METER

TEROS 06 INTEGRATOR GUIDE

SENSOR DESCRIPTION

The TEROS 06 is a precision temperature profile probe that measures soil temperature at six depths following the World Meteorological Organization recommendations. With a small diameter of 20 mm, the soil disturbance is minimized. The TEROS 06 is designed to be installed down an augered hole, plugged into a data logger, and left to log soil temperature data.

For a more detailed description of how this sensor makes measurements, refer to the TEROS 06 User Manual.

APPLICATIONS

- Soil temperature measurement

ADVANTAGES

- Three-wire sensor interface: power, ground, and data (for SDI-12)
- Digital sensor communicates multiple measurements over a serial interface
- Low-input voltage requirements
- Low-power design supports battery-operated data loggers
- SDI-12 or DDI serial communications protocols supported
- Modbus RTU or tensioLINK serial communications protocol supported

PURPOSE OF THIS GUIDE

METER Group provides this integrator guide to help TEROS 06 customers establish communication between these sensors and data acquisition equipment or field data loggers. Customers using data loggers that support SDI-12 sensor communications should consult the data logger user manual. This sensor is fully integrated into the METER system of plug-and-play sensors, cellular-enabled data loggers, and data analysis software. For more information about METER integrated systems, please contact Customer Support.

COMPATIBLE FIRMWARE VERSIONS

This guide is compatible with firmware versions 1.09 or newer.

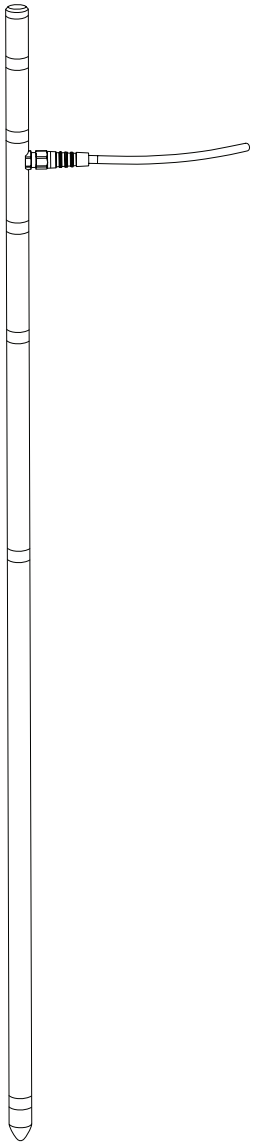

Figure 1 TEROS 06 Soil Profile Temperature Probe

METER Group, Inc.
2365 NE Hopkins Court, Pullman, WA 99163
T +1.509.332.2756 **F** +1.509.332.5158
E info@metergroup.com **W** metergroup.com

SPECIFICATIONS

MEASUREMENT SPECIFICATIONS

Temperature	
Range	–20 to +50 °C
Resolution	±0.03 °C
Accuracy	±0.1 °C (0 to 30 °C) ±0.2 °C (–20 to 50 °C)

Measurement Depths

5 cm, 10 cm, 20 cm, 30 cm, 50 cm, 100 cm

COMMUNICATION SPECIFICATIONS

Output

SDI-12 one-wire serial interface
- DDI serial communications protocol
- SDI-12 communications protocol

RS-485 two-wire serial interface (M12 connector only)
- tensioLINK serial communications protocol
- Modbus RTU communications protocol

Data Logger Compatibility

METER ZL6 data loggers and any data acquisition system capable of 3.9- to 28.0-VDC power and serial interface with SDI-12 and/or RS-485 interface, Modbus RTU, or tensioLINK communication

PHYSICAL SPECIFICATIONS

Dimensions	
Length	104.0 cm (40.94 in)
Diameter	2.0 cm (0.79 in)

Operating Temperature Range	
Minimum	–30 °C
Maximum	60 °C

Materials	
Shaft	PA66GF30
Sensor elements	Stainless steel

Cable Length

4.5 m (standard)
75 m (maximum custom cable length using M12 connector and elongation cables)

Connector Types

3.5-mm stereo plug connector

4-pin M12 plug connector

NOTE: If a stripped and tinned wire is needed, please contact Example Using a CR6 Datalogger and Modbus RTU.

ELECTRICAL AND TIMING CHARACTERISTICS

Supply Voltage (VCC to GND)	
Minimum	3.9 VDC continuous
Typical	
Maximum	28.0 VDC continuous

Digital Input Voltage (Logic High)	
Minimum	1.9 V
Typical	3.6 V
Maximum	5.0 V

Digital Input Voltage (Logic Low)	
Minimum	–0.3 V
Typical	0.0 V
Maximum	1.0 V

Digital Output Voltage (Logic High)	
Minimum	NA
Typical	3.6 V
Maximum	NA

Power Line Slew Rate	
Minimum	1.0 V/ms
Typical	NA
Maximum	NA

Current Drain (During Measurement)	
Minimum	2.0 mA
Typical	2.5 mA
Maximum	4.0 mA

Current Drain (While Asleep)	
Minimum	NA
Typical	80.0 µA
Maximum	NA

Power Up Time (DDI Serial Ready)	
Minimum	210 ms
Typical	NA
Maximum	400 ms

Power Up Time (SDI-12 Ready, DDI Disabled)	
Minimum	210 ms
Typical	NA
Maximum	400 ms

Power Up Time (RS-485 Ready, DDI Disabled)	
Minimum	210 ms
Typical	NA
Maximum	400 ms

Measurement Duration	
Minimum	30 ms
Typical	NA
Maximum	240 ms

COMPLIANCE

EM ISO/IEC 17050:2010 (CE Mark)

EQUIVALENT CIRCUIT AND CONNECTION TYPES

Refer to Figure 2 and Figure 4 to connect the TEROS 12 to a data logger. Figure 2 provides a low-impedance variant of the recommended SDI-12 specification.

