



CR1000X

Measurement and Control Datalogger

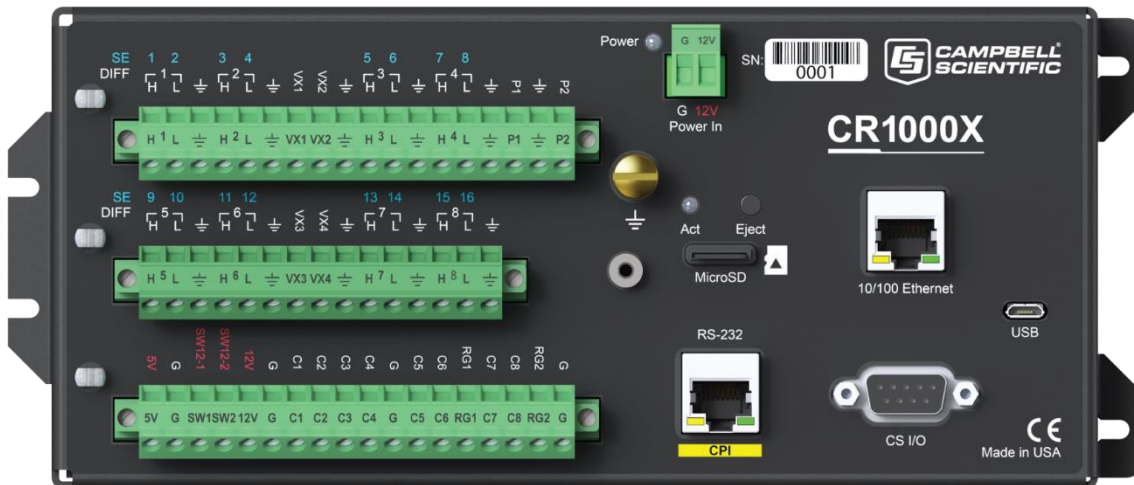


Table of Contents

DATA ACQUISITION SYSTEM COMPONENTS	1
Sensors.....	2
Supported Sensor Types	2
The CR1000X Datalogger	2
Components	2
Operations	3
Programs	3
CR1000X Wiring Panel	3
Power Input	5
Power Output.....	6
Grounds.....	7
Communication Ports	8
Programmable Logic Control	10
SETTING UP THE DATALOGGER	12
Setting up Communications with the Datalogger	12
Configuring Settings to Communicate over USB or RS-232.....	12
Configuring Settings to Communicate over Ethernet.....	13
Testing Communication and Completing EZ Setup	15
Connecting the Datalogger to a Computer	16
Creating a Program in Short Cut	16
Sending a Program to the Datalogger	18
Sending a Program Using Datalogger Support Software	18
Program Run Options	18
WORKING WITH DATA	19
Monitoring Data.....	19
Collecting Data.....	20
Collecting Data Using LoggerNet.....	20
Collecting Data Using PC200W or PC400W	20
Viewing Historic Data	20
About Data Tables	21
Table Definitions.....	21
First Header Row.....	21
Second Header Row.....	22
Third Header Row.....	22
Fourth Header Row	22
Data Records.....	23
Creating Data Tables in a Program	23
Memory and Data Storage	23
Memory Allocation	24
CPU Drive	24
SRAM.....	24
USR Drive	24
MicroSD (CRD: Drive) Storage	25
Managing Memory via Powerup.ini	26
MEASUREMENTS	29
Voltage Measurements	29
Single-Ended Measurements.....	29
Differential Measurements	30
Current Measurements.....	30

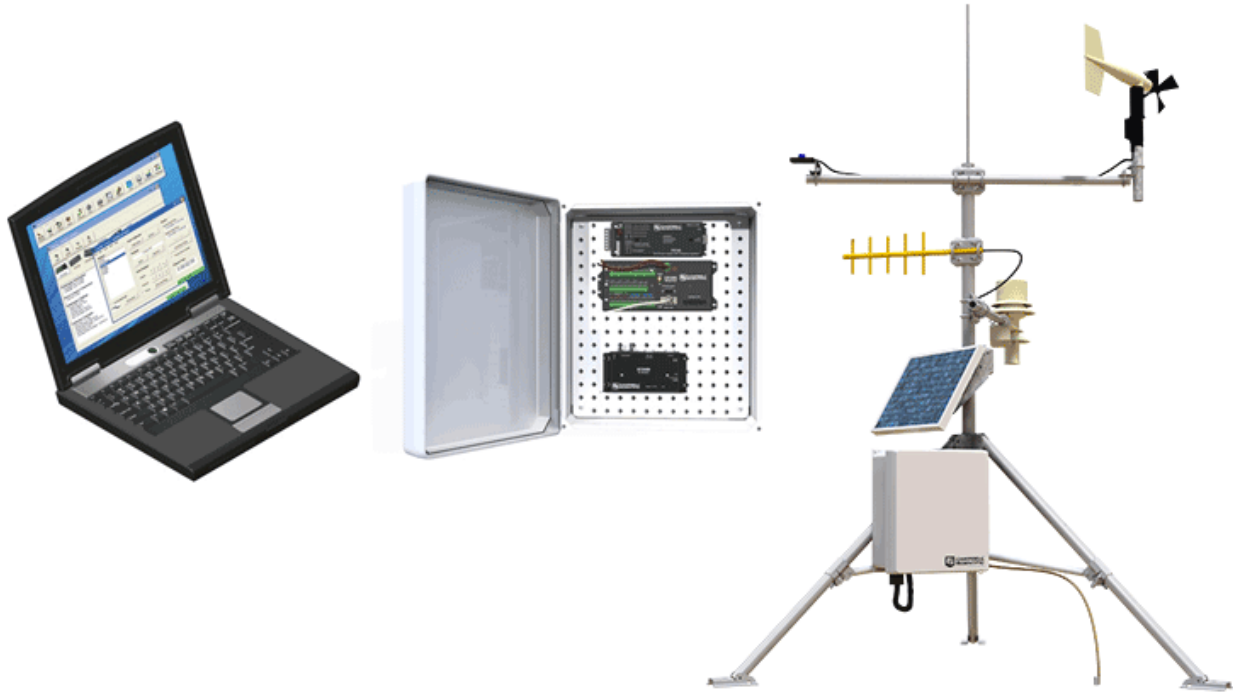
Example Current Measurement Connections.....	30
Ratiometric Resistance Measurements.....	32
Resistive-Bridge Measurements and Voltage Excitation.....	34
Strain Measurements.....	34
Ac Excitation.....	36
Accuracy for Ratiometric Resistance Measurements.....	36
Thermocouple Measurements.....	36
Period-Averaging Measurements.....	37
Pulse Measurements.....	37
Low-Level Ac Measurements.....	38
High Frequency Measurements.....	39
Switch-Closure and Open-Collector Measurements.....	39
Edge Timing and Edge Counting.....	40
Pulse Measurement Tips.....	41
Vibrating Wire Measurements.....	42
COMMUNICATION PROTOCOLS.....	43
General Serial Communications.....	43
Modbus Communications.....	44
Internet Communications.....	45
IP Address.....	45
HTTPS.....	45
DNP3 Communications.....	45
PakBus Communications.....	46
SDI-12 Communication.....	46
SDI-12 Transparent Mode.....	47
SDI-12 Programmed Mode/Recorder Mode.....	48
Programming the Datalogger to Act as an SDI-12 Sensor.....	48
SDI-12 Power Considerations.....	49
Serial Peripheral Interface (SPI) and I2C.....	49
MAINTAINING YOUR DATALOGGER.....	50
Datalogger Calibration.....	50
About Background Calibration.....	50
Datalogger Security.....	51
Security Codes.....	52
Creating a .csipasswd File.....	52
Datalogger Enclosures.....	53
Internal Battery.....	54
Internal Lithium Battery Specifications.....	54
Replacing the Internal Battery.....	55
Electrostatic Discharge and Lightning Protection.....	56
Power Budgeting.....	57
Updating the Operating System.....	57
TIPS AND TROUBLESHOOTING.....	59
Checking Station Status.....	59
Viewing Station Status.....	59
Watchdog Errors.....	60
Results for Last Program Compiled.....	60
Skipped Scans.....	60
Skipped Records.....	60
Variable Out of Bounds.....	60
Battery Voltage.....	60
Understanding NAN and INF Occurrences.....	60
Time Keeping.....	61

Clock Best Practices	61
Time Stamps	62
Avoiding Time Skew	62
CRBasic Program Errors	62
Program Does Not Compile	62
Program Compiles but Does Not Run Correctly	63
Resetting the Datalogger	63
Processor Reset	63
Program Send Reset	64
Manual Data Table Reset	64
Formatting Drives	64
Full Memory Reset	64
Troubleshooting Power Supplies	64
Minimizing Ground Loop Errors	65
Improving Voltage Measurement Quality	66
Deciding Between Single-Ended or Differential Measurements	66
Minimizing Ground Potential Differences	67
Detecting Open Inputs	68
Minimizing Power-Related Artifacts	68
Minimizing Settling Errors	69
Factors Affecting Accuracy	71
Minimizing Offset Voltages	72
File System Error Codes	75
File Name and Resource Errors	76
Background Calibration Errors	76
SPECIFICATIONS	77
System Specifications	77
Physical Specifications	77
Power Requirements	77
Ground Specifications	78
Power Output Specifications	78
Analog Specifications	80
Voltage Measurements	80
Ratiometric-Resistance Measurements Specifications	81
Period Averaging Specifications	81
Current Measurements Specifications	82
Pulse Counting Specifications	82
Switch Closure Input	82
High-Frequency Input	82
Low-Level Ac Input	82
Digital I/O Specifications	83
Switch Closure Input	83
High-Frequency Input	84
Edge Timing	84
Communications Specifications	84
SUPPORT	86
Product Assistance	86
Precautions	86
Warranty	87
Acknowledgements	88
GLOSSARY	89
INDEX	107

Data Acquisition System Components

A basic data acquisition system consists of sensors, measurement hardware, and a computer with programmable software. The objective of a data acquisition system should be high accuracy, high precision, and resolution as high as appropriate for a given application.

The concept of a data acquisition system is illustrated in the following figure.



Following is a list of typical data acquisition system components:

- **Sensors** - Electronic sensors convert the state of a phenomenon to an electrical signal (see "Sensors" on page 2 for more information).
- **Datalogger** - The datalogger measures electrical signals or reads serial characters. It converts the measurement or reading to engineering units, performs calculations, and reduces data to statistical values. Data are stored in memory to await transfer to a computer by way of an external storage device or a communication link.
- **Data Retrieval and Communications** - Data are copied (not moved) from the datalogger, usually to a computer, by one or more methods using datalogger support software. Most communication options are bi-directional, which allows programs and settings to be sent to the datalogger. For more information, see "Sending a Program to the Datalogger" on page 18.
- **Datalogger Support Software** - Software retrieves data, sends programs, and sets settings. The software manages the communication link and has options for data display.
- **Programmable Logic Control** - Some data acquisition systems require the control of external devices to facilitate a measurement or to control a device based on measurements. This datalogger is adept at programmable logic control. See "Programmable Logic Control" on page 9 for more information.
- **Measurement and Control Peripherals** - Sometimes, system requirements exceed the capacity of the datalogger. The excess can usually be handled by addition of input and output expansion modules.

Sensors

Sensors transduce phenomena into measurable electrical forms by modulating voltage, current, resistance, status, or pulse output signals. Suitable sensors do this with accuracy and precision. Smart sensors have internal measurement and processing components and simply output a digital value in binary, hexadecimal, or ASCII character form.

Most electronic sensors, regardless of manufacturer, will interface with the datalogger. Some sensors require external signal conditioning. The performance of some sensors is enhanced with specialized input modules. The datalogger, sometimes with the assistance of various peripheral devices, can measure or read nearly all electronic sensor output types.

The following list may not be comprehensive. A library of sensor manuals and application notes are available at www.campbellsci.com/support to assist in measuring many sensor types.

SUPPORTED SENSOR TYPES

- Analog
 - o Voltage
 - o Current
 - o Strain
 - o Thermocouples
 - o Resistive bridges
- Pulse
 - o High frequency
 - o Switch-closure
 - o Low-level ac
- Period average
- Vibrating wire (through interface modules)
- Smart sensors
 - o SDI-12
 - o RS-232
 - o Modbus
 - o DNP3
 - o TCP/IP
 - o RS-485

The CR1000X Datalogger

The CR1000X is used in a broad range of measurement and control functions. Rugged enough for extreme conditions and reliable enough for remote environments, it is also robust enough for complex configurations. Used in applications all over the world, it is a powerful core component for your data acquisition system.

COMPONENTS

The CR1000X datalogger is the main part of a data acquisition system (see "Components of Data Acquisition Systems" on page 1 for more information). It has a central-processing unit (CPU), analog and digital measurement inputs, analog and digital outputs, and memory. An operating system

(firmware) coordinates the functions of these parts in conjunction with the onboard clock and the CRBasic application program.

The CR1000X can simultaneously provide measurement and communication functions. Low power consumption allows the CR1000X to operate for extended time on a battery recharged with a solar panel, eliminating the need for ac power. The CR1000X suspends execution when primary power drops below 9.6 V, reducing the possibility of inaccurate measurements.

The electronics are RF shielded and glitch protected by the sealed, stainless-steel canister, making the CR1000X economical, small, and very rugged. A battery-backed clock assures accurate timekeeping.

OPERATIONS

The CR1000X measures almost any sensor with an electrical response, drives direct communications and telecommunications, reduces data to statistical values, performs calculations, and controls external devices. After measurements are made, data are stored in onboard, nonvolatile memory awaiting transfer to the computer. Because most applications do not require that every measurement be recorded, the program usually combines several measurements into computational or statistical summaries, such as averages and standard deviations.

Programs are run by the CR1000X in either sequential mode or pipeline mode. In sequential mode, each instruction is executed sequentially in the order it appears in the program. In pipeline mode, the CR1000X determines the order of instruction execution to maximize efficiency.

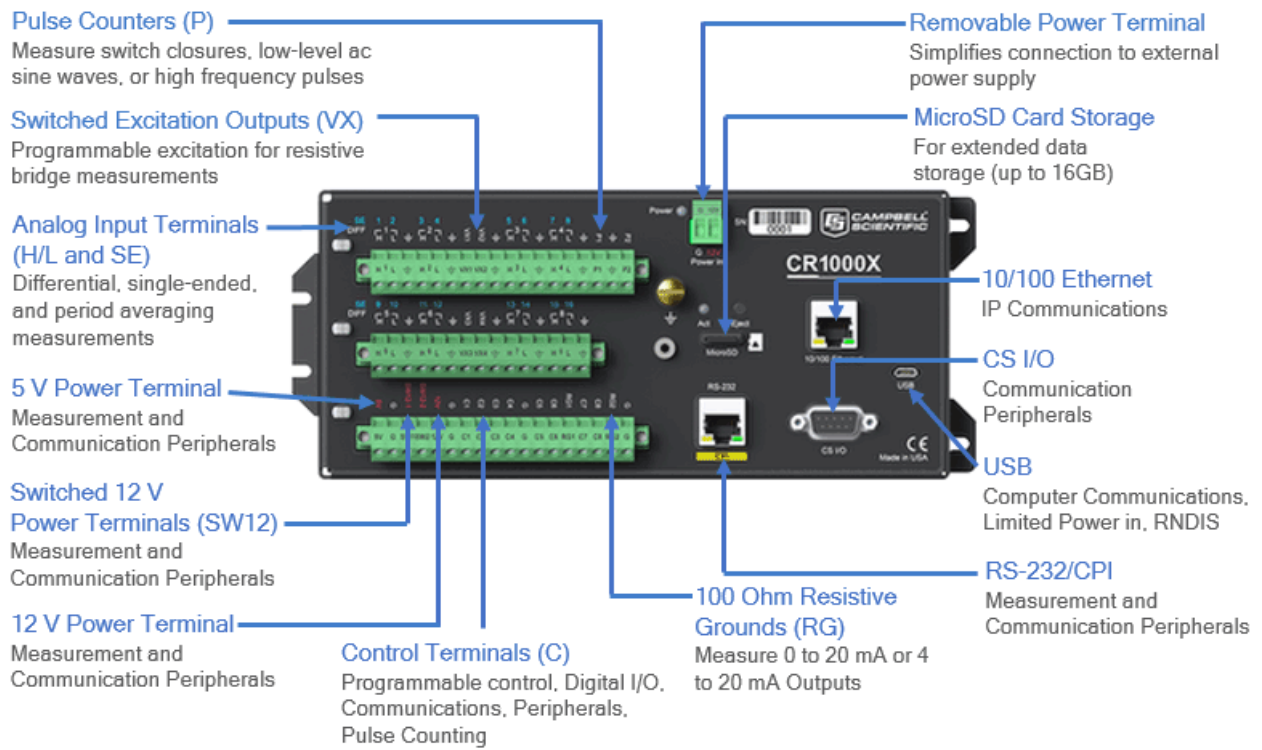
PROGRAMS

A program directs the datalogger on how and when sensors are to be measured, calculations made, and data stored. The application program for the CR1000X is written in CRBasic, which is a programming language that includes measurement, data processing, and analysis routines, as well as the standard BASIC instruction set. For simpler applications, Short Cut, a user-friendly program generator, can be used to write the program. For more demanding programs, use CRBasic Editor. If you are programming with CRBasic, you can utilize the extensive help available with the CRBasic Editor software. Formal training is also available from Campbell Scientific.

CR1000X Wiring Panel

The CR1000X wiring panel provides ports and removable terminals for connecting sensors, power, and communication devices. It is protected against surge, over-voltage, over-current, and reverse power. The wiring panel is the interface to most datalogger functions so studying it is a good way to get acquainted with the datalogger. Functions of the terminals are broken down into the following categories:

- Analog input
- Pulse counting
- Period averaging
- Analog output
- Communications
- Digital I/O
- Power input
- Power output
- Ground terminals



Analog Input Functions																			
SE DIFF	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	VX1-VX4	RG1	RG2
	r1	r2	r3	r4	r5	r6	r7	r8											
	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L			
Single-Ended Voltage	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Differential Voltage	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L			
Current Loop																		✓	✓
Ratiometric/Bridge	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Thermocouple	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Period Average	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			

Pulse Counting Functions	P1	P2	C1-C8
Switch-Closure	✓	✓	✓
High Frequency	✓	✓	✓
Low-level Ac	✓	✓	

Analog Output Functions	VX1-VX4	5V	12V	SW12-1	SW12-2
Switched Voltage Excitation	✓				
5 Vdc Regulated Supply	✓	✓			
3.3 Vdc Regulated Supply	✓				
12 Vdc Source			✓	✓	✓

Communications Functions	C1	C2	C3	C4	C5	C6	C7	C8	RS-232/CPI
SDI-12	✓		✓		✓		✓		
GPS Time Sync	PPS	Rx							
TTL 0-5 V	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	
LVTTL 0-3.3 V	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	
RS-232					Tx	Rx	Tx	Rx	✓
RS-485 (Half Duplex)					A-	B+	A-	B+	
RS-485 (Full Duplex)					Tx-	Tx+	Rx-	Rx+	
I2C	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	
SPI	SCLK	MOSI	MISO		SCLK	MOSI	MISO		
SDM ¹	Data	Clk	Enable		Data	Clk	Enable		
CPI/CDM									✓

¹SDM can be on either C1-C3 or C5-C7, but not both at the same time.

Communication Functions also include Ethernet via Ethernet port and USB via USB port.

Digital I/O Functions	C1-C8
General I/O	✓
Pulse-Width Modulation Output	✓
Timer Input	✓
Interrupt	✓

POWER INPUT

The datalogger requires a power supply (see "Troubleshooting Power Supplies" on page 64). The datalogger operates with external power connected to the green **POWER IN** port on the face of the wiring panel (see "CR1000X Wiring Panel" on page 3). The positive power lead connects to **12V**. The negative lead connects to **G**. The connection is internally reverse-polarity protected (it protects against accidental polarity reversal) and high voltage transients.

Warning: Sustained input voltages in excess of those listed in the "Power Output Specifications" on page 78, can damage the transient voltage suppression.

Be sure that power supply components match the specifications of the device to which they are connected. When connecting power, first switch off the power supply then insert the connector. Once you make the connection, turn the power supply on.

A UPS is often the best power source for long-term installations. An external UPS consists of a primary-power source, a charging regulator external to the datalogger, and an external battery. The primary power source, which is often a transformer, power converter, or solar panel, connects to the charging regulator, as does a nominal 12 Vdc sealed rechargeable battery. A third connection connects the charging regulator to the **12V** and **G** terminals of the **POWER IN** port.

If external alkaline power is used, the alkaline battery pack is connected directly to the **POWER IN** port.

The CR1000X can receive power via the **POWER IN** port as well as 5 Vdc via a **USB** connection. If both **POWER IN** and **USB** are connected, power will be supplied by whichever has the highest voltage. If **USB** is the only power source, then the **CS I/O** port and the **12V**, **SW12**, and **5V** terminals will not be operational. Functions that will be active with a 5 Vdc source (**USB**) include sending programs, adjusting datalogger settings, and making some measurements.

Powering A Datalogger with a Vehicle

If a datalogger is powered by a motor-vehicle power supply, a second power supply may be needed. When starting the motor of the vehicle, battery voltage often drops below the voltage required for datalogger operation. This may cause the datalogger to stop measurements until the voltage again equals or exceeds the lower limit. A second supply or charge regulator can be provided to prevent measurement lapses during vehicle starting.

In vehicle applications, the earth ground lug should be firmly attached to the vehicle chassis with 12 AWG wire or larger.

Power LED Indicator

When the datalogger is powered, the Power LED will turn on according to power and program states:

- **Off:** No power, no program running.
- **1 flash every 10 seconds:** Powered, program running.
- **3 flashes every 10 seconds:** Powered via USB, program running.
- **Always on:** Powered, no program running.

POWER OUTPUT

The datalogger can be used as a power source for sensors and peripherals. Take precautions to prevent damage to sensors or peripherals from over- or under-voltage conditions, and to minimize the error associated with the measurement of underpowered sensors. Exceeding current limits causes voltage output to become unstable. Voltage should stabilize once current is again reduced to within stated limits. The following are available:

- **12V** terminals: unregulated nominal 12 Vdc. This supply closely tracks the primary datalogger supply voltage, so it may rise above or drop below the power requirement of the sensor or peripheral. Precautions should be taken to prevent damage to sensors or peripherals from over- or under-voltage conditions, and to minimize the error associated with the measurement of underpowered sensors.
- **5V** terminals: regulated 5 Vdc. The 5 Vdc supply is regulated to within a few millivolts of 5 Vdc so long as the main power supply for the datalogger does not drop below the minimum supply voltage. It is intended for power sensors or devices requiring a 5 Vdc power supply. It is not intended as an excitation source for bridge measurements.
- **SW12** terminal: program-controlled, switched 12 Vdc terminal. Often used to power devices such as sensors that require 12 Vdc during measurement. Voltage on a **SW12** terminal will change with datalogger supply voltage. CRBasic instruction **sw12 ()** controls the **SW12** terminal.
- **CS I/O** port: used to communicate with and often supply power to Campbell Scientific peripheral devices

Caution: Voltage levels at the **12V** and switched **SW12** terminals, and pin **8** on the **CS I/O** port, are tied closely to the voltage levels of the main power supply. For example, if the power received at the **POWER IN 12V** and **G** terminals is 16 Vdc, the **12V** and **SW12** terminals, and pin **8** on the **CS I/O** port, will supply 16 Vdc to a connected peripheral. If the connected peripheral or sensor is not designed for that voltage level, it may be damaged.

- **VX** terminals: supply precise output voltage used by analog sensors to generate high resolution and accurate signals. In this case, these terminals are regularly used with resistive-bridge measurements (see "Resistance Measurements" on page 32 for more information). Using the

swvx () instruction, **VX** terminals can also be used to supply a selectable, switched, regulated 3.3 or 5 Vdc power source to power digital sensors and toggle control lines.

- **C** terminals: can be set low or high as output terminals. With limited drive capacity, digital output terminals are normally used to operate external relay-driver circuits.

See also "Power Output Specifications" on page 78.

GROUNDS

Proper grounding lends stability and protection to a data acquisition system. Grounding the datalogger with its peripheral devices and sensors is critical in all applications. Proper grounding will ensure maximum ESD protection and measurement accuracy. It is the easiest and least expensive insurance against data loss, and often the most neglected. The following terminals are provided for connection of sensor and datalogger grounds:

- **Signal Ground** (\oplus) - reference for single-ended analog inputs, excitation returns, and as a ground for sensor shield wires.
- **Power Ground (G)** - return for 5V, 12V, and digital sensors. Use of **G** grounds for these outputs minimizes potentially large current flow through the analog-voltage-measurement section of the wiring panel, which can cause single-ended voltage measurement errors.
- **Resistive Ground (RG)** - terminal for decoupling ground on RS-485 signals. Includes 101 Ω resistance to ground. Also used for measuring Current (see "Current Measurements" on page 30 for more information).
- **Earth Ground Lug** (\oplus) - connection point for heavy-gauge earth-ground wire. A good earth connection is necessary to secure the ground potential of the datalogger and shunt transients away from electronics. 14 AWG wire, minimum, is recommended.

Note: Several ground wires can be connected to the same ground terminal.

A good earth (chassis) ground will minimize damage to the datalogger and sensors by providing a low-resistance path around the system to a point of low potential. Campbell Scientific recommends that all dataloggers be earth (chassis) grounded. All components of the system (dataloggers, sensors, external power supplies, mounts, housings, etc.) should be referenced to one common earth (chassis) ground.

In the field, at a minimum, a proper earth ground will consist of a 5-foot copper-sheathed grounding rod driven into the earth and connected to the large brass ground lug on the wiring panel with a 14 AWG wire. In low-conductive substrates, such as sand, very dry soil, ice, or rock, a single ground rod will probably not provide an adequate earth ground. For these situations, search for published literature on lightning protection or contact a qualified lightning-protection consultant.

In laboratory applications, locating a stable earth ground is challenging, but still necessary. In older buildings, new Vac receptacles on older Vac wiring may indicate that a safety ground exists when, in fact, the socket is not grounded. If a safety ground does exist, good practice dictates the verification that it carries no current. If the integrity of the Vac power ground is in doubt, also ground the system through the building plumbing, or use another verified connection to earth ground.

In vehicle applications, the earth ground lug should be firmly attached to the vehicle chassis with 12 AWG wire or larger (see "Powering A Datalogger with a Vehicle" on page 6 for more information). See also:

- "Minimizing Ground Loop Errors" on page 65
- "Minimizing Ground Potential Differences" on page 67

COMMUNICATION PORTS

The datalogger is equipped with ports that allow communication with other devices and networks, such as:

- Computers
- Smart sensors
- Modbus and DNP3 networks
- Ethernet
- Modems
- Campbell Scientific PakBus networks
- Other Campbell Scientific dataloggers

Campbell Scientific datalogger communication ports include:

- CS I/O
- RS-232/CPI
- USB
- Ethernet
- C1-C8 terminals

USB Port

One micro-B USB port, labeled **USB**, for communicating with a computer through datalogger support software, supporting virtual Ethernet (RNDIS), and providing 5 Vdc power to the datalogger (powering through the USB port has limitations, see "Power Input" on page 5 for more information). The datalogger USB port does not support USB flash or thumb drives.

Ethernet Port

RJ45 10/100 Ethernet port used for IP communications.

C1 - C8 Terminals for Communications

C Terminals are configurable for the following communication types:

- SDI-12
- RS-232
- RS-485
- TTL (0 to 5 V)
- LVTTL (0 to 3.3 V)
- SDM

Some communication types require more than one terminal, and some are only available on specific terminals. This is shown in the datalogger specifications.

SDI-12 Port

SDI-12 is a 1200 baud protocol that supports many smart sensors. **C1**, **C3**, **C5**, and **C7** can be configured as **SDI-12** ports. SDI-12 standard 1.4 sensors accept addresses 0 through 9, a through z, and A through Z. Maximum cable lengths depend on the number of sensors connected, the type of cable used, and the environment of the application. Refer to the sensor manual for guidance.

For more information, see "SDI-12 Communication" on page 46.

RS-232, RS-485, TTL, and LVTTL Ports

RS-232, RS-485, TTL, and LVTTL communications are typically used for the following:

- Reading sensors with serial output
- Creating a multi-drop network
- Communication with other dataloggers or devices over long cables

Configure **C** terminals as serial ports using Device Configuration Utility or by using the `SerialOpen()` CRBasic instruction. Terminals **C1-C8** can be configured for RS-232 or RS-485 communications. The terminals are configured in pairs for RS-232 or half-duplex RS-485. For full-duplex RS-485, all four terminals are required. See also "Communication Protocols" on page 43.

SDM Port

SDM is a protocol proprietary to Campbell Scientific that supports several Campbell Scientific digital sensor and communication input and output expansion peripherals and select smart sensors. It uses a common bus and addresses each node. CRBasic SDM device and sensor instructions configure terminals **C1**, **C2**, and **C3** together to create an SDM port. Alternatively, terminals **C5**, **C6**, and **C7** can be configured together to be used as the SDM port by using the `SDMBeginPort()` instruction.

See also "Communications Specifications" on page 84.

CS I/O Port

CS I/O is a Campbell Scientific proprietary input/output port as well as the proprietary serial communication protocol that occurs over this port.

One nine-pin port, labeled **CS I/O**, for communicating with a computer or modem through Campbell Scientific communication interfaces, modems, or peripherals. It is recommended that you keep CS I/O cables short (maximum of a few feet). See also "Communications Specifications" on page 84.

RS-232/CPI Port

The datalogger includes one RJ45 module jack labeled RS-232-CPI. CPI is a proprietary interface for communications between Campbell Scientific dataloggers and Campbell Scientific CDM peripheral devices. It consists of a physical layer definition and a data protocol. CDM devices are similar to Campbell Scientific SDM devices in concept, but the use of the CPI bus enables higher data-throughput rates and use of longer cables. CDM devices require more power to operate in general than do SDM devices. CPI ports also enable networking between compatible Campbell Scientific dataloggers. Consult the manuals for CDM modules for more information.

CPI has the following power levels:

- **Off:** Not used.
- **High power:** Fully active.
- **Low-power standby:** Whenever possible.
- **Low-power bus:** Sets bus and modules to low power.

When used with an RJ45-to-DB9 converter cable, the RS-232/CPI port can be used as an RS-232 port. It defaults to 115200 bps (in autobaud mode), 8 data bits, no parity, and 1 stop bit. Use Device Configuration Utility or the `SerialOpen()` CRBasic instruction to change these options.

PROGRAMMABLE LOGIC CONTROL

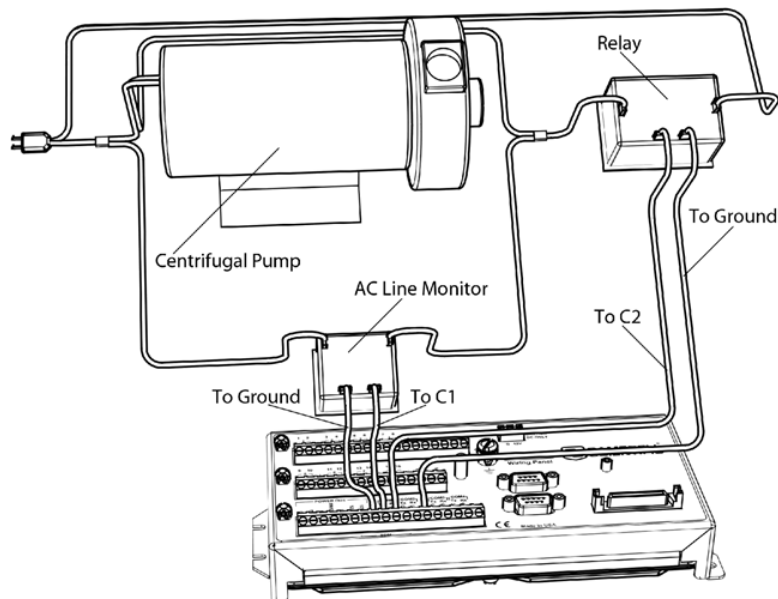
The datalogger can control instruments and devices such as:

- Controlling wireless cellular modem or GPS receiver to conserve power.
- Triggering a water sampler to collect a sample.
- Triggering a camera to take a picture.
- Activating an audio or visual alarm.
- Moving a head gate to regulate water flows in a canal system.
- Controlling pH dosing and aeration for water quality purposes.
- Controlling a gas analyzer to stop operation when temperature is too low.
- Controlling irrigation scheduling.

