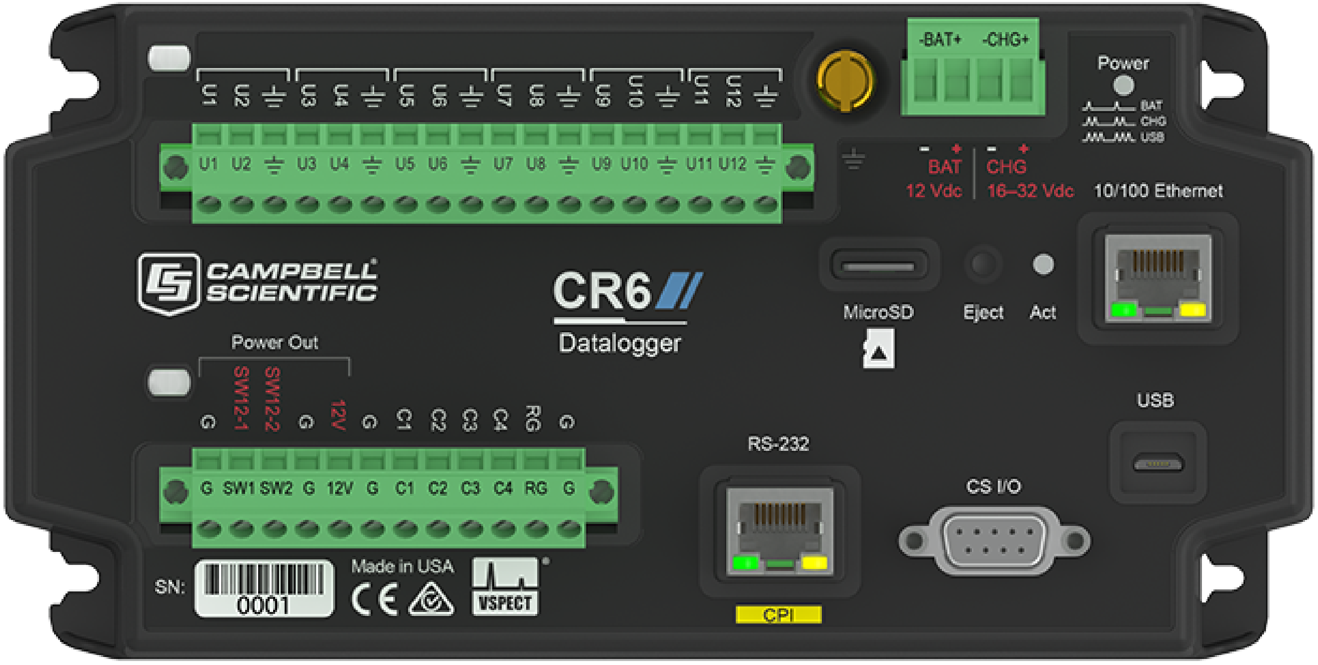
Product Manual

Datalogger

CR6

Measurement and Control Datalogger



Revision: 10/26/2018

RELIABLE SINCE 1974 MONITORING

CAMPBELL SCIENTIFIC®

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 110).
- Protect from ESD (see Electrostatic discharge and lightning protection on page 113).

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 (p. 111) 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 and Acknowledgements

The datalogger is warranted for three (3) years subject to this limited warranty: https://www.campbellsci.com/terms#warranty.

Acknowledgements

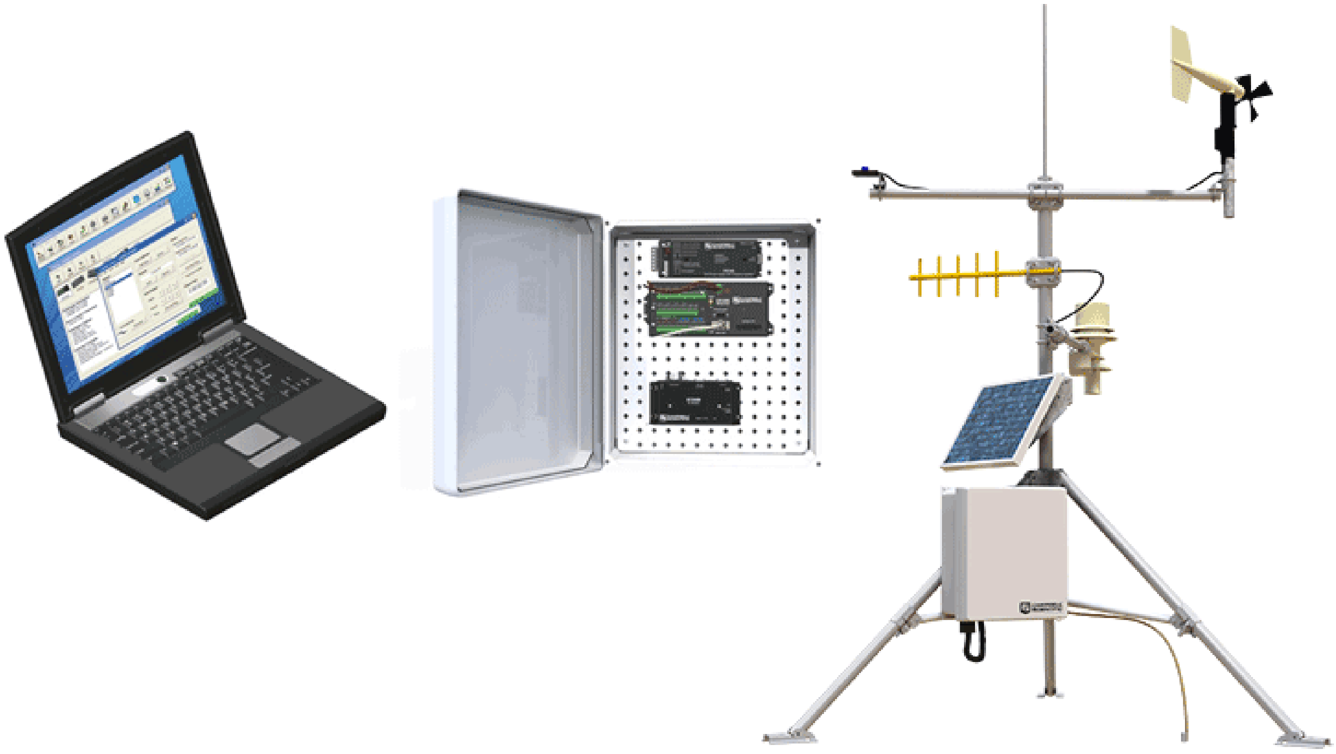
lwIP

Table of Contents

1. 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 components of a basic data acquisition system are shown 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 3 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 is stored in memory to await transfer to a computer by way of an external storage device or a communications link.

- **Data Retrieval and Communications** - Data is copied (not moved) from the datalogger, usually to a computer, by one or more methods using datalogger support software. Most communications 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 (p. 44).
- **Datalogger Support Software** - Software retrieves data, sends programs, and sets settings. The software manages the communications 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 (p. 15) 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.

1.1 The CR6 Datalogger

The CR6 datalogger provides fast communications, low power requirements, built-in USB, compact size, and high analog input accuracy and resolution. It includes universal (**U**) terminals, which allow connection to virtually any sensor - analog, digital, or smart. This multipurpose datalogger is also capable of doing static vibrating-wire measurements.

1.1.1 Overview

The CR6 datalogger is the main part of a data acquisition system (see Data acquisition system components 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 CR6 can simultaneously provide measurement and communications functions. Low power consumption allows the datalogger to operate for extended time on a battery recharged with a solar panel, eliminating the need for ac power. The CR6 temporarily suspends operations when primary power drops below 9.6 V, reducing the possibility of inaccurate measurements.

1.1.2 Communications Options

The CR6 can include Wi-Fi or the following radio options for different regions:

- RF407: 900 MHz (United States and Canada)
- RF412: 920 MHz (Australia and New Zealand)

- RF422: 868 MHz (Europe)
- RF451: 900 MHz, 1 Watt (United States, Canada, and Australia)

1.1.3 Operations

The CR6 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 is stored in onboard, nonvolatile memory. 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.

1.1.4 Programs

A program directs the datalogger on how and when sensors are measured, calculations are made, data is stored, and devices are controlled. The application program for the CR6 is written in CRBasic, a programming language that includes measurement, data processing, and analysis routines, as well as the standard BASIC instruction set. For simple applications, Short Cut, a user-friendly program generator, can be used to generate the program. For more demanding programs, use the full featured CRBasic Editor.

Programs are run by the CR6 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 CR6 determines the order of instruction execution to maximize efficiency.

1.2 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 is available at www.campbellsci.com/support to assist in measuring many sensor types.

- Analog
    - Voltage
    - Current
    - Strain
    - Thermocouple
    - Resistive bridge
- Pulse
    - High frequency
    - Switch-closure
    - Low-level ac
    - Quadrature
- Period average
- Vibrating wire
- Smart sensors
    - SDI-12
    - RS-232
    - Modbus
    - DNP3
    - TCP/IP
    - RS-485

2. Wiring panel and terminal functions

The CR6 wiring panel provides ports and removable terminals for connecting sensors, power, and communications 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
- Analog output
- Communications
- Digital I/O
- Power input
- Power output
- Power ground
- Signal ground

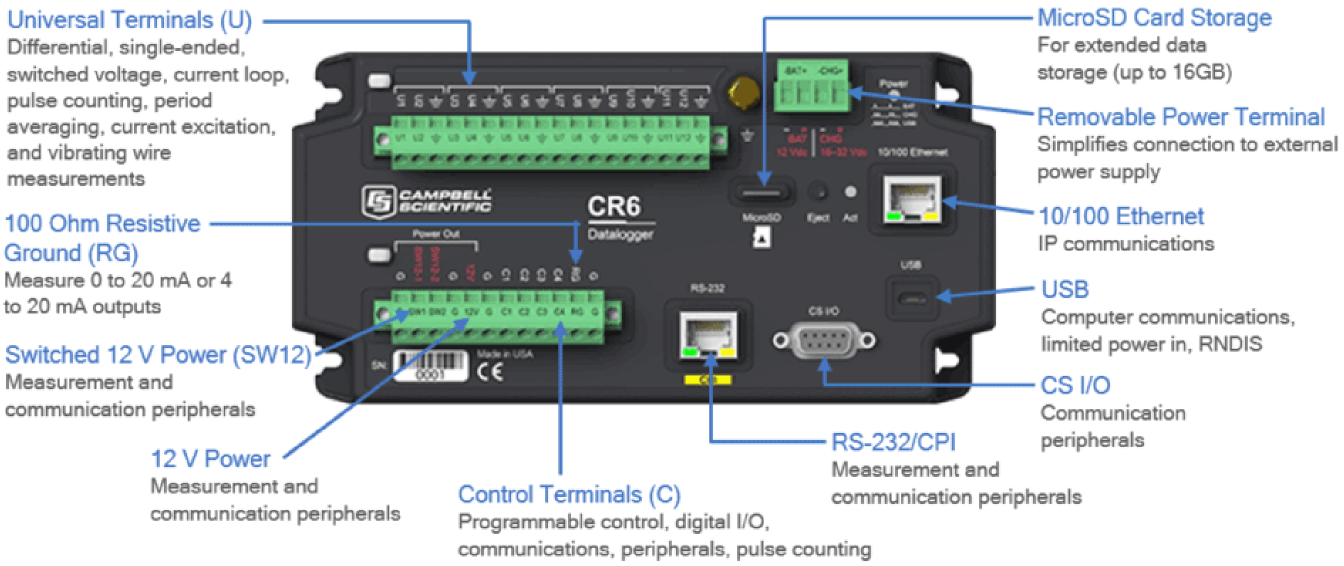

Universal Terminals (U)
Differential, single-ended, switched voltage, current loop, pulse counting, period averaging, current excitation, and vibrating wire measurements
100 Ohm Resistive Ground (RG)
Measure 0 to 20 mA or 4 to 20 mA outputs
Switched 12 V Power (SW12)
Measurement and communication peripherals
12 V Power
Measurement and communication peripherals
Control Terminals (C)
Programmable control, digital I/O, communications, peripherals, pulse counting
RS-232/CPI
Measurement and communication peripherals
CS I/O
Communication peripherals
USB
Computer communications, limited power in, RNDIS
10/100 Ethernet
IP communications
Removable Power Terminal
Simplifies connection to external power supply
MicroSD Card Storage
For extended data storage (up to 16GB)
CAMPBELL SCIENTIFIC
CR6
Datalogger