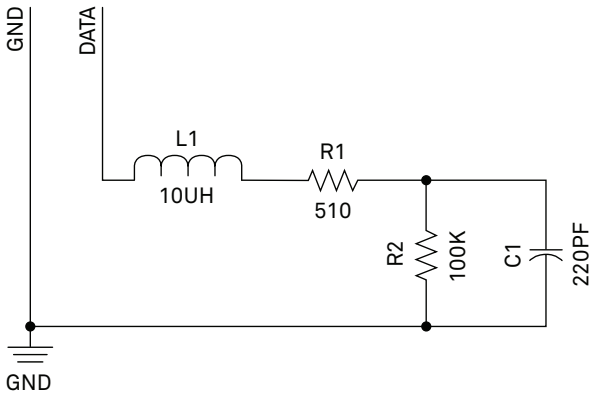

Figure 2 Equivalent circuit diagram

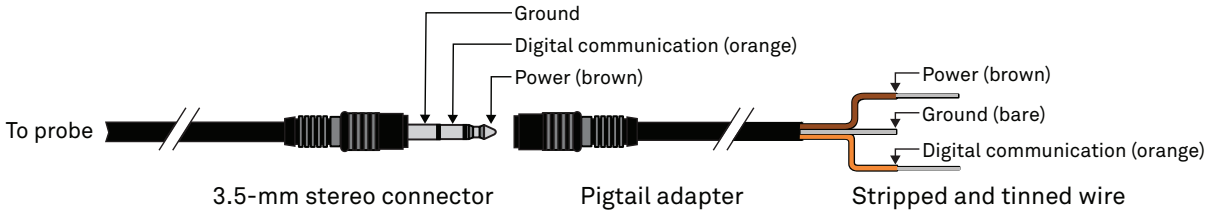

Figure 3 Three-wire stereo connector and pigtail adapter

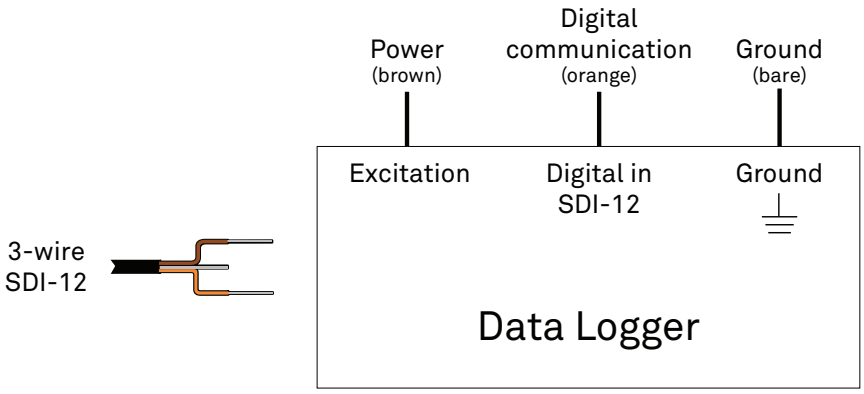

Figure 4 Three-wire SDI-12 pigtail wiring diagram

FOUR-WIRE VERSION

TEROS 06 sensors can also be ordered with a 4-pin M12 connector and optional pigtail adapter.

Connect the TEROS 06 wires to the data logger as listed below and illustrated in Figure 5, Figure 6, and Figure 7.

- Supply wire (brown) connected to the excitation.
- Digital out wire (white) connected to digital input (SDI-12 or RS-485 A).
- Digital out wire negative (black) connected to digital input (RS-485 B).
- Ground wire (blue) connected to the ground.
- Optionally, the screen wire (bare) can be connected to the ground for shielding when using long cables.

