to 17:00 Central European time.

Email: support.europe@metergroup.com
sales.europe@metergroup.com

Phone: +49 89 12 66 52 0

Fax: +49 89 12 66 52 20

Website: metergroup.com

If contacting METER by email, please include the following information:

Name	Email address
Address	Instrument serial number
Phone number	Description of problem

NOTE: For products purchased through a distributor, please contact the distributor directly for assistance.

REVISION HISTORY

The following table lists document revisions.

Revision	Date	Compatible Firmware	Description
00	6.2025	1.10	Initial release