Control decisions can be based on time, an event, or a measured condition. Controlled devices can be physically connected to **C**, **VX**, or **SW12** terminals. Short Cut has provisions for simple on/off control. Control modules and relay drivers are available to expand and augment datalogger control capacity.

- **C** terminals are selectable as binary inputs, control outputs, or communication ports. **C** terminals can be set low (0 Vdc) or high (3.3 or 5 Vdc) using the `PortSet()` or `WriteIO()` instructions. Other functions include device-driven interrupts, asynchronous communications and SDI-12 communications. The high voltage for **C** terminals defaults to 5 V, but it can be changed to 3.3 V using the `PortPairConfig()` instruction. Terminals **C4**, **C5**, and **C7** can also be configured for pulse width modulation with a maximum period of 36.4 s. A **C** terminal configured for digital I/O is normally used to operate an external relay-driver circuit because the port itself has limited drive capacity.
- **VX** terminals can be set low (0 V) or high (3.3 or 5 V) using the `swvx()` instruction.
- **SW12** terminals can be set low (0 V) or high (12 V) using the `sw12()` instruction.

The following image illustrates a simple application wherein a **C** terminal configured for digital input and another configured for control output are used to control a device (turn it on or off) and monitor the state of the device (whether the device is on or off).



Example: Turn Modem on for 10 Minutes Hourly

In the case of a cell modem, control is based on time. The modem requires 12 Vdc power, so connect its power wire to a datalogger **SW12** terminal. The following code snip turns the modem on for the first ten minutes of every hour using the **TimeIsBetween()** instruction embedded in an **If/Then** logic statement:

```
If TimeIsBetween(0,10,60,Min) Then
    SW12(SW12_1,1,1) 'Turn phone on.
Else
    SW12(SW12_1,0,1) 'Turn phone off.
EndIf
```

Setting Up the Datalogger

The basic steps to setting up your datalogger to measure data include:

1. Configuring your connection.
2. Testing communication (optional).
3. Connecting the datalogger to the computer.
4. Creating a program and sending that program to the datalogger.

Setting up Communications with the Datalogger

The first step in setting up and communicating with your datalogger is to configure your connection. Communication peripherals, dataloggers, and software must all be configured for communication. In addition to this instruction and manuals for your specific peripherals, refer to your datalogger support software manual and help for guidance.

The default settings for the datalogger allow it to communicate with a computer via USB, RS-232, or Ethernet, and to accept and execute user application programs. For other communication methods or more complex applications, some settings may need adjustment. Settings can be changed through Device Configuration Utility or through datalogger support software.

You can configure your connection using any of the following options. The simplest is via USB.

- "Configuring Settings to Communicate over USB or RS-232" on page 12
- "Configuring Settings to Communicate over Ethernet" on page 13

For other configurations, see the LoggerNet EZSetup Wizard help. Context-specific help is given in each step of the wizard by clicking on the **Help** button in the bottom right corner of the window. For complex datalogger networks, use Network Planner.



CONFIGURING SETTINGS TO COMMUNICATE OVER USB OR RS-232

Setting up a USB or RS-232/CPI connection is a good way to begin communicating with your datalogger. Because these connections do not require configuration (like an IP address), you need only set up the communication between your computer and the datalogger.

Watch a [video](#) or use the following instructions.

Initial setup instruction follows. These settings can be revisited using the datalogger support software

Edit Datalogger Setup option ()

1. Using datalogger support software, launch the EZSetup Wizard.
 - PC200W and PC400 users, click the **Add Datalogger** button ()
 - LoggerNet users, click the **Setup** () option, click the **View** menu to ensure you are in the **EZ (Simplified)** view, then click the **Add Datalogger** button.
2. Click **Next**.
3. Select your datalogger from the list, type a name for your datalogger (for example, a site or project name), and click **Next**.
4. If prompted, select the **Direct Connect** connection type and click **Next**.
5. If this is the first time connecting this computer to a CR1000X via USB, click the **Install USB Driver** button, select your datalogger, click **Install**, and follow the prompts to install the USB drivers.

6. Plug the datalogger into your computer using a USB or RS-232 cable. The USB connection supplies 5V power as well as a communication link, which is adequate for setup, but a 12V battery will be needed for field deployment. If using RS-232, external power must be provided to the datalogger and a CPI/RS-232 RJ45 to DB9 cable is required for connection to a serial cable.

Note: The **Power** LED on the datalogger indicates the program and power state. Because the datalogger ships with a program set to run on power-up, the **Power** LED flashes 3 times every 10 seconds when powered over USB. When powered with a 12 V battery, it flashes 1 time every 10 seconds.


7. From the **COM Port** list, select the COM port used for your datalogger.
8. USB and RS-232 connections do not typically require a **COM Port Communication Delay** - this allows time for the hardware devices to "wake up" and negotiate a communication link. Accept the default value of **00 seconds** and click **Next**.
9. The baud rate and PakBus address must match the hardware settings for your datalogger. A USB connection does not require a baud rate selection, RS-232 connections default to 115200 baud, and the default PakBus address is **1**.
 - Set an **Extra Response Time** if you have a difficult or marginal connection and you want the datalogger support software to wait a certain amount of time before returning a communication failure error.
 - LoggerNet and PC400 users can set a **Max Time On-Line** to limit the amount of time the datalogger remains connected. When the datalogger is contacted, communication with it is terminated when this time limit is exceeded. A value of **0** in this field indicates that there is no time limit for maintaining a connection to the datalogger.
10. Click **Next**.
11. By default, the datalogger does not use a security code or a PakBus encryption key. Therefore, the **Security Code** can be set to **0** and the **PakBus Encryption Key** can be left blank. If either setting has been changed, enter the new code or key. See "Datalogger Security" on page 51 for more information.
12. Click **Next**.
13. Review the **Communication Setup Summary**. If you need to make changes, click the **Previous** button to return to a previous window and change the settings.

Setup is now complete, and the EZSetup Wizard allows you to click **Finish** or click **Next** to test your communication, set the datalogger clock, and send a program to the datalogger.

CONFIGURING SETTINGS TO COMMUNICATE OVER ETHERNET

The CR1000X offers a 10/100 Ethernet connection you can configure using Device Configuration Utility. You will use this utility to enter the datalogger IP Address, Subnet Mask, and IP Gateway address. After this, you can use the EZSetup Wizard to set up communications with the datalogger (see "Configuring Ethernet Communication with the Datalogger" on page 14). If you already have the datalogger IP information, you can skip these steps and go directly to "Configuring Ethernet Communication with the Datalogger" on page 14.

1. Supply power to the datalogger. A USB connection supplies 5V power (as well as a communication link), which is adequate for setup, but a 12V battery will be needed for field deployment. If powering via USB, first install USB drivers during EZ Setup or by using Device Configuration Utility (select your datalogger, then on the main page, click **Install USB Driver**).
2. Connect an Ethernet cable to the **10/100 Ethernet** port on the datalogger. The yellow and green **Ethernet** port LEDs display activity approximately one minute after connecting. If you do not see activity, contact your network administrator. For more information, see "Ethernet LEDs" on page 14.

- Using datalogger support software (LoggerNet, PC400, or PC200W), enter the Device Configuration Utility ()
- Select the **CR1000X Series** datalogger from the list
- Select the port assigned to the datalogger from the **Communication Port** list. If connecting via Ethernet, check the **Use IP Connection** button.
- By default, this datalogger does not use a PakBus encryption key, so the **PakBus Encryption Key** box can be left blank. If this setting has been changed, enter the new code or key. See "Datalogger Security" on page 51 for more information.
- Click **Connect**.
- On the **Deployment** tab, click the **Ethernet** subtab.
- The **Ethernet Power** setting allows you to reduce the power consumption of the datalogger. If there is no Ethernet connection, the datalogger will turn off its Ethernet interface for the time specified before turning it back on to check for a connection. Select an option from the list.
- Enter the **IP Address**, **Subnet Mask**, and **IP Gateway**. These values should be provided by your network administrator. A static IP address is recommended. If you are using DHCP, note the IP address assigned to the datalogger on the right side of the window. When the IP Address is set to 0.0.0.0, the default, the information displayed on the right side of the window updates with the information obtained from the DHCP server. Note, however, that this address is not static and may change. An IP address here of 169.254.###.### means the datalogger was not able to obtain an address from the DHCP server. Contact your network administrator for help.
- Click **Apply** to save your changes.


Ethernet LEDs



When the datalogger is powered, the **10/100 Ethernet** port will turn on with Ethernet activity:

- **Solid Yellow:** Valid Ethernet link.
- **No Yellow:** Invalid Ethernet link.
- **Flashing Yellow:** Ethernet activity.
- **Solid Green:** 100 Mbps link.
- **No Green:** 10 Mbps link.

Configuring Ethernet Communication with the Datalogger

Once you have configured the Ethernet settings or obtained the IP information for your datalogger, you can set up communication between your computer and the datalogger over Ethernet.

Initial setup instruction follows. These settings can be revisited using the datalogger support software **Edit Datalogger Setup** option ()

- Using datalogger support software, launch the EZSetup Wizard.
 - PC400 users, click the **Add Datalogger** button ()
 - LoggerNet users, click the **Setup** () option, click the **View** menu to ensure you are in the **EZ (Simplified)** view, then click the **Add Datalogger** button.

Note: PC200W does not allow IP connections.

- Click **Next**.
- Select the **CR1000X Series** from the list, type a name for your datalogger (for example, a site or project name), and click **Next**.
- Select the **IP Port** connection type and click **Next**.

5. Supply power to the datalogger. A USB connection supplies 5V power (as well as a communication link), which is adequate for setup, but a 12V battery will be needed for field deployment. If powering via USB, first install USB drivers during EZ Setup or by using Device Configuration Utility (select your datalogger, then on the main page, click **Install USB Driver**).
6. Connect an Ethernet cable to the **10/100 Ethernet** port on the datalogger. The yellow and green **Ethernet** port LEDs display activity approximately one minute after connecting. If you do not see activity, contact your network administrator. For more information, see "Ethernet LEDs" on page 14.
7. Type the datalogger IP address followed by a colon, then the port number of the datalogger in the **Internet IP Address** field (these were set up when "Configuring Settings to Communicate over Ethernet" on page 13). They can be accessed in Device Configuration Utility on the **Ethernet** subtab. Leading 0s must be omitted. For example:
 - IPv4 addresses are entered as 192.168.1.2:6785
 - IPv6 addresses must be enclosed in square brackets. They are entered as [2001:db8::1234:5678]:6785
8. The PakBus address must match the hardware settings for your datalogger. The default PakBus address is 1.
 - Set an **Extra Response Time** if you want the datalogger support software to wait a certain amount of time before returning a communication failure error.
 - LoggerNet and PC400 users can set a **Max Time On-Line** to limit the amount of time the datalogger remains connected. When the datalogger is contacted, communication with it is terminated when this time limit is exceeded. A value of 0 in this field indicates that there is no time limit for maintaining a connection to the datalogger.
9. Click **Next**.
10. By default, the datalogger does not use a security code or a PakBus encryption key. Therefore the **Security Code** can be set to 0 and the **PakBus Encryption Key** can be left blank. If either setting has been changed, enter the new code or key. See "Datalogger Security" on page 51 for more information.
11. Click **Next**.
12. Review the **Communication Setup Summary**. If you need to make changes, click the **Previous** button to return to a previous window and change the settings.

Setup is now complete, and the EZSetup Wizard allows you click **Finish** or click **Next** to test your communication, set the datalogger clock, and send a program to the datalogger.




Testing Communication and Completing EZ Setup

1. Using datalogger support software EZ Setup, access the **Communication Test** window.
 - Accessed during EZ Setup (see "Configuring Settings to Communicate over USB or RS-232" on page 12 for more information). Alternatively, you can double-click a datalogger from the station list to open the EZ Setup Wizard and access the **Communication Test** step from the left side of the window.
2. Ensure the datalogger is connected to the computer, select **Yes** to test the communication, then click **Next** to initiate the test. To troubleshoot an unsuccessful test, see "Troubleshooting the Datalogger" on page 59.

3. With a successful connection, the **Datalogger Clock** window displays the time for both the datalogger and the computer.
 - The **Adjusted Server Date/Time** displays the current reading of the clock for the computer or server running your datalogger support software. If the **Datalogger Date/Time** and **Adjusted Server Date/Time** don't match, you can set the datalogger clock to the **Adjusted Server Date/Time** by clicking **Set Datalogger Clock**.
 - Use the **Time Zone Offset** to specify a positive or negative offset to apply to the computer time when setting the datalogger clock. This offset will allow you to set the clock for a datalogger that needs to be set to a different time zone than the time zone of the computer (or to accommodate for changes in daylight saving time).
4. Click **Next**.
5. The datalogger ships with a default **GettingStarted** program. If the datalogger does not have a program, you can choose to send one by clicking the **Select and Send Program** button. Click **Next**.
6. LoggerNet only - watch a [video](#) or use the following instructions:
 - The **Datalogger Table Output Files** window displays the data tables available to be collected from the datalogger and the output file name. To include a data table in scheduled collection, select the data table from the **Tables** list and check the **Table Collected During Data Collection** box. Select a **Data File Option**: **Append to End of File** adds new data to the end of the existing data file, **Overwrite Existing File** replaces the existing file with a newly created file, and **No Output File** results in no data file being written to disk. Make note of the **Output File Name** and location. Click **Next**.
 - Check **Scheduled Collection Enabled** to have LoggerNet collect data from the datalogger according to a schedule. Set the **Base Date** and **Time** to begin scheduled collections. Set a **Collection Interval**, then click **Next**.
7. Click **Finish**.

Connecting the Datalogger to a Computer

Once you have configured your connection (see "Setting up Communications with the Datalogger" on page 12 for more information), you can connect the datalogger to your computer.

- PC200W and PC400 users, select the datalogger from the list and click the **Connect** button (.
- LoggerNet users, select **Main** and click the **Connect** button () on the LoggerNet toolbar, select the datalogger from the **Stations** list, then click the **Connect** button (.



To disconnect, click the **Disconnect** button (.

See also "Communication Protocols" on page 43.

Creating a Program in Short Cut

Use the Short Cut software to generate a program for your datalogger. Short Cut is included with your datalogger support software.

Watch a [video](#) or use the following instructions.

1. Using datalogger support software, launch Short Cut.
 - PC200W and PC400 users, click the **Short Cut** button (.
 - LoggerNet users, click **Program** then click the **Short Cut** button (.

2. Click **Create New Program**.
3. Select the **CR1000XSeries** datalogger and click **Next**.

Note: The first time Short Cut is run, a prompt will appear asking for a choice of noise rejection. Select **60 Hz Noise Rejection** for North America and areas using 60 Hz ac voltage. **Select 50 Hz Noise Rejection** for most of the Eastern Hemisphere and areas that operate at 50 Hz.

A second prompt lists sensor support options. **Campbell Scientific, Inc. (US)** is probably the best fit if you are outside Europe.

To change the noise rejection or sensor support option for future programs, use the **Program** menu.

4. A list of **Available Sensors and Devices** and **Selected Measurements Available for Output** display. Battery voltage **BattV** and internal temperature **PTemp_C** are selected by default. During operation, battery and temperature should be recorded at least daily to assist in monitoring system status.
5. Use the Search feature or expand folders to locate your sensor or device. Double-click on a sensor or measurement in the **Available Sensors and Devices** list to configure the device (if needed) and add it to the **Selected** list.
6. If the sensor or device requires configuration, a window displays with configuration options. Click **Help** at the bottom of the window to learn more about any field or option.
7. Click **OK**.
8. Click **Wiring Diagram** on the left side of the window to see how to wire the sensor to the datalogger. With the power disconnected from the datalogger, insert the wires as directed in the diagram. Ensure you clamp the terminal on the conductor, not the wire insulation. Use the included flat-blade screwdriver to open/close the terminals.
9. Click **Sensors** on the left side of the window to return to the sensor selection window.
10. Use the **Output Setup** options to specify how often measurements are to be made and how often outputs are to be stored. Note that multiple output intervals can be specified, one for each output table (**Table1** and **Table2** tabs).
11. In the **Table Name** box, type a name for the table.
12. Select a **Data Output Storage Interval**.
13. Scroll down and optionally, select to copy the table to an external storage device.
14. Check the **Advanced Outputs** option if you want to specify the number of records and data events to store, set output intervals, specify measurements to evaluate, and set flags based on the value of a variable.
15. Click **Next**.
16. Select the measurement from the **Selected Measurements Available for Output** list, then click an output processing option to add the measurement to the **Selected Measurements for Output** list.

The screenshot displays two side-by-side windows from a software application. The left window, titled "Selected Measurements Available for Output", shows a tree view of sensors. Under "Datalogger" > "Default", "BattV" and "PTemp_C" are listed. "Type T TC" is selected, showing "Temp_C" as its measurement. The right window, titled "Selected Measurements for Output", shows a table with columns: Sensor, Measurement, Processing, Output Label, and Units. It has two tabs: "1 OneMin" and "2 Table2". The "Table2" tab is active, showing a table with three rows: "Default" (BattV, Average, BattV_AVG, Volts), "Default" (PTemp_C, Average, PTemp_C_AVG, Deg C), and "Type T TC" (Temp_C, Average, Temp_C_AVG, Deg C). To the left of this table is a vertical list of processing options: Average, ETo, Maximum, Minimum, Sample, StdDev, Total, and WindVector.

17. Click **Finish** to compile the program. Replace the **untitled.cr1x** default name and click **Save**.
18. If LoggerNet or other datalogger support software is running on your computer, and the datalogger is connected to the computer (see "Connecting the Datalogger to a Computer" on page 16 for more information), you can choose to send the program.


Note: A good practice is to always retrieve data from the datalogger before sending a program; otherwise, data may be lost. See "Collecting Data" on page 20 for detailed instruction.

If your data acquisition requirements are simple, you can probably create and maintain a datalogger program exclusively with Short Cut. If your data acquisition needs are more complex, the files that Short Cut creates are a great source for programming code to start a new program or add to an existing custom program using CRBasic. See the CRBasic Editor help for detailed information on program structure, syntax, and each instruction available to the datalogger.

Note: Once a Short Cut generated program has been edited with CRBasic Editor, it can no longer be modified with Short Cut.

Sending a Program to the Datalogger

The CR1000X datalogger requires a CRBasic program to direct measurement, processing, control, and data storage operations. The program file may use the extension `.CR1x` or `.d1d`.

A good practice is to always retrieve data from the datalogger before sending a program; otherwise, data may be lost. To collect data using LoggerNet, connect to your datalogger and click the **Collect Now** button (). Some methods of sending a program give the option to retain data when possible. Regardless of the program upload tool used, data will be erased when a new program is sent if any change occurs to one or more data table structure in the following list:

- Data table name(s)
- Data output interval or offset
- Number of fields per record
- Number of bytes per field
- Field type, size, name, or position
- Number of records in table

SENDING A PROGRAM USING DATALOGGER SUPPORT SOFTWARE

Watch an video on sending a program using [LoggerNet](#) or [PC200W](#) or use the following instructions.

1. Connect the datalogger to your computer (see "Connecting the Datalogger to a Computer" on page 16 for more information).
2. Using your datalogger support software, click **Send New** or **Send Program** (located in the Current Program section on the right side of the window).
3. Navigate to the location of the program, select it, and click **Open**.
4. Confirm that you would like to proceed and erase all data tables saved on the datalogger. The program will send, compile, then display results.

After sending a program, it is a good idea to monitor the data to make sure it is measuring as you expect. See "Monitoring Data" on page 19 for more information.

PROGRAM RUN OPTIONS

When sending a program via File Control, Short Cut, or CRBasic, you can choose to have programs **Run on Power-up** and **Run Now**. **Run Now** will run the program when it is sent to the datalogger. **Run on Power-up** runs the program when the datalogger is powered-up. Selecting both **Run Now** and **Run on Power-up** will invoke both options.



Working with Data

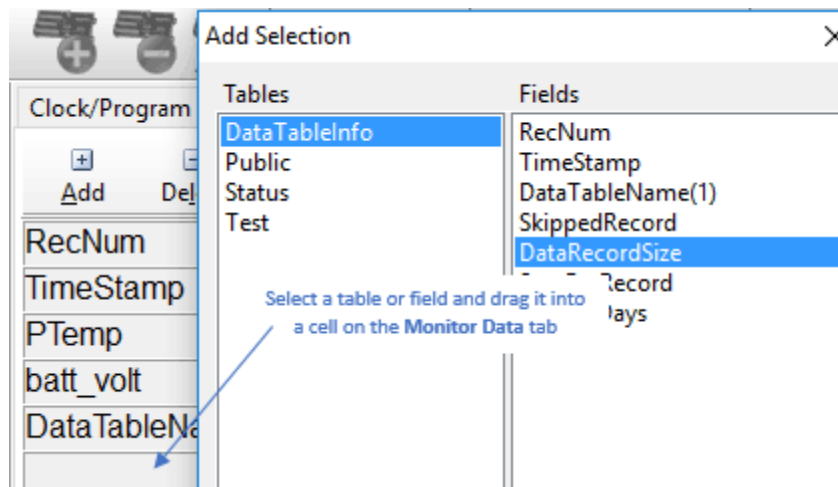
By default, the datalogger includes three tables: **Public**, **Status**, and **DataTableInfo**. Each of these tables only contains the most recent measurements and information.



- The **Public** table contains the measurements as they are made. It is updated at the scan interval set within the datalogger program.
- The **Status** table includes information on the health of the datalogger and is updated only when viewed.
- The **DataTableInfo** table reports statistics related to data tables. It also only updates when viewed.
- User-defined data tables update at the schedule set within the program.

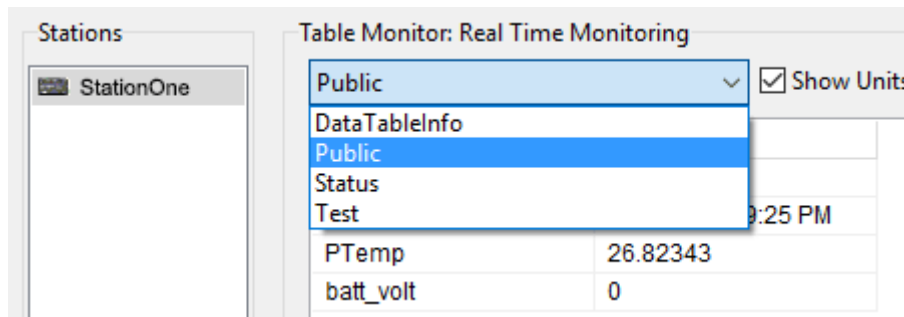
Monitoring Data

Follow a [tutorial](#) or use the following instructions:

PC200W and PC400 users, click the **Connect** button (), then click the **Monitor Data** tab. When this tab is first opened for a datalogger, values from the **Public** table are displayed. To view data from other tables, click the **Add** button (), select a table or field from the list, then drag it into a cell on the **Monitor Data** tab.






LoggerNet users, select **Main** and click the **Connect** button () on the LoggerNet toolbar, select the datalogger from the **Stations** list, then click the **Connect** button (). Once connected, you can select a table to view using the **Table Monitor** list.




Collecting Data

The datalogger writes to data tables according to the schedule set within the CRBasic program (see "Creating Data Tables in a Program" on page 23 for more information). After the program has been running for enough time to generate data records, data may be collected and reviewed via your datalogger support software. Collections may be done manually, or automatically through scheduled collections set in LoggerNet **Setup**. Follow a [tutorial](#) or use the following instructions.

COLLECTING DATA USING LOGGERNET

1. LoggerNet users, select **Main** and click the **Connect** button () on the LoggerNet toolbar, select the datalogger from the **Stations** list, then click the **Connect** button ()
2. Click the **Collect Now** button ()
3. After the data is collected, the **Data Collection Results** window displays the tables collected and where they are stored on the computer.
4. Click **View File** to view the data. See "Viewing Historic Data" on page 20 for more information.




COLLECTING DATA USING PC200W OR PC400W

1. PC200W and PC400 users, click the **Connect** button ()
2. Click the **Collect Data** tab.
3. Select an option for **What to Collect**. Either option creates a new file if one does not already exist.
 - **New data from datalogger (Append to data files)**: Collects only the data in the selected tables stored since the last data collection and appends this data to the end of the existing table files on the computer.
 - **All data from datalogger (Overwrite data files)**: Collects all of the data in the selected tables and replaces the existing table files on the computer.
4. Select the tables to collect from the list at the bottom of the window.
5. Click **Start Data Collection**.
6. The **Data Collection Results** window displays the tables collected and where they are stored on the computer.
7. Click **View File** to view the data.

Viewing Historic Data

View historic data in a spreadsheet format using View Pro. View Pro also contains tools for visualizing data in several graphical layouts. Follow a [tutorial](#) or use the following instructions:

Once the datalogger has had ample time to take multiple measurements, you can collect and review the data.

1. To view the most recent data, connect the datalogger to your computer and collect your data (see "Collecting Data" on page 20 for more information).
2. Open View Pro:
 - LoggerNet users select **Data** and click **View Pro** () on the LoggerNet toolbar.
 - PC200W and PC400 users click the **View Data Files via View Pro** toolbar button ()
3. Click the **Open** toolbar button () , navigate to the directory where you saved your tables (the default directory is `C:\Campbellsci\[your datalogger software application]`).

About Data Tables

A data table is essentially a file that resides in datalogger memory. The file consists of five or more rows. Each row consists of columns, or fields. The first four rows constitute the file header. Subsequent rows contain data records. Data tables may store individual measurements, individual calculated values, or summary data such as averages, maxima, or minima.

The file is written to each time data are directed to that file. The datalogger records data based on time or event. You can retrieve data based on a schedule or by manually choosing to collect data using datalogger support software (see "Collecting Data" on page 20). The number of data tables is limited to 250.

Example of a typical data table

TOA5, MyStation, CR1000X, 142, CRxxx.Std.01, CPU:MyTemperature.CR1X, 1958, OneMin				
TIMESTAMP	RECORD	BattV_Avg	PTemp_C_Avg	Temp_C_Avg
TS	RN	Volts	Deg C	Deg C
		Avg	Avg	Avg
2016-03-08 14:24:00	0	13.68	21.84	20.71
2016-03-08 14:25:00	1	13.65	21.84	20.63
2016-03-08 14:26:00	2	13.66	21.84	20.63
2016-03-08 14:27:00	3	13.58	21.85	20.62
2016-03-08 14:28:00	4	13.64	21.85	20.52
2016-03-08 14:29:00	5	13.65	21.85	20.64

By default, the datalogger includes three tables: **Public**, **Status**, and **DataTableInfo**. Each of these tables only contains the most recent measurements and information.

- The **Public** table contains the measurements as they are made. It is updated at the scan interval set within the datalogger program.
- The **Status** table includes information on the health of the datalogger and is updated only when viewed.
- The **DataTableInfo** table reports statistics related to data tables. It also only updates when viewed.
- User-defined data tables update at the schedule set within the program.

TABLE DEFINITIONS

Each data table is associated with overhead information, referred to as "table definitions," that becomes part of the file header (first few lines of the file) when data are downloaded to a computer. These include the table format, datalogger type and OS, name of the CRBasic program running in the datalogger, name of the data table (limited to 20 characters), and the alphanumeric field names to attach at the head of data columns,

FIRST HEADER ROW

The first header row of the data table is the environment line, which consists of eight fields. The following list describes the fields using the previous table entries as an example:

- 1: **TOA5** - Table output format. Changed via LoggerNet **Setup** (✖) **Standard View, Data Files** tab.
- 2: **MyStation** - Station name. Changed via LoggerNet **Setup**, Device Configuration Utility, or CRBasic program.
- 3: **CR1000X** - Datalogger model.
- 4: **142** - Datalogger serial number.

- 5: **CRxxx.Std.01** - Datalogger OS version. Changed via installation of a new OS.
- 6: **CPU:MyTemperature.CR1X** - Datalogger program name. Changed by sending a new program (see "Sending a Program to the Datalogger" on page 18 for more information).
- 7: **1958** - Datalogger program signature. Changed by revising a program or sending a new program (see "Sending a Program to the Datalogger" on page 18 for more information).
- 8: **OneMin** - Table name. Changed by revising a program (see "Creating Data Tables in a Program" on page 23 for more information).

SECOND HEADER ROW

The second header row reports field names. Default field names are a combination of the variable names (or aliases) from which data are derived, and a three-letter suffix. The suffix is an abbreviation of the data process that outputs the data to storage. A list of these abbreviations follows in "Data Process Names and Abbreviations" on page 22.

If a field is an element of an array, the field name will be followed by a list of subscripts within parentheses that identifies the array index. For example, a variable named `Values`, which is declared as a two-by-two array in the datalogger program, will be represented by four field names: `Values (1,1)`, `Values (1,2)`, `Values (2,1)`, and `Values (2,2)`. Scalar variables will not have array subscripts. There will be one value on this line for each scalar value defined by the table.

If the default field names are not acceptable to the programmer, the `FieldNames()` instruction can be used in the CRBasic program to customize the names. `TIMESTAMP`, `RECORD`, `BattV_Avg`, `PTemp_C_Avg`, and `Temp_C_Avg` are the default field names in the previous "Example of a typical data table" on page 21.

THIRD HEADER ROW

The third header row identifies engineering units for that field of data. These units are declared at the beginning of a CRBasic program, as shown in "Creating Data Tables in a Program" on page 23. In Short Cut, units are chosen when sensors or measurements are added. Units are strictly for documentation. The datalogger does not make use of declared units, nor does it check their accuracy.

FOURTH HEADER ROW

The fourth header row reports abbreviations of the data process used to produce the field of data.

Data Process Names and Abbreviations

- Totalize **Tot**
- Average **Avg**
- Maximum **Max**
- Minimum **Min**
- Sample at Max or Min **SMM**
- Standard Deviation **Std**
- Moment **MMT**
- Sample **No abbreviation**
- Histogram¹ **Hst**
- Histogram4D **H4D**
- FFT **FFT**
- Covariance **Cov**
- Level Crossing **LCr**

- WindVector **WVc**
- Median **Med**
- ET **ETsz**
- Solar Radiation (from ET) **RSO**
- Time of Max **TMx**
- Time of Min **TMn**

¹Hst is reported in the form `Hst, 20, 1.0000e+00, 0.0000e+00, 1.0000e+01` where `Hst` denotes a histogram, 20 = 20 bins, 1 = weighting factor, 0 = lower bound, 10 = upper bound.

DATA RECORDS

Subsequent rows are called data records. They include observed data and associated record keeping. The first field is a timestamp (**TS**), and the second field is the record number (**RN**).

The timestamp shown represents the time at which the data is written. Therefore, because the scan rate of the program `MyTemperature.CR1X` is 1 second, **Temp_C_Avg** in record number 3 in the previous "Example of a typical data table" on page 21, shows the average of the measurements taken over the minute beginning at 14:26:01 and ending at 14:27:00. As another example, consider rainfall measured every second with a daily total rainfall recorded in a data table written at midnight. The record timestamped 2016-03-08 00:00:00 will contain the total rainfall beginning at 2016-03-07 00:00:01 and ending at 2016-03-08 00:00:00.