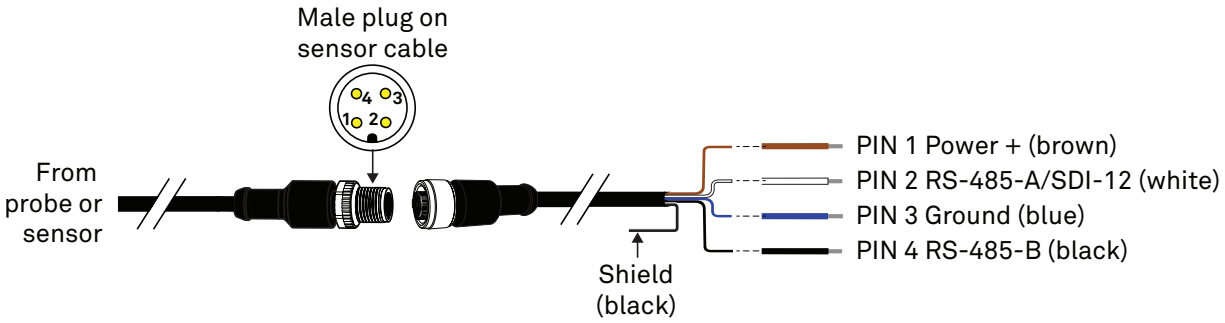

Figure 5 Four-wire M12 connector and pigtail adapter for use with screw terminal

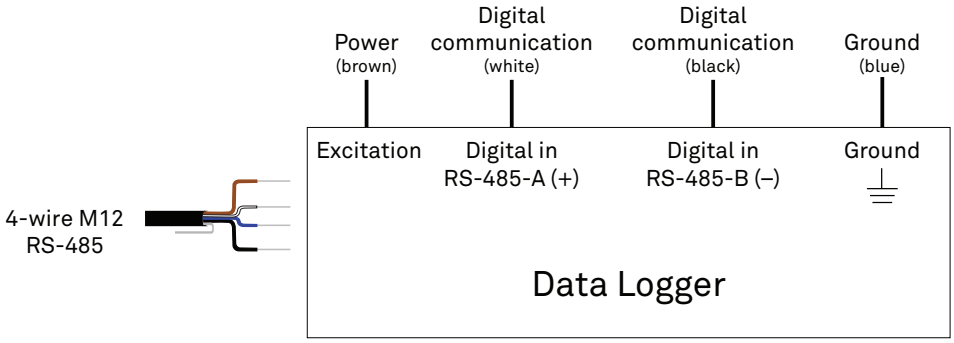

Figure 6 Four-wire M12 connector RS-485 wiring diagram

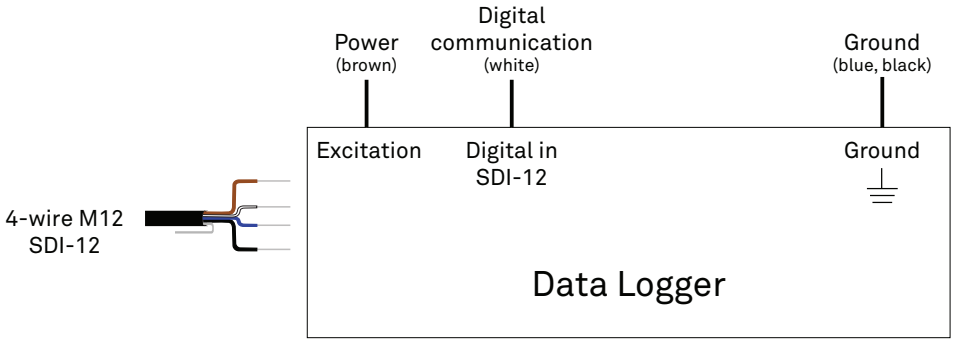

Figure 7 Four-wire M12 connector SDI-12 wiring diagram

⚠ 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.

SURGE CONDITIONS

Sensors have built-in circuitry that protects them against common surge conditions. Installations in lightning-prone areas, however, require special precautions, especially when sensors are connected to a well-grounded third-party logger.

Read the application note Lightning surge and grounding practices on the METER website for more information.

POWER AND GROUNDING

Ensure there is sufficient power to simultaneously support the maximum sensor current drain for all the sensors on the bus. The sensor protection circuitry may be insufficient if the data logger is improperly powered or grounded. Refer to the data logger installation instructions. Improper grounding may affect the sensor output as well as sensor performance.

Read the application note Lightning surge and grounding practices on the METER website for more information.

CABLES

Improperly protected cables can lead to severed cables or disconnected sensors. Cabling issues can be caused by many factors, including rodent damage, driving over sensor cables, tripping over the cable, not leaving enough cable slack during installation, or poor sensor wiring connections. To relieve strain on the connections and prevent loose cabling from being inadvertently snagged, gather and secure the cable travelling between the TEROS 06 and the data acquisition device to the mounting mast in one or more places. Install cables in conduit or plastic cladding when near the ground to avoid rodent damage. Tie excess cable to the data logger mast to ensure cable weight does not cause sensor to unplug.

SENSOR COMMUNICATIONS

METER digital sensors feature a serial interface with shared receive and transmit signals for communicating sensor measurements on the data wire (Figure 3 and Figure 4). The sensor supports two different protocols: SDI-12 and DDI serial. Each protocol has implementation advantages and challenges. Please contact 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.

DDI SERIAL INTRODUCTION

The DDI serial protocol is the method used by the METER data loggers for collecting data from the sensor. This protocol uses the data line configured to transmit data from the sensor to the receiver only (simplex). Typically, the receive side is a microprocessor UART or a general-purpose I/O pin using a bitbang method to receive data. Sensor measurements are triggered by applying power to the sensor.

RS-485 INTRODUCTION (4-WIRE VERSION ONLY)

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. TEROS 06 uses a 2-wire, half-duplex implementation of RS-485. Two wires are used for supply and two wires for the differential serial interface. One of the serial wires is also overlayed with the SDI-12 data wire. The sensor recognizes a command depending on the protocol that is used to issue a command. 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 the Wikipedia entry for more details on RS-485 or visit modbus.org.