Table 2-1: Analog input terminal functions

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	RG
Single-Ended Voltage	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Differential Voltage	H	L	H	L	H	L	H	L	H	L	H	L	
Ratiometric/Bridge	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Vibrating Wire (Static, VSPECT®)	✓		✓		✓		✓		✓		✓		
Vibrating Wire with Thermistor	✓				✓				✓				
Thermistor	✓		✓		✓		✓		✓		✓		
Thermocouple	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Current Loop													✓
Period Average	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Table 2-2: Pulse counting terminal functions

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	C1-C4
Switch-Closure	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
High Frequency	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Low-level Ac		✓		✓		✓		✓		✓		✓	
Quadrature	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Table 2-3: Analog output terminal functions

	U1-U12
Switched Voltage Excitation	✓
Switched Current Excitation	✓

Table 2-4: Voltage output terminal functions

	U1-U12	C1-C4	12V	SW12-1	SW12-2
3.3 Vdc	✓	✓			
5 Vdc	✓	✓			
12 Vdc			✓	✓	✓

C and even numbered U terminals have limited drive capacity. Voltage levels are configured in pairs.

Table 2-5: Communications terminal functions

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	C1	C2	C3	C4	RS-232/ CPI
SDI-12	✓		✓		✓		✓		✓		✓		✓		✓		
GPS Time Sync													PPS	Rx	Tx	Rx	
TTL 0-5 V	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	
LVTTL 0-3.3 V	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	
RS-232													Tx	Rx	Tx	Rx	✓
RS-485 (Half Duplex)													A-	B+	A-	B+	

Table 2-5: Communications terminal functions

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	C1	C2	C3	C4	RS-232/CPI
RS-485 (Full Duplex)													Tx-	Tx+	Rx-	Rx+	
I2C	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	
SPI	MOSI	SCLK	MISO		MOSI	SCLK	MISO		MOSI	SCLK	MISO		MOSI	SCLK	MISO		
SDM	Data	Clk	Enabl		Data	Clk	Enabl		Data	Clk	Enabl		Data	Clk	Enabl		
CPI/ CDM																	✓

Table 2-6: Digital I/O terminal functions

	U1-U12	C1-C4
General I/O	✓	✓
Pulse-Width Modulation Output	✓	✓
Timer Input	✓	✓
Interrupt	✓	✓

2.1 Power input

The datalogger requires a power supply. It can receive power from a variety of sources, operate for several months on non-rechargeable batteries, and supply power to many sensors and devices. The datalogger operates with external power connected to the green **BAT** and/or **CHG** terminals on the face of the wiring panel. The positive power wire connects to +. The negative wire connects to -. The power terminals are internally protected against polarity reversal and high voltage transients.

In the field, the datalogger can be powered in any of the following ways:

- 10 to 18 Vdc applied to the **BAT** + and – terminals
- 16 to 32 Vdc applied to the **CHG** + and – terminals

To establish an uninterruptible power supply (UPS), connect the primary power source (often a transformer, power converter, or solar panel) to the **CHG** terminals and connect a nominal 12 Vdc sealed rechargeable battery to the **BAT** terminals. See Power budgeting (p. 114) for more information. The **Status Table ChargeState** may display any of the following:

- **No Charge** - The charger input voltage is either less than +9.82V±2% or there is no charger attached to the terminal block.
- **Low Charge Input** – The charger input voltage is less than the battery voltage.
- **Current Limited** – The charger input voltage is greater than the battery voltage AND the battery voltage is less than the optimal charge voltage. For example, on a cloudy day, a solar panel may not be providing as much current as the charger would like to use.
- **Float Charging** – The battery voltage is equal to the optimal charge voltage.
- **Regulator Fault** - The charging regulator is in a fault condition.

> WARNING:
> Sustained input voltages in excess of 32 Vdc on **CHG** or **BAT** terminals can damage the transient voltage suppression.

Ensure that power supply components match the specifications of the device to which they are connected. When connecting power, switch off the power supply, insert the connector, then turn the power supply on. See Troubleshooting power supplies (p. 127) for more information.

Following is a list of CR6 power input terminals and the respective power types supported.

- **BAT terminals**: Voltage input is 10 to 18 Vdc. This connection uses the least current since the internal datalogger charging circuit is bypassed. If the voltage on the **BAT** terminals exceeds 19 Vdc, power is shut off to certain parts of the datalogger to prevent damaging connected sensors or peripherals.
- **CHG terminals**: Voltage input range is 16 to 32 Vdc. Connect a primary power source, such as a solar panel or Vac-to-Vdc transformer, to **CHG**. The voltage applied to **CHG** terminals must be at least 0.3 V higher than that needed to charge a connected battery. When within the 16 to 32 Vdc range, it will be regulated to the optimal charge voltage for a lead acid battery at the current datalogger temperature, with a maximum voltage of approximately 15 Vdc. A battery need not be connected to the **BAT** terminals to supply power to the datalogger through the **CHG** terminals. The onboard charging regulator is designed for efficiently charging lead-acid batteries. It will not charge lithium or alkaline batteries.
- **USB port**: 5 Vdc via USB connection. If power is also provided with **BAT** or **CHG**, 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** and **SW12** terminals will not be operational. When powered by USB (no other power supplies connected) **Status** field **Battery** = **0**. Functions that will be

active with a 5 Vdc source include sending programs, adjusting datalogger settings, and making some measurements.

> NOTE:
> The **Status** field **Battery** value and the destination variable from the `Battery()` instruction (often called `batt_volt` or `BattV`) in the **Public** table reference the external battery voltage. For information about the internal battery, see Internal battery (p. 111).

2.1.1 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.

2.1.2 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 from **BAT**, program running.
- **2 flashes every 10 seconds**: Powered from **CHG**, program running.
- **3 flashes every 10 seconds**: Powered via USB, program running.
- **Always on**: Powered, no program running.

2.2 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 errors. Additionally, 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**: 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 minimize the error associated with the measurement of underpowered sensors.

- **SW12**: program-controlled, switched 12 Vdc terminals. It is 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. Therefore, 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. The connected peripheral or sensor may be damaged if it is not designed for that voltage level.

- **C** or **U** 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. Drive current varies between terminals. See also Digital input/output specifications (p. 228).
- **U** terminals: can be configured to provide regulated ±2500 mV dc excitation.

See also Power output specifications (p. 217).

2.3 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 (⏚)** - reference for single-ended analog inputs, excitation returns, and as a ground for sensor shield wires.
- **Power Ground (G)** - return for 3.3 V, 5 V, 12 V, **U** or **C** terminals configured for control, 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 100 Ω resistance to ground. Also used for measuring current (see Current-loop measurements on page 64 for more information).
- **Earth Ground Lug (⏚)** - connection point for heavy-gage earth-ground wire. A good earth connection is necessary to secure the ground potential of the datalogger and shunt transients away from electronics. Campbell Scientific recommends 14 AWG wire, minimum.

> 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 grounded. All components of the system (dataloggers, sensors, external power supplies, mounts, housings) should be referenced to one common earth 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 to verify 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.

See also:

- Minimizing ground loop errors (p. 128)
- Minimizing ground potential differences (p. 131)

2.4 Communications ports

The datalogger is equipped with ports that allow communications 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 communications ports include:

- CS I/O
- RS-232/CPI
- USB
- Ethernet
- **C** and **U** terminals

2.4.1 USB port

One USB port supports communicating with a computer through datalogger support software or through virtual Ethernet (RNDIS), and provides 5 Vdc power to the datalogger (powering through the USB port has limitations - details are available in the specifications). The datalogger USB port does not support USB flash or thumb drives. Although the USB connection supplies 5 V power, a 12 Vdc battery will be needed for field deployment.

2.4.2 Ethernet port

The RJ45 **10/100 Ethernet** port is used for IP communications.

2.4.3 C and U terminals for communications

C and **U** terminals are configurable for the following communications types:

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

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

2.4.3.1 SDI-12 ports

SDI-12 is a 1200 baud protocol that supports many smart sensors. **C1**, **C3**, **U1**, **U3**, **U5**, **U7**, **U9**, and **U11** can each be configured as SDI-12 ports. 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 communications (p. 100).

2.4.3.2 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
- Communications with other dataloggers or devices over long cables

Configure **C** or **U** terminals as serial ports using Device Configuration Utility or by using the `SerialOpen()` CRBasic instruction. **C** and **U** terminals are configured in pairs for TTL and LVTTL communications, and **C** terminals are configured in pairs for RS-232 or half-duplex RS-485. For full-duplex RS-485, all four **C** terminals are required. See also Communications (p. 87).

> NOTE:
> RS-232 ports are not isolated.

2.4.3.3 SDM port

SDM is a protocol proprietary to Campbell Scientific that supports several Campbell Scientific digital sensor and communications 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 **U1**, **U2**, and **U3**; **U5**, **U6**, and **U7**; or **U9**, **U10**, and **U11** can be configured together to be used as SDM ports by using the `SDMBeginPort()` instruction.

See also Communications specifications (p. 230).

2.4.4 CS I/O port

One nine-pin port, labeled **CS I/O**, is available for communicating with a computer through Campbell Scientific communications interfaces, modems, and peripherals. Campbell Scientific recommends keeping CS I/O cables short (maximum of a few feet). See also Communications specifications (p. 230).

2.4.5 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 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.

> NOTE:
> RS-232/CPI port is not isolated.

The CPI port 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.

2.5 Programmable logic control

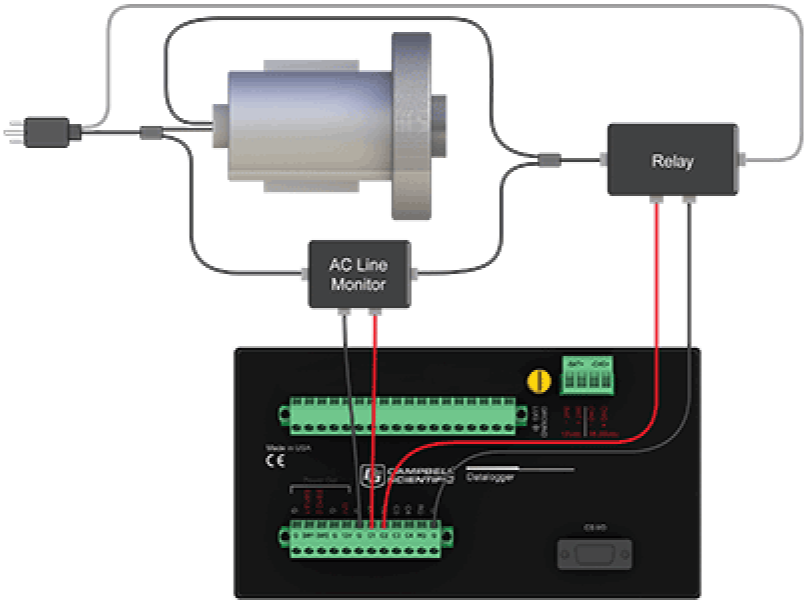
The datalogger can control instruments and devices such as:

- Controlling 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**, **U**, 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** and **U** terminals are selectable as binary inputs, control outputs, or communication ports. These 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 these terminals defaults to 5 V, but it can be changed to 3.3 V using the `PortPairConfig()` instruction. A **C** or **U** terminal configured for digital I/O is normally used to operate an external relay-driver circuit because the terminal itself has limited drive capacity.
- **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** or **U**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).