Creating Data Tables in a Program

Data are stored in tables as directed by the CRBasic program. In Short Cut, data tables are created in the **Output** steps (see "Creating a Program in Short Cut" on page 16). Data tables are created within the CRBasic datalogger program using the `DataTable()` / `EndTable` instructions. They are placed after variable declarations and before the `BeginProg` instruction. Between `DataTable()` and `EndTable()` are instructions that define what data to store and under what conditions data are stored. A data table must be called by the CRBasic program for data storage processing to occur. Typically, data tables are called by the `CallTable()` instruction once each Scan. These instructions include:

```
DataTable()
  'Output Trigger Condition(s)
  'Output Processing Instructions
EndTable
```

For more information, see the CRBasic help.

Memory and Data Storage

During data table initialization, memory sectors are assigned to each data table according to the parameters set in the program. Program options that affect the allocation of memory include the *Size* parameter of the `DataTable()` instruction, the *Interval* and *Units* parameters of the `DataInterval()` instruction, and the *Interval* and *Units* parameters of the `Scan()` instruction. The datalogger uses those parameters to assign sectors in a way that maximizes the life of its memory.

By default, data storage memory sectors are organized as ring memory. When the ring is full, oldest data are overwritten by newest data. Using the `FillStop` statement sets a program to stop writing to the data table when it is full, and no more data are stored until the table is reset. To see the total number of records that can be stored before the oldest data are overwritten, or to reset tables, go to **Station Status | Table Fill Times** in your datalogger support software.

Data concerning the datalogger memory are posted in the **Status** and **DataTableInfo** tables.

MEMORY ALLOCATION

Data table SRAM and the CPU drive are automatically partitioned for use in the datalogger. The USR drive can be partitioned as needed. The USB drive is automatically partitioned when a Campbell Scientific mass-storage device is connected. The CRD drive is automatically partitioned when a memory card is installed. Primary storage for measurement data are those areas in SRAM allocated to data tables.

The CPU and USR drives use the FAT32 file system. There is no limit, beyond practicality and available memory, to the number of files that can be stored. While a FAT file system is subject to fragmentation, performance degradation is not likely to be noticed since the drive has a relatively small amount of solid state RAM and so is accessed very quickly.

CPU DRIVE

The serial flash memory CPU drive contains datalogger programs and other files. It holds a copy of final-data memory when the `TableFile()` instruction is used. It provides memory for `FileRead()` and `FileWrite()` operations. This memory is managed in File Control. When writing to files under program control, take care to write infrequently in order to prevent premature failure of serial flash memory.

SRAM

Measurement data is stored in data tables within SRAM. Data are usually erased from this area when a program is sent to the datalogger.

USR DRIVE

SRAM can be partitioned to create a FAT32 USR drive, analogous to partitioning a second drive on a computer hard disk. Certain types of files are stored to USR to reserve limited CPU drive memory for datalogger programs and calibration files. Partitioning also helps prevent interference from data table SRAM. The USR drive holds any file type within the constraints of the size of the drive and the limitations on filenames. Files typically stored include image files from cameras, certain configuration files, files written for FTP retrieval, HTML files for viewing with web access, and files created with the `TableFile()` instruction. Measurement data can also be stored on USR as discrete files by using the `TableFile()` instruction. Files on USR can be collected using the datalogger support software **Retrieve** command in File Control, or automatically using the LoggerNet **Setup | File Retrieval** tab functions.

USR is not affected by program recompilation or formatting of other drives. It will only be reset if the USR drive is formatted, a new operating system is loaded, or the size of USR is changed. USR size is set manually by accessing it in the Settings Editor, or programmatically by loading a CRBasic program with a USR drive size entered in a `SetStatus()` or `SetSetting()` instruction. Partition the USR drive to at least 11264 bytes in 512-byte increments. If the value entered is not a multiple of 512 bytes, the size is rounded up. Maximum size of USR 2990000 bytes.

Note: Placing an optional USR size setting in the CRBasic program overrides manual changes to USR size. When USR size is changed manually, the CRBasic program restarts and the programmed size for USR takes immediate effect.

Files in the USR drive can be managed through datalogger support software **File Control** or through the `FileManage()` instruction in CRBasic program.

MICROSD (CRD: DRIVE) STORAGE

The datalogger has a microSD card slot for removable, supplemental storage. The card can be configured as an extension of the datalogger final storage memory or as a repository of discrete data files. In data file mode, sub folders are not supported.

The CRD: drive uses microSD cards exclusively. Its primary purpose is the storage of data files in a compact binary format. Campbell Scientific recommends and supports only the use of microSD cards obtained from Campbell Scientific. These cards are industrial-grade and have passed Campbell Scientific hardware testing. Use of consumer grade cards substantially increases the risk of data loss. Following are listed advantages Campbell Scientific cards have over less expensive commercial-grade cards:

- Less susceptible to failure and data loss.
- Match the datalogger operating temperature range.
- Provide faster read/write times.
- Include vibration and shock resistance.
- Have longer life spans (more read/write cycles).

A maximum of 30 data tables can be created on a microSD card. When a data table is sent to a microSD card, a data table of the same name in SRAM is used as a buffer for transferring data to the card. When the card is present, the **Status** table will show the size of the table on the card. If the card is removed, the size of the table in SRAM is shown.

When a new program is compiled that sends data to the card, the datalogger checks if a card is present and if the card has adequate space for the data tables. If no card is present, or if space is inadequate, the datalogger will warn that the card is not being used. However, the CRBasic program runs anyway and data are stored to SRAM. When a card is inserted later, data accumulated in the SRAM table are copied to the card.

A microSD card can also facilitate the use of **powerup.ini** (see "Managing Memory via Powerup.ini" on page 26 for more information).

Formatting MicroSD Cards

The datalogger accepts microSD cards formatted as FAT16 or FAT32; however, FAT32 is recommended. Otherwise, some functionality, such as the ability to manage large numbers of files (>254) is lost. The datalogger formats memory cards as FAT32.

Because of the way the FAT32 card format works, you can avoid long datalogger compile times with a freshly formatted card by first formatting the new card on a computer, then copying a small file to the card from the computer, and then deleting the file with the computer. When the small file is copied to the card, the computer updates a sector on the card that which allows the datalogger program to compile faster. This only needs to be done once when the card is formatted. If you have the datalogger update the card sector, the first datalogger program compile with the card can take as long as 30 minutes. After that, compile times will be normal.

MicroSD Card Precautions

Observe the following precautions when using optional storage cards:

- Before removing a card from the datalogger, disable the card by pressing the **Eject** button, and waiting for the green LED. You then have 15 seconds to pull the card before normal operations resume.
- Do not remove a storage card while the drive is active, or data corruption and damage to the card may result.
- Prevent data loss by collecting data before sending a program from the storage card to the datalogger. Sending a program from the storage card to the datalogger often erases all data.

MicroSD LED Indicator

When the datalogger is powered and a microSD is connected, the microSD LED will turn on according to power and program states:

- **Red flash:** Card read/write activity.
- **Solid green:** Formatted card inserted, powered up. This LED also indicates it is OK to remove card. The **Eject** button must be pressed before removing a card to allow the datalogger to store buffered data to the card and then power it off.
- **Solid orange:** Error.
- **Dim/flashing orange:** Card has been removed and has been out long enough that CPU memory has wrapped and data is being overwritten without being stored to the card.

MANAGING MEMORY VIA POWERUP.INI

Another way to upload a program, install a datalogger OS, or format a drive is to create a **powerup.ini** program. The program is created with a text editor and saved to a memory card or SC115, along with associated files. Alternatively, the **powerup.ini** program and associated files can be saved directly to the datalogger using the datalogger support software **File Control | Send** command. With the memory card or SC115 connected, or with the **powerup.ini** file saved in the datalogger memory, a power cycle to the datalogger begins the process chosen in the **powerup.ini** program.

When uploading a program, the following rules determine what datalogger program to run:

- If the **Run Now** program is changed, then it is the program that runs.
- If no change is made to the **Run Now** program, but the **Run on Power Up** program is changed, the new **Run on Power Up** program runs.
- If neither **Run on Power Up** nor **Run Now** programs are changed, the previous **Run on Power Up** program runs.

Syntax for the **powerup.ini** program and available options follow.

Syntax

Syntax for **powerup.ini** is:

```
Command, File, Device
```

where,

- **Command** is one of the numeric commands in the following table.
- **File** is the accompanying operating system or user program file. File name can be up to 22 characters long.

- `Device` is the datalogger memory drive to which the accompanying operating system or user program file is copied (usually CPU). If left blank or with an invalid option, default device will be CPU. Use the same drive designation as the transporting external device if the preference is to not copy the file.

Command	Action	Details
1 ¹	Run always, preserve data	Copies a program file to a drive and sets the program to both Run Now and Run on Power Up . Data on a memory card from the previously running program will be preserved if table structures have not changed.
2	Run on power up	Copies a program file to a drive and sets the program to Run Always unless command 6 or 14 is used to set a separate Run Now program.
5	Format	Formats a drive.
6 ¹	Run now, preserve data	Copies a program file to a drive and sets the program to Run Now . Data on a memory card from the previously running program will be preserved if table structures have not changed.
7	Copy files	Copies a file, such as an Include or program support file, to the specified drive.
9	Load OS (File= .obj)	Loads an .obj file to the CPU drive and then loads the .obj file as the new datalogger operating system.
13	Run always, erase data	Copies a program to a drive and sets the program to both Run Now and Run on Power Up . Data on a memory card from the previously running program will be erased.
14	Run now, erase data	Copies a program to a drive and sets the program to Run Now . Data on a memory card from the previously running program will be erased.
15	Move file	Moves a file, such as an Include or program support file, to the specified drive.

¹Use `PreserveVariables()` instruction in the CRBasic program in conjunction with powerup.ini script commands **1** and **6** to preserve data and variables.

Example Powerup.ini Programs

Comments can be added to the file by preceding them with a single-quote character ('). All text after the comment mark on the same line is ignored.

Caution: Test the `powerup.ini` file and procedures in the lab before going to the field. Always carry a laptop or mobile device (with datalogger support software) into difficult- or expensive-to-access places as backup.

Example: Code Format and Syntax

```
'Command = numeric power up command
'File = file associated with the action
'Device = device to which File is copied. Defaults to CPU

'Command,File,Device
13,Write2CRD_2.crlx,cpu:
```

Example: Run Program on Power Up

```
'Copy program file pwrup.crlx from the external drive to CPU:
'File will run only when the datalogger is powered-up later.
2,pwrup.crlx,cpu:
```

Example: Format the USB Drive

```
5,,usr:
```

Example: Send OS on Power Up

```
'Load an operating system (.obj) file into FLASH as the new OS
9,CR1000X.Std.01.obj
```

Example: Run Program from SC115 Flash Memory Drive

'A program file is carried on an SC115 Flash Memory drive.

'Do not copy program file from SC115. Run program always, erase data.

13, toobigforcpu.crlx, usb:

Example: Always Run Program, Erase Data

13, pwrap_1.crlx, cpu:

Example: Run Program Now and Erase Data Now

14, run.crlx, cpu:

Measurements

Voltage Measurements	29
Current Measurements	30
Ratiometric Resistance Measurements	32
Thermocouple Measurements	36
Period-Averaging Measurements	37
Pulse Measurements	37
Vibrating Wire Measurements	42

Voltage Measurements

Voltage measurements are made using an ADC. A high-impedance programmable-gain amplifier amplifies the signal. Internal multiplexers route individual terminals within the amplifier. The CRBasic measurement instruction controls the ADC gain and configuration - either single-ended or differential input. Information on the differences between single-ended and differential measurements can be found here: "Deciding Between Single-Ended or Differential Measurements" on page 66.

A voltage measurement proceeds as follows:

1. Set PGIA gain for the voltage range selected with the CRBasic measurement instruction parameter *Range*. Set the ADC for the first notch frequency selected with *fN1*.
2. If used, turn on excitation to the level selected with *ExmV*.
3. Multiplex selected terminals (*InChan*, *SEChan*, or *DiffChan*).
4. Delay for the entered settling time (*SettlingTime*).
5. Perform the ADC.
6. Repeat for input reversal as determined by parameters *RevEx* and *RevDiff*.
7. Apply multiplier (*Multi*) and offset (*Offset*) to measured result.

In general, use the smallest input range that accommodates the full-scale output of the sensor. This results in the best measurement accuracy and resolution (see "Analog Specifications" on page 80 for more information).

A set overhead reduces the chance of overrange. Overage limits are available in the specifications. The datalogger indicates a measurement overrange by returning a **NAN** for the measurement.

Warning: Sustained voltages in excess of ± 20 V applied to terminals configured for analog input will damage CR1000X circuitry.

SINGLE-ENDED MEASUREMENTS

A single-ended measurement measures the difference in voltage between the terminal configured for single-ended input and the reference ground. For example, single-ended channel 1 is comprised of terminals **SE 1** and ⏏ . Single-ended terminals are labeled in blue. For more information, see "CR1000X Wiring Panel" on page 3. The single-ended configuration is used with the following CRBasic instructions:

- **VoltSE()**
- **BrHalf()**
- **BrHalf3W**
- **TCSE()**
- **Therm107()**
- **Therm108()**
- **Therm109()**

DIFFERENTIAL MEASUREMENTS

A differential measurement measures the difference in voltage between two input terminals. For example, **DIFF** channel 1 is comprised of terminals **1H** and **1L**, with **1H** as high and **1L** as low. For more information, see "CR1000X Wiring Panel" on page 3. The differential configuration is used with the following CRBasic instructions:

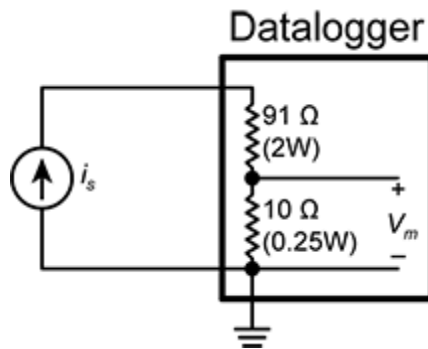
- `VoltDiff()`
- `BrFull()`
- `BrFull6W()`
- `BrHalf4W()`
- `TCDiff()`

Reverse Differential

Differential measurements have the advantage of an input reversal option, *RevDiff*. When *RevDiff* is set to `True`, two differential measurements are made, the first with a positive polarity and the second reversed. Subtraction of opposite polarity measurements cancels some offset voltages associated with the measurement.

Current Measurements

RG terminals can be configured to make analog current measurements using the `CurrentSE()` instruction. When configured to measure current, terminals each have an internal resistance of $101\ \Omega$ which is placed in the current measurement loop. The return path of the sensor must be connected directly to the **G** terminal closest to the terminal used. The following image shows a simplified schematic of a current measurement.



EXAMPLE CURRENT MEASUREMENT CONNECTIONS

The following table shows example schematics for connecting typical current sensors and devices.

Sensor Type	Connection Example
2-wire transmitter using datalogger power	

Sensor Type	Connection Example
2-wire transmitter using external power	
3-wire transmitter using datalogger power	
3-wire transmitter using external power	
4-wire transmitter using datalogger power	
4-wire transmitter using external power	

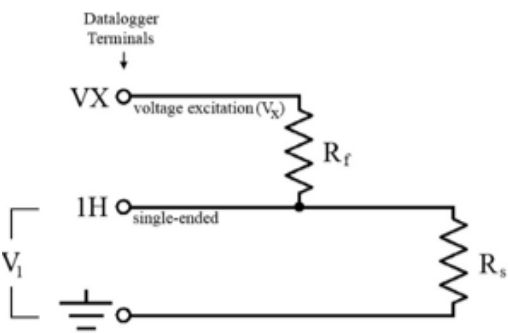
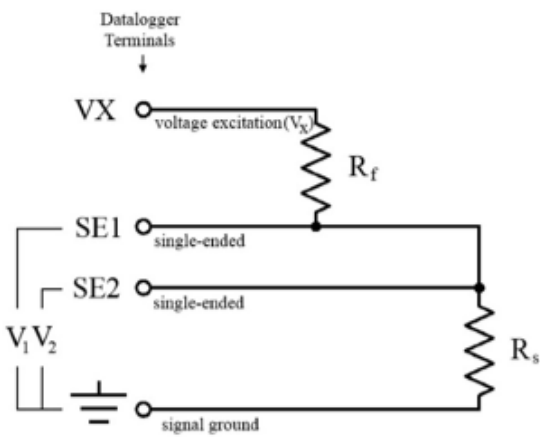
See also "Current Measurements Specifications" on page 83.

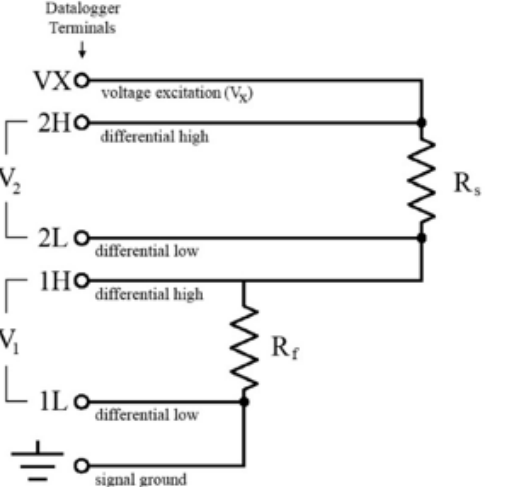
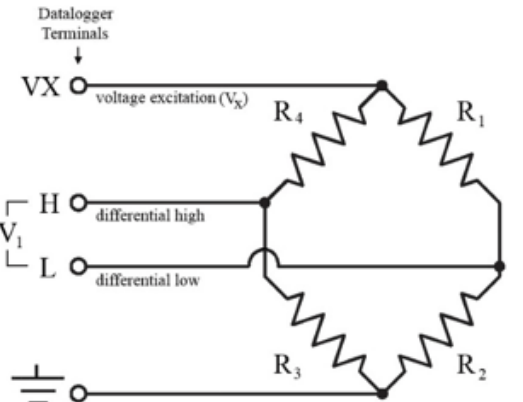
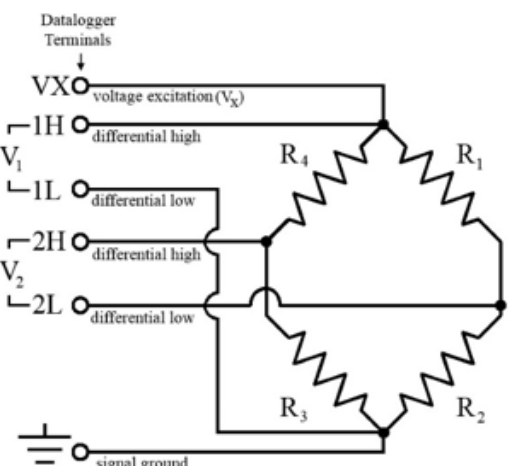
Ratiometric Resistance Measurements

Bridge resistance is determined by measuring the difference between a known voltage applied to the excitation (input) of a resistor bridge and the voltage measured on the output arm. The datalogger supplies a precise voltage excitation via **VX** terminals. Return voltage is measured on analog input terminals configured for single-ended (**SE**) or differential (**DIFF**) input. The result of the measurement is a ratio of measured voltages, as shown in the following table.

CRBasic instructions for measuring resistance include:

- **BrHalf()** - half-bridge
- **BrHalf3W()** - three-wire half-bridge
- **BrHalf4W()** - four-wire half-bridge
- **BrFull()** - four-wire full-bridge
- **BrFull6W()** - six-wire full-bridge

Resistive-Bridge Type and Circuit Diagram	CRBasic Instruction and Fundamental Relationship	Relational Formulas
<p>Half-Bridge¹</p> 	<p>CRBasic Instruction: BrHalf()</p> <p>Fundamental Relationship²:</p> $X = \frac{V_1}{V_x} = \frac{R_s}{R_s + R_f}$	$R_s = R_f \frac{X}{1 - X}$ $R_f = \frac{R_s(1 - X)}{X}$
<p>Three-Wire Half-Bridge^{1,3}</p> 	<p>CRBasic Instruction: BrHalf3W()</p> <p>Fundamental Relationship²:</p> $X = \frac{2V_2 - V_1}{V_x - V_1} = \frac{R_s}{R_f}$	$R_f = R_s / X$ $R_s = R_f X$

Resistive-Bridge Type and Circuit Diagram	CRBasic Instruction and Fundamental Relationship	Relational Formulas
<p>Four-Wire Half-Bridge^{1,3}</p> 	<p>CRBasic Instruction: <code>BrHalf4W()</code></p> <p>Fundamental Relationship²:</p> $X = \frac{V_2}{V_1} = \frac{R_s}{R_f}$	$R_s = R_f X$ $R_f = R_s / X$
<p>Full-Bridge^{1,3}</p> 	<p>CRBasic Instruction: <code>BrFull()</code></p> <p>Fundamental Relationship²:</p> $X = 1000 \frac{V_1}{V_x}$ $= 1000 \left(\frac{R_3}{R_3 + R_4} - \frac{R_2}{R_1 + R_2} \right)$	<p>These relationships apply to <code>BrFull()</code> and <code>BrFull6W()</code></p> $X_1 = \frac{-X}{1000} + \frac{R_3}{R_3 + R_4}$ $R_1 = \frac{R_2(1 - X_1)}{X_1}$ $R_2 = \frac{R_1 X_1}{1 - X_1}$
<p>Six-Wire Full Bridge¹</p> 	<p>CRBasic Instruction: <code>BrFull6W()</code></p> <p>Fundamental Relationship²:</p> $X = 1000 \frac{V_2}{V_1}$ $= 1000 \left(\frac{R_3}{R_3 + R_4} - \frac{R_2}{R_1 + R_2} \right)$	$X_2 = \frac{X}{1000} + \frac{R_2}{R_1 + R_2}$ $R_3 = \frac{R_4 X_2}{1 - X_2}$ $R_4 = \frac{R_3(1 - X_2)}{X_2}$

¹Key: V_x = excitation voltage; V_1 , V_2 = sensor return voltages; R_f = fixed, bridge or completion resistor; R_s = variable or sensing resistor.

²Where X = result of the CRBasic bridge measurement instruction with a multiplier of 1 and an offset of 0.

³Campbell Scientific offers a list of available terminal input modules to facilitate this measurement.

RESISTIVE-BRIDGE MEASUREMENTS AND VOLTAGE EXCITATION

Offset voltage compensation applies to bridge measurements. In addition to *RevDiff* and *MeasOff* parameters discussed in "Minimizing Offset Voltages" on page 72, CRBasic bridge measurement instructions include the *RevEx* parameter that provides the option to program a second set of measurements with the excitation polarity reversed. Much of the offset error inherent in bridge measurements is canceled out by setting *RevDiff*, *RevEx*, and *MeasOff* to **True**.

Measurement speed can be slowed when using *RevDiff*, *MeasOff*, and *RevEx*. When more than one measurement per sensor are necessary, such as occur with the *BrHalf3W()*, *BrHalf4W()*, and *BrFull16W* instructions, input and excitation reversal are applied separately to each measurement. For example, in the four-wire half-bridge (*BrHalf4W()*), when excitation is reversed, the differential measurement of the voltage drop across the sensor is made with excitation at both polarities and then excitation is again applied and reversed for the measurement of the voltage drop across the fixed resistor. The results of measurement instructions (X) must then be processed further to obtain the resistance value, which requires additional program execution time.

CRBasic Example: Four-Wire Full Bridge Measurement and Processing

```
'This program example demonstrates the measurement and  
'processing of a four-wire resistive full bridge.  
'In this example, the default measurement stored  
'in variable X is deconstructed to determine the  
'resistance of the R1 resistor, which is the variable  
'resistor in most sensors that have a four-wire  
'full-bridge as the active element.
```

```
'Declare Variables
```

```
Public X  
Public X_1  
Public R_1  
Public R_2 = 1000 'Resistance of fixed resistor R2  
Public R_3 = 1000 'Resistance of fixed resistor R2  
Public R_4 = 1000 'Resistance of fixed resistor R4
```

```
'Main Program
```

```
BeginProg
```

```
  Scan(500,mSec,1,0)
```

```
    'Full Bridge Measurement:
```

```
    BrFull(X,1,mV200,1,Vx1,1,4000,True,True,0,60,1.0,0.0)
```

```
    X_1 = ((-1 * X) / 1000) + (R_3 / (R_3 + R_4))
```

```
    R_1 = (R_2 * (1 - X_1)) / X_1
```

```
  NextScan
```

```
EndProg
```

See also "Ratiometric-Resistance Measurements Specifications" on page 81.

STRAIN MEASUREMENTS

A principal use of the four-wire full bridge is the measurement of strain gages in structural stress analysis. *StrainCalc()* calculates microstrain ($\mu\epsilon$) from the formula for the particular strain bridge configuration used. All strain gages supported by *StrainCalc()* use the full-bridge schematic. In strain-gage parlance, 'quarter-bridge', 'half-bridge' and 'full-bridge' refer to the number of active

elements in the full-bridge schematic. In other words, a quarter-bridge strain gage has one active element, a half-bridge has two, and a full-bridge has four.

`StrainCalc()` requires a bridge-configuration code. The following table shows the equation used by each configuration code. Each code can be preceded by a dash (-). Use a code without the dash when the bridge is configured so the output decreases with increasing strain. Use a dashed code when the bridge is configured so the output increases with increasing strain. A dashed code sets the polarity of V_r to negative.

StrainCalc() BrConfig Code	Configuration
1	Quarter-bridge strain gage ¹ : $\mu\varepsilon = \frac{-4 * 10^6 V_r}{GF(1 + 2V_r)}$
2	Half-bridge strain gage ¹ . One gage parallel to strain, the other at 90° to strain: $\mu\varepsilon = \frac{-4 * 10^6 V_r}{GF[(1 + \nu) - 2V_r(\nu - 1)]}$
3	Half-bridge strain gage. One gage parallel to + ε , the other parallel to - ε ¹ : $\mu\varepsilon = \frac{-2 * 10^6 V_r}{GF}$
4	Full-bridge strain gage. Two gages parallel to + ε , the other two parallel to - ε ¹ : $\mu\varepsilon = \frac{-10^6 V_r}{GF}$
5	Full-bridge strain gage. Half the bridge has two gages parallel to + ε and - ε , and the other half to + $\nu\varepsilon$ and - $\nu\varepsilon$ 1: $\mu\varepsilon = \frac{-2 * 10^6 V_r}{GF(\nu + 1)}$
6	Full-bridge strain gage. Half the bridge has two gages parallel to + ε and - $\nu\varepsilon$, and the other half to - $\nu\varepsilon$ and + ε ¹ : $\mu\varepsilon = \frac{-2 * 10^6 V_r}{GF[(\nu + 1) - V_r(\nu - 1)]}$

- ν : Poisson's Ratio (0 if not applicable).
- GF: Gage Factor.
- V_r : 0.001 (Source-Zero) if **BRConfig** code is positive (+).
- V_r : -0.001 (Source-Zero) if **BRConfig** code is negative (-).

and where:

- "source": the result of the full-bridge measurement ($X = 1000 \cdot V1 / Vx$) when multiplier = 1 and offset = 0.
- "zero": gage offset to establish an arbitrary zero.

AC EXCITATION

Some resistive sensors require ac excitation. Ac excitation is defined as excitation with equal positive (+) and negative (-) duration and magnitude. These include electrolytic tilt sensors, soil moisture blocks, water-conductivity sensors, and wetness-sensing grids. The use of single polarity dc excitation with these sensors can result in polarization of sensor materials and the substance measured. Polarization may cause erroneous measurement, calibration changes, or rapid sensor decay.

Other sensors, for example, LVDTs (linear variable differential transformers), require ac excitation because they require inductive coupling to provide a signal. Dc excitation in an LVDT will result in no measurement.

CRBasic bridge-measurement instructions have the option to reverse polarity to provide ac excitation by setting the *RevEx* parameter to **True**.

Note: Take precautions against ground loops when measuring sensors that require ac excitation. See also "Minimizing Ground Loop Errors" on page 65.

ACCURACY FOR RATIOMETRIC RESISTANCE MEASUREMENTS

Consult the following technical papers for in-depth treatments of several topics addressing voltage measurement quality:

- [Preventing and Attacking Measurement Noise Problems](#)
- [Benefits of Input Reversal and Excitation Reversal for Voltage Measurements](#)
- [Voltage Measurement Accuracy, Self-Calibration, and Ratiometric Measurements](#)

Note: Error discussed in this section and error-related specifications of the CR1000X do not include error introduced by the sensor or by the transmission of the sensor signal to the datalogger.

For accuracy specifications for ratiometric resistance measurements, see "Ratiometric-Resistance Measurements Specifications" on page 81. Voltage measurement is variable V_1 or V_2 in "Resistance Measurements" on page 32. Offset is the same as that for simple analog voltage measurements.

Assumptions that support the ratiometric-accuracy specification include:

- Datalogger is within factory calibration specification.
- Input reversal for differential measurements and excitation reversal for excitation voltage are within specifications.
- Effects due to the following are not included in the specification:
 - o Bridge-resistor errors
 - o Sensor noise
 - o Measurement noise

Thermocouple Measurements

Thermocouple measurements are special case voltage measurements.

Note: Thermocouples are inexpensive and easy to use. However, they pose several challenges to the acquisition of accurate temperature data, particularly when using external reference junctions.

A thermocouple consists of two wires, each of a different metal or alloy, joined at one end to form the measurement junction. At the opposite end, each lead connects to terminals of a voltage measurement

device, such as the datalogger. These connections form the reference junction. If the two junctions (measurement and reference) are at different temperatures, a voltage proportional to the difference is induced in the wires. This phenomenon is known as the Seebeck effect.

Measurement of the voltage between the positive and negative terminals of the voltage-measurement device provides a direct measure of the temperature difference between the measurement and reference junctions. A third metal (e.g., solder or datalogger terminals) between the two dissimilar-metal wires form parasitic-thermocouple junctions, the effects of which cancel if the two wires are at the same temperature. Consequently, the two wires at the reference junction are placed in close proximity so they remain at the same temperature.

Knowledge of the reference junction temperature provides the determination of a reference junction compensation voltage, corresponding to the temperature difference between the reference junction and 0°C. This compensation voltage, combined with the measured thermocouple voltage, can be used to compute the absolute temperature of the thermocouple junction.

`TCDiff()` and `TCSe()` thermocouple instructions determine thermocouple temperatures using the following sequence. First, the temperature (°C) of the reference junction is determined. Next, a reference junction compensation voltage is computed based on the temperature difference between the reference junction and 0°C. If the reference junction is the datalogger analog-input terminals, the temperature is conveniently measured with the `PanelTemp()` instruction. The actual thermocouple voltage is measured and combined with the reference junction compensation voltage. It is then used to determine the thermocouple-junction temperature based on a polynomial approximation of NIST thermocouple calibrations.

Period-Averaging Measurements

Period-averaging measurements use a high-frequency digital clock to measure time differences between signal transitions, whereas pulse-count measurements simply accumulate the number of counts. As a result, period-average measurements offer much better frequency resolution per measurement interval than pulse-count measurements. See also "Pulse Measurements" on page 37.

SE terminals on the datalogger are configurable for measuring the period of a signal.

The measurement is performed as follows: low-level signals are amplified prior to a voltage comparator. The internal voltage comparator is referenced to the programmed threshold. The threshold parameter allows referencing the internal voltage comparator to voltages other than 0 V. For example, a threshold of 2500 mV allows a 0 to 5 Vdc digital signal to be sensed by the internal comparator without the need for additional input conditioning circuitry. The threshold allows direct connection of standard digital signals, but it is not recommended for small-amplitude sensor signals.