TENSIOLINK RS-485 INTRODUCTION (4-WIRE VERSION ONLY)

The 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 (4-WIRE VERSION ONLY)

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 or PROCHECK handheld devices can operate as a computer-to-sensor interface for making on-demand sensor measurements. For more information, please contact Customer Support .

METER SDI-12 IMPLEMENTATION

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 9–10. The `aR3` and `aR4` commands are used by METER systems and as a result use a space delimiter, instead of a sign delimiter as required by the SDI-12 standard.

Out of the factory, all METER sensors start with SDI-12 address `0` and print out the DDI serial startup string during the power-up time. This can be interpreted by non-METER SDI-12 sensors as a pseudo-break condition followed by a random series of bits.

The TEROS 06 will omit the DDI serial startup string when the SDI-12 address is nonzero. Changing the address to a nonzero address is recommended for this reason.

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, but METER does not recommend scheduling power cycling events on an SDI-12 bus more than once or twice per day. Many factors influence the effectiveness of the bus configuration. Visit metergroup.com for articles and virtual seminars containing more information.

SENSOR ERROR CODE

The TEROS 06 has one error code: -9999. This error code is output in place of the measured value if the sensor detects that the measurement function has been compromised and the subsequent measurement values have no meaning.

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 8 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>`). SDI-12 commands and responses will follow the timing shown in Figure 9.

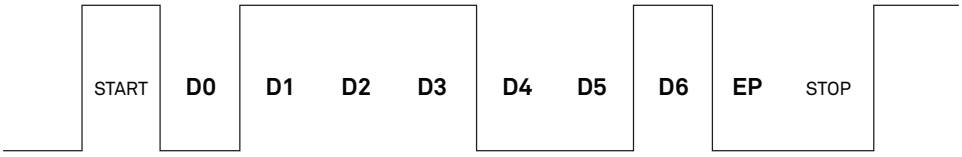

Figure 8 Example SDI-12 transmission of the character 1 (`0x31`)

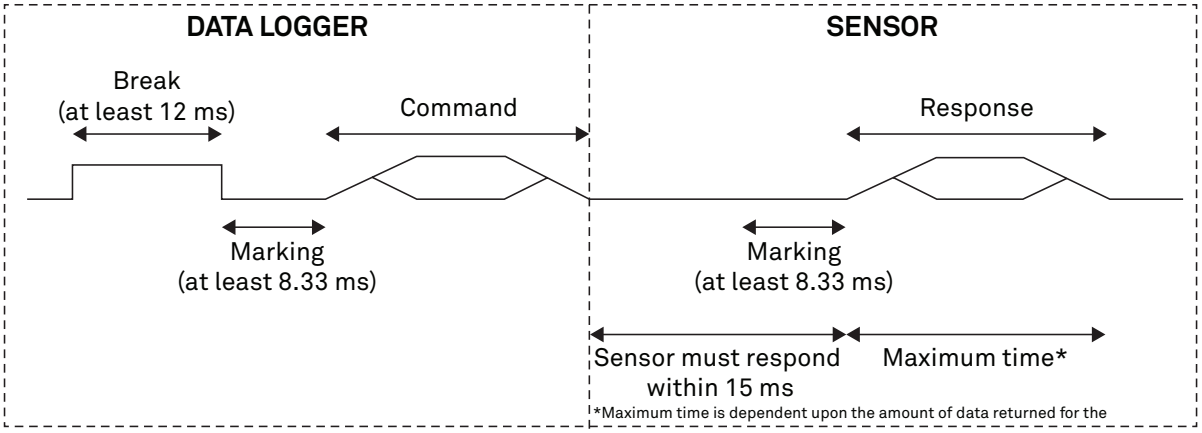

Figure 9 Example data logger and sensor communication

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␣␣␣TER06_100T06-32165`

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)
`TER06␣␣`	6	Sensor model string. This string is specific to the sensor type. For the TEROS 06 the string is TER06.
`100`	3	Sensor version. This number divided by 100 is the METER sensor version (e.g., 100 is version 1.00).
`T06-321656`	≤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`, `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 9 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±<Temp1>±<Temp2>±<Temp3>±<Temp4>±<Temp5>±<Temp6>

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 9 for an explanation of response parameters.

Table 3 aC! measurement command sequence

Command	Response
This command reports instantaneous values.	
aC!	atttnn
aD0!	a±<Temp1>±<Temp2>±<Temp3>±<Temp4>±<Temp5>±<Temp6>

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.

VERIFICATION COMMAND IMPLEMENTATION

The Verification (V) command is intended to give users a means to determine information about the current state of the sensor. The V command is sent first, followed by D commands to read the response.

Table 4 aV! measurement command sequence

Command	Response
This command reports instantaneous values.	
aV!	attn
aD0!	a+<meta>

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!, aR3!, and aR4! return more characters in their responses 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 5 and Table 6 for an explanation of the command sequence and see Table 9 for an explanation of response parameters.