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
```

3. Setting up the datalogger

The basic steps for setting up your datalogger to take measurements and store data include:

1. Configuring your communications connection.
2. Testing communications (optional).
3. Connecting the datalogger to the computer.
4. Creating a program.
5. Sending that program to the datalogger.

3.1 Setting up communications with the datalogger

The first step in setting up and communicating with your datalogger is to configure your connection. Communications peripherals, dataloggers, and software must all be configured for communications. Additional information is found in your specific peripheral manual, and the datalogger support software manual and help.

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

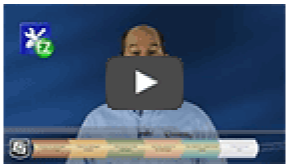
You can configure your connection using any of the following options. The simplest is via USB. For detailed instruction, see:

- USB or RS-232 communications (p. 18)
- Virtual Ethernet over USB (RNDIS) (p. 19)
- Ethernet communications (p. 20)
- Wi-Fi communications (p. 23) (WIFI models only)
- Radio communications (p. 27) (RF models only)

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

3.1.1 USB or RS-232 communications

Setting up a USB or RS-232 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 communications between your computer and the datalogger. Use the following instructions or watch a video.



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 **Add Datalogger** .
   - LoggerNet users, click **Setup** , click the **View** menu to ensure you are in the **EZ (Simplified)** view, then click **Add Datalogger**.
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 CR6 via USB, click **Install USB Driver**, 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 5 V power as well as a communications 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 to connect to the computer.

   > 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 communications 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. The default PakBus address is **1**. A USB connection does not require a baud rate selection. RS-232 connections default to 115200 baud.

   > **NOTE:**
   > Unlike the RS-232 port on some other Campbell Scientific dataloggers that autobaud, the CR300 RS-232 port does not. If the hardware and software settings for baud rate and PakBus address do not match, you will not be able to connect.

10. 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.
11. 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.
12. Click **Next**.
13. 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 (p. 106) for more information.
14. Click **Next**.
15. Review the **Communication Setup Summary**. If you need to make changes, click **Previous** to return to a previous window and change the settings.

Setup is now complete, and the EZSetup Wizard allows to you click **Finish** or click **Next** to test communications, set the datalogger clock, and send a program to the datalogger. See Testing communications and completing EZ Setup (p. 40) for more information.

3.1.2 Virtual Ethernet over USB (RNDIS)

CR6 series dataloggers with OS version 7 or greater support RNDIS (virtual Ethernet over USB). This allows the datalogger to communicate via TCP/IP over USB. Watch a video or use the following instructions.

3.1.2.1 Connecting to your datalogger via RNDIS

1. Supply power to the datalogger. If connecting via USB for the first time, you must first install USB drivers by using Device Configuration Utility (select your datalogger, then on the main page, click **Install USB Driver**). Alternately, you can install the USB drivers using EZ Setup. A USB connection supplies 5 V power (as well as a communication link), which is adequate for setup, but a 12 V battery will be needed for field deployment.

   > NOTE:
   > Ensure the datalogger is connected directly to the computer USB port (not to a USB hub). We recommended always using the same USB port on your computer.

2. Physically connect your datalogger to your computer using a USB cable, then open Device Configuration Utility and select your datalogger.
3. Retrieve your IP Address. On the bottom, left side of the screen, select **Use IP Connection**, then click the browse button next to the **Communication Port** box. Note the IP Address (default is **192.168.66.1**). If you have multiple dataloggers in your network, more than one datalogger may be returned. Ensure you select the correct datalogger by verifying the datalogger serial number or station name (if assigned).
4. A virtual IP address can be used to connect to the datalogger using Device Configuration Utility or other computer software, or to view the datalogger internal web page in a browser. To view the web page, open a browser and enter www.linktodevice.com or the IP address you retrieved in the previous step (for example, **192.168.66.1**) into the address bar.

To secure your datalogger from others who have access to your network, we recommend that you set security and establish access permissions using `.csipasswd`. For more information, see Creating a .csipasswd file (p. 109).

3.1.3 Ethernet communications

The CR6 offers a 10/100 Ethernet connection. Use Device Configuration Utility to enter the datalogger IP Address, Subnet Mask, and IP Gateway address. After this, use the EZSetup Wizard to set up communications with the datalogger. If you already have the datalogger IP information, you can skip these steps and go directly to Setting up Ethernet communications between the datalogger and computer (p. 22).

3.1.3.1 Configuring datalogger Ethernet settings

1. Supply power to the datalogger. If connecting via USB for the first time, you must first install USB drivers by using Device Configuration Utility (select your datalogger, then on the main page, click **Install USB Driver**). Alternately, you can install the USB drivers using EZ

Setup. A USB connection supplies 5 V power (as well as a communication link), which is adequate for setup, but a 12 V battery will be needed for field deployment.

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 (p. 21).
3. Using datalogger support software (LoggerNet, PC400, or PC200W), open Device Configuration Utility ().
4. Select the **CR6 Series** datalogger from the list
5. Select the port assigned to the datalogger from the **Communication Port** list. If connecting via Ethernet, select **Use IP Connection**.
6. 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 (p. 106) for more information.
7. Click **Connect**.
8. On the **Deployment** tab, click the **Ethernet** subtab.
9. 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 **Always On**, **1 Minute**, or **Disable**.
10. 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 the default, 0.0.0.0, 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.
11. **Apply** to save your changes.

3.1.3.2 Ethernet LEDs

When the datalogger is powered, and **Ethernet Power** setting is not disabled, the **10/100 Ethernet** LEDs will show the 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.

3.1.3.3 Setting up Ethernet communications between the datalogger and computer

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

This procedure only needs to followed once per datalogger. However, these settings can be revised using the datalogger support software **Edit Datalogger Setup** option ().

1. Using datalogger support software, open **EZSetup**.
   - PC400 users, click the **Add Datalogger** button ().
   - LoggerNet users, select **Setup** () from the **Main** category on the toolbar, click the **View** menu to ensure you are in the **EZ (Simplified)** view, then click the **Add Datalogger** button.

   > **NOTE:**
   > PC200W does not support IP connections.
2. Click **Next**.
3. Select the **CR6 Series** from the list, enter a name for your datalogger (for example, a site or project name), **Next**.
4. Select the **IP Port** connection type and click **Next**.
5. Type the datalogger IP address followed by a colon, then the port number of the datalogger in the **Internet IP Address** box (these were set up through the Ethernet communications (p. 20)) step. 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*

6. 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 communications 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, communications 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. **Next**.
7. 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 (p. 106) for more information. **Next**.
8. Review the **Communication Setup Summary**. If you need to make changes, click **Previous** to return to a previous window and change the settings.

Setup is now complete, and the EZSetup Wizard allows you **Finish** or select **Next**. The **Next** steps take you through testing communications, setting the datalogger clock, and sending a program to the datalogger. See Testing communications and completing EZ Setup (p. 40) for more information.

3.1.4 Wi-Fi communications

By default, CR6-WIFI dataloggers are configured to host a Wi-Fi network. The LoggerLink mobile app for iOS and Android can be used to connect with a CR6-WIFI. Up to eight devices can connect to a network created by a CR6. The setup follows the same steps shown in this video: CR6-WIFI Datalogger - Setting Up a Network.

> NOTE:
> The user is responsible for emissions if changing the antenna type or increasing the gain.

See also Communications specifications (p. 230).

3.1.4.1 Configuring the datalogger to host a Wi-Fi network

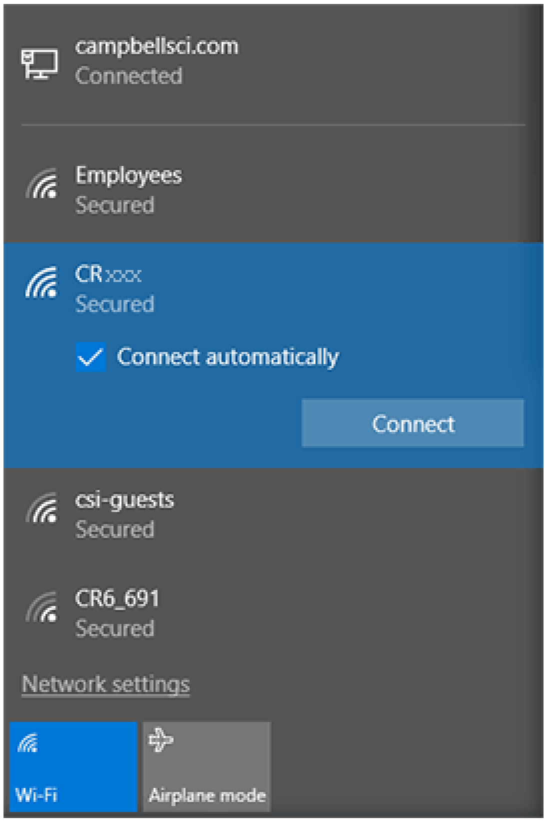
By default, CR6-WIFI dataloggers are configured to host a Wi-Fi network. If the settings have changed, you can follow these instructions to reconfigure the datalogger:

1. Ensure your CR6-WIFI is connected to an antenna and power.
2. Using Device Configuration Utility, connect to the datalogger.

3. On the **Deployment** tab, click the **Wi-Fi** sub-tab.
4. In the **Configuration** list, select the **Create a Network** option.
5. Optionally, set security on the network to prevent unauthorized access by typing a password in the **Password** box (recommended).
6. Apply your changes.

3.1.4.2 Connecting your computer to the datalogger over Wi-Fi

1. Open the Wi-Fi network settings on your computer.



2. Select the Wi-Fi-network hosted by the datalogger. The default name is **CR6** followed by the serial number of the datalogger. In the previous image, the Wi-Fi network is **CRxxx**.
3. If you set a password, select the **Connect Using a Security Key** option (instead of a PIN) and type the password you chose.
4. Connect to this network.

3.1.4.3 Setting up Wi-Fi communications between the datalogger and the datalogger support software

1. Using LoggerNet or PC400, click **Add Datalogger** to launch the EZSetup Wizard. For LoggerNet users, you must first click **Setup**, then **View** menu to ensure you are in the **EZ (Simplified)** view, then click **Add Datalogger** .

   > NOTE:
   > PC200W does not support IP connections.

2. Select the **IP Port** connection type and click **Next**.
3. In the **Internet IP Address** field, type **`192.168.67.1`**. This is the default datalogger IP address created when the CR6-WIFI creates a network.
4. Click **Next**.
5. 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. This can usually be left at **00 seconds**.
   - You 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.
6. Click **Next**.
7. By default, the datalogger does not use a security code or a PakBus encryption key. Therefore, the **Security Code** can be left at **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 (p. 106) for more information.
8. Click **Next**.
9. 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 communications, set the datalogger clock, and send a program to the datalogger. See Testing communications and completing EZ Setup (p. 40) for more information.