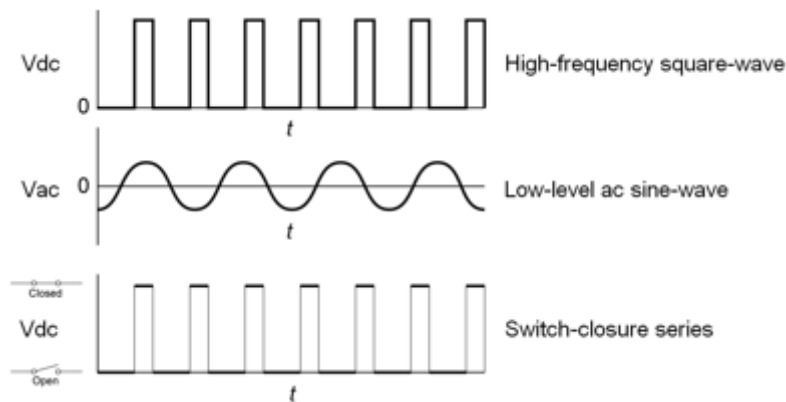
A threshold other than zero results in offset voltage drift, limited accuracy ($\approx \pm 10$ mV) and limited resolution (≈ 1.2 mV).

See also "Period Averaging Specifications" on page 81.

Pulse Measurements

The output signal generated by a pulse sensor is a series of voltage waves. The sensor couples its output signal to the measured phenomenon by modulating wave frequency. The datalogger detects the state transition as each wave varies between voltage extremes (high-to-low or low-to-high). Measurements are processed and presented as counts, frequency, or timing data.

The datalogger includes terminals that are configurable for pulse input to measure counts or frequency as shown in the following image.



Pulse input terminals and the input types they can measure:

Input Type	Pulse Input Terminal	Data Option
High-frequency	P1 P2 C (all)	<ul style="list-style-type: none"> • Counts • Frequency • Running average of frequency
Low-level ac	P1 P2	
Switch-closure	P1 P2 C (all)	

Using the `PulseCount()` instruction, **P** and **C** terminals are configurable for pulse input to measure counts or frequency. Maximum input frequency is dependent on input voltage (see "Pulse Counting Specifications" on page 82 for more information). If pulse input voltages exceed the maximum voltage, third-party external-signal conditioners should be employed. Under no circumstances should voltages greater than 20 V be measured.

P and **C** terminals configured for pulse input have internal filters that reduce electronic noise, and thus reduce false counts. Internal ac coupling is used to eliminate dc offset voltages. High-frequency pulse inputs are routed to an inverting CMOS input buffer with input hysteresis. For tips on working with pulse measurements, see "Pulse Measurement Tips" on page 41.

For more information, see "Pulse Counting Specifications" on page 82.

LOW-LEVEL AC MEASUREMENTS

Low-level ac (sine-wave) signals can be measured on **P** terminals. Ac generator anemometers typically output low-level ac.

Measurements include the following:

- Counts
- Frequency (Hz)
- Running average

Rotating magnetic-pickup sensors commonly generate ac voltage ranging from thousandths of volts at low-rotational speeds to several volts at high-rotational speeds.

CRBasic instruction: `PulseCount()`

Low-level ac signals cannot be measured directly by **C** terminals. Peripheral terminal expansion modules are available for converting low-level ac signals to square-wave signals measurable by **C** terminals.

For more information, see "Pulse Counting Specifications" on page 82.

HIGH FREQUENCY MEASUREMENTS

High-frequency (square-wave) signals can be measured on **P** or **C** terminals. Common sensors that output high-frequency include:

- Photo-chopper anemometers
- Flow meters

Measurements include counts, frequency in hertz, and running average. Note that the resolution of a frequency measurement can be different depending on whether the measurement is made with the `PulseCount()` or `TimerIO()` instruction. See the CRBasic help for more information.

P Terminals

- CRBasic instructions: `PulseCount()`

High-frequency pulse inputs are routed to an inverting CMOS input buffer with input hysteresis.

C Terminals

- CRBasic instructions: `PulseCount()`, `TimerIO()`

See "Specifications" on page 77 for more information.

SWITCH-CLOSURE AND OPEN-COLLECTOR MEASUREMENTS

Switch-closure and open-collector signals can be measured on **P** or **C** terminals. Mechanical switch-closures have a tendency to bounce before solidly closing. Unless filtered, bounces can cause multiple counts per event. The datalogger automatically filters bounce. Because of the filtering, the maximum switch-closure frequency is less than the maximum high-frequency measurement frequency. Sensors that commonly output a switch-closure or open-collector signal include:

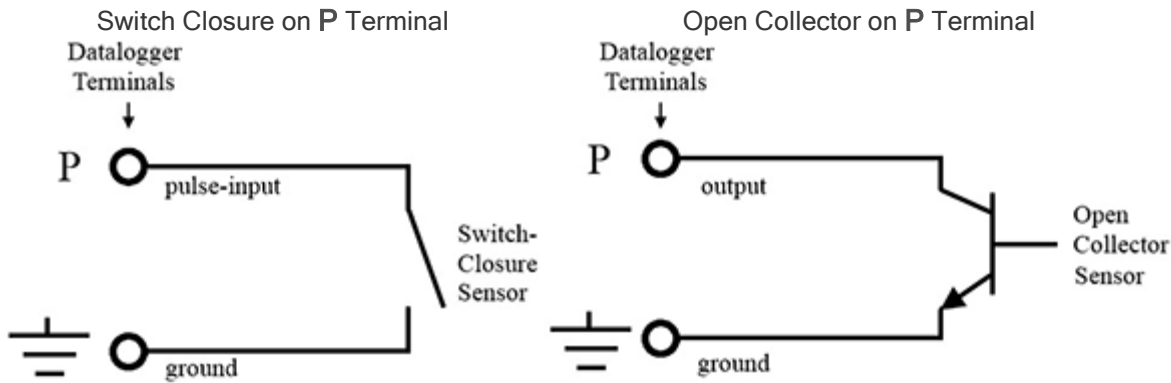
- Tipping-bucket rain gages
- Switch-closure anemometers
- Flow meters

Data output options include counts, frequency (Hz), and running average.

P Terminals

An internal 100 k Ω pull-up resistor pulls an input to 5 Vdc with the switch open, whereas a switch-closure to ground pulls the input to 0 V. An internal hardware debounce filter has a 3.3 ms time constant.

- CRBasic instruction: `PulseCount()`



C Terminals

Switch-closure mode is a special case edge-count function that measures dry-contact switch-closures or open collectors. The operating system filters bounces.

- CRBasic instruction: `PulseCount()`

See also "Pulse Counting Specifications" on page 82.

EDGE TIMING AND EDGE COUNTING

Edge time, period, and counts can be measured on **P** or **C** terminals. Measurements for feedback control using pulse-width modulation (PWM) are an example of an edge timing application.

Single Edge Timing

A single edge or state transition can be measured on **P** or **C** terminals. Measurements can be expressed as a frequency (Hz) or period (μs):

CRBasic instructions:

- `TimerInput()` - available on **C1-C8**
- `PulseCount()` - available on **C1-C8** and **P1-P2**
- `PeriodAvg()` - available on **SE1-SE16**

Multiple Edge Counting

Time between edges, time from an edge on the previous channel, and edges that span the scan interval can be measured on **C** terminals:

- CRBasic instruction: `TimerInput()` - available on **C1-C8**

Timer Input on NAN Conditions

NAN is the result of a `TimerInput()` measurement if one of the following occurs:

- Measurement timer expires.
- The signal frequency is too fast.

For more information, see:

- "Pulse Counting Specifications" on page 82
- "Digital I/O Specifications" on page 83
- "Period Averaging Specifications" on page 81

PULSE MEASUREMENT TIPS

The `PulseCount()` instruction uses dedicated 32-bit counters to accumulate all counts over the programmed scan interval. The resolution of pulse counters is one count or 1 Hz. Counters are read at the beginning of each scan and then cleared. Counters will overflow if accumulated counts exceed 4,294,967,296 (2^{32}), resulting in erroneous measurements.

- Counts are the preferred `PulseCount()` output option when measuring the number of tips from a tipping-bucket rain gage or the number of times a door opens. Many pulse-output sensors, such as anemometers and flow meters, are calibrated in terms of frequency (Hz) so are usually measured using the `PulseCount()` frequency-output option.
- Use the LLAC4 module to convert non-TTL-level signals, including low-level ac signals, to TTL levels for input into C terminals.

Understanding the signal to be measured and compatible input terminals and CRBasic instructions is helpful. See “Pulse input terminals and the input types they can measure” on page 38. **Pulse input terminals and the input types they can measure**

Input Filters and Signal Attenuation

P and C terminals configured for pulse input have internal filters that reduce electronic noise. The electronic noise can result in false counts. However, input filters attenuate (reduce) the amplitude (voltage) of the signal. Attenuation is a function of the frequency of the signal. Higher-frequency signals are attenuated more. If a signal is attenuated enough, it may not pass the detection thresholds required by the pulse count circuitry.

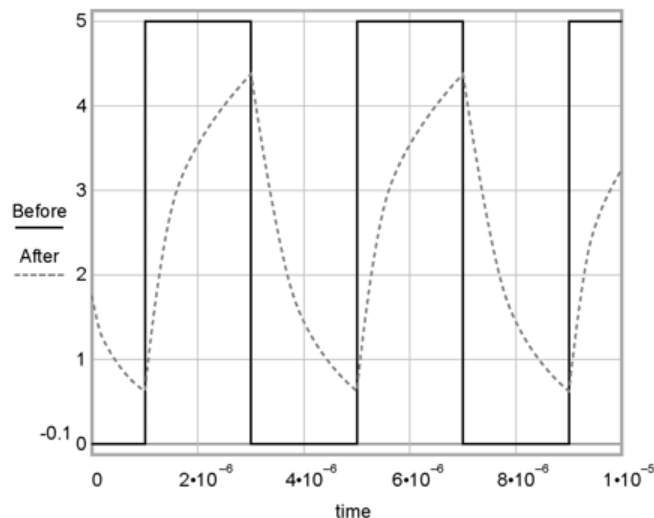
The metric for filter effectiveness is τ , the filter time constant. The higher the τ value, the less noise that gets through the filter, but the lower the signal frequency must be to pass the detection thresholds.

While a C terminal measured with the `TimerIO()` frequency measurement may be superior for clean signals, a P terminal filter (much higher τ) may be required to get a measurement on an electronically noisy signal.

For example, increasing voltage is required for low-level ac inputs to overcome filter attenuation on P terminals configured for low-level ac:

- 8.5 ms time constant filter (19 Hz 3 dB frequency) for low-amplitude signals.
- 1 ms time constant (159 Hz 3 dB frequency) for larger (> 0.7 V) amplitude signals.

An example showing the amplitude reduction that results from τ in high-frequency pulse input mode is illustrated in the following image:



Vibrating Wire Measurements

The datalogger can measure vibrating wire sensors through vibrating-wire interface modules. Vibrating wire sensors are the sensor of choice in many environmental and industrial applications that need sensor stability over very long periods, such as years or even decades. A thermistor included in most sensors can be measured to compensate for temperature errors.

Measuring the resonant frequency by means of period averaging is the classic technique, but Campbell Scientific has developed static and dynamic spectral-analysis techniques (VSPECT) that produce superior noise rejection, higher resolution, diagnostic data, and, in the case of dynamic VSPECT, measurements up to 333.3 Hz. For detailed information on VSPECT, see [Vibrating Wire Spectral Analysis Technology](#).

Communication Protocols

Dataloggers communicate with datalogger support software and other Campbell Scientific dataloggers using a number of protocols including PakBus, Modbus, DNP3, CPI, SPI, and TCP/IP. Several industry-specific protocols are also supported. CAN bus is supported when using the Campbell Scientific SDM-CAN communication module. See also "Communications Specifications" on page 84.

General Serial Communications	43
Modbus Communications	44
Internet Communications	45
DNP3 Communications.....	45
PakBus Communications.....	46
SDI-12 Communication.....	46
Serial Peripheral Interface (SPI) and I2C	49

Some communication services, such as satellite networks, can be expensive to send and receive information. Best practices for reducing expense include:

- Declare **Public** only those variables that need to be public. Other variables should be declared as **Dim**.
- Be conservative with use of string variables and string variable sizes. Make string variables as big as they need to be and no more. The default size, if not specified, is 24 bytes, but the minimum is 4 bytes. Specify the size if 24 bytes of possibly unused space is not desired. Declare string variables **Public** and sample string variables into data tables only as needed.
- When using `GetVariables ()` / `SendVariables ()` to send values between dataloggers, put the data in an array and use one command to get the multiple values. Using one command to get 10 values from an array and swath of 10 is much more efficient (requires only 1 transaction) than using 10 commands to get 10 single values (requires 10 transactions).
- Set the datalogger to be a PakBus router only as needed. When the datalogger is a router, and it connects to another router like LoggerNet, it exchanges routing information with that router and, possibly (depending on your settings), with other routers in the network. Network Planner set this appropriately when it is used. This is also set through the **IsRouter** setting in the Settings Editor.
- Set PakBus beacons and verify intervals properly. For example, there is no need to verify routes every five minutes if communications are expected only every 6 hours. Network Planner will set this appropriately when it is used. This is also set through the **Beacon** and **Verify** settings in the Settings Editor.

For information on Designing a PakBus network using the Network Planner tool in LoggerNet, watch the following [video](#).

General Serial Communications

The datalogger supports two-way serial communication. These communication ports can be used with smart sensors that deliver measurement data through serial data protocols or with devices, such as modems, that communicate using serial data protocols. See "Communication Ports" on page 12 for information on port configuration options.

CRBasic instructions for general serial communications include:

- `SerialOpen()`
- `SerialClose()`
- `SerialIn()`
- `SerialInRecord()`
- `SerialInBlock()`
- `SerialOut()`
- `SerialOutBlock()`

To communicate over a serial port, it is important to be familiar with the protocol of the sensor or device. Refer to the manual of the sensor or device to find its protocol and then select the appropriate options for each CRBasic parameter. See the application note [Interfacing Serial Sensors with Campbell Scientific Dataloggers](#) for more programming details and examples.

Modbus Communications

The datalogger supports Modbus RTU, Modbus ASCII, and Modbus TCP protocols and can be programmed as a Modbus master or Modbus slave. These protocols are often used in Modbus SCADA networks. Dataloggers can communicate with Modbus on all available communication ports. The datalogger supports RTU and ASCII communication modes on RS-232 and RS-485 connections.

Modbus has a set command structure. It uses a common bus and addresses each node.

CRBasic Modbus instructions include (see CRBasic Editor help for the most recent information on each of these instructions and for program examples):

- `ModbusMaster()`
- `ModbusSlave()`
- `MoveBytes()`

For additional information on Modbus, see:

- [Why Modbus Matters: An Introduction](#)
- [How to Access Live Measurement Data Using Modbus](#)
- [Using Campbell Scientific Dataloggers as Modbus Slave Devices in a SCADA Network](#)

Because Modbus has a set command structure, programming the datalogger to get data from field instruments is much simpler than from serial sensors. Because Modbus uses a common bus and addresses each node, field instruments are effectively multiplexed to a datalogger without additional hardware.

A datalogger goes into sleep mode after 40 seconds of communication inactivity. Once asleep, two packets are required before it will respond. The first packet awakens the datalogger; the second packet is received as data. This would make a Modbus master fail to poll the datalogger, if not using retries. The datalogger, through Device Configuration Utility or the **Status** table, can be set to keep communication ports open and awake, but at higher power usage.

Further information on Modbus can be found at:

- www.simplyModbus.ca/FAQ.htm
- www.Modbus.org/tech.php
- www.lammertbies.nl/comm/info/modbus.html

Internet Communications

See the "Communications Specifications" on page 84 for a list of the internet protocols supported by the datalogger .

CRBasic instructions for internet communications include:

- `EmailRelay()`
- `EmailSend()`
- `EmailRecv()`
- `FTPClient()`
- `HTTPGet()`
- `HTTPOut()`
- `HTTPPost()`
- `HTTPPut`
- `IPInfo()`
- `PPPOpen()`
- `PPPClose()`
- `TCPOpen()`
- `TCPClose()`

Once the hardware has been configured, basic PakBus communication over TCP/IP is possible. These functions include the following:

- Sending programs
- Retrieving programs
- Setting the datalogger clock
- Collecting data
- Displaying the current record in a data table

Data callback and datalogger-to-datalogger communications are also possible over TCP/IP. For details and example programs for callback and datalogger-to-datalogger communications, consult your network link manual (see www.campbellsci.com/internet-ip-networks).

IP ADDRESS

When connected to a server with a list of IP addresses available for assignment, the datalogger will automatically request and obtain an IP address through DHCP. Once the address is assigned, look in the **Settings Editor: Ethernet** | {information box} to see the assigned IP address.

The CR1000X provides a DNS client that can query a DNS server to determine if an IP address has been mapped to a hostname. If it has, then the hostname can be used interchangeably with the IP address in some datalogger instructions.

HTTPS

The datalogger has the ability to act as an HTTPS server. This can be configured using Device Configuration Utility.

DNP3 Communications

DNP3 is designed to optimize transmission of data and control commands from a master computer to one or more remote devices or outstations. The datalogger allows DNP3 communication on all available communication ports. CRBasic DNP3 instructions include:

- `DNP()`
- `DNPUpdate()`
- `DNPVariable()`

See the CRBasic help for detailed information and program examples.

For additional information on DNP3 see:

- [DNP3 with Campbell Scientific Dataloggers](#)
- [Getting to Know DNP3](#)
- [How to Access Your Measurement Data Using DNP3](#)

PakBus Communications

PakBus is a protocol similar in concept to IP (Internet Protocol). By using signed data packets, PakBus increases the number of communication and networking options available to the datalogger. The datalogger allows PakBus communication on all available communication ports. For additional information, see [The Many Possibilities of PakBus Networking](#).

Advantages of PakBus are as follows:

- Simultaneous communication between the datalogger and other devices.
- Peer-to-peer communication - no computer required. Special CRBasic instructions simplify transferring data between dataloggers for distributed decision making or control.
- Data consolidation - other PakBus dataloggers can be used as "sensors" to consolidate all data into one datalogger.
- Routing - the datalogger can act as a router, passing on messages intended for another Campbell Scientific datalogger. PakBus supports automatic route detection and selection.
- Short distance networks - a datalogger can talk to another datalogger over distances up to 30 feet by connecting transmit, receive, and ground wires between the dataloggers.

In a PakBus network, each datalogger is set to a unique address. The default PakBus address in most devices is 1. To communicate with the datalogger, the datalogger support software must know the datalogger PakBus address. The PakBus address is changed using Device Configuration Utility, datalogger **Status** table, Settings Editor, or PakBusGraph software.

CRBasic PakBus instructions include:

- `GetDataRecord()`
- `GetVariables()`
- `SendData()`
- `SendGetVariables()`
- `SendVariables()`

SDI-12 Communication

SDI-12 is a 1200 baud protocol that supports many smart sensors. The datalogger supports SDI-12 communication through two modes – transparent mode and programmed mode (see "SDI-12 Port" on page 8 for wiring terminal information).

- Transparent mode facilitates sensor setup and troubleshooting. It allows commands to be manually issued and the full sensor response viewed. Transparent mode does not record data. See "SDI-12 Transparent Mode" on page 47 for more information.
- Programmed mode automates much of the SDI-12 protocol and provides for data recording. See "SDI-12 Programmed Mode/Recorder Mode" on page 48 for more information.

CRBasic SDI-12 instructions include:

- `SDI12Recorder()`
- `SDI12SensorSetup()`
- `SDI12SensorResponse()`


SDI-12 TRANSPARENT MODE

System operators can manually interrogate and enter settings in probes using transparent mode. Transparent mode is useful in troubleshooting SDI-12 systems because it allows direct communication with probes.

Transparent mode may need to wait for commands issued by the programmed mode to finish before sending responses. While in transparent mode, the datalogger programs may not execute. Datalogger security may need to be unlocked before transparent mode can be activated.

Transparent mode is entered while the computer is in communication with the datalogger through a terminal emulator program. It is easily accessed through a terminal emulator available with Device Configuration Utility and other datalogger support software. Keyboard displays cannot be used. For how-to instructions for communicating directly with an SDI-sensor using a terminal emulator, watch this [video](#).

To enter the SDI-12 transparent mode, enter the datalogger support software terminal emulator:



```
Deployment | Logger Control | Data Monitor | File Control | Send OS | Settings Editor | Terminal
CR1000>
CR1000>SDI12
Enter Cx Port 1,2,3 or 7
1
Entering SDI12 Terminal
←
Exit SDI12 Terminal
```

1. Press **Enter** until the datalogger responds with the prompt `CR1000X>`.
2. Type `SDI12` at the prompt and press **Enter**.
3. In response, the query `Enter Cx Port` is presented with a list of available ports. Enter the port number assigned to the terminal to which the SDI-12 sensor is connected, and press **Enter**. For example, 1 is entered for terminal **C1**.
4. An `Entering SDI12 Terminal` response indicates that SDI-12 transparent mode is active and ready to transmit SDI-12 commands and display responses.

The terminal-mode utility allows monitoring of SDI-12 traffic by using the watch command (sniffer mode). Watch an instructional this [video](#). or use the following instructions.

1. Enter the terminal mode as described previously.
2. Press **Enter** until a `CR1000X>` prompt appears.
3. Type `w` and then press **Enter**.
4. In response, the query `Select:` is presented with a list of available ports. Enter the port number assigned to the terminal to which the SDI-12 sensor is connected, and press **Enter**.
5. In answer to `Enter timeout (secs):` type `100` and press **Enter**.
6. In response to the query `ASCII (Y)?`, type `Y` and press **Enter**.
7. SDI-12 communications are then opened for viewing.

SDI-12 Transparent Mode Commands

SDI-12 commands and responses are defined by the SDI-12 Support Group (www.sdi-12.org) and are available in the [SDI-12 Specification](#). Sensor manufacturers determine which commands to support. Commands have three components:

- Sensor address (**a**): A single character and the first character of the command. Sensors are usually assigned a default address of zero by the manufacturer. The wildcard address (?) is used in the `Address Query` command. Some manufacturers may allow it to be used in other commands.
- Command body (for example, **M1**): An upper case letter (the “command”) followed by alphanumeric qualifiers.
- Command termination (**!**): An exclamation mark.

An active sensor responds to each command. Responses have several standard forms and terminate with `<CR><LF>` (carriage return-line feed).

SDI-12 PROGRAMMED MODE/RECORDER MODE

The datalogger can be programmed to act as an SDI-12 data recorder or as an SDI-12 sensor using a terminal configured for SDI-12. The `SDI12Recorder()` instruction automates sending commands and recording responses. With this instruction, the commands to poll sensors and retrieve data are done automatically with proper elapsed time between the two. The datalogger automatically issues retries. See CRBasic Editor help for more information on this instruction.

Commands entered into the `SDIRecorder()` instruction differ slightly in function from similar commands entered in transparent mode. In transparent mode, for example, the operator manually enters `aM!` and `aD0!` to initiate a measurement and get data, with the operator providing the proper time delay between the request for measurement and the request for data. In programmed mode, the datalogger provides command and timing services within a single line of code. For example, when the `SDI12Recorder()` instruction is programmed with the `M!` command (note that the SDI-12 address is a separate instruction parameter), the datalogger issues the `aM!` and `aD0!` commands with proper elapsed time between the two. The datalogger automatically issues retries and performs other services that make the SDI-12 measurement work as trouble free as possible.

For troubleshooting purposes, responses to SDI-12 commands can be captured in programmed mode by placing a variable declared `As String` in the variable parameter. Variables not declared `As String` will capture only numeric data.

PROGRAMMING THE DATALOGGER TO ACT AS AN SDI-12 SENSOR

The `SDI12SensorSetup()` / `SDI12SensorResponse()` instruction pair programs the datalogger to behave as an SDI-12 sensor. A common use of this feature is the transfer of data from the datalogger to other Campbell Scientific dataloggers over a single-wire interface (terminal configured for SDI-12 to terminal configured for SDI-12), or to transfer data to a third-party SDI-12 recorder.

Details of using the `SDI12SensorSetup()` / `SDI12SensorResponse()` instruction pair can be found in the CRBasic Editor help.

When programmed as an SDI-12 sensor, the datalogger will respond to SDI-12 commands `M`, `MC`, `C`, `CC`, `R`, `RC`, `V`, `?`, and `I`. The following rules apply:

A datalogger can be assigned only one SDI-12 address per SDI-12 port. For example, a datalogger will not respond to both `0M!` AND `1M!` on SDI-12 port `C1`. However, different SDI-12 ports can have unique SDI-12 addresses. Use a separate `sLowSequence` for each SDI-12 port configured as a sensor.

SDI-12 POWER CONSIDERATIONS

When a command is sent by the datalogger to an SDI-12 probe, all probes on the same SDI-12 port will wake up. However, only the probe addressed by the datalogger will respond. All other probes will remain active until the timeout period expires.

Example:

Probe: Water Content

Power Usage:

- Quiescent: 0.25 mA
- Active: 66 mA
- Measurement: 120 mA

Measurement time: 15 s

Timeout: 15 s

Probes 1, 2, 3, and 4 are connected to SDI-12 port C1.

The time line in the following table shows a 35-second power-usage profile example.

For most applications, total power usage of 318 mA for 15 seconds is not excessive, but if 16 probes were wired to the same SDI-12 port, the resulting power draw would be excessive. Spreading sensors over several SDI-12 terminals helps reduce power consumption.

Example Power Usage Profile for a Network of SDI-12 Probes

Time into Measurement Processes	Command	All Probes Awake	Time Out Expires	Probe 1 (mA)	Probe 2 (mA)	Probe 3 (mA)	Probe 4 (mA)	Total (mA)
Sleep				0.25	0.25	0.25	0.25	1
1	1M!	Yes		120	66	66	66	318
2-14				120	66	66	66	318
15			Yes	120	66	66	66	318
16	1D0!	Yes		66	66	66	66	264
17-29				66	66	66	66	264
30			Yes	66	66	66	66	264
Sleep				0.25	0.25	0.25	0.25	1

Serial Peripheral Interface (SPI) and I2C

Serial Peripheral Interface is a clocked synchronous interface, used for short distance communications, generally between embedded devices. I2C is a multi-master, multi-slave, packet switched, single-ended, serial computer bus. I2C is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication. I2C and SPI are protocols supported by the operating system. See CRBasic Editor help for instructions that support these protocols.

For additional information on I2C, see www.i2c-bus.org.

Maintaining Your Datalogger

Protect the datalogger from humidity and moisture. When humidity levels reach the dewpoint, condensation occurs, and damage to datalogger electronics can result. Adequate desiccant should be placed in the instrumentation enclosure to provide protection and control humidity, and it should be changed periodically.

If sending the datalogger to Campbell Scientific for calibration or repair, consult first with Campbell Scientific. If the datalogger is malfunctioning, be prepared to perform some troubleshooting procedures (see "Troubleshooting the Datalogger" on page 59). Many problems can be resolved with a telephone conversation. If calibration or repair is needed, the procedure shown here should be followed when sending the product.

Maintenance for your datalogger components include:

Datalogger Calibration	50
Datalogger Security	51
Datalogger Enclosures.....	51
Internal Battery	54
Electrostatic Discharge and Lightning Protection	56
Power Budgeting	57
Updating the Operating System	57

Datalogger Calibration

Campbell Scientific recommends factory recalibration every three years. During calibration, all the input terminals, peripheral and communications terminals, operating system, and Flash EEPROM are checked; and the internal battery is replaced. The datalogger is checked to ensure that all hardware operates within published specifications before it is returned. For calibration information, see www.campbellsci.com/repair.

It is recommended that you maintain a level of calibration appropriate to the application of the datalogger. Consider the following factors when setting a calibration schedule:

- The importance of the measurements.
- How long the datalogger will be used.
- The operating environment.
- How the datalogger will be handled.

See also "About Background Calibration" on page 50.

ABOUT BACKGROUND CALIBRATION

The datalogger uses an internal voltage reference to routinely self-calibrate to compensate for changes caused by changing operating temperatures and aging. Background calibration calibrates only the coefficients necessary to run the current CRBasic program. These coefficients are reported in the **Status** table as `CalVolts()`, `CalGain()`, `CalOffset()`, and `CalCurrent()`.

Auto background calibration will be disabled automatically when the scan rate is too high for the background calibration measurements to occur in addition to the measurements contained within the program. The `Calibrate()` instruction can be used to override or disable background calibration. Disable this background calibration when it interferes with execution of very fast programs and less accuracy can be tolerated. With background calibration disabled, measurement accuracy over the

operational temperature range is specified as less accurate by a factor of 10. That is, over the extended temperature range of -55 °C to 85 °C, the accuracy specification of ±0.08 % of reading can degrade to ±0.8 % of reading with background calibration disabled. If the temperature of the datalogger remains the same, there is little calibration drift if background calibration is disabled.

Datalogger Security

Datalogger security concerns include:

- Collection of sensitive data
- Operation of critical systems
- Networks accessible by many individuals

Some options to secure your datalogger from mistakes or tampering include:

- Sending the latest operating system to the datalogger. See "Updating the Operating System" on page 57 for more information.
- Disabling unused services and securing those that are used. This includes disabling Ping to prevent discovery of a datalogger on your network.
- Setting security codes (see following information under "Security Codes").
- Setting a PakBus/TCP password. The PakBus TCP password controls access to PakBus communication over a TCP/IP link. PakBusTCP passwords can be set in Device Configuration Utility. The following CRBasic instructions also offer provisions for PakBus password protection:
 - o `ModemCallBack()`
 - o `SendVariables()`
 - o `SendGetVariables()`
 - o `SendFile()`
 - o `GetVariables()`
 - o `GetFile()`
 - o `GetDataRecord()`
- Setting an FTP username and password in Device Configuration Utility.
- Setting an AES-128 PakBus encryption key in Device Configuration Utility. This enables encrypted PakBus data transmission.
- Creating a `.csipasswd` file for securing HTTP/HTTPS and for providing web interface permissions (see "Creating a `.csipasswd` File" on page 52 for more information).
- Enabling HTTPS and disabling HTTP.
- Tracking signatures.
- Encrypting program files if they contain sensitive information (see CRBasic help `Include()` instruction or use the CRBasic Editor **File** menu, **Save and Encrypt** option).
- Hiding program files for extra protection (see CRBasic help `FileManage()` instruction).
- Securing the physical datalogger and power supply under lock and key.
- Monitoring your datalogger for changes by tracking program and operating system signatures, as well as CPU, USR, and CRD file contents.

Warning: All security features can be subverted through physical access to the datalogger. If absolute security is a requirement, the physical datalogger must be kept in a secure location.

SECURITY CODES

The datalogger employs a security scheme that includes three levels of security. Security codes can effectively lock out innocent tinkering and discourage wannabe hackers on non-IP based communication links. However, any serious hacker with physical access to the datalogger or to the communications hardware can, with only minimal trouble, overcome the five-digit security codes. Systems adequately secured with security codes are probably limited to the following:

- Private, non-IP radio networks.
- Direct links (hardwire RS-232, short haul, multidrop, fiber optic).
- Non-IP satellite.
- Landline, non-IP based telephone, where the telephone number is not published.
- Cellular phone wherein IP has been disabled, providing a strictly serial connection.

Methods of enabling security codes include the following:

- **Settings:** **Security(1)**, **Security(2)**, and **Security(3)** are writable variables in the **Status** table.
- Device Configuration Utility: Security codes are set on the **Deployment** tab.
- **SetSecurity()** instruction in CRBasic: **SetSecurity()** is only executed at program compile time.

Note: Deleting **SetSecurity()** from a CRBasic program is not equivalent to **SetSecurity(0,0,0)**. Settings persist when a new program is downloaded that has no **SetSecurity()** instruction.

Up to three levels of security codes can be set. Valid security codes are 1 through 65535 (0 confers no security). **Security 1** must be set before **Security 2**. **Security 2** must be set before **Security 3**. If any one of the codes is set to 0, any security code level greater than it will be set to 0. For example, if **Security 2** is 0 then **Security 3** is automatically set to 0. Security codes are unlocked in reverse order: **Security 3** before **Security 2**, **Security 2** before **Security 1**.