Table 5 aR0! measurement command sequence

Command	Response
This command reports average, accumulated, or maximum values.	
aR0!	a±<Temp1>±<Temp2>±<Temp3>±<Temp4>±<Temp5>±<Temp6>

NOTE: This command does not adhere to the SDI-12 response timing. See METER SDI-12 Implementation for more information.

EXTENDED COMMANDS IMPLEMENTAITON

Extended (X) commands provide sensors with a means of performing manufacturer-specific functions. Additionally, the extended commands are utilized by METER systems and use a different response format than standard SDI-12 commands. X commands are required to be prefixed with the address and terminated with an exclamation point. Responses are required to be prefixed with the address and terminated with <CR><LF>.

METER implements the following X commands:

- aXRx! to trigger a sensor measurement and return the data automatcially after the readings are completed without needing to send additional commands.
- aXO! (with capital O) to supress the DDI serial string.

Please refer to Table 6 through Table 8 for an explanation of the command sequence and see Table 9 for an explanation of response parameters.

Table 6 aXO! measurement command sequence

Command	Response
aXO!	a<supressionState>
aXO<supressionState	aOK

NOTE: This command uses capital O as in Oscar (not zero). See METER SDI-12 Implementation for more information.

Table 7 aXR3! measurement command sequence

Command	Response
This command reports instantaneous values.	
aXR3!	a<TAB><temp_D1>±<temp_D2>±<temp_D3>±<temp_D4>±<temp_D5>±<temp_D6><CR><sensorType><Checksum><CRC>

NOTE: This command does not adhere to the SDI-12 response format or timing. The values in this command are space delimited. As such a + sign is not assigned between values, and a – sign is only present if the value is negative. See METER SDI-12 Implementation for more information.

Table 8 aXR4! measurement command sequence

Command	Response
This command reports instantaneous values.	
aXR4!	a<TAB><temp_D1>±<temp_D2>±<temp_D3>±<temp_D4>±<temp_D5>±<temp_D6>CR><sensorType><Checksum><CRC>

NOTE: This command does not adhere to the SDI-12 response format or timing. The values in this command are space delimited. As such a + sign is not assigned between values, and a – sign is only present if the value is negative. See METER SDI-12 Implementation for more information.

PARAMETERS

Table 9 lists the parameters, unit measurement, and a description of the parameters returned in command responses for TEROS 06.

Table 9 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
<temp_Dx>	°C	Sensor temperature at depth Dx
<meta>	—	Auxiliary sensor information 0: no sensor error 1: sensor has experienced temperatures below freezing 16: sensor refill orientation error 17: both 1 and 16
<suppressionState>	—	0: DDI serial string unsuppressed 1: DDI serial string suppressed
<sensorType>	—	ASCII character denoting the sensor type. For TEROS 06, the character is 3.
<Checksum>	—	METER serial checksum
<CRC>	—	METER 6-bit CRC

SENSOR METADATA VALUE

The sensor metadata value contains information to help alert users to sensor-identified conditions that may compromise optimal sensor operation. The output of the aV!, aDO! sequence will output a <meta> integer value. This integer represents a binary bitfield, with each individual bit representing an error flag. Below are the possible error flags that can be set by the TEROS 06. If multiple error flags are set, the sensor metadata integer value will be the sum of their individual values. To decode an integer value not explicitly called out in Table 11, find the largest error flag value in the table that will fit in the integer value and accept that error as being present. Then, subtract that error flag value from the integer value and repeat the process on the remainder until the result is 0. For example, a sensor metadata integer value of 273 is the sum of individual error flag values 256 + 16 + 1, so this sensor has a freezing error flag, sensor misorientation error flag, and sensor calibrations lost or corrupted error flag.

Table 10 Error flag values and issue resolution

Error flag value	Issue present	Resolution
0	No issue present	NA
128	Sensor firmware is corrupt	Refer to Example Using a CR6 Datalogger and Modbus RTU for instructions on reloading firmware.
256	Sensor calibrations lost or corrupted	Refer to Example Using a CR6 Datalogger and Modbus RTU for instructions on reloading sensor calibrations.

DDI SERIAL COMMUNICATION

The DDI serial communications protocol is ideal for systems that have dedicated serial signaling lines for each sensor or use a multiplexer to handle multiple sensors. The serial communications are compatible with many TTL serial implementations that support active-high logic levels using 0.0–3.6 V signal levels. When the sensor is first powered, it automatically makes measurements of the integrated transducers then outputs a response over the data line. Systems using this protocol control the sensor excitation to initiate data transfers from the sensor. This protocol is subject to change as METER improves and expands the line of digital sensors and data loggers.

The TEROS 06 will omit the DDI serial startup string when the SDI-12 address is nonzero.

NOTE: Out of the factory, all METER sensors start with SDI-12 address 0 and print out the startup string when power cycled.

DDI SERIAL TIMING

Table 11 lists the DDI serial communication configuration.

Table 11 DDI serial communication configuration

Baud Rate	1,200
Start Bits	1
Data Bits	8 (LSB first)
Parity Bits	0 (none)
Stop Bits	1
Logic	Standard (active high)

At power up, the sensor will pull the data line high within 100 ms to indicate that the sensor is taking a reading (Figure 10). When the reading is complete, the sensor begins sending the serial signal out the data line adhering to the format shown in Figure 11. Once the data is transmitted, the sensor goes into SDI-12 communication mode. To get another serial signal, the sensor must be power cycled.