3.1.4.4 Configuring dataloggers to join a Wi-Fi network

By default, the CR6-WIFI dataloggers are configured to host a Wi-Fi network. To set them up to join a network:

1. Ensure your CR6-WIFI is connected to an antenna and power.
2. Using Device Configuration Utility, connect to the datalogger.
3. On the **Deployment** tab, click the **Wi-Fi** sub-tab.
4. In the **Configuration** list, select the **Join a Network** option.
5. Next to the **Network Name (SSID)** box, click **Browse** [...] to search for and select a Wi-Fi network.
6. If the network is a secured network, you must enter the password in the **Password** box and add any additional security in the **Enterprise** section of the window.
7. Enter the **IP Address, Network Mask**, and **Gateway**. These values should be provided by your network administrator. A static IP address is recommended.
    - Alternatively, you can use an IP address assigned to the datalogger via DHCP. To do this, make sure the IP Address is set to `0.0.0.0`. Click **Apply** to save the configuration changes. Then reconnect. The IP information obtained through DHCP is updated and displayed in the **Status** section of the **Wi-Fi** subtab. 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.
8. Apply your changes.
9. For each datalogger you want to connect to network, you must follow the instruction in Setting up Wi-Fi communications between the datalogger and the datalogger support software (p. 25), using the IP address used to configure that datalogger (step 7 in this instruction).

3.1.4.5 Wi-Fi LED indicator

When the datalogger is powered, the Wi-Fi LED will turn on according to Wi-Fi communication states:

- **Off**: Insufficient power, Wi-Fi disabled, or datalogger failed to join or create a network (periodic retries will occur).
- **Solid for 2 seconds**: Attempting to join or create a network.

- **Flashing**: Successfully joined or created a network. Flashes with network activity and once every four seconds.

3.1.5 Radio communications

CR6-RF dataloggers include radio options. The RF407-series frequency-hopping spread-spectrum (FHSS) radio options include the RF407, RF412, RF422, and RF427. The RF451, another radio option, has a 902-to-928 MHz operating-frequency range typically used for long-range communications. RF407-series are designed for license-free use in several countries:

- The RF407 option has a 902 to 928 MHz operating-frequency range appropriate for use in the United States and Canada (FCC / IC compliant).
- The RF412 option has a 915 to 928 MHz operating-frequency range appropriate for use in Australia and New Zealand (ACMA compliant).
- The RF422 option has an 863 to 873 MHz operating-frequency range appropriate for use in most of Europe and some of Asia (ETSI compliant).
- The RF427 option has a 902 to 907.5 MHz/915 to 928 MHz operating-frequency range appropriate for use in Brazil.
- The RF451 option has a 902 to 928 MHz operating-frequency range appropriate for use in the United States and Canada (FCC / IC compliant). This radio option is typically used for long-range communications.

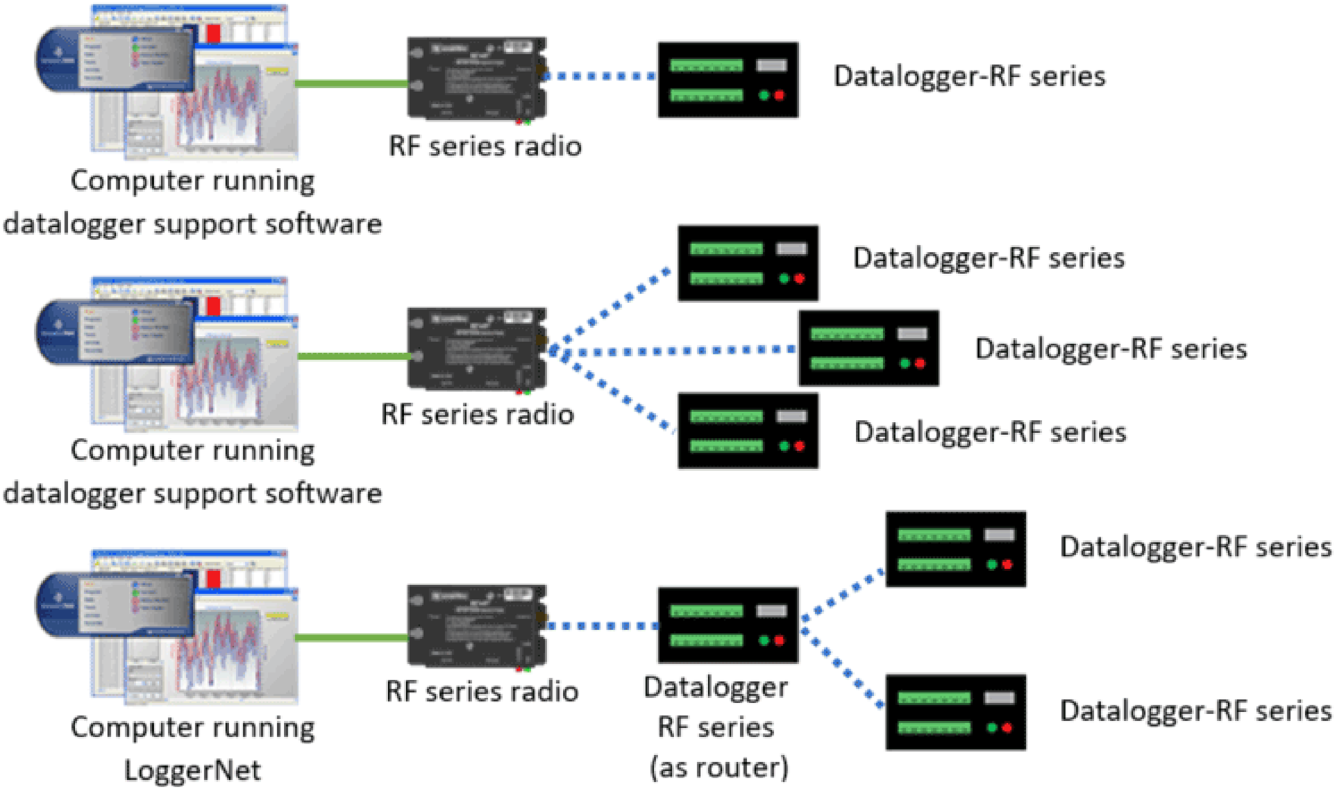
> NOTE:
> This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his or her own expense.

Radio options cannot be mixed within a network. An RF407 can only be used with other RF407-type radios, an RF412 can only be used with other RF412-type radios, an RF422 can only be used with other RF422-type radios, RF427 can only be used with other RF427-type radios, and an RF451 can only be used with other RF451-type radios.

Throughout these instructions, RF407-series represents each of the RF407, RF412, RF422, and RF427 radio options, unless otherwise noted. Similarly, the RF407-series standalone, or independent radio represents each of the RF407, RF412, RF422, and RF427 models, unless otherwise noted.

3.1.5.1 Configuration options

The most frequently used configurations with the RF-series datalogger and RF-series radio include the following:



For detailed instruction, see:

- RF407-Series radio communications with one or more dataloggers (p. 29)
- RF407-Series radio communications with multiple dataloggers using one datalogger as a router (p. 31)
- RF451 radio communications with one or more dataloggers (p. 34)
- RF451 radio communications with multiple dataloggers using one datalogger as a repeater (p. 37)

See also RF radio option specifications (p. 233).

3.1.5.2 RF407-Series radio communications with one or more dataloggers

To configure an RF407-series radio to communicate with the datalogger, you must complete the following steps (instruction follows):

- Ensure your datalogger and RF407-series radio are connected to an antenna and power.
- Configure the connection to the RF407-series device using Device Configuration Utility.
- If you are connecting to multiple dataloggers, you will have to assign unique PakBus addresses to each datalogger using Device Configuration Utility. (Connect to each datalogger, set the **PakBus Address** on the **Deployment** | **Datalogger** tab.)
- Use datalogger support software to set up communications between the RF407-series radio and the dataloggers.

> NOTE:
> This procedure assumes the RF407 series devices are using factory default settings.

Configuring the RF407-Series radio

Configure the RF407-Series radio connected to the computer (see image in Configuration options (p. 28) for reference).

1. Ensure your RF407-series radio is connected to an antenna and power.
2. If connecting via USB for the first time, you must first install USB drivers using Device Configuration Utility (select your radio, then on the main page, click **Install USB Driver**). Plug the RF407-series radio to your computer using a USB or RS-232 cable.
3. Using Device Configuration Utility, select the **Communication Port** used for your radio and connect to the RF407-series radio.
4. On the **Main** tab, set the **Active Interface** to **USB** or **RS-232** (depending on how your computer will be connected to the RF407-series radio).
5. Apply the changes.
6. Connect the RF407-Series radio to the computer communication port selected in the previous step.

Setting up communications between the RF407-Series datalogger and the computer

These instructions provide an easy way to set up communications between the RF407-series datalogger and the computer connected to the RF407-series radio (as configured in previous instructions). Follow these instructions multiple times to set up multiple dataloggers. In this case, each datalogger must be given a unique PakBus address (see PakBus communications on page 99 for more information). For more complicated networks, it is recommended that you use Network Planner.

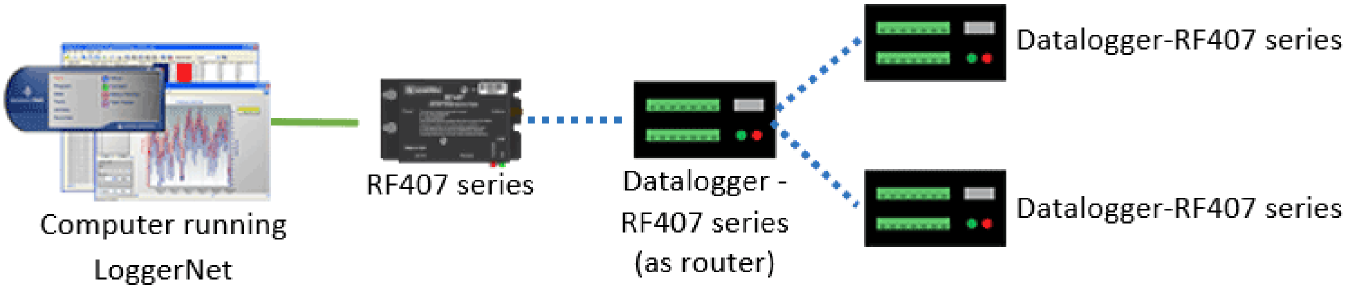
1. Supply 12 Vdc power to the datalogger.
2. Ensure the datalogger antenna is connected.
3. Using datalogger support software, launch the EZSetup Wizard and add the datalogger.
    - PC200W and PC400 users, click **Add Datalogger** .
    - LoggerNet users, click **Setup** , click the **View** menu to ensure you are in the **EZ (Simplified)** view, then click **Add Datalogger** .
4. Click **Next**.
5. Select the **CR6Series** datalogger from the list, type a name for your datalogger (for example, a site or project name), and click **Next**.
6. If prompted, select the **Direct Connect** connection type and click **Next**.
7. Select the communication port used to communicate with the RF407-series radio from the **COM Port** list. (Note that the RF407-series radio to RF407-series datalogger link is not indicated in the LoggerNet Setup Standard View.)
8. Accept the default value of **00 seconds** in the **COM Port Communication Delay** - this box is used to allow time for hardware devices to "wake up" and negotiate a communications link. Click **Next**.
9. In the previous instruction "Configuring a Connection to an RF407-Series Radio," you were asked to select an active interface option of USB or RS-232. If you selected USB as the active interface for the radio, you do not need to select a baud rate. If you selected RS-232, set the baud rate to the one chosen during that step. The radio's default baud rate is 115200. The PakBus address must match the hardware settings for your datalogger. The default **PakBus Address** is **1**.
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 left at **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 (p. 106) 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 communications, set the datalogger clock, and send a program to the datalogger. See Testing communications and completing EZ Setup (p. 40) for more information.

If you experience network communications problems, see Troubleshooting Radio Communications (p. 125) for assistance.

3.1.5.3 RF407-Series radio communications with multiple dataloggers using one datalogger as a router

This type of network configuration is useful for communicating around an obstacle, such as a hill or building, or to reach longer distances.


Computer running LoggerNet
RF407 series
Datalogger - RF407 series (as router)
Datalogger-RF407 series
Datalogger-RF407 series

To configure an RF407-series radio to communicate with multiple dataloggers through a router, you must complete the following steps (instruction follows):

- Ensure your dataloggers and RF407-series radios are each connected to an antenna and power.
- Configure your connection to the RF407-series devices using Device Configuration Utility.
- Assign unique PakBus addresses to each datalogger using Device Configuration Utility. (Connect to each datalogger, and set the **PakBus Address** on the **Deployment** | **Datalogger** tab.)
- Configure the datalogger acting as a router.
- Use datalogger support software to set up communications between the computer and the dataloggers.

Configuring the RF407-Series radio

Configure the RF407-Series radio connected to the computer (see previous image for reference).

1. Ensure your RF407-series radio is connected to an antenna and power.
2. If connecting via USB for the first time, you must first install USB drivers using Device Configuration Utility (select your radio, then on the main page, click **Install USB Driver**). Plug the RF407-series radio to your computer using a USB or RS-232 cable.

3. Using Device Configuration Utility, select the **Communication Port** used for your radio and connect to the RF407-series radio.
4. On the **Main** tab, set the **Active Interface** to **USB** or **RS-232** (depending on how your computer will be connected to the RF407-series radio).
5. Apply the changes.
6. Connect the RF407-Series radio to the computer communication port selected in the previous step.

Configuring the datalogger acting as a router

1. Supply power to the datalogger. If connecting via USB for the first time, you must first install USB drivers by using Device Configuration Utility (select your datalogger, then on the main page, click **Install USB Driver**). Alternately, you can install the USB drivers using EZ Setup. A USB connection supplies 5 V power (as well as a communication link), which is adequate for setup, but a 12 V battery will be needed for field deployment. Ensure the datalogger antenna is connected.
2. Using Device Configuration Utility, connect to the RF407-series datalogger that will serve as a router.
3. On the **Deployment|Datlogger** tab, assign a unique PakBus Address (see PakBus communications on page 99 for more information).
4. On the **Deployment** tab, click the **Com Ports Settings** sub-tab.
5. From the **Select the ComPort** list, select **RF**.
6. Set the **Beacon Interval** to **60** seconds (or the amount of time you are willing to wait for the leaf dataloggers in the network to be discovered).

> NOTE:
> A beacon is a packet broadcast at a specified interval intended to discover neighbor devices.