Functions affected by each level of security are:

- **Security 1:** Collecting data, setting the clock, and setting variables in the **Public** table are unrestricted, requiring no security code. If **Security 1** code is entered, read/write values in the **Status**, **Settings**, and **DataTableInfo** tables can be changed; fields can be edited in the **ConstTable**; and the datalogger program can be changed or retrieved.
- **Security 2:** Data collection is unrestricted, requiring no security code. If the user enters the **Security 2** code, the datalogger clock can be changed, values in the **Public** table can be changed, and read/write **DataTable Sample** fields can be edited.
- **Security 3:** When this level is set, all communication with the datalogger is prohibited if no security code is entered. If **Security 3** code is entered, data can be viewed and collected from the datalogger (except data suppressed by the **TableHide()** instruction).

CREATING A .CSIPASSWD FILE

The datalogger employs a security scheme that includes three levels of security (see "Datalogger Security" on page 51 for more information). This scheme can be used to limit access to a datalogger that is publicly available. However, the security code is visible in Device Configuration Utility and in the program. In addition, the range of codes is relatively small. To provide a more robust means of security, Basic Access Authentication was implemented with the HTTP API interface in the form of an encrypted password file named `.csipasswd`. In order to provide read/write access to the web interface, you must create a `.csipasswd` file.

When a file named `.csipasswd` is stored on the datalogger's CPU drive, basic access authentication is enabled in the datalogger and read/write access to the web interface can be defined. Four levels of access are available: all access denied (0), all access allowed (1), set variables allowed (2), and read-only access (3). Multiple user accounts/levels of access can be defined for one datalogger.

Create an encrypted password file or modify an existing password file using Device Configuration Utility:

1. Connect to your device in Device Configuration Utility.
2. Click the **Network Services** tab, then click the **Edit .csipasswd File** button.
3. Define your user accounts and access levels.
4. Click **Apply**. The `.csipasswd` file is automatically saved to the datalogger's CPU drive.

When a `.csipasswd` file enables basic access authentication, the datalogger's PakBus/TCP Password security setting is not used when accessing the datalogger via HTTP. If the `.csipasswd` file is blank or does not exist, the default user name is "anonymous" with no password and a user level of read-only.

When access to the datalogger web server is attempted without the appropriate security level, the datalogger will return a 401 Authorization Required response, which will prompt the web client to display a user name/password request dialog. If an invalid username or password is entered, the datalogger web server will default to the level of access assigned to "anonymous". As noted above, anonymous is assigned a user level of read-only, though this can be changed using Device Configuration Utility.

If the numeric security code has been enabled using Device Configuration Utility, the `SetSecurity()`, instruction, or the **Security() Status** table settings, and no `.csipasswd` file is on the datalogger, then that numeric security code must be entered to access the datalogger. If a `.csipasswd` file is on the datalogger, the User Name and Password employed by the Basic Access Authentication will eliminate the need for entering the numeric security code.

Command Syntax

Syntax for the commands sent to the web server generally follows the form of:

```
URL?command=CommandName&uri=DataSource&arguments
```

Arguments are appended to the command string using an ampersand (&). Some commands have optional arguments, where omitting the argument results in a default being used. When applicable, optional arguments and their defaults are noted and examples are provided in the CRBasic help (search Web Server/API Commands).

Datalogger Enclosures

The datalogger and most of its peripherals must be protected from moisture and humidity. Moisture in the electronics will seriously damage the datalogger. In most cases, protection from moisture is easily accomplished by placing the datalogger in a weather-tight enclosure with desiccant and elevating the enclosure above the ground. Desiccant in enclosures should be changed periodically.

Warning: Do not completely seal the enclosure if lead-acid batteries are present; hydrogen gas generated by the batteries may build up to an explosive concentration.

The following details a typical installation using a Campbell Scientific enclosure. The datalogger has mounting holes through which small screws are inserted into nylon anchors in the backplate.

1. Insert the included nylon anchors into the backplate. Position them to align with the mounting holes on the base of the CR1000X.
2. Holding the CR1000X to the backplate, screw the screws into the nylon anchors.

Internal Battery

The lithium battery powers the internal clock and SRAM when the datalogger is not powered. This voltage is displayed in the LithiumBattery field in the **Status** table. Replace the battery when voltage is approximately 2.7 Vdc. The internal lithium battery has a three-year life when no external power source is applied. Its life is extended when the datalogger is installed with an external power source. If the datalogger is used in a high-temperature application, the battery life is shortened.

To prevent clock and memory issues, it is recommended you proactively replace the battery every 2-3 years, or more frequently when operating continuously in high temperatures.

Note: The battery is replaced during regular factory recalibration, which is recommended every 3 years. For more information, see "Datalogger Calibration" on page 50.

When the lithium battery is removed (or is depleted and primary power to the datalogger is removed), the CRBasic program and most settings are maintained, but the following are lost:

- Run-now and run-on power-up settings.
- Routing and communication logs (relearned without user intervention).
- Time. Clock will need resetting when the battery is replaced.
- Final-memory data tables.

A replacement lithium battery can be purchased from Campbell Scientific or another supplier. See "Power Requirements" on page 77 for more information.

Warning: Misuse or improper installation of the internal lithium battery can cause severe injury. Fire, explosion, and severe burns can result. Do not recharge, disassemble, heat above 100 °C (212 °F), solder directly to the cell, incinerate, or expose contents to water. Dispose of spent lithium batteries properly.

INTERNAL LITHIUM BATTERY SPECIFICATIONS

- Manufacturer: Tadiran
- Tadiran Model Number: TL-5903/S
- Voltage: 3.6 V
- Capacity: 2.4 Ah
- Self-discharge Rate: 1%/year @ 20 °C
- Operating Temperature Range: -55 to 85 °

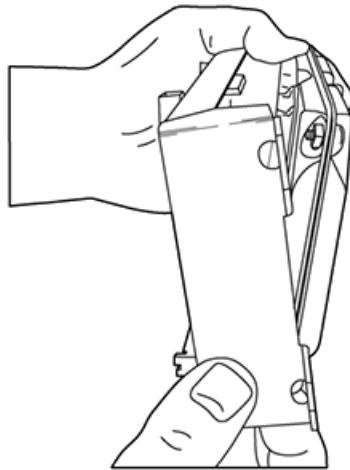
REPLACING THE INTERNAL BATTERY

It is recommended that you send the datalogger in for scheduled calibration, which includes internal battery replacement (see "Datalogger Calibration" on page 50).

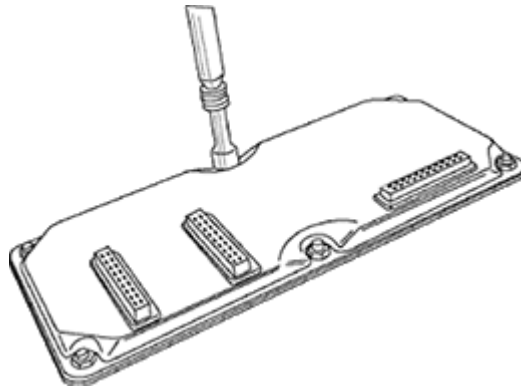
Warning: Any damage made to the datalogger during replacement of the internal battery is not covered under warranty.

Remove the two Phillips screws from the back of the panel.

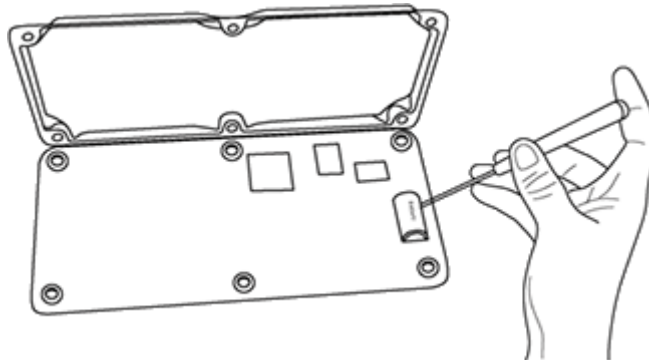
2. Pull one edge of the canister away from the wiring panel to loosen it from the internal connector seatings.



3. Lift the canister edge out of the enclosure tabs.
4. Remove the nuts, then open the clam shell.



5. Remove the lithium battery by gently prying it out with a small slot head screwdriver.

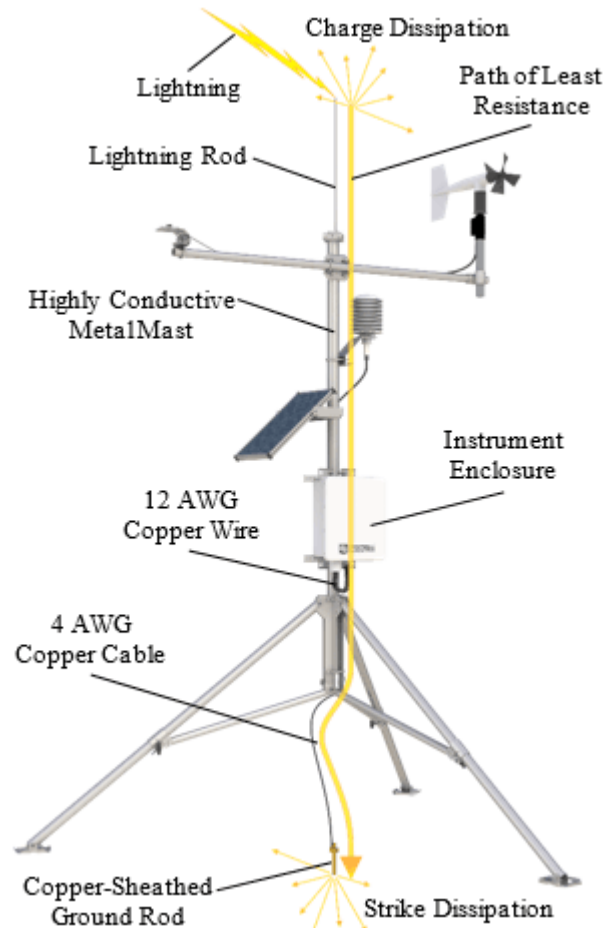


6. Reassemble the datalogger. Take particular care to ensure the canister is reseated tightly into the connectors by firmly pressing them together by hand.

Electrostatic Discharge and Lightning Protection

Warning: Lightning strikes may damage or destroy the datalogger and associated sensors and power supplies.

Electrostatic discharge (ESD) can originate from several sources, the most common and destructive being primary and secondary lightning strikes. Primary lightning strikes hit instrumentation directly. Secondary strikes induce voltage in power lines or wires connected to instrumentation. While elaborate, expensive, and nearly infallible lightning protection systems are on the market, Campbell Scientific, for many years, has employed a simple and inexpensive design that protects most systems in most circumstances. The system employs a lightning rod, metal mast, heavy-gauge ground wire, and ground rod to direct damaging current away from the datalogger. This system, however, is not infallible. The following image displays a typical application of the system:



The primary devices for protection against ESD are gas-discharge tubes. All critical inputs and outputs on the datalogger are protected with gas-discharge tubes or transient voltage suppression diodes. Gas-discharge tubes fire at 75 V to allow current to be diverted to the earth ground lug. To be effective, the earth ground lug must be properly connected to earth (chassis) ground.

Communication ports are another path for transients. You should provide communication paths, such as telephone or short-haul modem lines, with spark-gap protection. Spark-gap protection is usually an option with these products, so request it when ordering. Spark gaps must be connected to either the earth ground lug, the enclosure ground, or to the earth (chassis) ground.

For detailed information on grounding, see "Grounds" on page 7.

Power Budgeting

In low-power situations, the datalogger can operate for several months on non-rechargeable batteries. Power systems for longer-term remote applications typically consist of a charging source, a charge controller, and a rechargeable battery. When ac line power is available, a Vac-to-Vdc wall adapter, the onboard charging regulator, and a rechargeable battery can be used to construct an uninterruptible power supply (UPS).

When designing a power supply, consider worst-case power requirements and environmental extremes. For example, the power requirement of a weather station may be substantially higher during extreme cold, while at the same time, the extreme cold constricts the power available from the power supply. System operating time for batteries can be determined by dividing the battery capacity (ampere hours) by the average system current drain (amperes).

For more information see:

- [Application Note - Power Supplies](#)
- [Video Tutorial - Power Budgeting](#)

See also:

- "Power Input" on page 5
- "Power Output" on page 6
- "Power Requirements" on page 77
- "Power Output Specifications" on page 78

Updating the Operating System

Campbell Scientific posts operating system (OS) updates at www.campbellsci.com/downloads when they become available. Before deploying instruments, check operating system versions and update as needed. The datalogger operating system version is shown in the **Status** table, **Station Status Summary**, and Device Configuration Utility **Deployment | Datalogger**. An operating system may be sent through Device Configuration Utility or through program-send procedures.

Sending an operating system through Device Configuration Utility resets all settings on the datalogger to factory defaults and clears its memory. To send an OS through Device Configuration Utility, follow the procedure described on the **Send OS** tab.

Watch a video: [Sending an OS to a \(Local\) Datalogger](#).

In most instances, sending an operating system as a program preserves settings. This allows for sending supported operating systems remotely (check the release notes). However, this should be done with great caution as updating the OS may reset the datalogger settings, even settings critical to supporting the telecommunication link. To send an OS this way, use the **Send New** button in the datalogger support software, as instructed in the topic "Sending a Program to the Datalogger" on page 18.

Note the following precautions when sending as a program:

- Because sending an OS resets datalogger memory, data loss will certainly occur. Depending on several factors, the datalogger may also become incapacitated for a time.
- While the programs are retained in memory, no program will be set to run after the operating system is sent. To overcome this, create a "default.CR1" program and store it on the datalogger. Default.CR1 can be edited to preserve critical datalogger settings, such as communication settings, but cannot be more than a few lines of code.

- Any peripherals being powered through the **SW12** terminals will be turned off until the program logic turns them on again.
- Operating systems are very large files. Be cautious of data charges. Sending over a direct serial or USB connection is recommended, when possible.

Tips and Troubleshooting

The following basic procedure can be used if a system is not operating properly.


1. Using a voltmeter, check the voltage of the primary power source at the **POWER IN** terminals on the face of the datalogger.
2. Check wires and cables for the following:
 - Incorrect wiring connections. Make sure each sensor and device are wired to the channels assigned in the program. If the program was written in Short Cut, check wiring against the generated wiring diagram. If written in CRBasic Editor, check wiring against each measurement and control instruction.
 - Loose connection points.
 - Faulty connectors.
 - Cut wires.
 - Damaged insulation, which allows water to migrate into the cable. Water, whether or not it comes in contact with wire, can cause system failure. Water may increase the dielectric constant of the cable sufficiently to impede sensor signals, or it may migrate into the sensor, which will damage sensor electronics.
3. Check the CRBasic program. If the program was written solely with Short Cut, the program is probably not the source of the problem. If the program was written or edited with CRBasic Editor, logic and syntax errors could easily have crept into the code. To troubleshoot, create a stripped-down version of the program, or break it up into multiple smaller units to test individually. For example, if a sensor signal-to-data conversion is faulty, create a program that only measures that sensor and stores the data, absent from all other inputs and data.
4. Reset the datalogger. Sometimes the easiest way to resolve a problem is by resetting the datalogger (see "Resetting the Datalogger" on page 63 for more information).

Checking Station Status

View the condition of the datalogger using **Station Status**. Here you can see the operating system version of the datalogger, the name of the current program, program compile results, and other key indicators. Items that may need your attention appear in **red** or **blue**. The following information describes the significance of some entries in the station status window. Watch a [video](#) or use the following instructions.

VIEWING STATION STATUS

Using your datalogger support software, access the **Station Status** to view the condition of the datalogger.

- From LoggerNet: Click **Connect** () , then click the **Station Status** button to view the **Summary** tab.
- From PC200W and PC400: Click the **Datalogger** menu and select **Station Status** to view the **Summary** tab.

WATCHDOG ERRORS

Watchdog errors indicate that the datalogger has crashed and reset itself. This could be due to:

- Transient voltage.
- Miswired or malfunctioning sensor.
- Poor ground connection on the power supply.
- Running the CRBasic program very fast.
- Many `PortSet()` instructions back-to-back with no delay.
- High-speed serial data on multiple ports with very large data packets or bursts of data.

If any of these are not the apparent cause, contact Campbell Scientific for assistance (see <https://www.campbellsci.com/repair>). Causes that may require assistance include:

- Memory corruption
- Operating System problem
- Hardware problem

RESULTS FOR LAST PROGRAM COMPILED

Messages generated by the datalogger at program upload and as the program runs are reported here. Warnings indicate that an expected feature may not work, but the program will still operate. Errors indicate that the program cannot run. For more information, see "CRBasic Program Errors" on page 61.

SKIPPED SCANS

Skipped scans are caused when a program takes longer to process than the scan rate allows. If any scan skips repeatedly, the datalogger program may need to be optimized or reduced. For more information, see: [How to Prevent Skipped Scans and a Sunburn](#).

SKIPPED RECORDS

Skipped records usually occur because a scan is skipped. They indicate that a record was not stored to the data table when it should have been.

VARIABLE OUT OF BOUNDS

Variable-out-of-bounds errors happen when an array is not sized to the demands of the program. The datalogger attempts to catch all out-of-bounds errors at compile time and document the error. However, it is not always possible, so these errors may occur during runtime.

BATTERY VOLTAGE

If powering through USB, reported battery voltage should be 0 V. If connecting to an external power source, battery voltage should be reported at or near 12 V. See also:

- "Power Input" on page 5.
- "Power Requirements" on page 77.

Understanding NAN and INF Occurrences

NAN (not a number) and INF (infinite) are data words indicating an exceptional occurrence in datalogger function or processing. **INF** indicates that the program has encountered an arithmetic

expression that is undefined. **NAN** indicates an invalid measurement. For more information, see [Tips and Tricks: Who's NAN?](#)

NANs are expected in the following conditions:

- Input signals exceed the voltage range chosen for the measurement.
- An invalid SDI-12 command is sent
- An SDI-12 sensor does not respond or aborts without sending data
- Undefined arithmetic expressions, such as $0 \div 0$.

NAN is a constant that can be used in expressions. This is shown in the following code snip that sets a CRBasic control feature (a flag) if the wind direction is **NAN**:

```
If WindDir = NAN Then
  WDFlag = False
Else
  WDFlag = True
EndIf
```

If a **NAN** is included in the values being processed, **NAN** will be stored. Note that since there is no such thing as **NAN** for integers, values that are converted from float to integer will be expressed in data tables as the most negative number for a given data type. For example, the most negative number of data type FP2 is -7999, so **NAN** for FP2 data will appear in a data table as -7999. If the data type is Long, **NAN** will appear in the data table as -2,147,483,648.

Because **NAN** is a constant, it can be used in conjunction with the disable variable parameter (*DisableVar*) in output processing instructions. Use the *DisableVar* parameter to call the output table conditionally (for example, do not call the table if a variable = **NAN**) to keep **NANs** from affecting the other good values.

Time Keeping

Measurement of time is an essential function of the datalogger. Time measurement with the onboard clock enables the datalogger to attach time stamps to data, measure the interval between events, and time the initiation of control functions. Details on clock accuracy and resolution are available in the "System Specifications" on page 77. An internal lithium battery backs the clock when the datalogger is not externally powered (see "Internal Battery" on page 54 for more information).

CLOCK BEST PRACTICES

If you are going to set the clock with LoggerNet, initiate it manually during a maintenance period when you are already disrupting site data.

If you are going to use automated clock check with LoggerNet, it is recommended that you do this on the order of days (not hours). Set an allowed clock deviation that is appropriate for the expected jitter in the network, and use the initial time setting to offset the clock check away from storage and measurement intervals.

Take into account the amount of time that is required for the **CLockCheck** command to reach the datalogger, be processed, and for it to send its response (round-trip time, or time-of-flight). LoggerNet maintains a history of the round trip times for up to the ten previous successful clock check transactions in order to calculate an estimate of this time of flight. It adds this average to the time values received from the datalogger and subtracts it from any adjustment that it might make.

The `ClockChange` instruction can be used to:

- Detect, track, and log externally triggered clock change events.
- Characterize the jitter in your clock-syncing method.

TIME STAMPS

A measurement without an accurate time reference has little meaning. Data on the datalogger are stored with time stamps. How closely a time stamp corresponds to the actual time a measurement is taken depends on several factors.

The time stamp in common CRBasic programs matches the time at the beginning of the current scan as measured by the real-time clock in the datalogger. If a scan starts at 15:00:00, data output during that scan will have a time stamp of **15:00:00** regardless of the length of the scan or when in the scan a measurement is made. The possibility exists that a scan will run for some time before a measurement is made. For instance, a scan may start at 15:00:00, execute time-consuming code, then make a measurement at 15:00:00.51. The time stamp attached to the measurement, if the `CallTable()` instruction is called from within the `Scan()` / `NextScan` construct, will be **15:00:00**, resulting in a time-stamp skew of 510 ms.

AVOIDING TIME SKEW

Time skew between consecutive measurements is a function of settling and integration times, ADC, and the number entered into the `Reps` parameter of CRBasic instructions.

Time-stamp skew is not a problem with most applications because:

- Program execution times are usually short, so time stamp skew is only a few milliseconds. Most measurement requirements allow for a few milliseconds of skew.
- Data processed into averages, maxima, minima, and so forth are composites of several measurements. Associated time stamps only reflect the time the last measurement was made and processing calculations were completed, so the significance of the exact time a specific sample was measured diminishes.

Applications measuring and storing sample data wherein exact time stamps are required can be adversely affected by time-stamp skew. Skew can be avoided by:

- Making measurements in the scan before time-consuming code.
- Programming the datalogger such that the time stamp reflects the system time rather than the scan time using the `DataTime()` instruction. See topics concerning data table declarations in CRBasic Editor help for more information.

CRBasic Program Errors

Analyze data soon after deployment to ensure the datalogger is measuring and storing data as intended. Most measurement and data-storage problems are a result of one or more CRBasic program bugs. Watch a video: [CRBasic | Common Errors - Identifying and fixing common errors in the CRBasic programming language](#).

PROGRAM DOES NOT COMPILE

Although the CRBasic Editor compiler states that a program compiles OK, the program may not run or even compile in the datalogger. This is rare, but reasons may include:

- The datalogger has a different operating system that is not fully compatible with the computer compiler. Check the two versions if in doubt. The computer compiler version is shown on the first line of the compile results. Update the computer compiler by first downloading the executable OS file from www.campbellsci.com. When run, the executable file updates the computer compiler. To update the datalogger operating system, see "Updating the Operating System" on page 57.
- The program has large memory requirements for data tables or variables and the datalogger does not have adequate memory. This normally is flagged at compile time in the compile results. If this type of error occurs, check:
 - o Copies of old programs on the CPU drive. The datalogger keeps copies of all program files unless they are deleted, the drive is formatted, or a new operating system is loaded with Device Configuration Utility.
 - o The USR drive size, is it too large? The USR drive may be using memory needed for the program.
 - o Ensure the memory card is available when a program is attempting to access the CRD drive.

PROGRAM COMPILES BUT DOES NOT RUN CORRECTLY

If the program compiles but does not run correctly, timing discrepancies are often the cause. If a program is tight on time, look further at the execution times. Check the measurement and processing times in the **Status** table (**MeasureTime**, **ProcessTime**, **MaxProcTime**) for all scans, then try experimenting with the **InstructionTimes ()** instruction in the program. Analyzing **InstructionTimes ()** results can be difficult due to the multitasking nature of the logger, but it can be a useful tool for fine-tuning a program.

Resetting the Datalogger

A reset is referred to as a "memory reset." Be sure to back up the current datalogger configuration before a reset in case you need to revert to the old settings. To back up the datalogger configuration, connect to the datalogger using Device Configuration Utility, and click **Backup | Back Up Datalogger**. To restore a configuration after the datalogger has been reset, connect and click **Backup | Restore Datalogger**.

The following features are available for complete or selective reset of datalogger memory:

- Processor reset.
- Program send reset.
- Manual data table reset.
- Formatting memory drives.
- Full memory reset.

PROCESSOR RESET

To reset the processor, simply power cycle the datalogger. This resets its short-term memory, restarts the current program, sets variables to their starting values, and clears communication buffers. This does not clear data tables but may result in a skipped record. If the datalogger is remote, a power cycle can be mimicked in the **Terminal Emulator** program (type REBOOT <Enter>).

PROGRAM SEND RESET

Final-storage data are erased when user programs are uploaded, unless preserve / erase data options are used and the program was not altered. Preserve / erase data options are presented when sending programs using File Control **Send** command and CRBasic Editor **Compile, Save and Send**.

When a program compiles, all variables are initialized. A program is recompiled after a power failure or a manual stop. For instances that require variables to be preserved through a program recompile, the CR1000X has the `PreserveVariables()` instruction.

MANUAL DATA TABLE RESET

Data table memory is selectively reset from:

- Datalogger support software: Station Status **Table Fill Times** tab, **Reset Tables**.
- Device Configuration Utility: **Data Monitor** tab, **Reset Table** button.

FORMATTING DRIVES

CPU, USB, CRD (memory card required), and USB (module required) drives can be formatted individually. Formatting a drive erases all files on that drive. If the currently running user program is on the drive to be formatted, the program will cease running and any SRAM data associated with the program are erased. Drive formatting is performed through the datalogger support software **File Control | Format** command.

FULL MEMORY RESET

Full memory reset occurs when an operating system is sent to the datalogger using Device Configuration Utility or when entering 98765 in the **Status** table field **FullMemReset**. A full memory reset does the following:

- Clears and formats CPU drive (all program files erased).
- Clears SRAM data tables.
- Clears **Status** table elements.
- Restores settings to default.
- Initializes system variables.
- Clears communication memory.

Full memory reset does not affect the CRD drive directly. Subsequent user program uploads, however, can erase CRD. See “Updating the Operating System” on page 57 for more information.

Troubleshooting Power Supplies

Power supply systems may include batteries, charging regulators, and a primary power source such as solar panels or ac/ac or ac/dc transformers attached to mains power. All components may need to be checked if the power supply is not functioning properly. Check connections and check polarity of connections.

Base diagnostic: connect the datalogger to a new 12 V battery. (A small 12 V battery carrying a full charge would be a good thing to carry in your troubleshooting tool kit.) Ensure correct polarity of the connection. If the datalogger powers up and works, troubleshoot the datalogger power supply.

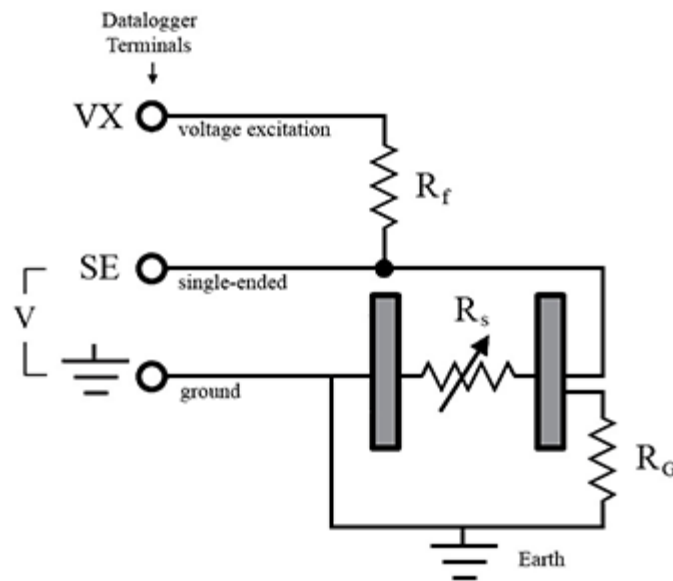
When diagnosing or adjusting power equipment supplied by Campbell Scientific, it is recommended you consider:

- Battery-voltage test.
- Charging-circuit test (when using an unregulated solar panel).
- Charging-circuit test (when using a transformer).
- Adjusting charging circuit.

If power supply components are working properly and the system has peripherals with high current drain, such as a satellite transmitter, verify that the power supply is designed to provide adequate power.

Minimizing Ground Loop Errors

When measuring soil moisture with a resistance block, or water conductivity with a resistance cell, the potential exists for a ground loop error. In the case of an ionic soil matric potential (soil moisture) sensor, a ground loop arises because soil and water provide an alternate path for the excitation to return to datalogger ground. This example is modeled in the following image:



With R_g in the resistor network, the signal measured from the sensor is described by the following equation:

$$V_1 = V_x \frac{R_s}{(R_s + R_f) + R_s R_f / R_g}$$

where

- V_x is the excitation voltage
- R_f is a fixed resistor
- R_s is the sensor resistance
- R_g is the resistance between the excited electrode and datalogger earth ground.

$R_s R_f / R_g$ is the source of error due to the ground loop. When R_g is large, the error is negligible. Note that the geometry of the electrodes has a great effect on the magnitude of this error. The Delmhorst gypsum block used in the Campbell Scientific 227 probe has two concentric cylindrical electrodes. The center

electrode is used for excitation; because it is encircled by the ground electrode, the path for a ground loop through the soil is greatly reduced. Moisture blocks which consist of two parallel plate electrodes are particularly susceptible to ground loop problems. Similar considerations apply to the geometry of the electrodes in water conductivity sensors.

The ground electrode of the conductivity or soil moisture probe and the datalogger earth ground form a galvanic cell, with the water/soil solution acting as the electrolyte. If current is allowed to flow, the resulting oxidation or reduction will soon damage the electrode, just as if dc excitation was used to make the measurement. Campbell Scientific resistive soil probes and conductivity probes are built with series capacitors to block this dc current. In addition to preventing sensor deterioration, the capacitors block any dc component from affecting the measurement.

See also "Grounds" on page 7.

Improving Voltage Measurement Quality

The following topics discuss methods of generally improving voltage measurements:

- "Deciding Between Single-Ended or Differential Measurements" on page 66
- "Minimizing Ground Potential Differences" on page 67
- "Minimizing Power-Related Artifacts" on page 68
- "Minimizing Settling Errors" on page 69
- "Detecting Open Inputs" on page 68
- "Factors Affecting Accuracy" on page 71
- "Minimizing Offset Voltages" on page 72

Read More: Consult the following technical papers at www.campbellsci.com/app-notes for in-depth treatments of several topics addressing voltage measurement quality:

- [Preventing and Attacking Measurement Noise Problems](#)
- [Benefits of Input Reversal and Excitation Reversal for Voltage Measurements](#)
- [Voltage Accuracy, Self-Calibration, and Ratiometric Measurements](#)

DECIDING BETWEEN SINGLE-ENDED OR DIFFERENTIAL MEASUREMENTS

Deciding whether a differential or single-ended measurement is appropriate is usually, by far, the most important consideration when addressing voltage measurement quality. The decision requires trade-offs of accuracy and precision, noise cancellation, measurement speed, available measurement hardware, and fiscal constraints.

In broad terms, analog voltage is best measured differentially because these measurements include noise reduction features, listed below, that are not included in single-ended measurements.

- Passive Noise Rejection
 - No voltage reference offset
 - Common-mode noise rejection, which filters capacitively coupled noise
- Active Noise Rejection
 - Input reversal
 - For more information, see "Compensating for Offset Voltage" on page 73.

Reasons for using single-ended measurements, however, include:

- Not enough differential terminals are available. Differential measurements use twice as many analog input terminals as do single-ended measurements.
- Rapid sampling is required. Single-ended measurement time is about half that of differential measurement time.
- Sensor is not designed for differential measurements. Many Campbell Scientific sensors are not designed for differential measurement, but the drawbacks of a single-ended measurement are usually mitigated by large programmed excitation and/or sensor output voltages.

Sensors with a high signal-to-noise ratio, such as a relative-humidity sensor with a full-scale output of 0 to 1000 mV, can normally be measured as single-ended without a significant reduction in accuracy or precision.

Sensors with a low signal-to-noise ratio, such as thermocouples, should normally be measured differentially. However, if the measurement to be made does not require high accuracy or precision, such as thermocouples measuring brush-fire temperatures, which can exceed 2500 °C, a single-ended measurement may be appropriate. If sensors require differential measurement, but adequate input terminals are not available, an analog multiplexer should be acquired to expand differential input capacity.