NOTE: Sometimes the signaling from the sensor can confuse typical microprocessor UARTs. The sensor holds the data line low while taking measurements. The sensor raises the line high to signal the logger that it will send a measurement. Then the sensor may take some additional measurements before starting to clock out the first data byte starting with a typical start bit (low). Once the first start bit is sent, typical serial timing is valid; however, the signal transitions before this point are not serial signaling and may be misinterpreted by the UART.

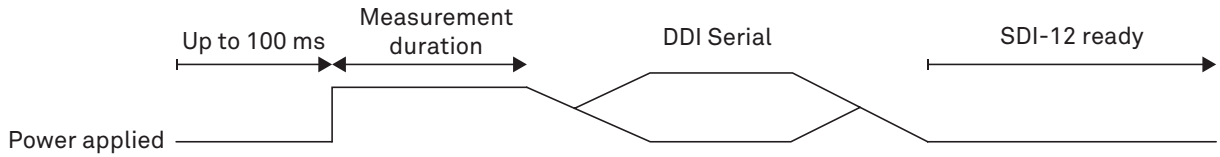

Figure 10 Data line DDI serial timing

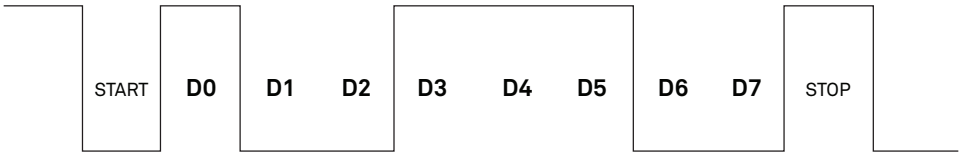

Figure 11 Example DDI serial transmission of the character 9 (0x39)

DDI SERIAL RESPONSE

Table 12 details the DDI serial response.

Table 12 DDI serial response

COMMAND	RESPONSE
NA	`<TAB>±<Temp1> ±<Temp2> ±<Temp3> ±<Temp4> ±<Temp5> ±<Temp6> <CR><sensorType><Checksum><CRC>`

NOTE: There is no actual command. The response is returned automatically upon power up.

DDI SERIAL CHECKSUM

These checksums are used in the continuous commands `R3` and `R4` as well as the DDI serial response. The legacy checksum is computed from the start of the transmission to the sensor identification character, excluding the sensor address.

Example input is `<TAB>2749.0 23.8 660<CR>g` and the resulting checksum output is `8`.

```
uint8_t LegacyChecksum(const char * response)
{
    uint16_t length;
    uint16_t i;
    uint16_t sum = 0;

    // Finding the length of the response string
    length = strlen(response);

    // Adding characters in the response together
    for( i = 0; i < length; i++ )
    {
        sum += response[i];
        if(response[i] == '\r')
        {
            // Found the beginning of the metadata section of the response
            break;
        }
    }

    // Include the sensor type into the checksum
    sum += response[++i];

    // Convert checksum to a printable character
    sum = sum % 64 + 32;
    return sum;
}
```

The more robust `CRC6` utilizes the `CRC-6-CDMA2000-A` polynomial with the value `48` added to the results to make this a printable character and is computed from the start of the transmission to the legacy checksum character, excluding the sensor address.

`CRC6` checksum example input is `<TAB>2749.0 23.8 660<CR>g8` and the resulting checksum output is `O`. (uppercase O as in Oscar).

```
uint8_t CRC6_Offset(const char *buffer)
{
    uint16_t byte;
    uint16_t i;
    uint16_t bytes;
    uint8_t bit;
    uint8_t crc=0xfc; // Set upper 6 bits to 1s

    // Calculate total message length—updated once the metadata section if found
    bytes=strlen(buffer);

    // Loop through all the bytes in the buffer for(byte=0; byte < bytes; byte++);

   {
         // Get the next bye in the buffer and XOR it with the crc
            crc^=buffer[byte];
         // Loop through all the bits in the current byte for
            bit=8; bit>0; bit(--)
         {
              // If the uppermost bit is a 1... if(crc & 0x80)
              {
                  // Shift to the next bit and XOR it with a polynomial
                  crc = (crc<<1)^0x9c;
              }
              else
              {
                 // Shift to the next bit
                 crc = crc<<1;
             }
        }
        if(buffer[byte] == '\r')
        {
             // Found the beginning of the metadata section of the reponse
             // both sensor type and legacy checksum are part of the crc6
             // this requires only two more iterations of the loop so reset
             // "bytes"

            // bytes is incremented at the beginning of the loops, so 3 is added
            bytes = byte+3;
       }

    }

    // Shift upper 6 bits down for crc
    crc = crc<<2;

   // Add 48 to shift crc to printable character avoiding \r\n and ! return (crc+48)
}
```

METER MODBUS RTU SERIAL IMPLEMENTATION

Modbus over Serial Line is specified in two versions—ASCII and RTU. TEROS 06 sensors communicate using RTU mode exclusively. The following explanation is always related to RTU.