7. Set the **Verify Interval** to something slightly greater than the expected communications interval between the router and the other (leaf) dataloggers in the network (for example, 90 seconds).
8. Click the **Advanced** sub-tab and set **Is Router** to **True**.
9. Apply your changes.

Adding routing datalogger to LoggerNet network

1. Using LoggerNet, click **Setup** and click the **View** menu to ensure you are in the **Standard** view.

2. Click **Add Root** .
3. Click **ComPort**, then **PakBusPort (PakBus Loggers)**, then **CR6Series**.
4. Click **Close**.
5. In the Entire Network pane on the left side of the window, select the **ComPort**.
6. On the **Hardware** tab on the right, click the **ComPort Connection** list and select the communication port assigned to the RF407-series radio.
7. In the Entire Network pane on the left side of the window, select **PakBusPort**.
8. On the **Hardware** tab on the right, select the **PakBus Port Always Open** check box.
   - If you would like to prevent the possibility of LoggerNet communicating with any other dataloggers in the network without going through the router, set the **Beacon Interval** to **00 h 00 m 00s**.
9. In the Entire Network pane on the left side of the window, select the router datalogger (**CR6Series**) from the list.
10. On the **Hardware** tab on the right, type the **PakBus Address** you assigned to the router datalogger in Device Configuration Utility.
11. Optionally, click the **Rename** button ( ) to provide the datalogger a descriptive name.
12. Apply your changes.

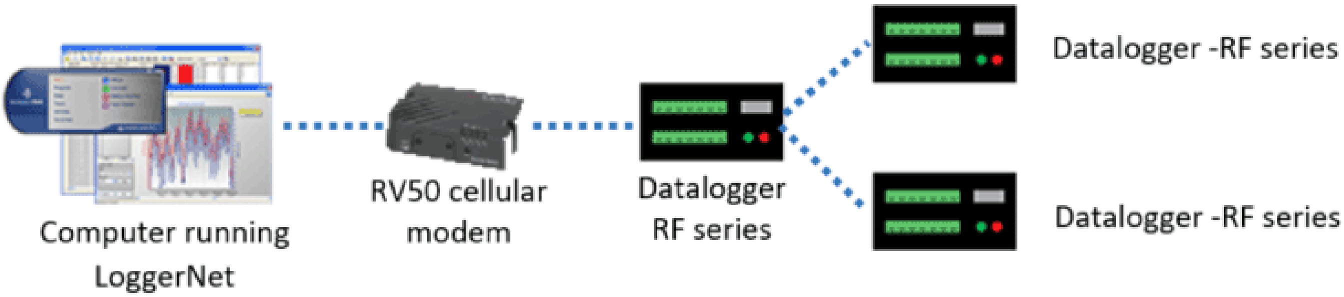
Adding leaf dataloggers to the network

1. In the LoggerNet **Standard Setup** view (click the **Setup** ( ) option and click the **View** menu to ensure you are in the **Standard** view), right-click on the router datalogger in the Entire Network pane on the left side of the window and select **CR6Series**.
2. With the newly added datalogger selected in the **Entire Network** pane, set the **PakBus Address** to the address that was assigned to the leaf datalogger in Device Configuration Utility.
3. Click **Rename**. Enter a descriptive name for the datalogger.
4. Apply your changes.
5. Repeat these steps for each leaf datalogger in the network.

If you experience problems with network communications, see Troubleshooting Radio Communications (p. 125) for assistance.

Using additional communications methods

Using similar instructions, a RF407-series datalogger can be used in a system with additional communication methods. For example, in the following image, the router RF407-series datalogger communicates with LoggerNet through an RV50 cellular modem connected to RF407-series datalogger using the **RS-232** port. The router RF407-series datalogger communicates with the leaf RF407-series dataloggers over RF.



3.1.5.4 RF451 radio communications with one or more dataloggers

To configure an RF451 radio to communicate with the datalogger, you must complete the following steps (detailed instruction follows):

- Ensure your datalogger and RF451 radio are connected to an antenna and power.
- Configure the RF451 radio that will be connected to the computer using Device Configuration Utility.
- If you are connecting to multiple dataloggers, you will have to assign unique PakBus addresses to each datalogger using Device Configuration Utility. (Connect to each datalogger, set the **PakBus Address** on the **Deployment** | **Datalogger** tab.)
- Use datalogger support software to set up communications between the computer with the RF451 radio and the dataloggers.

Configuring the RF451 radio connected to the computer

Configure the RF451 radio connected to the computer (see image in Configuration options (p. 28) for reference).

1. Ensure your RF451 radio is connected to an antenna and power.
2. If connecting via USB for the first time, you must first install USB drivers using Device Configuration Utility (select your radio, then on the main page, click **Install USB Driver**). Plug the RF451 radio to your computer using a USB cable.
3. Using Device Configuration Utility, connect to the RF451 radio.

4. On the **Deployment** tab, set the **Active Interface** to **USB** or **RS-232** (depending on how your computer will be connected to the radio).
5. Set the **Radio Operation Mode** to **Multi-Point Master**.
6. In the **Network ID** field, type a unique number between 0 and 4095 (excluding 255). This is used to communicate with RF451 devices in the network. Make note of this number. All radios in the network require the same Network ID.
7. Select a **Frequency Key** between 0 and 14 and make note of this number. Generally all radios in the network will have the same Frequency Key.
8. Apply the changes.
9. Connect the RF451 radio to the computer communication port selected in the previous step.

Configuring slave RF451 dataloggers

Follow these instructions multiple times to set up multiple dataloggers. In this case, each must be given a unique name and PakBus address. For more complicated networks, it is recommended that you use Network Planner.

1. Ensure your RF451 datalogger is connected to an antenna and power.
2. Using Device Configuration Utility, connect to the RF451 datalogger.
3. On the **Deployment | Datalogger** tab, assign a unique PakBus address.
4. On the **Deployment | RF451** tab, set the **Radio Operation Mode** to **Point to Multi-Point Slave**.
5. In the **Network ID** field, type the same number set in the previous instruction: "Configuring a Connection to an RF451 Radio."
6. Select the same **Frequency Key** set in the previous instruction: "Configuring a Connection to an RF451 Radio."
7. Apply the changes.

Setting up communications between the RF451 datalogger and the computer

These instructions provide an easy way to set up communications between the RF451 datalogger and the computer with an RF451 radio. Follow these instructions multiple times to set up multiple dataloggers. For more complicated networks, it is recommended that you use Network Planner.

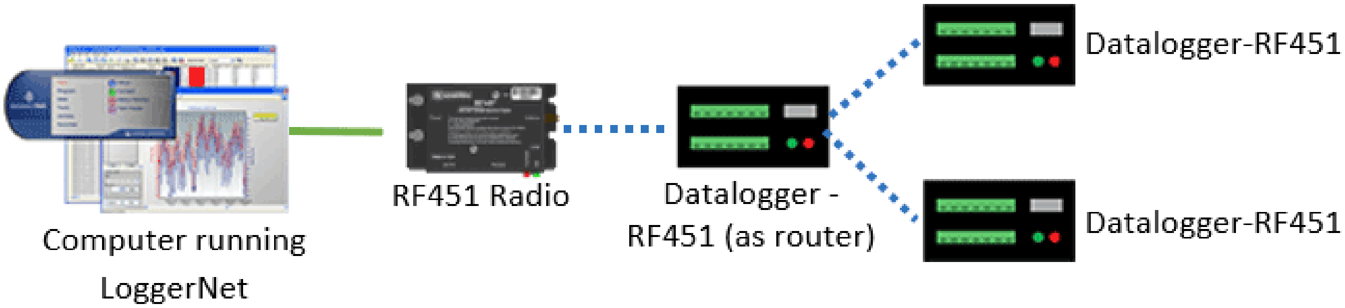
1. Ensure the datalogger is connected to an antenna and power.
2. Connect the RF451 datalogger to the computer communication port that was selected as the active interface in Device Configuration Utility.
3. Using datalogger support software, launch the EZSetup Wizard and add the datalogger.
    - PC200W and PC400 users, click **Add Datalogger** .
    - LoggerNet users, click **Setup** , then the **View** menu to ensure you are in the **EZ (Simplified)** view, then click the **Add Datalogger** button.
4. Click **Next**.
5. Select the **CR6Series** datalogger from the list, type a name for your datalogger (for example, a site or project name), and click **Next**.
6. If prompted, select the **Direct Connect** connection type and click **Next**.
7. Select the communication port used to communicate with the RF451 from the **COM Port** list.
8. Accept the default value of **00 seconds** in the **COM Port Communication Delay** - this box is used to allow time for hardware devices to "wake up" and negotiate a communications link. Click **Next**.
9. You do not need to select a baud rate. The PakBus address must match the hardware settings for your 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 left at **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 (p. 106) 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 communications, set the datalogger clock, and send a program to the datalogger. See Testing communications and completing EZ Setup (p. 40) for more information.

If you experience problems with network communications, see Troubleshooting Radio Communications (p. 125) for assistance.

3.1.5.5 RF451 radio communications with multiple dataloggers using one datalogger as a repeater

This type of network configuration is useful for communicating around an obstacle, such as a hill or building, or to reach longer distances.



To configure an RF451 radio to communicate with multiple dataloggers, you must complete the following steps (instruction follows):

- Ensure your dataloggers and RF451 radios are each connected to an antenna and power.
- Configure your connection to the RF451 devices using Device Configuration Utility.
- Assign unique PakBus addresses to each datalogger using Device Configuration Utility. (Connect to each datalogger, and set the **PakBus Address** on the **Deployment** | **Datalogger** tab.)
- Configure one datalogger to act as a repeater.
- Use datalogger support software to set up communication between the computer and the dataloggers.

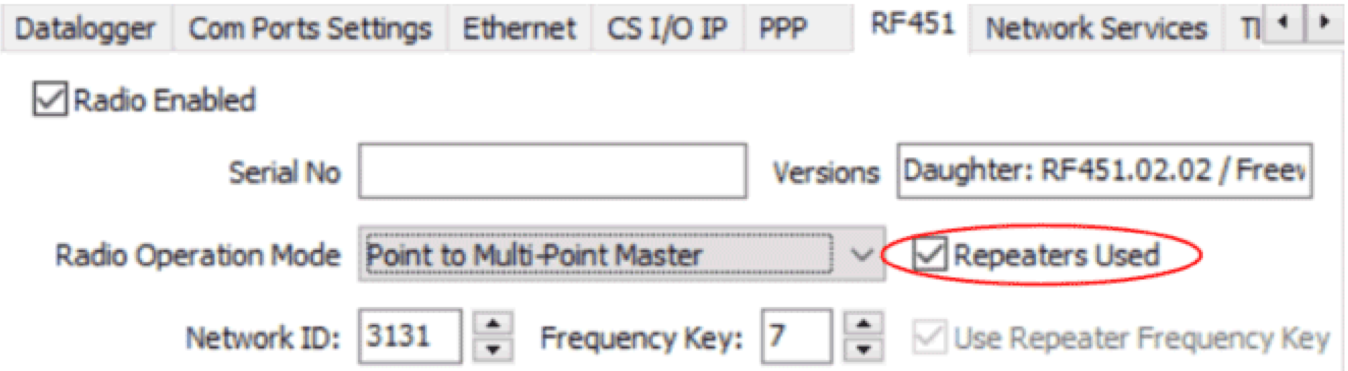
Configuring the RF451 radio connected to the computer

Configure the RF451 radio connected to the computer (see previous image for reference).

1. Ensure your RF451 radio is connected to an antenna and power.
2. Using Device Configuration Utility, connect to the RF451 radio.
3. On the **Main** tab, set the **Active Interface** to **USB** or **RS-232** (depending on how your computer will be connected to the radio).
4. Set the **Radio Operation Mode** to **Multi-Point Master**. In the **Network ID** box, type a unique number between 0 and 4095 (excluding 255). This is used to communicate with RF451 devices in the network. Make note of this number. All radios in the network require the same Network ID.

5. In the **Frequency Key** box, type a number between 0 and 14, and make note of this number. Generally, all radios in the nework will have the same Frequency Key.

6. Check the **Repeaters Used** box.



☑ Radio Enabled

Serial No

Versions Daughter: RF451.02.02 / Freev

Radio Operation Mode Point to Multi-Point Master

☑ Repeaters Used

Network ID: 3131

Frequency Key: 7

☑ Use Repeater Frequency Key

7. Apply your changes.

Configuring the datalogger acting as a repeater

1. Supply power to the datalogger. If connecting via USB for the first time, you must first install USB drivers by using Device Configuration Utility (select your datalogger, then on the main page, click **Install USB Driver**). Alternately, you can install the USB drivers using EZ Setup. A USB connection supplies 5 V power (as well as a communication link), which is adequate for setup, but a 12 V battery will be needed for field deployment. Ensure the datalogger antenna is connected.

2. Using Device Configuration Utility, connect to the RF451 datalogger.

3. On the **Deployment** tab, click the **Com Ports Settings** sub-tab.

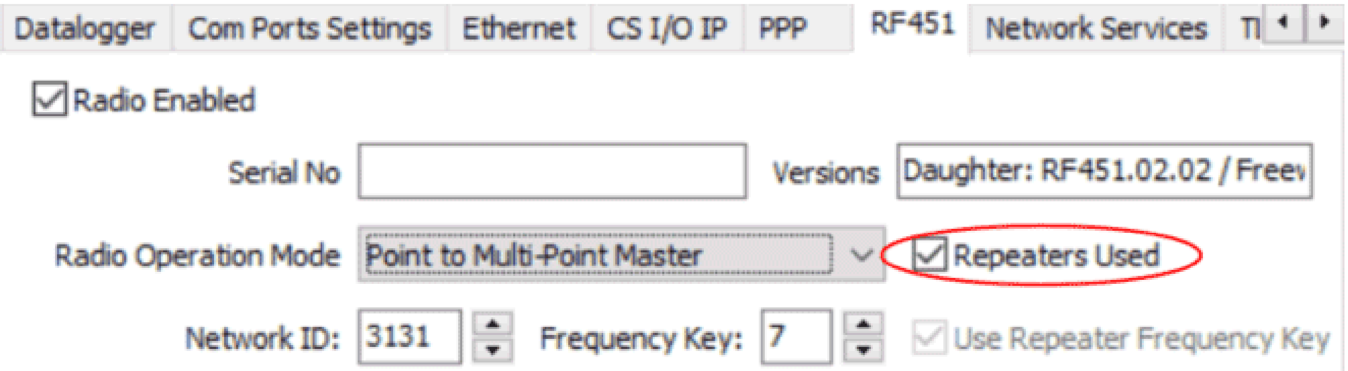
4. From the **Select the ComPort** list, select **RF**.

5. Set the **Beacon Interval** to **60** seconds (or the amount of time you are willing to wait for the leaf dataloggers in the network to be discovered).

NOTE:
A beacon is a packet broadcast at a specified interval intended to discover neighbor devices.

6. Set the **Verify Interval** to something slightly greater than the expected communication interval between the repeater and the other (leaf) dataloggers in the network (for example, 90 seconds).

7. Click the **RF451** sub-tab and check the **Repeaters Used** box.



☑ Radio Enabled

Serial No

Versions Daughter: RF451.02.02 / Freev

Radio Operation Mode Point to Multi-Point Master

☑ Repeaters Used

Network ID: 3131

Frequency Key: 7

☑ Use Repeater Frequency Key

8. Also on the **RF451** sub-tab, set the **Radio Operation Mode** to **Point to Multi-Point Slave/Repeater**. In the **Network ID** box, type the same number set in the previous instruction ("Configuring the RF451 Radio Connected to the Computer").
9. In the **Frequency Key** box, type the same number set in the previous instruction ("Configuring the RF451 Radio Connected to the Computer").
10. Apply your changes.

Adding the repeater datalogger to the LoggerNet network

To add the repeater datalogger to the LoggerNet network, follow the instructions found in Setting up communications between the RF451 datalogger and the computer (p. 35).

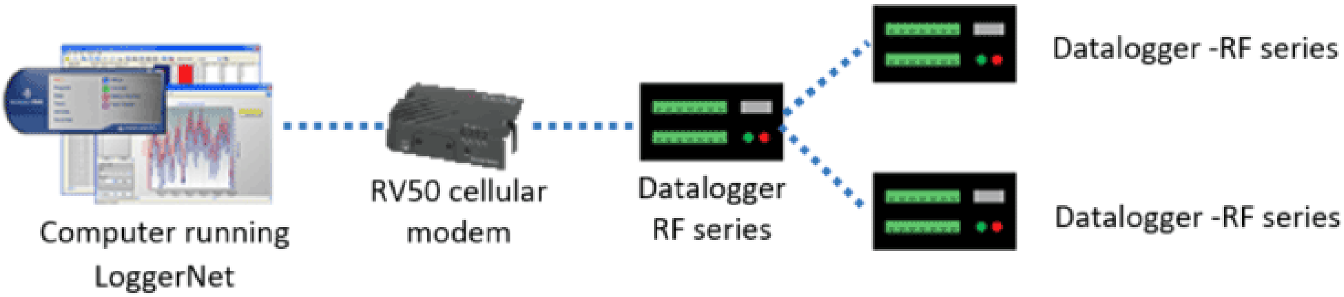
Adding leaf dataloggers to the network