Because a single-ended measurement is referenced to datalogger ground, any difference in ground potential between the sensor and the datalogger will result in an error in the measurement. For more information on grounds, see "Grounds" on page 7 and "Minimizing Ground Potential Differences" on page 67.

MINIMIZING GROUND POTENTIAL DIFFERENCES

Low-level, single-ended voltage measurements (<200 mV) are sensitive to ground potential fluctuation due to changing return currents from **5V**, **12V**, **SW12**, and **C** terminals. The datalogger grounding scheme is designed to minimize these fluctuations by separating signal grounds (⚡) from power grounds (**G**). For more information on datalogger grounds, see "Grounds" on page 7. To take advantage of this design, observe the following rules:

- Connect grounds associated with **5V**, **12V**, **SW12**, and **C** terminals to **G** terminals.
- Connect excitation grounds to the nearest ⚡ terminal on the same terminal block.
- Connect the low side of single-ended sensors to the nearest ⚡ terminal on the same terminal block.
- Connect shield wires to the ⚡ terminal nearest the terminals to which the sensor signal wires are connected.

If offset problems occur because of shield or ground leads with large current flow, tying the problem leads into terminals next to terminals configured for excitation and pulse-count should help. Problem leads can also be tied directly to the ground lug to minimize induced single-ended offset voltages.

Ground Potential Differences

Because a single-ended measurement is referenced to datalogger ground, any difference in ground potential between the sensor and the datalogger will result in a measurement error. Differential measurements **MUST** be used when the input ground is known to be at a different ground potential from datalogger ground.

Ground potential differences are a common problem when measuring full-bridge sensors (strain gages, pressure transducers, etc), and when measuring thermocouples in soil.

- **Soil Temperature Thermocouple:** If the measuring junction of a thermocouple is not insulated when in soil or water, and the potential of earth ground is, for example, 1 mV greater at the sensor than at the point where the datalogger is grounded, the measured voltage is 1 mV greater than the thermocouple output. With a copper-constantan thermocouple, 1 mV equates to approximately 25 °C measurement error.
- **External Signal Conditioner:** External instruments with integrated signal conditioners, such as an infrared gas analyzer (IRGA), are frequently used to make measurements and send analog information to the datalogger. These instruments are often powered by the same Vac-line source as the datalogger. Despite being tied to the same ground, differences in current drain and lead resistance result in different ground potentials at the two instruments. For this reason, a differential measurement should be made on the analog output from the external signal conditioner.

For additional information, see "Minimizing Offset Voltages" on page 72.

DETECTING OPEN INPUTS

A useful option available to single-ended and differential measurements is the detection of open inputs due to a broken or disconnected sensor wire. This prevents otherwise undetectable measurement errors. Range codes appended with **C** enable open-input detection. For detailed information, see the CRBasic help (`VolTSE()` and `VolTDiff()` instructions, *Range* parameter)

The **C** option may not work, or may not work well, in the following applications:

- If the input is not a truly open circuit, such as might occur on a wet cut cable end, the open circuit may not be detected because the input capacitor discharges to a normal voltage through external leakage to ground within the settling time of the measurement. This problem is worse when a long settling time is selected, as more time is given for the input capacitors to discharge to a "normal" level.
- If the open circuit is at the end of a very long cable, the test pulse may not charge the cable (with its high capacitance) up to a voltage that generates NAN or a distinct error voltage. The cable may even act as an aerial and inject noise which also might not read as an error voltage.
- The sensor may "object" to the test pulse being connected to its output, even for 100 µs. There is little or no risk of damage, but the sensor output may be caused to temporarily oscillate. Programming a longer settling time in the CRBasic measurement instruction to allow oscillations to decay before the ADC may mitigate the problem.

MINIMIZING POWER-RELATED ARTIFACTS

Some Vac-to-Vdc power converters produce switching noise or ac ripple as an artifact of the ac-to-dc rectification process. Excessive switching noise on the output side of a power supply can increase measurement noise, and so increase measurement error. Noise from grid or mains power also may be transmitted through the transformer, or induced electromagnetically from nearby motors, heaters, or power lines.

High-quality power regulators typically reduce noise due to power regulation. Using the optional 50 Hz or 60 Hz rejection arguments for CRBasic analog input measurement instructions often improves rejection of noise sourced from power mains. The CRBasic standard deviation instruction, `StdDev()`, can be used to evaluate measurement noise.

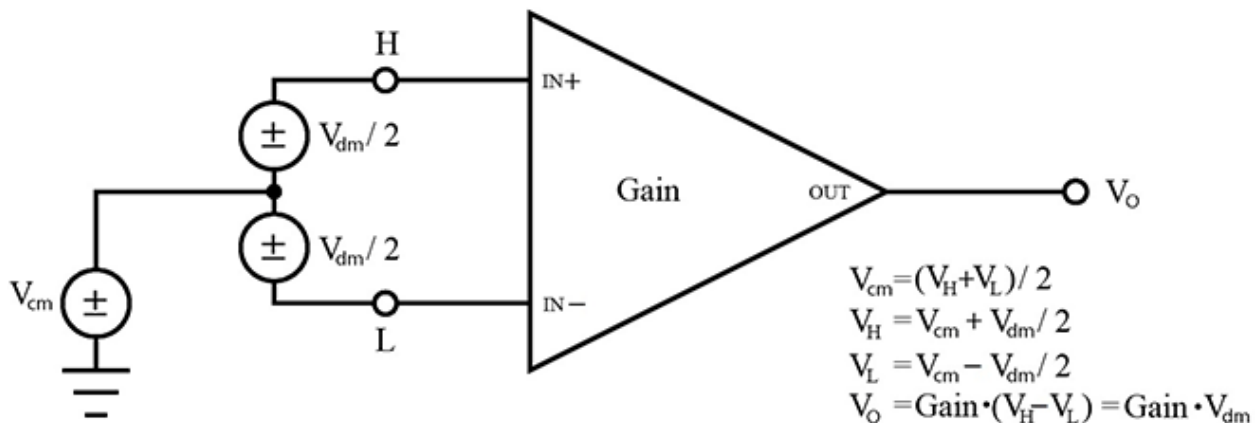
The datalogger includes **EN1** digital filtering, which serves two purposes:

- Arrive as close as possible to the true input signal
- Filter out measurement noise at specific frequencies, the most common being noise at 50 Hz or 60 Hz, which originate from mains-power lines.

Filtering time is inversely proportional to the frequency being filtered.

Minimizing Electronic Noise

Electronic noise can cause significant error in a voltage measurement, especially when measuring voltages less than 200 mV. So long as input limitations are observed, the PGIA ignores voltages, including noise, that are common to each side of a differential-input pair. This is the common-mode voltage. Ignoring (rejecting or canceling) the common-mode voltage is an essential feature of the differential input configuration that improves voltage measurements. The following image illustrates the common-mode component (V_{cm}) and the differential-mode component (V_{dm}) of a voltage signal. V_{cm} is the average of the voltages on the $V+$ and $V-$ inputs. So, $V_{cm} = (V+ + V-)/2$ or the voltage remaining on the inputs when $V_{dm} = 0$. The total voltage on the $V+$ and $V-$ inputs is given as $V_H = V_{cm} + V_{dm}/2$, and $V_L = V_{cm} - V_{dm}/2$, respectively.



MINIMIZING SETTILING ERRORS

Settling time allows an analog voltage signal to rise or fall closer to its true magnitude prior to measurement. Default settling times (those resulting when `settlingTime = 0`) provide sufficient settling in most cases. Additional settling time is often programmed when measuring high-resistance (high-impedance) sensors or when sensors connect to the input terminals by long cables. The time to complete a measurement increases with increasing settling time. For example, a 1 ms increase in settling time for a bridge instruction with input reversal and excitation reversal results in a 4 ms increase in time to perform the instruction.

When sensors require long lead lengths, use the following general practices to minimize settling errors:

- Do not use wire with PVC-insulated conductors. PVC has a high dielectric constant, which extends input settling time.
- Where possible, run excitation leads and signal leads in separate shields to minimize transients.
- When measurement speed is not a prime consideration, additional time can be used to ensure ample settling time.
- In difficult cases where measurement speed is a consideration, an appropriate settling time can be determined through testing.

Measuring Settling Time

Settling time for a particular sensor and cable can be measured with the CR1000X. Programming a series of measurements with increasing settling times will yield data that indicate at what settling time a further increase results in negligible change in the measured voltage. The programmed settling time at this point indicates the settling time needed for the sensor / cable combination.

The following **CRBasic Example: Measuring Settling Time** presents CRBasic code to help determine settling time for a pressure transducer using a high-capacitance semiconductor. The code consists of a series of full-bridge measurements (`BrFull()`) with increasing settling times. The pressure transducer is placed in steady-state conditions so changes in measured voltage are attributable to settling time rather than changes in pressure.

CRBasic Example: Measuring Settling Time

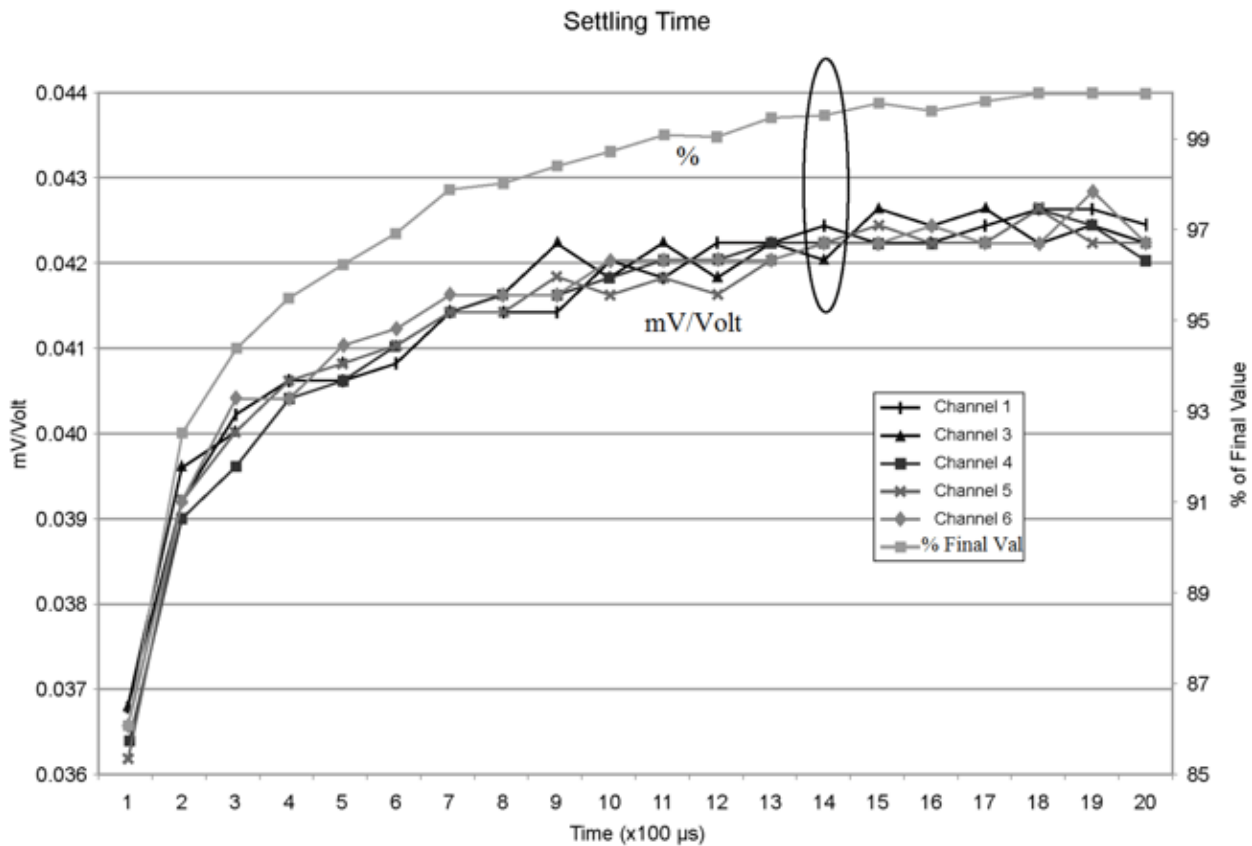
```
'This program example demonstrates the measurement of settling time  
'using a single measurement instruction multiple times in succession.
```

```
Public PT(20) 'Variable to hold the measurements  
DataTable(Settle,True,100)  
  Sample(20,PT(),IEEE4)  
EndTable  
  
BeginProg  
  Scan(1,Sec,3,0)  
  
    BrFull(PT(1),1,mV200,Vx1,1,2500,True,True,100,15000,1.0,0)  
    BrFull(PT(2),1,mV200,Vx1,1,2500,True,True,200,15000,1.0,0)  
    BrFull(PT(3),1,mV200,Vx1,1,2500,True,True,300,15000,1.0,0)  
    BrFull(PT(4),1,mV200,Vx1,1,2500,True,True,400,15000,1.0,0)  
    BrFull(PT(5),1,mV200,Vx1,1,2500,True,True,500,15000,1.0,0)  
    BrFull(PT(6),1,mV200,Vx1,1,2500,True,True,600,15000,1.0,0)  
    BrFull(PT(7),1,mV200,Vx1,1,2500,True,True,700,15000,1.0,0)  
    BrFull(PT(8),1,mV200,Vx1,1,2500,True,True,800,15000,1.0,0)  
    BrFull(PT(9),1,mV200,Vx1,1,2500,True,True,900,15000,1.0,0)  
    BrFull(PT(10),1,mV200,Vx1,1,2500,True,True,1000,15000,1.0,0)  
    BrFull(PT(11),1,mV200,Vx1,1,2500,True,True,1100,15000,1.0,0)  
    BrFull(PT(12),1,mV200,Vx1,1,2500,True,True,1200,15000,1.0,0)  
    BrFull(PT(13),1,mV200,Vx1,1,2500,True,True,1300,15000,1.0,0)  
    BrFull(PT(14),1,mV200,Vx1,1,2500,True,True,1400,15000,1.0,0)  
    BrFull(PT(15),1,mV200,Vx1,1,2500,True,True,1500,15000,1.0,0)  
    BrFull(PT(16),1,mV200,Vx1,1,2500,True,True,1600,15000,1.0,0)  
    BrFull(PT(17),1,mV200,Vx1,1,2500,True,True,1700,15000,1.0,0)  
    BrFull(PT(18),1,mV200,Vx1,1,2500,True,True,1800,15000,1.0,0)  
    BrFull(PT(19),1,mV200,Vx1,1,2500,True,True,1900,15000,1.0,0)  
    BrFull(PT(20),1,mV200,Vx1,1,2500,True,True,2000,15000,1.0,0)  
  
  CallTable Settle  
  
NextScan  
EndProg
```

The first six measurements are shown in the following table:

Timestamp	Rec	PT(1) Smp	PT(2) Smp	PT(3) Smp	PT(4) Smp	PT(5) Smp	PT(6) Smp
8/3/2017 23:34	0	0.03638599	0.03901386	0.04022673	0.04042887	0.04103531	0.04123745
8/3/2017 23:34	1	0.03658813	0.03921601	0.04002459	0.04042887	0.04103531	0.0414396
8/3/2017 23:34	2	0.03638599	0.03941815	0.04002459	0.04063102	0.04042887	0.04123745
8/3/2017 23:34	3	0.03658813	0.03941815	0.03982244	0.04042887	0.04103531	0.04103531
8/3/2017 23:34	4	0.03679027	0.03921601	0.04022673	0.04063102	0.04063102	0.04083316

Each trace in the following image contains all twenty $PT()$ (mV/V) values (left axis) for a given record number, along with an average value showing the measurements as percent of final reading (right axis). The reading has settled to 99.5% of the final value by the fourteenth measurement, which is contained in variable $PT(14)$. This is suitable accuracy for the application, so a settling time of 1400 μ s is determined to be adequate.



FACTORS AFFECTING ACCURACY

Accuracy describes the difference between a measurement and the true value. Many factors affect accuracy. This topic discusses the effect percent-of-reading, offset, and resolution have on the accuracy of the measurement of an analog voltage sensor signal. Accuracy is defined as follows:

$$\text{accuracy} = \text{percent-of-reading} + \text{offset}$$

where percents-of-reading and offsets are displayed in the "Analog Specifications" on page 80.

Note: Error discussed in this section and error-related specifications of the datalogger do not include error introduced by the sensor or by the transmission of the sensor signal to the datalogger.

Measurement Accuracy Example

The following example illustrates the effect percent-of-reading and offset have on measurement accuracy. The effect of offset is usually negligible on large signals.

Example:

- Sensor-signal voltage: ≈ 1050 mV
- CRBasic measurement instruction: `Voltdiff()`
- Programmed input-voltage range (*Range*): `mV5000` (± 5000 mV)
- Input measurement reversal (*RevDiff*): `True`
- Datalogger circuitry temperature: 10° C

Accuracy of the measurement is calculated as follows:

accuracy = percent-of-reading + offset

where

$$\begin{aligned}\text{percent-of-reading} &= 1050 \text{ mV} \cdot \pm 0.04\% \\ &= \pm 0.42 \mu\text{V}\end{aligned}$$

and

$$\text{offset} = 0.5 \mu\text{V}$$

Therefore,

$$\text{accuracy} = \pm(0.42 \text{ mV} + 0.5 \mu\text{V}) = \pm 0.4205 \text{ mV}$$

MINIMIZING OFFSET VOLTAGES

Voltage offset can be the source of significant error. For example, an offset of $3 \mu\text{V}$ on a 2500 mV signal causes an error of only 0.00012% , but the same offset on a 0.25 mV signal causes an error of 1.2% .

Measurement offset voltages are unavoidable, but can be minimized. Offset voltages originate with:

- Ground currents (see "Minimizing Ground Potential Differences" on page 67 for more information).
- Seebeck effect.
- Residual voltage from a previous measurement.

Remedies include:

- Connecting power grounds to power ground terminals (**G**).
- Using input reversal (`RevDiff = True`) with differential measurements.
- Automatic offset compensation for differential measurements when `RevDiff = False`.
- Automatic offset compensation for single-ended measurements when `MeasOff = False`.
- Using `MeasOff = True` for better offset compensation.
- Using excitation reversal (`RevEx = True`) with bridge measurements.
- Programming longer settling times.

Single-ended measurements are susceptible to voltage drop at the ground terminal caused by return currents from another device that is powered from the CR1000X wiring panel, such as another manufacturer's communication modem, or a sensor that requires a lot of power. Currents >5 mA are usually undesirable. The error can be avoided by routing power grounds from these other devices to a power ground **G** terminal, rather than using a signal ground (\oplus) terminal. Ground currents can be

caused by the excitation of resistive-bridge sensors, but these do not usually cause offset error. These currents typically only flow when a voltage excitation is applied. Return currents associated with voltage excitation cannot influence other single-ended measurements because the excitation is usually turned off before the CR1000X moves to the next measurement. However, if the CRBasic program is written in such a way that an excitation terminal is enabled during an unrelated measurement of a small voltage, an offset error may occur.

The Seebeck effect results in small thermally induced voltages across junctions of dissimilar metals as are common in electronic devices. Differential measurements are more immune to these than are single-ended measurements because of passive voltage cancellation occurring between matched high and low pairs such as **1H/1L**. So, use differential measurements when measuring critical low-level voltages, especially those below 200 mV, such as are output from pyranometers and thermocouples.

When analog voltage signals are measured in series by a single measurement instruction, such as occurs when `VOLTSE()` is programmed with `Reps = 2` or more, measurements on subsequent terminals may be affected by an offset, the magnitude of which is a function of the voltage from the previous measurement. While this offset is usually small and negligible when measuring large signals, significant error, or NAN, can occur when measuring very small signals. This effect is caused by dielectric absorption of the integrator capacitor and cannot be overcome by circuit design. Remedies include the following:

- Programing longer settling times.
- Using an individual instruction for each input terminal, the effect of which is to reset the integrator circuit prior to filtering.
- Avoiding preceding a very small voltage input with a very large voltage input in a measurement sequence if a single measurement instruction must be used.

The following **Offset Voltage Compensation Options** table lists some of the tools available to minimize the effects of offset voltages:

CRBasic Measurement Instruction	Input Reversal (RevDiff=True)	Excitation Reversal (RevEx=True)	Measure Offset During Measurement (MeasOff=True)	Measure Offset During Background Calibration (RevDiff=False) (RevEx=False) (MeasOff=False)
BrHalf()		✓		✓
BrHalf3W()		✓		✓
BrHalf4W()	✓	✓		✓
BrFull()	✓	✓		✓
BrFull6W()	✓	✓		✓
TCDiff()	✓			✓
TcSe()			✓	✓
VoltDiff()	✓			✓
VoltSe()			✓	✓

Compensating for Offset Voltage

Differential measurements also have the advantage of an input reversal option, *RevDiff*. When *RevDiff* is *True*, two differential measurements are made, the first with a positive polarity and the second reversed. Subtraction of opposite polarity measurements cancels some offset voltages associated with the measurement.

Ratiometric measurements use an excitation voltage to excite the sensor during the measurement process. Reversing excitation polarity also reduces offset voltage error. Setting the *RevEx* parameter to

True programs the measurement for excitation reversal. Excitation reversal results in a polarity change of the measured voltage so that two measurements with opposite polarity can be subtracted and divided by 2 for offset reduction similar to input reversal for differential measurements.

For example, if 3 μV offset exists in the measurement circuitry, a 5 mV signal is measured as 5.003 mV. When the input or excitation is reversed, the second sub-measurement is -4.997 mV. Subtracting the second sub-measurement from the first and then dividing by 2 cancels the offset:

$$5.003 \text{ mV} - (-4.997 \text{ mV}) = 10.000 \text{ mV}$$
$$10.000 \text{ mV} / 2 = 5.000 \text{ mV}$$

Ratiometric differential measurement instructions allow both **RevDiff** and **RevEx** to be set **True**. This results in four measurement sequences, which the datalogger processes into the reported measurement:

- positive excitation polarity with positive differential input polarity
- negative excitation polarity with positive differential input polarity
- positive excitation polarity with negative differential input polarity
- negative excitation polarity with negative differential input polarity

For ratiometric single-ended measurements, such as a **BrHalf()**, setting **RevEx = True** results in two measurements of opposite excitation polarity that are subtracted and divided by 2 for offset voltage reduction. For **RevEx = False** for ratiometric single-ended measurements, an offset-voltage measurement is determined from self-calibration.

When the datalogger reverses differential inputs or excitation polarity, it delays the same settling time after the reversal as it does before the first sub-measurement. So, there are two delays per measurement when either **RevDiff** or **RevEx** is used. If both **RevDiff** and **RevEx** are **True**, four sub-measurements are performed; positive and negative excitations with the inputs one way and positive and negative excitations with the inputs reversed. The automatic procedure then is as follows:

1. Switch to the measurement terminals.
2. Set the excitation, settle, and then measure.
3. Reverse the excitation, settle, and then measure.
4. Reverse the excitation, reverse the input terminals, settle, measure.
5. Reverse the excitation, settle, measure.

There are four delays per measurement. In cases of excitation reversal, excitation time for each polarity is exactly the same to ensure that ionic sensors do not polarize with repetitive measurements.

Read More: [The Benefits of Input Reversal and Excitation Reversal for Voltage Measurements](#).

Measuring Ground Reference Offset Voltage

Single-ended and differential measurements without input reversal use an offset voltage measurement with the PGIA inputs grounded. This offset voltage is subtracted from the subsequent measurement. For differential measurements without input reversal, this offset voltage measurement is performed as part of the routine background calibration of the datalogger (see "About Background Calibration" on page 50 for more information). Single-ended measurement instructions **VoltsE()** and **TCSe()** **MeasOff** parameter determines whether the offset voltage measured is done at the beginning of the measurement instruction, or as part of self-calibration. This option provides you with the opportunity to weigh measurement speed against measurement accuracy. When **MeasOff = True**, a measurement of the single-ended offset voltage is made at the beginning of the **VoltsE()** instruction. When **MeasOff = False**, measurements will be corrected for the offset voltage determined during

self-calibration. For installations experiencing fluctuating offset voltages, choosing *MeasOff* = *True* for the `VOLTSE()` instruction results in better offset voltage performance.

If *RevDiff*, *RevEx*, or *MeasOff* is disabled (= *False*), offset voltage compensation is automatically performed, albeit less effectively, by using measurements from the background calibration. Disabling *RevDiff*, *RevEx*, or *MeasOff* speeds up measurement time; however, the increase in speed comes at the cost of accuracy because of the following:

- *RevDiff*, *RevEx*, and *MeasOff* are more effective.
- Background calibrations are performed only periodically, so more time skew occurs between the background calibration offsets and the measurements to which they are applied.

Note: When measurement duration must be minimal to maximize measurement frequency, consider disabling *RevDiff*, *RevEx*, and *MeasOff* when datalogger temperatures and return currents are slow to change.

File System Error Codes

Errors can occur when attempting to access files on any of the available drives. All occurrences are rare, but they are most likely to occur when using optional memory cards. Often, formatting the drive will resolve the error. The errors display in the **File Control** messages box or in the **Status** table.

- 1 Invalid format
- 2 Device capabilities error
- 3 Unable to allocate memory for file operation
- 4 Max number of available files exceeded
- 5 No file entry exists in directory
- 6 Disk change occurred
- 7 Part of the path (subdirectory) was not found
- 8 File at EOF
- 9 Bad cluster encountered
- 10 No file buffer available
- 11 Filename too long or has bad chars
- 12 File in path is not a directory
- 13 Access permission, opening DIR or LABEL as file, or trying to open file as DIR or mkdir existing file
- 14 Opening read-only file for write
- 15 Disk full (can't allocate new cluster)
- 16 Root directory is full
- 17 Bad file ptr (pointer) or device not initialized
- 18 Device does not support this operation
- 19 Bad function argument supplied
- 20 Seek out-of-file bounds
- 21 Trying to mkdir an existing dir
- 22 Bad partition sector signature
- 23 Unexpected system ID byte in partition entry
- 24 Path already open
- 25 Access to uninitialized ram drive
- 26 Attempted rename across devices
- 27 Subdirectory is not empty
- 31 Attempted write to Write Protected disk
- 32 No response from drive (Door possibly open)
- 33 Address mark or sector not found
- 34 Bad sector encountered

- 35 DMA memory boundary crossing error
- 36 Miscellaneous I/O error
- 37 Pipe size of 0 requested
- 38 Memory-release error (relmem)
- 39 FAT sectors unreadable (all copies)
- 40 Bad BPB sector
- 41 Time-out waiting for filesystem available
- 42 Controller failure error
- 43 Pathname exceeds _MAX_PATHNAME

File Name and Resource Errors

The maximum size of the file name that can be stored, run as a program, or FTP transferred in the datalogger is 59 characters. If the name + file extension is longer than 59 characters, an **Invalid Filename** error is displayed. If several files are stored, each with a long file name, memory allocated to the root directory can be exceeded before the actual memory of storing files is exceeded. When this occurs, an **Insufficient resources or memory full** error is displayed.

Background Calibration Errors

Background calibration errors are rare. When they do occur, the cause is usually an analog input that exceeds the input limits of the datalogger.

- Check all analog inputs to make sure they are not greater than ± 5 Vdc by measuring the voltage between the input and a **G** terminal. Do this with a multimeter.
- Check for condensation, which can sometimes cause leakage from a 12 Vdc source terminal.
- Check for a loose ground wire on a sensor powered from a **12V** or **SW12** terminal.
- If a multimeter is not available, disconnect sensors, one at a time, that require power from 9 to 16 Vdc. If measurements return to normal, you have found the cause.

Specifications

Electrical specifications are valid over a -40 to +70 °C, non-condensing environment, unless otherwise specified. Extended electrical specifications (noted as XT in specifications) are valid over a -55 to +85 °C non-condensing environment. Recalibration recommended every three years. Critical specifications and system configuration should be confirmed with Campbell Scientific before purchase.

System Specifications	77
Physical Specifications	77
Power Requirements	77
Ground Specifications	78
Power Output Specifications	78
Analog Specifications	80
Current Measurements Specifications	83
Pulse Counting Specifications	82
Digital I/O Specifications	83
Communications Specifications	84

System Specifications

Processor: Renesas RX63N (32-bit with hardware FPU, running at 100 MHz)

Memory:

- 128 MB of flash + 4 MB battery-backed SRAM
- Data Storage: 4 MB SRAM + 72 MB flash
- Data Storage Expansion: removable microSD flash memory, up to 8 GB

Program Execution: 1 ms to one day

Real-Time Clock: Battery backed while external power is disconnected

- **Resolution:** 1 ms
- **Accuracy:** ±3 min. per year, optional correction using GPS (±10 µs), PakBus, or NTP

Wiring Panel Temperature: A 10K3A1A BetaTHERM thermistor, located between the two rows of analog input channels, is measured when reporting wiring panel temperature.

Physical Specifications

Dimensions: 23.8 x 10.1 x 6.2 cm (9.36 x 3.98 x 2.42 in); additional clearance required for cables and leads.

Weight/Mass: 0.86 kg (1.9 lb)

Power Requirements

Protection: Power inputs are protected against surge, overvoltage, overcurrent, and reverse power.

Power In terminal:

- **Voltage Input:** 10 to 16 Vdc.
- **Input Current Limit at 12 Vdc:**
 - o 4.35 A at -40°C
 - o 3 A at 20 °C
 - o 1.56 A at 85 °C

- 30 Vdc sustained voltage limit without damage. Transient voltage suppressor (TVS) diodes at the **Power In** terminal clamps transients to 36 to 40 V. Input voltages greater than 18 V and less than 32 V are tolerated; however, the 12 V output **SW12-1** and **SW12-2** are disabled and will not function until the input voltage falls below 16 V. Sustained input voltages in excess of 32 V can damage the TVS diodes.

USB Power: USB 5 Vdc supports programming, settings, and analog measurement. When powered by USB (no other power supplies connected) **Status** field **Battery** = 0.

Internal Lithium Battery: AA, 2.4 Ah, 3.6 Vdc (Tadiran TL 5903/S) for battery-backed SRAM and clock. 3 year life with no external power source.

Average Current Drain (assumes 12 Vdc on **POWER IN** terminals):

- **Idle:** <1 mA
- **Active 1 Hz Scan:** 1 mA
- **Active 20 Hz Scan:** 55 mA
- **Serial (RS-232/RS-485):** Active + 25 mA
- **Ethernet Power Requirements**
 - o Ethernet 1 Minute: Active + 1 mA
 - o Ethernet Idle: Active + 4 mA
 - o Ethernet Link: Active + 48 mA

Vehicle Power Connection: When primary power is pulled from the vehicle power system, a second power supply OR charge regulator may be required to overcome the voltage drop at vehicle startup.

Ground Specifications

Signal Ground (⚡) - reference for single-ended analog inputs, excitation returns, and as a ground for sensor shield wires.

- 11 common terminals

Power Ground (G) - return for 5V, 12V, and digital sensors.

- 4 common terminals

Earth Ground Lug (⚡) - connection point for heavy-gauge earth-ground wire. 14 AWG wire, minimum, is recommended.

Resistive Ground (RG) - two resistance-to-ground inputs that can be used for non-isolated 0-20 mA and 4-20 mA current loop measurements or for terminating the ground reference of an RS-485 serial connection. Maximum voltage for RG terminals is ± 16 V. See also "Current Measurements Specifications" on page 83.

Power Output Specifications

Power Output Terminals:

- **12V, SW12-1, and SW12-2:** Provide unregulated 12 Vdc power with voltage equal to the Power Input supply voltage. These are disabled when operating on USB power only. Two levels of thermal fuses regulate current sourcing. In total (**12V** + **SW12-1** + **SW12-2**) the hold current is limited to 2.68 A @ -40°C, 0.96 A @ 80°C.