Table 13 lists the Modbus RTU communication configuration.

Table 13 Modbus communication configuration

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

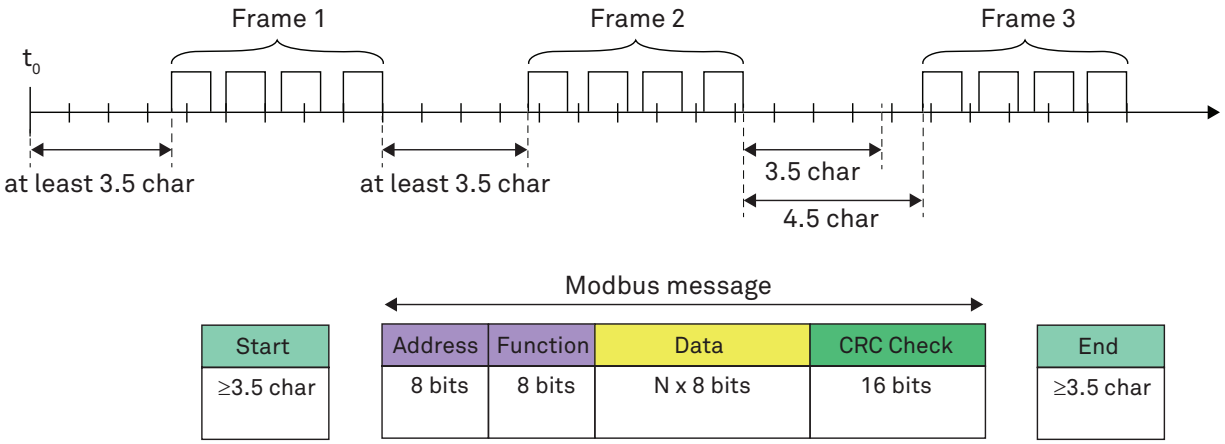

Figure 12 Modbus RTU message frame

A message in Modbus RTU format is shown in Figure 12. The length of the message is determined by the size of the data. The format of each byte in the message has 10 bits, including the 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 that are divided into three categories—public, user-defined, and reserved. This document covers TEROS sensor-supported public functions that are well-defined function codes documented in the Modbus Organization, Inc. (modbus.org)community.

For a reliable interaction between the TEROS sensors and a Modbus Master, a minimum 50 ms delay is required between every Modbus command sent on the RS-485 bus. An additional timeout is required for every Modbus query. This timeout is device specific and depends on the quantity of the polled registers. Generally 100 ms is sufficient for most TEROS sensors.

SUPPORTED MODBUS FUNCTIONS

Table 14 lists the Modbus function codes, action, and description.

Table 14 Modbus function definition

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	Read 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 multiple coils/ports 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 TEROS sensors uses both 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 of the different data types. This is an approach to the Modbus Enron implementation. Table 15 shows the four main tables used by the TEROS sensors with their respective access rights. Table 16 describes the subblocks for each different data type representation.

Please note that some Modbus dataloggers use addresses with a +1 offset. The offset can cause confusion and is based on a Modbus specification void. If problems occur implementing the Modbus program on a datalogger, always try to test different register offsets and data types. Use a known value to test, e.g., temperature.

Table 15 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

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 16 Modbus virtual Sub-blocks

Register Number	Access	Size	Description
X001-X009	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 17 Holding registers

4100 (4101*)	**Modbus Slave Address**
Detailed description	Read or update the sensor's Modbus address
Data type	Unsigned integer
Allowed range	0-247
Unit	–
Comments	Updated slave address will be stored in the senor's nonvolatile memory

Table 18 Input registers

3200 (3201*)	**Temperature at 5 cm depth**
Detailed description	<Temp1>
Data type	32-bit floating big-endian
Allowed range	-20 to 50
Unit	decC
Comments	–
3201 (3202*)	**Temperature at 10 cm depth**
Detailed description	<Temp2>
Data type	32-bit floating big-endian
Allowed range	-20 to 50
Unit	decC
Comments	–
3202 (3203*)	**Temperature at 20 cm depth**
Detailed description	<Temp3>
Data type	32-bit floating big-endian
Allowed range	-20 to 50
Unit	decC
Comments	–
3203 (3204*)	**Temperature at 30 cm depth**
Detailed description	<Temp4>
Data type	32-bit floating big-endian
Allowed range	-20 to 50
Unit	decC
Comments	–
3204 (3205*)	**Temperature at 50 cm depth**
Detailed description	<Temp5>
Data type	32-bit floating big-endian
Allowed range	-20 to 50
Unit	decC
Comments	–

Table 18 Input registers

3205 (3206*)	**Temperature at 100 cm depth**
Detailed description	<Temp6>
Data type	32-bit floating big-endian
Allowed range	-20 to 50
Unit	decC
Comments	–
3206 (3207*)	**Reserved**
Detailed description	<Temp7> reserved for probes with more than 6 measurement depths
Data type	32-bit floating big-endian
Allowed range	-20 to 50
Unit	decC
Comments	–
3207 (3208*)	**Reserved**
Detailed description	<Temp8> reserved for probes with more than 6 measurement depths
Data type	32-bit floating big-endian
Allowed range	-20 to 50
Unit	decC
Comments	–
3209 (3209*)	**Supply voltage**
Detailed description	Probe actual supply voltage
Data type	32-bit floating big-endian
Allowed range	2 to 24
Unit	volts
Comments	–

*Some devices report Modbus register addresses with an offset of +1. Specifically, Campbell Scientific Loggers and Dataker loggers report Modbus register addresses this way. In order to read the desired register use the number in paranthesis.