1. In the LoggerNet **Standard Setup** view (click **Setup** and then the **View** menu to ensure you are in the **Standard** view), right-click on the repeater datalogger in the Entire Network pane on the left side of the window and select **CR6Series**.
2. With the newly added datalogger selected in the **Entire Network** pane, set the **PakBus Address** to the address that was assigned to the leaf datalogger in Device Configuration Utility.
3. Click **Rename**. Enter a descriptive name for the datalogger.
4. Apply your changes.
5. Repeat these steps for each leaf datalogger in the network.

If you experience problems with network communications, see Troubleshooting Radio Communications (p. 125) for assistance.

Using additional communication methods

Using similar instructions, an RF451 datalogger can be used in a system with additional communication methods. For example, in the following image, the router RF451 datalogger communicates with LoggerNet through an RV50 cellular modem connected to RF451 datalogger using the **RS-232** port. The router RF451 datalogger communicates with the leaf RF451 dataloggers over RF.



3.2 Testing communications and completing EZ Setup

1. Using datalogger support software EZ Setup, access the **Communication Test** window. This window is accessed during EZ Setup (see USB or RS-232 communications on page 18 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 communications, then click **Next** to initiate the test. To troubleshoot an unsuccessful test, see Tips and troubleshooting (p. 118).
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 **Select and Send Program**. 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. By default, all data tables set up in the datalogger program will be included for collection. Make note of the **Output File Name** and location. Click **Next**.
   - Check **Scheduled Collection Enabled** to have LoggerNet automatically collect data from the datalogger according to a schedule. When the **Base Date** and **Time** are in the past, scheduled collection will begin on the first **Collection Interval**. Click **Next** twice.
7. Click **Finish**.

3.3 Connecting the datalogger to a computer

Once you have configured your hardware connection (see Setting up communications with the datalogger on page 17 for more information), your datalogger and computer can communicate.

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

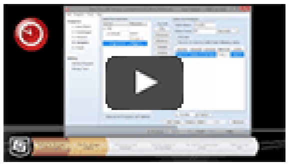
To disconnect, click **Disconnect** .

See also Communications (p. 87).

3.4 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.

Use the following instructions or watch a video.

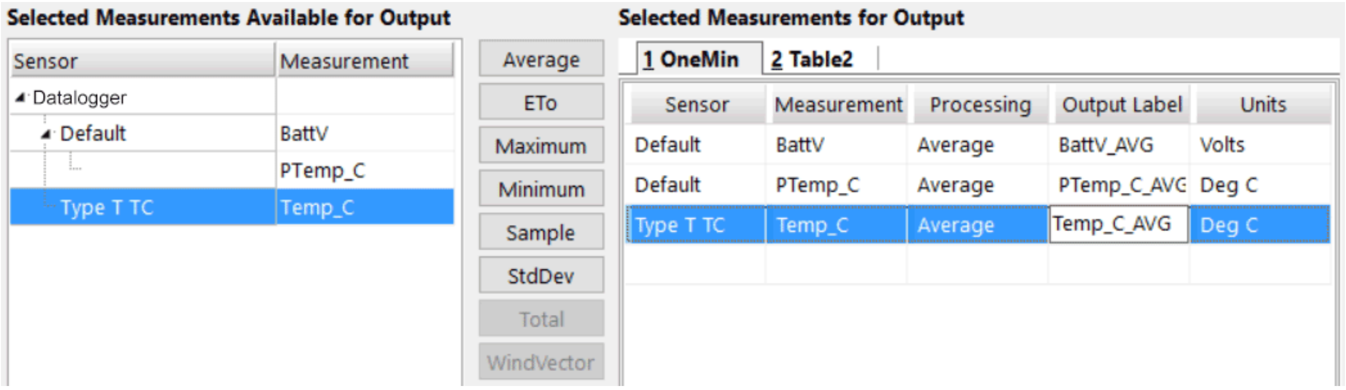


1. Using datalogger support software, launch Short Cut.
   - PC200W and PC400 users, click **Short Cut**.
   - LoggerNet users, click **Program** then **Short Cut**.
2. Click **Create New Program**.
3. Select **CR6 Series** datalogger and click **Next**.

> NOTE:
> The first time Short Cut is run, a prompt will ask for a noise rejection choice. 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 usually the best fit outside of 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. Click **Next**.
14. 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.



Selected Measurements Available for Output

Sensor	Measurement
Datalogger	
Default	BattV
	PTemp_C
Type T TC	Temp_C

Selected Measurements for Output

Sensor	Measurement	Processing	Output Label	Units
Default	BattV	Average	BattV_AVG	Volts
Default	PTemp_C	Average	PTemp_C_AVG	Deg C
Type T TC	Temp_C	Average	Temp_C_AVG	Deg C

15. Click **Finish** to compile the program. Replace the **untitled.cr6** default name and click **Save**.
16. 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 41 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 (p. 46) 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.

3.5 Sending a program to the datalogger

The datalogger requires a CRBasic program to direct measurement, processing, control, and data storage operations. The program file may use the extension `.CR6` or `.DLD`.

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 **Collect Now**. 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 structures 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

Use the following instructions or watch a video.

1. Connect the datalogger to your computer (see Connecting the datalogger to a computer on page 41 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 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 sensors are taking good measurements. See Working with data (p. 45) for more information.

3.5.0.1 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.

4. Working with data

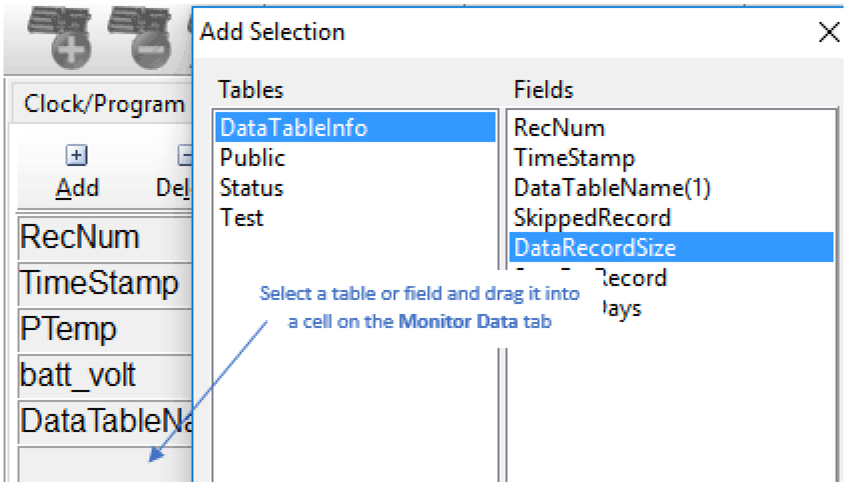
By default, the datalogger includes three tables: **Public**, **Status**, and **DataTableInfo**. Each of these tables only contains the most recent measurements and information.

- The **Public** table is configured by the datalogger program, and updated at the scan interval set within the datalogger program, It shows measurement and calculation results as they are made.
- The **Status** table includes information about 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.

For information on collecting your data, see Collecting data (p. 46).

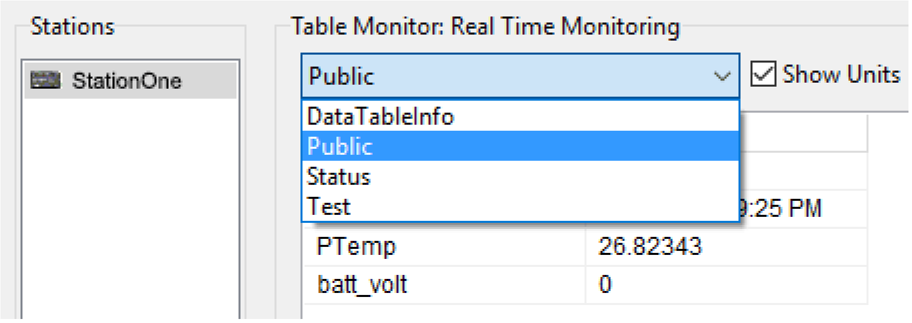
Use these instructions or follow a tutorial to monitor real-time data.

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



Select a table or field and drag it into a cell on the **Monitor Data** tab

LoggerNet users, select the **Main** category and **Connect** on the LoggerNet toolbar, select the datalogger from the **Stations** list, then click **Connect** . Once connected, select a table to view using the **Table Monitor** box.



PTemp	26.82343
batt_volt	0

4.1 Collecting data

The datalogger writes to data tables based on intervals and conditions set in the CRBasic program (see Creating data tables in a program on page 51 for more information). After the program has been running for enough time to generate data records, data may be collected by using datalogger support software. During data collection, data is copied to the computer but still remains on the datalogger. Collections may be done manually, or automatically through scheduled collections set in LoggerNet **Setup**. Use these instruction or follow a tutorial.

4.1.1 Collecting data using LoggerNet

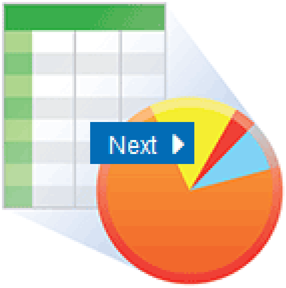
1. LoggerNet users, select **Main** and **Connect** on the LoggerNet toolbar, select the datalogger from the **Stations** list, then click **Connect** .
2. Click **Collect Now** .
3. After the data is collected, the **Data Collection Results** window displays the tables collected and where they are stored on the computer.
4. Select a data file, then **View File** to view the data. See Viewing historic data (p. 47) for more information.

4.1.2 Collecting data using PC200W or PC400

1. PC200W and PC400 users, click **Connect**.
2. Select 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. This is the default, and most often used option.
   - **All data from datalogger (Overwrite data files)**: Collects all of the data in the selected tables and replaces the existing data files on the computer.
4. By default, all tables set up in the datalogger program are selected for collection.
5. Click **Start Data Collection**.
6. After the data is collected, the **Data Collection Results** window displays the tables collected and where they are stored on the computer.
7. Select a data file, then **View File** to view the data. See Viewing historic data (p. 47) for more information.

4.2 Viewing historic data

Open data files using View Pro. View Pro contains tools for reviewing data in tabular form as well as several graphical layouts for visualization. Use these instructions or follow a tutorial.



Once the datalogger has had enough time to store multiple records, you should 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 46 for more information).

2. Open View Pro:
    - LoggerNet users select **Data** and click **View Pro** on the LoggerNet toolbar.
    - PC200W and PC400 users click **View Data Files via View Pro**.
3. Click **Open**, navigate to the directory where you saved your tables (the default directory is **C:\Campbellsci\***[your datalogger software application]*).

4.3 About data tables

A data table is essentially a file that resides in datalogger memory (for information on data table storage, see Data memory (p. 53)). 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, maximums, or minimums.

Typically, files are written to based on time or event. The number of data tables is limited to 250. You can retrieve data based on a schedule or by manually choosing to collect data using datalogger support software (see Collecting data on page 46).

Table 4-1: Example data

TOA5, MyStation, CR6, 1142, CR6.Std.01, CPU:MyTemperature.CR6, 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

4.3.1 Table definitions

Each data table is associated with descriptive information, referred to as a“table definition,” that becomes part of the file header (first few lines of the file) when data is downloaded to a computer. Table definitions include the datalogger type and OS version, name of the CRBasic program associated with the data, name of the data table (limited to 20 characters), and alphanumeric field names.

4.3.1.1 Header rows

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:

- **TOA5** - Table output format. Changed via LoggerNet **Setup** () **Standard View**, **Data Files** tab.
- **MyStation** - Station name. Changed via LoggerNet **Setup**, Device Configuration Utility, or CRBasic program.
- **CR6** - Datalogger model.
- **1142** - Datalogger serial number.
- **CR6.Std.01** - Datalogger OS version.
- **CPU:MyTemperature.CR6** - Datalogger program name. Changed by sending a new program (see Sending a program to the datalogger on page 44 for more information).
- **1958** - Datalogger program signature. Changed by revising a program or sending a new program (see Sending a program to the datalogger on page 44 for more information).
- **OneMin** - Table name as declared in the running program (see Creating data tables in a program on page 51 for more information).

The second header row reports field names. Default field names are a combination of the variable names (or aliases) from which data is 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 processing abbreviations (p. 50).

If a field is an element of an array, the field name will be followed by a indices within parentheses that identify the element in the array. 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)`. There will be one value in the second header row 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 data (p. 48).

The third header row identifies engineering units for that field. These units are declared at the beginning of a CRBasic program using the optional `Units()` declaration. 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.

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

Table 4-2: Data processing abbreviations

Data processing name	Abbreviation
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`

4.3.1.2 Data records

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

The time stamp shown represents the time at the beginning of the scan in which the data is written. Therefore, in record number 3 in the previous Example data (p. 48), **Temp_C_Avg** 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 time stamped 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.

4.4 Creating data tables in a program

Data is 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 41). 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 is stored. A data table must be called by the CRBasic program for data processing and storage 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
```