SW12-1 and **SW12-2** can be independently set under program control. Each SW12 terminal has

a hold current limited to 1.3 A @ -40°C, 0.47 A @ 80°C. See the **VX** terminal information for additional regulated voltage outputs.

Terminal	Source Limit ¹
SW12-1 or SW12-2	1310 mA @ -40 °C 1040 mA @ 0 °C 900 mA @ 20 °C 690 mA @ 50 °C 550 mA @ 70 °C 470 mA @ 80 °C
12V + SW12-1 +SW12-2	2680 mA @ -40 °C 2130 mA @ 0 °C 1850 mA @ 20 °C 1420 mA @ 50 °C 1130 mA @ 70 °C 960 mA @ 85 °C

¹Thermal fuse hold current. Overload causes voltage drop. Disconnect and let cool to reset. Operate at limit if the application can tolerate some fluctuation.

- **5V:** One regulated 5V output. Supply is shared between the **5V** terminal and **CS I/O DB9 5 V** output. (See the **VX** terminal information for additional regulated voltage outputs.)
 - o **Voltage Output:** Regulated 5 V output (±5%)
 - o **Current Limit:** 230 mA
- **C1-C8:** Values for **C** terminals reflect a thermal fuse limit current. Circuit holds current at the maximum by dropping the voltage when the load is too great. To reset, disconnect and allow circuit to cool. Operating at the current limit is OK so long some fluctuation can be tolerated. Drive capacity is determined by the logic level of the Vdc supply and the output resistance (R_o) of the terminal.
 - o **Output Resistance (R_o):** 150 Ω
 - o **5 V Logic Level Drive Capacity:** 10 mA @ 3.5 Vdc; $V_{out} = 5 V - (I_{out} \times 150 \Omega)$
 - o **3.3 V Logic Level Drive Capacity:** 10 mA @ 1.8 Vdc; $V_{out} = 3.3 V - (I_{out} \times 150 \Omega)$
- **VX:** Four independently configurable voltage terminals (**VX1-VX4**). When providing voltage excitation, a single 16-bit DAC shared by all **VX** outputs produces a user-specified voltage during measurement only. In this case, these terminals are regularly used with resistive-bridge measurements (see "Resistance Measurements" on page 32 for more information). **VX** terminals can also be used to supply a selectable, switched, regulated 3.3 or 5 Vdc power source to power digital sensors and toggle control lines.

	Range	Resolution	Accuracy ¹²	Maximum Source/Sink Current ³
Voltage Excitation	±4 V	0.06 mV	±(0.1% of setting + 2 mV)	±40 mA
Switched, Regulated	+3.3 or 5 V	3.3 or 5 V	±5%	50 mA

¹ Valid over -55 to +85 °C temperature range

² Note that ratiometric accuracy, rather than the absolute accuracy of excitation or analog measurement, determines the accuracy of ratiometric resistance measurements.

³ Exceeding current limits causes voltage output to become unstable. Voltage should stabilize when current is reduced to within stated limits.

System Power Out Limits:

Temperature (°C)	Current Limit at 12 Vdc ¹ (Amperes)
-40°	4.53
20°	3.00
70°	1.83
85°	1.56

¹Limited by self-resetting thermal fuse

Analog Specifications

16 single-ended (SE) or 8 differential (DIFF) inputs individually configurable for voltage, thermocouple, ratiometric, and period average measurements, using a 24-bit ADC. One channel at a time is measured in numeric succession.

VOLTAGE MEASUREMENTS

Input Resistance: 20 GΩ

Input Limits: ±5 V

Sustained Input Voltage without Damage: ±20 Vdc

Dc Common Mode Rejection:

- > 120 dB with input reversal
- ≥ 86 dB without input reversal

Normal Mode Rejection: > 70 dB @ 60 Hz

Input Current: ±1 nA typical @ 25 °C

Filter First Notch Frequency (f_{N1}) Range: 0.5 Hz to 31.25 kHz

Analog Range and Resolution:

		Differential with Input Reversal		Single-Ended and Differential without Input Reversal	
Notch Frequency (f _{N1}) ¹ (Hz)	Range ² (mV)	RMS (μV)	Bits ³	RMS (μV)	Bits ³
15000	±5000	8.2	20	11.8	19
	±1000	1.9	20	2.6	19
	±200	0.75	19	1.0	18
50/60 ⁴	±5000	0.6	24	0.88	23
	±1000	0.14	23	0.2	23
	±200	0.05	22	0.08	22
5	±5000	0.18	25	0.28	25
	±1000	0.04	25	0.07	24
	±200	0.02	24	0.03	23

¹ Valid frequencies are 0.5 Hz to 31.25 kHz.

² Range overhead of ~5% on all ranges guarantees that full-scale values will not cause over range.

³ Typical effective resolution (ER) in bits; computed from ratio of full-scale range to RMS resolution.

⁴ 50/60 correspond to rejection of 50 and 60 Hz ac power mains noise.

Accuracy (does not include sensor or measurement noise):

- 0 to 40 °C: ±(0.04% of measurement + offset)
- -40 to 70 °C: ±(0.06% of measurement + offset)
- -55 to 85 °C (XT): ±(0.08% of measurement + offset)

Voltage Measurement Accuracy Offsets:

Range (mV)	Typical Offset (μV RMS)	
	Differential with Input Reversal	Single-Ended or Differential without Input Reversal
±5000	±0.5	±2
±1000	±0.25	±1
±200	±0.15	±0.5

Multiplexed Measurement Time: $(450 \mu\text{s} + \text{settling time} + (1/fN1)) \cdot \text{reps}$
 (Default settling time of 500 μs . These are not maximum speeds. Multiplexed denotes circuitry inside the datalogger that switches signals into the ADC.)

Multiplexed Measurement Time (ms) with 500 μs Settling Time:

	Differential with Input Reversal	Single-Ended or Differential without Input Reversal
Example $fN1^1$ (Hz)	Time (ms)	Time (ms)
15000	2.04	1.02
60	35.24	17.62
50	41.9	20.95
5	401.9	200.95

¹Notch frequency (1/integration time).

Measurement Settling Time: 20 μs to 600 ms; 500 μs default

RATIOMETRIC-RESISTANCE MEASUREMENTS SPECIFICATIONS

The datalogger makes ratiometric-resistance measurements for four- and six-wire full bridge and two-, three-, and four-wire half bridge using voltage excitation. Excitation polarity reversal available to minimize dc error. Typically, at least one terminal is configured for excitation output. Multiple sensors can usually use a common excitation terminal.

Accuracy: Assumes input reversal for differential measurements along with excitation reversal along with excitation reversal for excitation voltage <1000 mV. Does not include bridge resistor errors or sensor and measurement noise.

Ratiometric accuracy, rather than absolute accuracy, determines overall measurement accuracy. Offset is the same as specified for analog voltage measurements.

- 0 to 40 °C: $\pm(0.01\%$ of voltage measurement + offset)
- -40 to 70 °C: $\pm(0.015\%$ of voltage measurement+ offset)
- -55 to 85 °C (XT): $\pm(0.02\%$ of voltage measurement+ offset)

PERIOD AVERAGING SPECIFICATIONS

Up to 16 analog inputs can be used for period averaging.

Accuracy: $\pm(0.01\%$ of reading + resolution), where resolution is $(0.13 \mu\text{s} / n)$, where n = number of cycles to be measured

Ranges:

Minimum signal centered around specified period average threshold.

Maximum signal centered around datalogger ground.

Maximum frequency = $1/(2 \cdot (\text{minimum pulse width}))$ for 50% duty cycle signals

Gain Code Option	Minimum Peak to Peak Signal (mV)	Maximum Peak to Peak Signal (V)	Minimum Pulse Width (μs)	Maximum Frequency (kHz)
0	500	10	2.5	200
1	50	2	10	50
2	10	2	62	8
3	2	2	100	5

Current Measurements Specifications

The datalogger makes current loop measurements by measuring across a current-sense resistor associated with the RS-485 resistive ground terminals RG1 & RG2.

Maximum Input Voltage: ± 16 V

Resistance to Ground: 101 Ω

Current Measurement Shunt Resistance: 10 Ω

Maximum Current Measurement Range: ± 80 mA

Absolute Maximum Current: ± 160 mA

Current Measurement Resolution: ≤ 20 nA

Current Measurement Accuracy: $\pm(0.1\%$ of Reading + 100 nA) @ -40 to 70 °C

See also "Current Measurements" on page 30.

Pulse Counting Specifications

Two inputs (P1-P2) individually configurable for switch closure, high-frequency pulse, or low-level AC measurements. See also "Digital I/O Specifications" on page 83. Each terminal has its own independent 32-bit counter.

Maximum Input Voltage: ± 20 Vdc

Accuracy: $\pm(0.02\%$ of reading + 1/scan)

SWITCH CLOSURE INPUT

Pull-Up Resistance: 100 k Ω to 5 V

Event: Low (<0.8 V) to High (>2.5 V)

Minimum Switch Closed Time: 5 ms

Minimum Switch Open Time: 6 ms

Maximum Bounce Time: 1 ms open without being counted

HIGH-FREQUENCY INPUT

Pull-Up Resistance: 100 k Ω to 5 V

Event: Low (<0.8 V) to High (>2.5 V)

Internal Pull-Up Resistance: 100 k Ω to 5 V

Maximum Input Frequency: 250 kHz

LOW-LEVEL AC INPUT

Minimum Pull-Down Resistance: 10 k Ω to ground

Dc-offset Rejection: Internal ac coupling eliminates dc-offset voltages up to ± 0.5 Vdc

Input Hysteresis: 12 mV @ 1 Hz

Low-Level Ac Pulse Input Ranges:

- Sine Wave Input 20 mV RMS, Input Frequency Range 1.0 to 20 Hz
- Sine Wave Input 200 mV RMS, Input Frequency Range 0.5 to 200 Hz
- Sine Wave Input 2000 mV RMS, Input Frequency Range 0.3 to 10,000 Hz
- Sine Wave Input 5000 mV RMS, Input Frequency Range 0.3 to 20,000 Hz

Digital I/O Specifications

Eight terminals (C1-C8) configurable for digital input and output including status high/low, pulse width modulation, external interrupt, edge timing, switch closure pulse counting, high-frequency pulse counting, UART, RS-232, RS-485, SDM, SDI-12, I2C, and SPI function. Terminals are configurable in pairs for 5 V or 3.3 V logic for some functions.

Maximum Input Voltage: ± 20 V

Logic Levels and Drive Current:

Terminal Pair Configuration	Logic Low	Logic High	Current Source
5 V	≤ 1.5 V	≥ 3.5 V	10 mA @ 3.5 V
3.3 V	≤ 0.8 V	≥ 2.5 V	10 mA @ 1.85 V

Digital I/O Function	C1	C2	C3	C4	C5	C6	C7	C8
Switch Closure	✓	✓	✓	✓	✓	✓	✓	✓
High-Frequency	✓	✓	✓	✓	✓	✓	✓	✓
General I/O	✓	✓	✓	✓	✓	✓	✓	✓
Pulse-Width Modulation	✓	✓	✓	✓	✓	✓	✓	✓
Timer Input	✓	✓	✓	✓	✓	✓	✓	✓
Interrupt	✓	✓	✓	✓	✓	✓	✓	✓
SDI-12	✓		✓		✓		✓	
GPS Time Sync	PPS	Rx						
5 V or 3.3 V TTL or RS-232	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx
RS-232					Tx	Rx	Tx	Rx
RS-485 Half Duplex					A-	B+	A-	B+
RS-485 Full Duplex					Tx-	Tx+	Rx-	Rx+
SDM ¹	DATA	CLK	ENABLE		DATA	CLK	ENABLE	
I2C	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA
SPI	CLX	MOSI	MISO		CLX	MOSI	MISO	

¹SDM can be on either C1-C3 or C5-C7, but not both at the same time.

SWITCH CLOSURE INPUT

Accuracy: $\pm(0.02\%$ of reading + 1/scan)

Resistance: Configurable in terminal pairs with 100 k Ω pull-up or pull-down

Software Debounce Time: 3 ms

Maximum Bounce Time: 1 ms open w/o being counted

Maximum Input Frequency: 150 Hz

HIGH-FREQUENCY INPUT

Accuracy: $\pm(0.02\%$ of reading + 1/scan)

Resistance: Configurable in terminal pairs with 100 k Ω pull-up or pull-down

Maximum Input Frequency: 1 MHz

EDGE TIMING

Maximum Input Frequency: ≤ 2.3 Hz

Resolution: 500 ns

Communications Specifications

A datalogger is normally part of a two-way conversation started by a computer. In applications with some types of interfaces, the datalogger can also initiate the call (callback) when needed. In satellite applications, the datalogger may simply send bursts of data at programmed times without waiting for a response.

Ethernet Port: RJ45/ jack 10/100Base MBps, full and half duplex, Auto-MDIX, magnetic isolation, and TVS surge protection

See also "Configuring Settings to Communicate over Ethernet" on page 13.

Internet Protocols: Ethernet, PPP, CS I/O IP, RNDIS, ICMP/Ping, Auto-IP(APIPA), IPv4, IPv6, UDP, TCP, TLS, DNS, DHCP, SLAAC, SNMPv2, NTP, Telnet, HTTP(S), FTP(S), SMTP/TLS, POP3/TLS

Additional Protocols: PakBus, PakBus Encryption, CPI, SDM, SDI-12, Modbus RTU / ASCII / TCP, DNP3, NTCIP, NMEA 0183, I2C, SPI, custom user definable over serial, UDP

Data File Formats: CSV, XML, JSON, binary, encrypted, custom user definable

USB: Micro-B device for computer connectivity

CS I/O: 9-pin D-sub multidrop interface to interface with Campbell Scientific CS I/O peripherals

CS I/O Pinout:

Pin Number	Function	Input (I) Output(o)	Description
1	5 Vdc	O	5 Vdc: sources 5 Vdc, used to power peripherals.
2	SG		Signal ground: provides a power return for pin 1 (5V), and is used as a reference for voltage levels.
3	RING	I	Ring: raised by a peripheral to put the CR1000X in the telecom mode.
4	RXD	I	Receive data: serial data transmitted by a peripheral are received on pin 4.
5	ME	O	Modem enable: raised when the CR1000X determines that a modem raised the ring line.
6	SDE	O	Synchronous device enable: addresses synchronous devices (SD); used as an enable line for printers.
7	CLK/HS	I/O	Clock/handshake: with the SDE and TXD lines addresses and transfers data to SDs. When not used as a clock, pin 7 can be used as a handshake line; during printer output, high enables, low disables.
8	12 Vdc		Nominal 12 Vdc power. Same power as 12V and SW12 terminals.
9	TXD	O	Transmit data: transmits serial data from the datalogger to peripherals on pin 9; logic-low marking (0V), logic-high spacing (5V), standard-asynchronous ASCII: eight data bits, no parity, one start bit, one stop bit. User selectable baud rates: 300, 1200, 2400, 4800, 9600, 19200, 38400, 115200.

RS-232/CPI: Single RJ45 module port that can operate in one of two modes: RS-232 or CPI. RS-232 is used to connect computer, sensor, or communication devices serially. CPI is used to connect Campbell Scientific CDM measurement peripherals and sensors.

CPI/RS-232 Pinout:

Pin Number	Description
1	RS-232: Transmit (Tx)
2	RS-232: Receive (Rx)
3	100 Ω Res Ground
4	CPI: Data
5	CPI: Data
6	100 Ω Res Ground
7	RS-232 CTS CPI: Sync
8	RS-232 DTR CPI: Sync
9	Not Used

RJ45 Interface to Campbell Scientific CDM Measurement Peripherals Pinout:

Pin Number	Description
1	Not Used
2	Not Used
3	100 Ω Res Ground
4	CAN Data Pair
5	CAN Data Pair
6	100 Ω Res Ground
7	RS485 Sync Pair
8	RS485 Sync Pair
9	Not used

Support

Product Assistance

For technical assistance with Campbell Scientific equipment, go to www.campbellsci.com/support, or contact technical support at 435-227-9100.

For repair and calibration services, visit www.campbellsci.com/repair.

Precautions

DANGER – MANY HAZARDS ARE ASSOCIATED WITH INSTALLING, USING, MAINTAINING, AND WORKING ON OR AROUND TRIPODS, TOWERS, AND ANY ATTACHMENTS TO TRIPODS AND TOWERS SUCH AS SENSORS, CROSSARMS, ENCLOSURES, ANTENNAS, ETC. FAILURE TO PROPERLY AND COMPLETELY ASSEMBLE, INSTALL, OPERATE, USE, AND MAINTAIN TRIPODS, TOWERS, AND ATTACHMENTS, AND FAILURE TO HEED WARNINGS, INCREASES THE RISK OF DEATH, ACCIDENT, SERIOUS INJURY, PROPERTY DAMAGE, AND PRODUCT FAILURE. TAKE ALL REASONABLE PRECAUTIONS TO AVOID THESE HAZARDS. CHECK WITH YOUR ORGANIZATION'S SAFETY COORDINATOR (OR POLICY) FOR PROCEDURES AND REQUIRED PROTECTIVE EQUIPMENT PRIOR TO PERFORMING ANY WORK.

Use tripods, towers, and attachments to tripods and towers only for purposes for which they are designed. Do not exceed design limits. Be familiar and comply with all instructions provided in product manuals. Manuals are available at www.campbellsci.com or by telephoning 435-227-9000 (USA). You are responsible for conformance with governing codes and regulations, including safety regulations, and the integrity and location of structures or land to which towers, tripods, and any attachments are attached. Installation sites should be evaluated and approved by a qualified engineer. If questions or concerns arise regarding installation, use, or maintenance of tripods, towers, attachments, or electrical connections, consult with a licensed and qualified engineer or electrician.

General

- Prior to performing site or installation work, obtain required approvals and permits. Comply with all governing structure-height regulations, such as those of the FAA in the USA.
- Use only qualified personnel for installation, use, and maintenance of tripods and towers, and any attachments to tripods and towers. The use of licensed and qualified contractors is highly recommended.
- Read all applicable instructions carefully and understand procedures thoroughly before beginning work.
- Wear a hardhat and eye protection, and take other appropriate safety precautions while working on or around tripods and towers.
- Do not climb tripods or towers at any time, and prohibit climbing by other persons. Take reasonable precautions to secure tripod and tower sites from trespassers.
- Use only manufacturer recommended parts, materials, and tools.

Utility and Electrical

- You can be killed or sustain serious bodily injury if the tripod, tower, or attachments you are installing, constructing, using, or maintaining, or a tool, stake, or anchor, come in contact with overhead or underground utility lines.

- Maintain a distance of at least one-and-one-half times structure height, or 20 feet, or the distance required by applicable law, whichever is greater, between overhead utility lines and the structure (tripod, tower, attachments, or tools).
- Prior to performing site or installation work, inform all utility companies and have all underground utilities marked.
- Comply with all electrical codes. Electrical equipment and related grounding devices should be installed by a licensed and qualified electrician.

Elevated Work and Weather

- Exercise extreme caution when performing elevated work.
- Use appropriate equipment and safety practices.
- During installation and maintenance, keep tower and tripod sites clear of un-trained or non-essential personnel. Take precautions to prevent elevated tools and objects from dropping.
- Do not perform any work in inclement weather, including wind, rain, snow, lightning, etc.

Maintenance

- Periodically (at least yearly) check for wear and damage, including corrosion, stress cracks, frayed cables, loose cable clamps, cable tightness, etc. and take necessary corrective actions.
- Periodically (at least yearly) check electrical ground connections.

DANGER: Fire, explosion, and severe-burn hazard. Misuse or improper installation of the internal lithium battery can cause severe injury. Do not recharge, disassemble, heat above 100 °C (212 °F), solder directly to the cell, incinerate, or expose contents to water. Dispose of spent lithium batteries properly.

WARNING:

- Protect from over-voltage.
- Protect from water (see "Datalogger Enclosures" on page 51).
- Protect from ESD (see "Electrostatic Discharge and Lightning Protection" on page 56).

IMPORTANT: Note the following about the internal battery:

- When primary power is continuously connected to the datalogger, the battery will last up to 10 years or more.
- When primary power is NOT connected to the datalogger, the battery will last about three years.
- See "Internal Battery" on page 54 for more information.

IMPORTANT: Maintain a level of calibration appropriate to the application. Campbell Scientific recommends factory recalibration of the datalogger every three years.

WHILE EVERY ATTEMPT IS MADE TO EMBODY THE HIGHEST DEGREE OF SAFETY IN ALL CAMPBELL SCIENTIFIC PRODUCTS, THE CUSTOMER ASSUMES ALL RISK FROM ANY INJURY RESULTING FROM IMPROPER INSTALLATION, USE, OR MAINTENANCE OF TRIPODS, TOWERS, OR ATTACHMENTS TO TRIPODS AND TOWERS SUCH AS SENSORS, CROSSARMS, ENCLOSURES, ANTENNAS, ETC.

Warranty

The datalogger is warranted for three (3) years subject to this limited warranty:

“Products manufactured by CSI are warranted by CSI to be free from defects in materials and workmanship under normal use and service for twelve months from the date of shipment unless otherwise specified in the corresponding product manual. (Product manuals are available for review

online at www.campbellsci.com.) Products not manufactured by CSI, but that are resold by CSI, are warranted only to the limits extended by the original manufacturer. Batteries, fine-wire thermocouples, desiccant, and other consumables have no warranty. CSI's obligation under this warranty is limited to repairing or replacing (at CSI's option) defective Products, which shall be the sole and exclusive remedy under this warranty. The Customer assumes all costs of removing, reinstalling, and shipping defective Products to CSI. CSI will return such Products by surface carrier prepaid within the continental United States of America. To all other locations, CSI will return such Products best way CIP (port of entry) per Incoterms ® 2010. This warranty shall not apply to any Products which have been subjected to modification, misuse, neglect, improper service, accidents of nature, or shipping damage. This warranty is in lieu of all other warranties, expressed or implied. The warranty for installation services performed by CSI such as programming to customer specifications, electrical connections to Products manufactured by CSI, and Product specific training, is part of CSI's product warranty. CSI EXPRESSLY DISCLAIMS AND EXCLUDES ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CSI hereby disclaims, to the fullest extent allowed by applicable law, any and all warranties and conditions with respect to the Products, whether express, implied or statutory, other than those expressly provided herein.”

Acknowledgements

lwIP

Copyright (c) 2001-2004 Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Glossary

A

ac

Alternating current (see Vac).

accuracy

The degree to which the result of a measurement, calculation, or specification conforms to the correct value or a standard.

ADC

Analog to digital conversion. The process that translates analog voltage levels to digital values.

amperes (A)

Base unit for electric current. Used to quantify the capacity of a power source or the requirements of a power-consuming device.

analog

Data presented as continuously variable electrical signals.

argument

Part of a procedure call (or command execution).

ASCII/ANSI

Abbreviation for American Standard Code for Information Interchange / American National Standards Institute. An encoding scheme in which numbers from 0-127 (ASCII) or 0-255 (ANSI) are used to represent pre-defined alphanumeric characters. Each number is usually stored and transmitted as 8 binary digits (8 bits), resulting in 1 byte of storage per character of text.

asynchronous

The transmission of data between a transmitting and a receiving device occurs as a series of zeros and ones. For the data to be "read" correctly, the receiving device must begin reading at the proper point in the series. In asynchronous communication, this coordination is accomplished by having each character surrounded by one or more start and stop bits which designate the beginning and ending points of the information. Also indicates the sending and receiving devices are not synchronized using a clock signal.

AWG

AWG ("gauge") is the accepted unit when identifying wire diameters. Larger AWG values indicate smaller cross-sectional diameter wires. Smaller AWG values indicate large-diameter wires. For example, a 14 AWG wire is often used for grounding because it can carry large currents. 22 AWG wire is often used as sensor leads since only small currents are carried when measurements are made.

B

baud rate

The rate at which data are transmitted.

beacon

A signal broadcasted to other devices in a PakBus® network to identify "neighbor" devices. A beacon in a PakBus network ensures that all devices in the network are aware of other devices that are viable. If configured to do so, a clock-set command may be transmitted with the beacon. This function can be used to synchronize the clocks of devices within the PakBus network.

binary

Describes data represented by a series of zeros and ones. Also describes the state of a switch, either being on or off.

BOOL8

A one-byte data type that holds eight bits (0 or 1) of information. BOOL8 uses less space than the 32 bit BOOLEAN data type.

boolean

Name given a function, the result of which is either true or false.

boolean data type

Typically used for flags and to represent conditions or hardware that have only two states (true or false) such as flags and control ports.

burst

Refers to a burst of measurements. Analogous to a burst of light, a burst of measurements is intense, such that it features a series of measurements in rapid succession, and is not continuous.

C

calibration wizard

The calibration wizard facilitates the use of the CRBasic field calibration instructions FieldCal() and FieldCalStrain(). It is found in LoggerNet (4.0 or higher) or RTDAQ.

callback

A name given to the process by which the datalogger initiates communications with a computer running appropriate Campbell Scientific datalogger support software. Also known as "Initiate Comms."

CardConvert software

A utility to retrieve binary final-storage data from memory cards and convert the data to ASCII or other formats.

CD100

An optional enclosure mounted keyboard/display for use with dataloggers.

CDM/CPI

CPI is a proprietary interface for communications between Campbell Scientific dataloggers and Campbell Scientific CDM peripheral devices. It consists of a physical layer definition and a data protocol. CDM devices are similar to Campbell Scientific SDM devices in concept, but the use of the CPI bus enables higher data-throughput rates and use of longer cables. CDM devices require more power to operate in general than do SDM devices.

CF

CompactFlash®

code

A CRBasic program, or a portion of a program.

Collect button

Button or command in datalogger support software that facilitates collection-on-demand of final-data memory. This feature is found in PC200W, PC400, LoggerNet, and RTDAQ software.

Collect Now button

Button or command in datalogger support software that facilitates collection-on-demand of final-data memory. This feature is found in PC200W, PC400, LoggerNet, and RTDAQ software.

COM port

COM is a generic name given to physical and virtual serial communication ports.

COM1

When configured as a communication port, terminals C1 and C2 act as a pair to form Com1.

command

An instruction or signal that causes a computer to perform one of its basic functions (usually in CRBasic).

command line

One line in a CRBasic program. Maximum length, even with the line continuation characters <space> <underscore> (_), is 512 characters. A command line usually consists of one program statement, but it may consist of multiple program statements separated by a <colon> (:).

CompactFlash

CompactFlash® (CF) is a memory-card technology used in some Campbell Scientific card-storage modules.

compile

The software process of converting human-readable program code to binary machine code. Datalogger user programs are compiled internally by the datalogger operating system.

conditioned output

The output of a sensor after scaling factors are applied.

connector

A connector is a device that allows one or more electron conduits (wires, traces, leads, etc) to be connected or disconnected as a group. A connector consists of two parts – male and female. For example, a common household ac power receptacle is the female portion of a connector. The plug at the end of a lamp power cord is the male portion of the connector.

constant

A packet of memory given an alpha-numeric name and assigned a fixed number.

control I/O

C terminals configured for controlling or monitoring a device.

CoraScript

CoraScript is a command-line interpreter associated with LoggerNet datalogger support software.

CPU

Central processing unit. The brains of the datalogger.

cr

Carriage return.

CRBasic

Campbell Scientific's BASIC-like programming language that supports analog and digital measurements, data processing and analysis routines, hardware control, and many communication protocols.

CRBasic Editor

The CRBasic programming editor; supplied as part of LoggerNet, PC400, and RTDAQ software.

CRD

An optional memory drive that resides on a memory card.

CS I/O

Campbell Scientific proprietary input/output port. Also the proprietary serial communication protocol that occurs over the CS I/O port.

CVI

Communication verification interval. The interval at which a PakBus® device verifies the accessibility of neighbors in its neighbor list. If a neighbor does not communicate for a period of time equal to 2.5 times the CVI, the device will send up to four Hellos. If no response is received, the neighbor is removed from the neighbor list.

D

DAC

Digital to analog conversion. The process that translates digital voltage levels to analog values.

data bits

Number of bits used to describe the data and fit between the start and stop bit. Sensors typically use 7 or 8 data bits.

data cache

The data cache is a set of binary files kept on the hard disk of the computer running the datalogger support software. A binary file is created for each table in each datalogger. These files mimic the storage areas in datalogger memory, and by default are two times the size of the datalogger storage area. When the software collects data from a datalogger, the data are stored in the binary file for that datalogger. Various software functions retrieve data from the data cache instead of the datalogger directly. This allows the simultaneous sharing of data among software functions.

data output interval

The interval between each write of a record to a final-storage memory data table.

data output processing instructions

CRBasic instructions that process data values for eventual output to final-data memory. Examples of output-processing instructions include Totalize(), Maximize(), Minimize(), and Average(). Data sources for these instructions are values or strings in variable memory. The results of intermediate calculations are stored in data output processing memory to await the output trigger. The ultimate destination of data generated by data output processing instructions is usually final-storage memory, but the CRBasic program can be written to divert to variable memory by the CRBasic program for further processing. The transfer of processed summaries to final-data memory takes place when the Trigger argument in the DataTable() instruction is set to True.

data output processing memory

SRAM memory automatically allocated for intermediate calculations performed by CRBasic data output processing instructions. Data output processing memory cannot be monitored.

data point

A data value which is sent to final-storage memory as the result of a data-output processing instruction. Strings of data points output at the same time make up a record in a data table.

data table

A concept that describes how data are organized in memory, or in files that result from collecting data in memory. The fundamental data table is created by the CRBasic program as a result of the DataTable() instruction and resides in binary form in main-memory SRAM. The data table structure also resides in the data cache, in discrete data files on the CPU, USB, CRD, and USB memory drives, and in binary or ASCII files that result from collecting final-storage memory with datalogger support software.

datalogger support software

LoggerNet, PC400, and PC200W - these Campbell Scientific software applications includes at least the following functions: datalogger communications, downloading programs, clock setting, and retrieval of measurement data.

DC

Direct current.

DCE

Data Communication Equipment. While the term has much wider meaning, in the limited context of practical use with the datalogger, it denotes the pin configuration, gender, and function of an RS-232 port. The RS-232 port on the datalogger is DCE. Interfacing a DCE device to a DCE device requires a null-modem cable.

desiccant

A hygroscopic material that absorbs water vapor from the surrounding air. When placed in a sealed enclosure, such as a datalogger enclosure, it prevents condensation.

Device Configuration Utility

Configuration tool used to set up dataloggers and peripherals, and to configure PakBus settings before those devices are deployed in the field and/or added to networks.

DHCP

Dynamic Host Configuration Protocol. A TCP/IP application protocol.

differential

A sensor or measurement terminal wherein the analog voltage signal is carried on two leads. The phenomenon measured is proportional to the difference in voltage between the two leads.

Dim

A CRBasic command for declaring and dimensioning variables. Variables declared with Dim remain hidden during datalogger operations.

dimension

To code a CRBasic program for a variable array as shown in the following examples: DIM example(3) creates the three variables example(1), example(2), and example(3); DIM example(3,3) creates nine variables; DIM example(3,3,3) creates 27 variables.

DNP3

Distributed Network Protocol is a set of communications protocols used between components in process automation systems. Its main use is in utilities such as electric and water companies.

DNS

Domain name server. A TCP/IP application protocol.

DTE

Data Terminal Equipment. While the term has much wider meaning, in the limited context of practical use with the datlogger, it denotes the pin configuration, gender, and function of an RS-232 port. The RS-232 port on the datlogger is DCE. Attachment of a null-modem cable to a DCE device effectively converts it to a DTE device.

duplex

A serial communication protocol. Serial communications can be simplex, half-duplex, or full-duplex.

duty cycle

The percentage of available time a feature is in an active state. For example, if the datlogger is programmed with 1 second scan interval, but the program completes after only 100 millisecond, the program can be said to have a 10% duty cycle.