EXAMPLE USING A CR6 DATALOGGER AND MODBUS RTU

The Campbell Scientific, Inc. CR6 Measurement and Control Datalogger supports Modbus master and Modbus slave communication for integration in 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 in RTU mode exclusively. In a Modbus network, each slave device has a unique address. Therefore, sensor devices must be properly configured before being connected 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) dataloggers 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 be edited using the CRBasic Editor, both available for downloading as a stand-alone application on the official Campbell Scientific website.

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 quired
- Function call to the data tables

CR6 DATALOGGER 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 13 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 TEROS sensor used for this example is connected to (`C1`–`C2`) terminals.

TEROS 06 to CR6 Datalogger Wiring Diagram

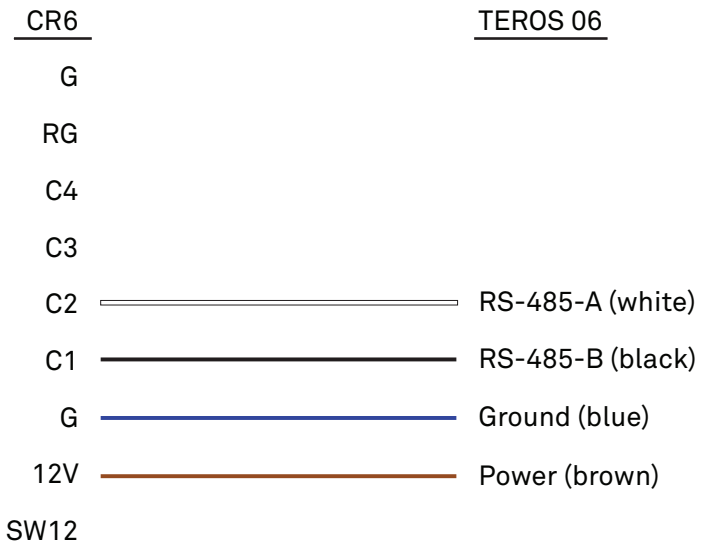

Figure 13 RS-485 interface

After assigning the TEROS sensor a unique Modbus Slave Address, it can be wired according to Figure 13 to the CR6 logger. Make sure to connect the black and the white wires according to their signals respectively to `C1` and `C2` ports. The brown wire to `12 V` (`V+`) and the blue to `G` (`GND`). If the power supply must be controlled through the program, connect the brown wire directly to one of the `SW12` terminals (switched `12 V` outputs).

EXAMPLE PROGRAMS

```
'CR6 Datalogger
'This is an example program for reading out the Teros06 Soil Water Content Profile Probe  'using a
CR6 datalogger and the MODBUS RTU protocol over a RS-485 Bus. The measurement 'values polled from
the sensor will be: Soil Volumetric Water Content and Temperature
'This program runs a scan every 1 Min and stores the data in a 1 Min table.

'Declare Constants
Const TEROS_MB_ADDR=1     'Teros06 Modbus slave address
Const MB_TIMEOUT=15       '150ms timeout (value * 0.01 sec)
Const MB_RETRIES=1
'Declare Public Variables
Public PTemp, batt_volt
Public mb_status        ' variable used for monitoring the modbus poll status

'Declare Private variables
Dim Temperature(6)  'array with temperature values

'Declare Aliases used for the Teros06
Alias Temperature(1)= TEMP_05CM
Alias Temperature(2)= TEMP_10CM
Alias Temperature(3)= TEMP_20CM
Alias Temperature(4)= TEMP_30CM
Alias Temperature(5)= TEMP_50CM
Alias Temperature(6)= TEMP_100CM

'Declare Units
Units temperature=degC

'Define Data Tables
DataTable (Teros_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 (6,temperature(),IEEE4)   'temperature values
EndTable

'Main Program
BeginProg
SW12(2,1)    'Switch ON the SW12-2 terminal (if used for powering the Teros sensor)
SerialOpen(ComC1,9600,3,2,50,4)         'open communication port, setup for RS-485
                                  'BaudRate, Format, TXDelay, BufferSize, CommsMode
Scan (1,Min,0,0)                '1 minute scan loop
PanelTemp (PTemp,15000)
Battery (batt_volt)
ModbusMaster(mb_status,ComC1,9600,TEROS_MB_ADDR,4,MB_dataset(),3201,16,MB_RETRIES,MB_TIMEOUT,2)

'Call Output Tables
CallTable Teros_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
03	9.2024	1.09	Modbus address updates
02	11.2023	1.09	Added new modbus, 4-write and RS-485 descriptions and options available.
01	9.22.2022	1.00	LegacyChecksum revision
00	8.31.2020	1.00	Initial release