Use the `DataTable()` instruction to define the number of records, or rows, allocated to a data table. You can set a specific number of records, which is recommended for conditional tables, or allow your datalogger to auto-allocate table size. With auto-allocation, the datalogger balances the memory so the tables “fill up” (newest data starts to overwrite the oldest data) at about the same time. It is recommended you reserve the use of auto-allocation for data tables that store data based only on time (tables that store data based on the `DataInterval()` instruction). Event or conditional tables are usually set to a fixed number of records. View data table fill times for your program on the **Station Status | Table Fill Times** tab (see Checking station status (p. 119) for more information). An example of the Table Fill Times tab follows. For information on data table storage, see the CRBasic help and Data memory (p. 53).

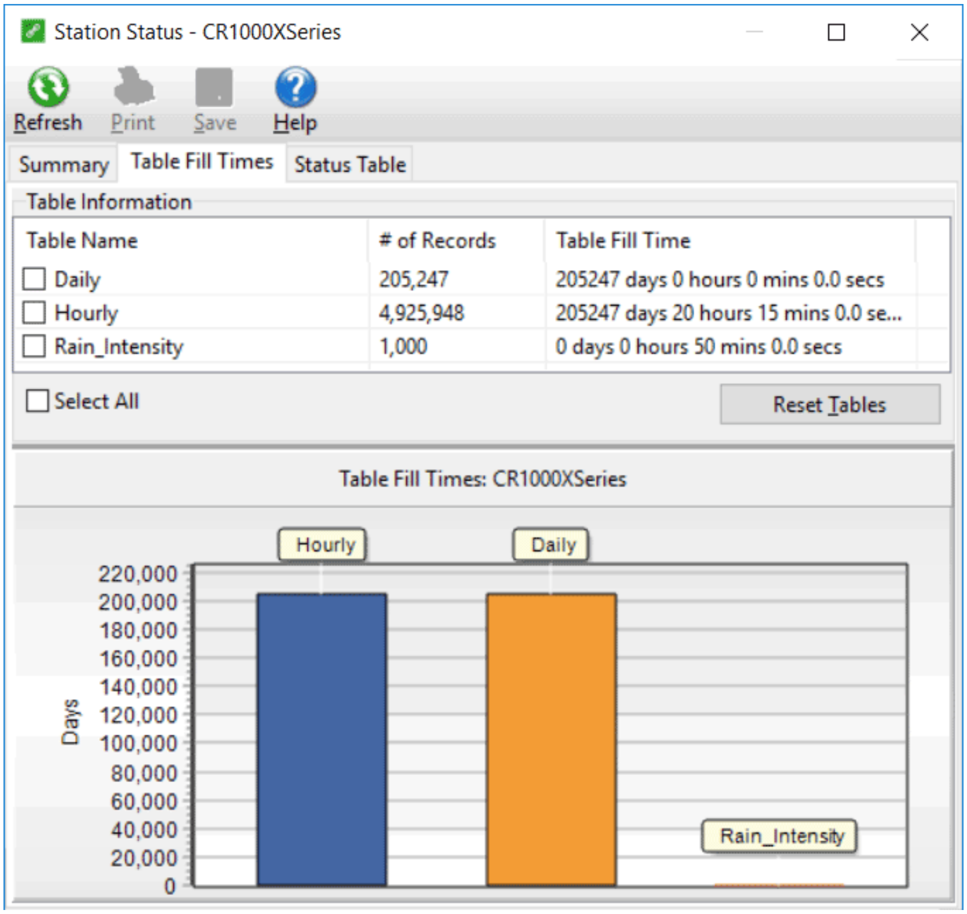


Table Name	# of Records	Table Fill Time
□ Daily	205,247	205247 days 0 hours 0 mins 0.0 secs
□ Hourly	4,925,948	205247 days 20 hours 15 mins 0.0 se...
□ Rain_Intensity	1,000	0 days 0 hours 50 mins 0.0 secs

5. Data memory

The datalogger includes three types of memory: SRAM, Flash, and Serial Flash. A memory card slot is also available for an optional microSD card. Note that the datalogger USB port does not support USB flash or thumb drives (see Communications ports on page 12 for more information).