E

earth ground

A grounding rod or other suitable device that electrically ties a system or device to the earth. Earth ground is a sink for electrical transients and possibly damaging potentials, such as those produced by a nearby lightning strike. Earth ground is the preferred reference potential for analog voltage measurements. Note that most objects have a "an electrical potential" and the potential at different places on the earth - even a few meters away - may be different.

engineering units

Units that explicitly describe phenomena, as opposed to, for example, the datlogger base analog-measurement unit of milliVolts.

ESD

Electrostatic discharge.

ESS

Environmental sensor station.

excitation

Application of a precise voltage, usually to a resistive bridge circuit.

execution interval

The time interval between initiating each execution of a given Scan() of a CRBasic program. If the Scan() Interval is evenly divisible into 24 hours (86,400 seconds), it is synchronized with the 24 hour clock, so that the program is executed at midnight and every Scan() Interval thereafter. The program is executed for the first time at the first occurrence of the Scan() Interval after compilation. If the Scan() Interval does not divide evenly into 24 hours, execution will start on the first even second after compilation.

execution time

Time required to execute an instruction or group of instructions. If the execution time of a program exceeds the Scan() Interval, the program is executed less frequently than programmed and the Status table SkippedScan field will increment.

expression

A series of words, operators, or numbers that produce a value or result.

F

FFT

Fast Fourier Transform. A technique for analyzing frequency-spectrum data.

field

Final-storage data tables are made up of records and fields. Each row in a table represents a record and each column represents a field. The number of fields in a record is determined by the number and configuration of output processing instructions that are included as part of the DataTable() declaration.

File Control

File Control is a feature of LoggerNet, PC400 and RTDAQ datalogger support software. It provides a view of the datalogger file system and a menu of file management commands.

fill and stop memory

A memory configuration for data tables forcing a data table to stop accepting data when full.

final-storage data

Data that resides in final-data memory.

final-storage memory

The portion of SRAM memory allocated for storing data tables with output arrays. Once data are written to final-data memory, they cannot be changed but only overwritten when they become the oldest data. Final-data memory is configured as ring memory by default, with new data overwriting the oldest data.

Flash

A type of memory media that does not require battery backup. Flash memory, however, has a lifetime based on the number of writes to it. The more frequently data are written, the shorter the life expectancy.

FLOAT

Four-byte floating-point data type. Default datalogger data type for Public or Dim variables. Same format as IEEE4.

FP2

Two-byte floating-point data type. Default datalogger data type for stored data. While IEEE four-byte floating point is used for variables and internal calculations, FP2 is adequate for most stored data. FP2 provides three or four significant digits of resolution, and requires half the memory as IEEE4.

frequency domain

Frequency domain describes data graphed on an X-Y plot with frequency as the X axis. VSPECT vibrating wire data are in the frequency domain.

frequency response

Sample rate is how often an instrument reports a result at its output; frequency response is how well an instrument responds to fast fluctuations on its input. By way of example, sampling a large gage thermocouple at 1 kHz will give a high sample rate but does not ensure the measurement has a high frequency response. A fine-wire thermocouple, which changes output quickly with changes in temperature, is more likely to have a high frequency response.

FTP

File Transfer Protocol. A TCP/IP application protocol.

full-duplex

A serial communication protocol. Simultaneous bi-directional communications. Communications between a serial port and a computer is typically full duplex.

G

garbage

The refuse of the data communication world. When data are sent or received incorrectly (there are numerous reasons why this happens), a string of invalid, meaningless characters (garbage) often results. Two common causes are: 1) a baud-rate mismatch and 2) synchronous data being sent to an asynchronous device and vice versa.

global variable

A variable available for use throughout a CRBasic program. The term is usually used in connection with subroutines, differentiating global variables (those declared using Public or Dim) from local variables, which are declared in the Sub() and Function() instructions.

ground

Being or related to an electrical potential of 0 volts.

ground currents

Pulling power from the datalogger wiring panel, as is done when using some communication devices from other manufacturers, or a sensor that requires a lot of power, can cause voltage potential differences between points in datalogger circuitry that are supposed to be at ground or 0 Volts. This difference in potentials can cause errors when measuring single-ended analog voltages.

H

half-duplex

A serial communication protocol. Bi-directional, but not simultaneous, communications. SDI-12 is a half-duplex protocol.

handshake

The exchange of predetermined information between two devices to assure each that it is connected to the other. When not used as a clock line, the CLK/HS (pin 7) line in the datalogger CS I/O port is primarily used to detect the presence or absence of peripherals.

hello exchange

The process of verifying a node as a neighbor.

hertz

SI unit of frequency. Cycles or pulses per second.

HTML

Hypertext Markup Language. Programming language used for the creation of web pages.

HTTP

Hypertext Transfer Protocol. A TCP/IP application protocol.

hysteresis

The dependence of the state of the system on its history.

Hz

SI unit of frequency. Cycles or pulses per second.

I

I2C

Inter-Integrated Circuit is a multi-master, multi-slave, packet switched, single-ended, serial computer bus.

IEEE4

Four-byte, floating-point data type. IEEE Standard 754. Same format as Float.

Include file

A file containing CRBasic code to be included at the end of the current CRBasic program, or it can be run as the default program.

INF

A data word indicating the result of a function is infinite or undefined.

initiate comms

A name given to a processes by which the datalogger initiates communications with a computer running LoggerNet. Also known as Callback.

input/output instructions

Used to initiate measurements and store the results in input storage or to set or read control/logic ports.

instruction

Usually refers to a CRBasic command.

integer

A number written without a fractional or decimal component. 15 and 7956 are integers; 1.5 and 79.56 are not.

intermediate memory

SRAM memory automatically allocated for intermediate calculations performed by CRBasic data output processing instructions. Data output processing memory cannot be monitored.

IP

Internet Protocol. A TCP/IP internet protocol.

IP address

A unique address for a device on the internet.

IP trace

Function associated with IP data transmissions. IP trace information was originally accessed through the CRBasic instruction IPTrace() and stored in a string variable. Files Manager setting is now modified to allow for creation of a file on a datalogger memory drive, such as USB:, to store information in ring memory.

isolation

Hardwire communication devices and cables can serve as alternate paths to earth ground and entry points into the datalogger for electromagnetic noise. Alternate paths to ground and electromagnetic noise can cause measurement errors. Using opto-couplers in a connecting device allows communication signals to pass, but breaks alternate ground paths and may filter some electromagnetic noise. Campbell Scientific offers optically isolated RS-232 to CS I/O interfaces as an accessory for use on the CS I/O port.

J

JSON

Java Script Object Notation. A data file format available through the datalogger or LoggerNet.

K

keep memory

keep memory is non-volatile memory that preserves some settings during a power-up or program start up reset. Examples include PakBus address, station name, beacon intervals, neighbor lists, routing table, and communication timeouts.

keyboard/display

The datalogger has an optional external keyboard/display.

L

leaf node

A PakBus node at the end of a branch. When in this mode, the datalogger is not able to forward packets from one of its communication ports to another. It will not maintain a list of neighbors, but it still communicates with other PakBus dataloggers and wireless sensors. It cannot be used as a means of reaching (routing to) other dataloggers.

lf

Line feed. Often associated with carriage return (<cr>). <cr><lf>.

linearity

The quality of delivering identical sensitivity throughout the measurement.

local variable

A variable available for use only by the subroutine in which it is declared. The term differentiates local variables, which are declared in the Sub() and Function() instructions, from global variables, which are declared using Public or Dim.

LoggerLink

Mobile applications that allow a mobile device to communicate with IP, wi-fi, or Bluetooth enabled dataloggers.

LoggerNet

Campbell Scientific's datalogger support software for programming, communication, and data retrieval between dataloggers and a computer.

LONG

Data type used when declaring integers.

loop

A series of instructions in a CRBasic program that are repeated a the programmed number of times. The loop ends with an end instruction.

loop counter

Increments by one with each pass through a loop.

LSB

Least significant bit (the trailing bit).

LVDT

The linear variable differential transformer (LVDT) is a type of electrical transformer used for measuring linear displacement (position).

M

mains power

The national power grid.

manually initiated

Initiated by the user, usually with a Keyboard/Display, as opposed to occurring under program control.

mass storage device

A mass storage device may also be referred to as an auxiliary storage device. The term is commonly used to describe USB mass storage devices.

MD5 digest

16 byte checksum of the TCP/IP VTP configuration.

micro SD

A removable memory-card technology used in CR6 and CR1000X dataloggers.

milli

The SI prefix denoting 1/1000 of a base SI unit.

Modbus

Communication protocol published by Modicon in 1979 for use in programmable logic controllers (PLCs).

modem/terminal

Any device that has the following: ability to raise the ring line or be used with an optically isolated interface to raise the ring line and put the datalogger in the communication command state, or an asynchronous serial communication port that can be configured to communicate with the datalogger.

modulo divide

A math operation. Result equals the remainder after a division.

MSB

Most significant bit (the leading bit).

multimeter

An inexpensive and readily available device useful in troubleshooting data acquisition system faults.

multiplier

A term, often a parameter in a CRBasic measurement instruction, that designates the slope (aka, scaling factor or gain) in a linear function. For example, when converting °C to °F, the equation is $^{\circ}\text{F} = ^{\circ}\text{C} * 1.8 + 32$. The factor 1.8 is the multiplier.

mV

The SI abbreviation for millivolts.

N

NAN

Not a number. A data word indicating a measurement or processing error. Voltage over-range, SDI-12 sensor error, and undefined mathematical results can produce NAN.

neighbor device

Device in a PakBus network that communicate directly with a device without being routed through an intermediate device.

Network Planner

Campbell Scientific software designed to help set up dataloggers in PakBus networks so that they can communicate with each other and the LoggerNet server. For more information, see <https://www.campbellsci.com/loggernet>.

NIST

National Institute of Standards and Technology.

node

Devices in a network – usually a PakBus network. The communication server dials through, or communicates with, a node. Nodes are organized as a hierarchy with all nodes accessed by the same device (parent node) entered as child nodes. A node can be both a parent and a child.

NSEC

Eight-byte data type divided up as four bytes of seconds since 1990 and four bytes of nanoseconds into the second.

null-modem

A device, usually a multi-conductor cable, which converts an RS-232 port from DCE to DTE or from DTE to DCE.

Numeric Monitor

A digital monitor in datalogger support software or in a keyboard/display.

O

offset

A term, often a parameter in a CRBasic measurement instruction, that designates the y-intercept (aka, shifting factor or zeroing factor) in a linear function. For example, when converting °C to °F, the equation is °F = °C*1.8 + 32. The factor 32 is the offset.

ohm

The unit of resistance. Symbol is the Greek letter Omega (Ω). 1.0 Ω equals the ratio of 1.0 volt divided by 1.0 ampere.

Ohm's Law

Describes the relationship of current and resistance to voltage. Voltage equals the product of current and resistance ($V = I \cdot R$).

on-line data transfer

Routine transfer of data to a peripheral left on-site. Transfer is controlled by the program entered in the datalogger.

operating system

The operating system (also known as "firmware") is a set of instructions that controls the basic functions of the datalogger and enables the use of user written CRBasic programs. The operating system is preloaded into the datalogger at the factory but can be re-loaded or upgraded by you using Device Configuration Utility software. The most recent datalogger operating system .obj file is available at www.campbellsci.com/downloads.

output

A loosely applied term. Denotes a) the information carrier generated by an electronic sensor, b) the transfer of data from variable memory to final-data memory, or c) the transfer of electric power from the datalogger or a peripheral to another device.

output array

A string of data values output to final-data memory. Output occurs when the data table output trigger is True.

output interval

The interval between each write of a record to a final-storage memory data table.

output processing instructions

CRBasic instructions that process data values for eventual output to final-data memory. Examples of output-processing instructions include Totalize(), Maximize(), Minimize(), and Average(). Data sources for these instructions are values or strings in variable memory. The results of intermediate calculations are stored in data output processing memory to await the output trigger. The ultimate destination of data generated by data output processing instructions is usually final-storage memory, but the CRBasic program can be written to divert to variable memory by the CRBasic program for further processing. The transfer of processed summaries to final-data memory takes place when the Trigger argument in the DataTable() instruction is set to True.

output processing memory

SRAM memory automatically allocated for intermediate calculations performed by CRBasic data output processing instructions. Data output processing memory cannot be monitored.

P

PakBus

A proprietary communication protocol similar to IP protocol developed by Campbell Scientific to facilitate communications between Campbell Scientific instrumentation.

PakBusGraph

Software that shows the relationship of various nodes in a PakBus network and allows for monitoring and adjustment of some registers in each node. A PakBus node is typically a Campbell Scientific datalogger, a computer, or a communication device.

parameter

Part of a procedure (or command) definition.

PC200W

Basic datalogger support software for direct connect. It supports a connection between computer and datalogger and includes Short Cut for creating datalogger programs. Tools for setting the datalogger clock, sending programs, monitoring sensors, and on-site viewing and collection of data are also included.

PC400

Datalogger support software that supports a variety of communication options, manual data collection, and data monitoring displays. Short Cut and CRBasic Editor are included for creating datalogger programs. PC400 does not support complex communication options, such as phone-to-RF, PakBus® routing, or scheduled data collection.

period average

A measurement technique using a high-frequency digital clock to measure time differences between signal transitions. Sensors commonly measured with period average include water-content reflectometers.

peripheral

Any device designed for use with the datalogger. A peripheral requires the datalogger to operate. Peripherals include measurement, control, and data retrieval and communication modules.

PGIA

Programmable Gain Input Amplifier

ping

A software utility that attempts to contact another device in a network.

pipeline mode

A CRBasic program execution mode wherein instructions are evaluated in groups of like instructions, with a set group prioritization.

PLC

Programmable Logic Controllers

Poisson ratio

A ratio used in strain measurements.

ppm

Parts per million.

precision

The amount of agreement between repeated measurements of the same quantity (AKA repeatability).

PreserveVariables

CRBasic instruction that protects Public variables from being erased when a program is recompiled.

print device

Any device capable of receiving output over pin 6 (the PE line) in a receive-only mode. Printers, "dumb" terminals, and computers in a terminal mode fall in this category.

print peripheral

Any device capable of receiving output over pin 6 (the PE line) in a receive-only mode. Printers, "dumb" terminals, and computers in a terminal mode fall in this category.

processing instructions

CRBasic instructions used to further process input-data values and return the result to a variable where it can be accessed for output processing. Arithmetic and transcendental functions are included.

program control instructions

Modify the execution sequence of CRBasic instructions. Also used to set or clear flags.

Program Send command

Program Send is a feature of datalogger support software.

program statement

A complete program command construct confined to one command line or to multiple command lines merged with the line continuation characters <space><underscore> (_). A command line, even with line continuation, cannot exceed 512 characters.

public

A CRBasic command for declaring and dimensioning variables. Variables declared with Public can be monitored during datalogger operation.

pulse

An electrical signal characterized by a rapid increase in voltage followed by a short plateau and a rapid voltage decrease.

R

ratiometric

Describes a type of measurement or a type of math. Ratiometric usually refers to an aspect of resistive-bridge measurements - either the measurement or the math used to process it. Measuring ratios and using ratio math eliminates several sources of error from the end result.

record

A record is a complete line of data in a data table or data file. All data in a record share a common time stamp. Final-storage data tables are made up of records and fields. Each row in a table represents a record and each column represents a field. The number of fields in a record is determined by the number and configuration of output processing instructions that are included as part of the DataTable() declaration.

regulator

A setting, a Status table element, or a DataTableInformation table element. Also a device for conditioning an electrical power source. Campbell Scientific regulators typically condition ac or dc voltages greater than 16 Vdc to about 14 Vdc.

Reset Tables command

Reset Tables command resets data tables configured for fill and stop. Location of the command varies among datalogger support software according to the following: LoggerNet - Connect window Station Status tab|Table Fill Times tab|Reset Tables PC400 - command sequence: Datalogger|Station Status|Table Fill Times|Reset Tables RTDAQ - command sequence: Datalogger|Station Status|Table Fill Times|Reset Tables PC200W - command sequence: Datalogger|Station Status|Table Fill Times|Reset Tables.

resistance

A feature of an electronic circuit that impedes or redirects the flow of electrons through the circuit.

resistor

A device that provides a known quantity of resistance.

resolution

The smallest interval measurable.

ring line

Ring line is pulled high by an external device to notify the datalogger to commence RS-232 communications. Ring line is pin 3 of a DCE RS-232 port.

ring memory

A memory configuration that allows the oldest data to be overwritten with the newest data. This is the default setting for final-storage data tables.

ringing

Oscillation of sensor output (voltage or current) that occurs when sensor excitation causes parasitic capacitances and inductances to resonate.

RMS

Root-mean square, or quadratic mean. A measure of the magnitude of wave or other varying quantities around zero.

RNDIS

Remote Network Driver Interface Specification - a Microsoft protocol that provides a virtual Ethernet link via USB.

router

Device configured as a router is able to forward PakBus packets from one port to another. To perform its routing duties, a datalogger configured as a router maintains its own list of neighbors and sends this list to other routers in the PakBus network. It also obtains and receives neighbor lists from other routers.

RS-232

Recommended Standard 232. A loose standard defining how two computing devices can communicate with each other. The implementation of RS-232 in Campbell Scientific dataloggers to computer communications is quite rigid, but transparent to most users. Features in the datalogger that implement RS-232 communication with smart sensors are flexible.

RS-485

Recommended Standard 485. A standard defining how two computing devices can communicate with each other.

RTDAQ

Datalogger support software for industrial and real time applications.

RTU

Remote Telemetry Units

Rx

Receive

S

sample rate

The rate at which measurements are made by the datalogger. The measurement sample rate is of interest when considering the effect of time skew, or how close in time are a series of measurements, or how close a time stamp on a measurement is to the true time the phenomenon being measured occurred. A 'maximum sample rate' is the rate at which a measurement can repeatedly be made by a single CRBasic instruction. Sample rate is how often an instrument reports a result at its output; frequency response is how well an instrument responds to fast fluctuations on its input. By way of example, sampling a large gage thermocouple at 1 kHz will give a high sample rate but does not ensure the measurement has a high frequency response. A fine-wire thermocouple, which changes output quickly with changes in temperature, is more likely to have a high frequency response.

scan interval

The time interval between initiating each execution of a given Scan() of a CRBasic program. If the Scan() Interval is evenly divisible into 24 hours (86,400 seconds), it is synchronized with the 24 hour clock, so that the program is executed at midnight and every Scan() Interval thereafter. The program is executed for the first time at the first occurrence of the Scan() Interval after compilation. If the Scan() Interval does not divide evenly into 24 hours, execution will start on the first even second after compilation.

scan time

When time functions are run inside the Scan() / NextScan construct, time stamps are based on when the scan was started according to the datalogger clock. Resolution of scan time is equal to the length of the scan.

SDI-12

Serial Data Interface at 1200 baud. Communication protocol for transferring data between the datalogger and SDI-12 compatible smart sensors.

SDK

Software Development Kit

SDM

Synchronous Device for Measurement. A processor-based peripheral device or sensor that communicates with the datalogger via hardware over a short distance using a protocol proprietary to Campbell Scientific.

Seebeck effect

Induces microvolt level thermal electromotive forces (EMF) across junctions of dissimilar metals in the presence of temperature gradients. This is the principle behind thermocouple temperature measurement. It also causes small, correctable voltage offsets in datalogger measurement circuitry.

semaphore

(Measurement semaphore.) In sequential mode, when the main scan executes, it locks the resources associated with measurements. In other words, it acquires the measurement semaphore. This is at the scan level, so all subscans within the scan (whether they make measurements or not), will lock out measurements from slow sequences (including the auto self-calibration). Locking measurement resources at the scan level gives non-interrupted measurement execution of the main scan.

send button

Send button in datalogger support software. Sends a CRBasic program or operating system to a datalogger.

sequential mode

A CRBasic program execution mode wherein each statement is evaluated in the order it is listed in the program.

serial

A loose term denoting output of a series of ASCII, HEX, or binary characters or numbers in electronic form.

Settings Editor

An editor for observing and adjusting settings. Settings Editor is a feature of LoggerNet|Connect, PakBusGraph, and Device Configuration Utility.

Short Cut

A CRBasic programming wizard suitable for many datalogger applications. Knowledge of CRBasic is not required to use Short Cut.

SI

Système Internationale. The uniform international system of metric units. Specifies accepted units of measure.

signature

A number which is a function of the data and the sequence of data in memory. It is derived using an algorithm that assures a 99.998% probability that if either the data or the data sequence changes, the signature changes.

simplex

A serial communication protocol. One-direction data only. Serial communications between a serial sensor and the datalogger may be simplex.

single-ended

Denotes a sensor or measurement terminal wherein the analog voltage signal is carried on a single lead and measured with respect to ground (0 V).

skipped scans

Occur when the CRBasic program is too long for the scan interval. Skipped scans can cause errors in pulse measurements.

slow sequence

A usually slower secondary scan in the CRBasic program. The main scan has priority over a slow sequence.

SMTP

Simple Mail Transfer Protocol. A TCP/IP application protocol.

SNP

Snapshot file.

SP

Space.

SPI

Serial Peripheral Interface - a clocked synchronous interface, used for short distance communications, generally between embedded devices.

start bit

The bit used to indicate the beginning of data.

state

Whether a device is on or off.

Station Status command

A command available in most datalogger support software.

stop bit

The end of the data bits. The stop bit can be 1, 1.5, or 2.

string

A datum or variable consisting of alphanumeric characters.

support software

Campbell Scientific software that includes at least the following functions: datalogger communications, downloading programs, clock setting, and retrieval of measurement data.

synchronous

The transmission of data between a transmitting and a receiving device occurs as a series of zeros and ones. For the data to be "read" correctly, the receiving device must begin reading at the proper point in the series. In synchronous communication, this coordination is accomplished by synchronizing the transmitting and receiving devices to a common clock signal (see also asynchronous).

system time

When time functions are run outside the Scan() / NextScan construct, the time registered by the instruction will be based on the system clock, which has a 10 ms resolution.

T

table

Final-storage data tables are made up of records and fields. Each row in a table represents a record and each column represents a field. The number of fields in a record is determined by the number and configuration of output processing instructions that are included as part of the DataTable() declaration.

task

Grouping of CRBasic program instructions automatically by the datalogger compiler. Tasks include measurement, SDM or digital, and processing. Tasks are prioritized when the CRBasic program runs in pipeline mode. Also a user-customized function defined through LoggerNet Task Master.

TCP/IP

Transmission Control Protocol / Internet Protocol.

TCR

Temperature Coefficient of Resistance. TCR tells how much the resistance of a resistor changes as the temperature of the resistor changes. The unit of TCR is ppm/°C (parts-per-million per degree Celsius). A positive TCR means that resistance increases as temperature increases. For example, a resistor with a specification of 10 ppm/°C will not increase in resistance by more than 0.000010 Ω per ohm over a 1 °C increase of the resistor temperature or by more than .00010 Ω per ohm over a 10 °C increase.

Telnet

A software utility that attempts to contact and interrogate another specific device in a network. Telnet is resident in Windows OS.

terminal

Point at which a wire (or wires) connects to a wiring panel or connector. Wires are usually secured in terminals by screw- or lever-and-spring actuated gates with small screw- or spring-loaded clamps.

terminal emulator

A command-line shell that facilitates the issuance of low-level commands to a datalogger or some other compatible device. A terminal emulator is available in most datalogger support software available from Campbell Scientific.

thermistor

A thermistor is a temperature measurement device with a resistive element that changes in resistance with temperature. The change is wide, stable, and well characterized. The output of a thermistor is usually non-linear, so measurement requires linearization by means of a Steinhart-Hart or polynomial equation. CRBasic instructions Therm107(), Therm108(), and Therm109() use Steinhart-Hart equations.

throughput rate

Rate that a measurement can be taken, scaled to engineering units, and the stored in a final-memory data table. The datalogger has the ability to scan sensors at a rate exceeding the throughput rate. The primary factor determining throughput rate is the processing programmed into the CRBasic program. In sequential-mode operation, all processing called for by an instruction must be completed before moving on to the next instruction.

time domain

Time domain describes data graphed on an X-Y plot with time on the X axis. Time series data are in the time domain.

TLS

Transport Layer Security. An Internet communication security protocol.

toggle

To reverse the current power state.

TTL

Transistor-to-Transistor Logic. A serial protocol using 0 Vdc and 5 Vdc as logic signal levels.

Tx

Transmit

U

UART

Universal Asynchronous Receiver/Transmitter for asynchronous serial communication.

UINT2

Data type used for efficient storage of totalized pulse counts, port status (status of 16 ports stored in one variable, for example) or integer values that store binary flags.

unconditioned output

The fundamental output of a sensor, or the output of a sensor before scaling factors are applied.

UPS

Uninterruptible Power Supply. A UPS can be constructed for most datalogger applications using ac line power, an ac/ac or ac/dc wall adapter, a charge controller, and a rechargeable battery. The datalogger needs and external charge controller.

URI

Uniform Resource Identifier

URL

Uniform Resource Locator

user program

The CRBasic program written by you in Short Cut program wizard.

USR drive

A portion of memory dedicated to the storage of image or other files.

V

Vac

Volts alternating current.

variable

A packet of SRAM given an alphanumeric name. Variables reside in variable memory.

Vdc

Volts direct current.

VisualWeather

Datalogger support software specialized for weather and agricultural applications. The software allows you to initialize the setup, interrogate the station, display data, and generate reports from one or more weather stations.

volt meter

An inexpensive and readily available device useful in troubleshooting data acquisition system faults.

voltage divider

A circuit of resistors that ratiometrically divides voltage. For example, a simple two-resistor voltage divider can be used to divide a voltage in half. So, when fed through the voltage divider, 1 mV becomes 500 μ V, 10 mV becomes 5 mV, and so forth. Resistive-bridge circuits are voltage dividers.

volts

SI unit for electrical potential.

VSPECT®

Trademark for Campbell Scientific's proprietary spectral-analysis, frequency domain, vibrating wire measurement technique.

W

watchdog timer

An error-checking system that examines the processor state, software timers, and program-related counters when the CRBasic program is running. The following will cause watchdog timer resets, which reset the processor and CRBasic program execution: processor bombed, processor neglecting standard system updates, counters are outside the limits, voltage surges, and voltage transients. When a reset occurs, a counter is incremented in the WatchdogTimer entry of the Status table. A low number (1 to 10) of watchdog timer resets is of concern, but normally indicates that the situation should just be monitored. A large number of errors (>10) accumulating over a short period indicates a hardware or software problem. Consult with a Campbell Scientific support engineer.

weather-tight

Describes an instrumentation enclosure impenetrable by common environmental conditions. During extraordinary weather events, however, seals on the enclosure may be breached.

web API

Application Programming Interface

wild card

A character or expression that substitutes for any other character or expression.

Index

A	
accuracy factors.....	36, 71
auto self-calibration.....	50
B	
bandwidth.....	43
battery.....	54
voltage status.....	59
bridge resistance.....	32, 36
bridge strain.....	34
C	
C terminals.....	5, 8, 10
programmable logic control.....	10
calibrating.....	50
clock.....	61
collecting data.....	20
communication options.....	8, 12, 44
Ethernet.....	13, 14
internet.....	84
RS-232.....	12
SDI-12.....	46, 47, 48, 49
SDM.....	9
USB.....	12
communication ports.....	8, 10
CS I/O.....	9
serial.....	44
communication protocols.....	43
DNP3.....	45
Modbus.....	44
PakBus.....	46
TCP/IP.....	45
configuring communications.....	12
Ethernet.....	13, 14
USB.....	12
CPI port.....	9
CS I/O.....	9
csipasswd.....	52
current measurements.....	30
D	
data	
collecting.....	20
historic.....	20
monitoring.....	19, 21
viewing.....	20
data acquisition system.....	1
data records.....	21
data storage.....	23
data tables.....	21
example program.....	23
instructions.....	23
memory.....	63
datalogger clock.....	13, 15
dataloggers	
current status.....	59
maintenance.....	50
protection.....	53
resetting.....	59, 63
DataTableInfo.....	19, 21
desiccant.....	53
differential measurements.....	29, 66
digital I/O.....	7
DNP3.....	45
drives	
formatting.....	63
E	
earth ground.....	7
edge counting.....	40
edge timing.....	40
electronic noise.....	68
electrostatic discharges.....	56
enclosures.....	53
Ethernet.....	13, 14, 84
Ethernet LEDs.....	14
extra response time.....	13, 15
F	
file names.....	76
file systems.....	75
formatting drives.....	63
G	
G terminals.....	7
ground terminals.....	7, 65, 67, 72
H	
high frequency signal measurements.....	39
historic data.....	20
humidity protection.....	53
I	
I2C.....	49
INF.....	60
internal battery.....	54
internet communications.....	45, 84
L	
LEDs	
Ethernet.....	14

lightening protection.....	56
lithium battery	54
low-level ac measurements.....	38
LVTTTL.....	9

M

maintenance	50
max time online.....	13, 15
measurements	
0-20mA	30
4-20mA	30
current.....	30
differential	29, 66
high-frequency signal.....	39
low-level ac	38
period-averaging.....	37
pulse	37, 38, 39, 40, 41
resistance.....	32, 36
single-ended	29, 66
strain	34
thermocouple	36
voltage	29
memory	23
allocation.....	24
insufficient.....	76
resetting	63
micro SD.....	25
Modbus.....	44
moisture protection	53
mutli-drop network	9

N

NAN.....	60
noise rejection	16, 68

O

offset voltages	67, 71, 72
open collector	39
open input detection	68
OS updates.....	57

P

PakBus	46
PakBus encryption key	13, 15
password	51
percent-of-reading	71
period-averaging measurements	37
PLC	10
ports	
communication.....	8, 9
CPI	9
CS I/O	9
Ethernet	13, 84
RS-232.....	9, 12
USB.....	12
power.....	6

12V.....	6
5V.....	6
budgeting	57
ground.....	7
I/O	5
inputs.....	5
noise.....	68
output	6
supplies.....	5, 57, 64
USB.....	5

powerup.ini.....	26
process names.....	21
programmable logic control	10
programmed mode.....	46, 48
programs	
creating	16
errors.....	62
run options	18
running on power-up	26
sending.....	18
sending without computer	26
protection	53
Public table	19, 21
pulse measurements	10, 37, 38, 39, 40, 41
pulse width modulation.....	10

R

ratiometric-resistance measurements.....	36
recorder mode.....	46, 48
resetting	63
resistance measurements	32, 36
resolution	71
RS-232.....	9, 12
RS-485.....	9

S

SCADA.....	44
scheduling collections	16
SDI-12.....	8, 46, 47, 48, 49
SDM.....	9
SE terminals.....	5
security.....	51, 52
security code.....	13, 15
Seebeck effect	72
sending a program	15
sending OS	57
sending programs	18
sensors	2
serial.....	9
wiring diagram.....	16
serial communications.....	44
serial peripheral interface	49
settling errors	69
settling time.....	69
Short Cut.....	16

signal ground	7
signal settling	69
single-ended measurements	29, 66
sink limits	6
skipped records	59
skipped scans	59
source limits	6
spark-gap protection	56
specifications	77
SPI	49
station status	59
Status table	19, 21
strain measurements	34
SW12	6, 10
SW12 terminals	
programmable logic control	10
switch closure	39
switching noise	68
T	
tables	21
DataTableInfo	19, 21
Public	19, 21
Status	19, 21
TCP/IP	45
temperature measurements	36
terminals	
C	5, 10, 39, 40, 41
defined	3
digital I/O	7
ground	7, 65, 67, 72
P	39, 40, 41
pulse	37, 38

SE	5
VX	5
testing communication	15
thermocouple measurements	36
time	
keeping	61
skew	62
stamps	61, 62
TOA5	21
transparent mode	47
troubleshooting	59
TTL	9

U

updating OS	57
USB	5, 8, 12

V

variable out of bounds	59
View Pro	20
voltage measurements	29
improving	66
VX terminals	5
programmable logic control	10

W

watchdog error	59
wiring	3
power I/O	5
wiring diagram	
0-20mA devices	30
4-20mA devices	30
sensors	16
wiring panel	3