5.1 Memory allocation

Data table SRAM and the CPU drive are automatically partitioned by the datalogger. The USR drive can be partitioned as needed. The CRD drive is automatically partitioned when a memory card is installed.

The CPU and USR drives use the FAT 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 is accessed very quickly.

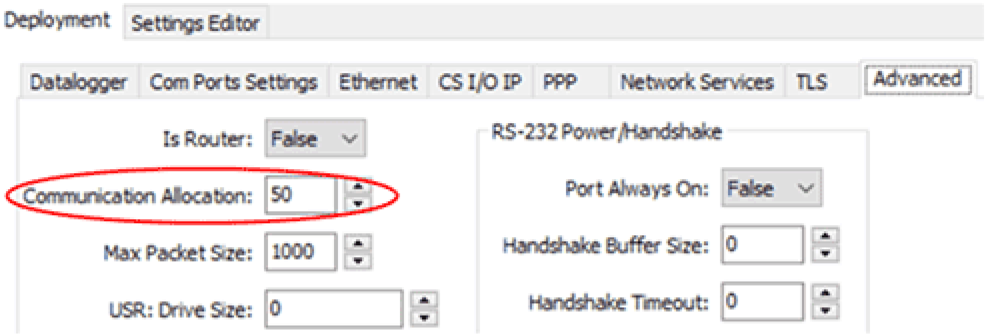
5.2 SRAM

SRAM holds program variables, communications buffers, final-data memory, and, if allocated, the USR drive. An internal lithium battery retains this memory when primary power is removed.

The structure of the datalogger SRAM memory is as follows:

- **Static Memory**: This is memory used by the operating system, regardless of the running program. This sector is rebuilt at power-up, program recompile, and watchdog events.
- **Operating Settings and Properties**: Also known as the "Keep" memory, this memory is used to store settings such as PakBus address, station name, beacon intervals, and allowed neighbor lists. This memory also stores dynamic properties such as known routes and communications timeouts.
- **CRBasic Program Operating Memory**: This memory stores the currently compiled and running user program. This sector is rebuilt on power-up, recompile, and watchdog events.
- **Variables & Constants**: This memory stores constants and public variables used by the CRBasic program. Variables may persist through power-up, recompile, and watchdog events if the `PreserveVariables` instruction is in the running program.

- **Final-Data Memory**: This memory stores data. Auto-allocated tables fill whatever memory remains after all other demands are satisfied. A compile error occurs if insufficient memory is available for user-allocated data tables. This memory is given lowest priority in SRAM memory allocation.
- **Communication Memory 1**: Memory used for construction and temporary storage of PakBus packets.
- **Communication Memory 2:** Memory used to store the list of known nodes and routes to nodes. Routers use more memory than leaf nodes because routes store information about other routers in the network. You can increase the **Communication Allocation** field in Device Configuration Utility to increase this memory allocation.



Is Router: False

Communication Allocation: 50

Max Packet Size: 1000

USR: Drive Size: 0

RS-232 Power/Handshake

Port Always On: False

Handshake Buffer Size: 0

Handshake Timeout: 0

- **USR drive:** Optionally allocated. Holds image files. Holds a copy of final-data memory when `TableFile()` instruction used. Provides memory for `FileRead()` and `FileWrite()` operations. Managed in **File Control**. Status reported in **Status** table fields **USRDriveSize** and **USRDriveFree**.

5.2.1 Data memory

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

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. The datalogger uses those parameters to assign sectors in a way that maximizes the life of its memory.

By default, data memory sectors are organized as ring memory. When the ring is full, oldest data is 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 is stored until the table is reset. To see the total number of records that can be stored before the oldest data is 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.

For additional information on datalogger memory, visit the Campbell Scientific blog article, "How to Know when Your Datalogger Memory is Getting Full."

5.2.2 USR drive

Battery-backed SRAM can be partitioned to create a FAT 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 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 `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.

5.3 Flash memory

The datalogger operating system is stored in a separate section of flash memory. To update the operating system, see Updating the operating system (p. 115).

5.4 Serial Flash memory

Serial flash memory holds the CPU drive, the web page, and datalogger settings. Because flash memory has a limited number of write/erase cycles, care must be taken to avoid continuously writing to files on the CPU drive.

5.4.1 CPU drive

The serial flash memory CPU drive contains datalogger programs and other files. This memory is managed in File Control.

> NOTE:
> When writing to files under program control, take care to write infrequently to prevent premature failure of serial flash memory. Internal chip manufacturers specify the flash technology used in Campbell Scientific CPU: drives at about 100,000 write/erase cycles. While Campbell Scientific's in-house testing has found the manufacturers' specifications to be very conservative, it is prudent to note the risk associated with repeated file writes via program control.

Also, see System specifications (p. 213) for information on datalogger memory.

5.5 MicroSD (CRD: drive) memory

The datalogger has a microSD card slot for removable, supplemental memory. The card can be configured as an extension of the datalogger final-data 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. For more information, see File system error codes (p. 143).

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 is stored to SRAM. When a card is inserted later, data accumulated in the SRAM table is copied to the card.

A microSD card can also facilitate the use of **powerup.ini** (see File management via powerup.ini on page 58 for more information).

5.5.1 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 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 up to 10 minutes. After that, compile times will be normal.

5.5.2 MicroSD card precautions

Observe the following precautions when using optional memory cards:

- Before removing a card from the datalogger, disable the card by pressing the **Eject** button and wait for the green LED. You then have 15 seconds to remove the card before normal operations resume.
- Do not remove a memory 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. Sending a program to the datalogger often erases all data.

5.5.3 Act LED indicator

When the datalogger is powered and a microSD card installed, the Act (Activity) 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.

5.6 File management via powerup.ini

Another way to upload a program, install a datalogger OS, or format a drive is to create a **powerup.ini** file. The file is created with a text editor and saved to a memory card or SC115 with the associated files. Alternatively, the **powerup.ini** file and associated files can be saved 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** file.

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** file and available options follow.

5.6.1 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.

- `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.

WARNING:
Uploading a program, installing a datalogger OS, or formatting a drive may result in data loss. Depending on several factors, the datalogger may also become incapacitated for a time. It is recommended that you retrieve data from the datalogger and back up your programs before sending a powerup.ini file; otherwise, data may be lost. To collect data using LoggerNet, connect to your datalogger and click **Collect Now** (📱). To backup your datalogger, connect to it in Device Configuration Utility, click the **Backup** menu and select **Backup Datalogger**.

Table 5-1: Powerup.ini commands

Command	Action	Details
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	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 support 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.

Table 5-1: Powerup.ini commands

Command	Action	Details
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.

5.6.2 Example powerup.ini files

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.

TIP:
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.cr6cpu:
```

Example: Run Program on Power Up

```
'Copy program file pwrup.cr6 from the external drive to CPU:
'File will run only when the datalogger is powered-up later.
2,pwrup.cr6,cpu:
```

Example: Format the USR Drive

5,,usr:

Example: Send OS on Power Up

```
'Load an operating system (.obj) file into FLASH as the new OS
9,CR6.Std.07.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.cr6,usb:
```

Example: Always Run Program, Erase Data

```
13,pwrup_1.cr6,cpu:
```

Example: Run Program Now and Erase Data Now

```
 14,run.cr6,cpu:
```

6. Measurements

6.1 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 (p. 129).

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 analog-to-digital conversion.
6. Repeat for input reversal as determined by parameters `RevEx` and `RevDiff`.
7. Apply multiplier (`Mult`) and offset (`Offset`) to measured result.

Conceptually, analog voltage sensors output two signals: high and low. For example, a sensor that outputs 1000 mV on the high signal and 0 mV on the low has an overall output of 1000 mV. A sensor that outputs 2000 mV on the high signal and 1000 mV on the low also has an overall output of 1000 mV. Sometimes, the low signal is simply sensor ground (0 mV). A single-ended measurement measures the high signal with reference to ground; the low signal is tied to ground. A differential measurement measures the high signal with reference to the low signal. Each configuration has a purpose, but the differential configuration is usually preferred.

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 measurements specifications on page 221 for more information).

A set overhead reduces the chance of overrange. Overrange 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 CR6 circuitry.

6.1.1 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 **U1** and ⏚. For more information, see Wiring panel and terminal functions (p. 5). The single-ended configuration is used with the following CRBasic instructions:

- `VoltSE()`
- `BrHalf()`
- `BrHalf3W()`
- `TCSE()`
- `Therm107()`
- `Therm108()`
- `Therm109()`
- `Thermistor()`

6.1.2 Differential measurements

A differential measurement measures the difference in voltage between two input terminals. For example, differential channel **1** is comprised of terminals **U1** and **U2**, with **U1** as high and **U2** as low. For more information, see Wiring panel and terminal functions (p. 5). The differential configuration is used with the following CRBasic instructions:

- `VoltDiff()`
- `BrFull()`

- `BrFull6W()`
- `BrHalf4W()`
- `TCDiff()`

6.1.2.1 Reverse differential

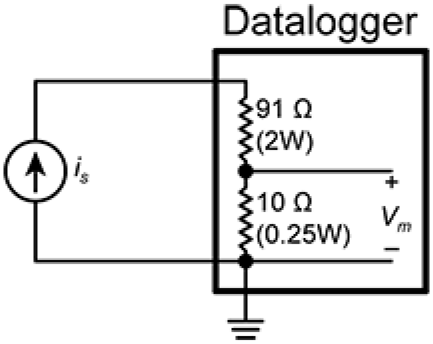
Differential measurements have the advantage of an input reversal option, `RevDiff`. When `RevDiff` is set to **True**, two differential measurements are made, the first with a positive polarity and the second reversed. Subtraction of opposite polarity measurements cancels some offset voltages associated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

For more information on voltage measurements, see Improving voltage measurement quality (p. 129).

6.2 Current-loop measurements

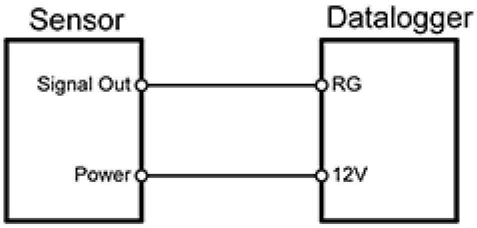
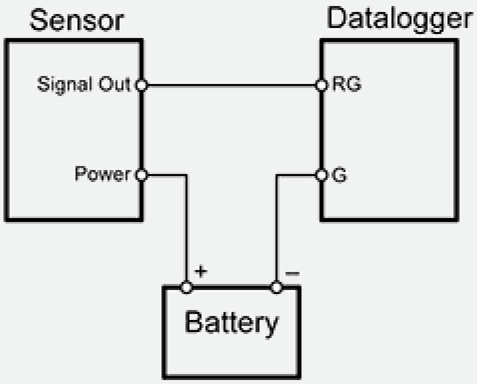
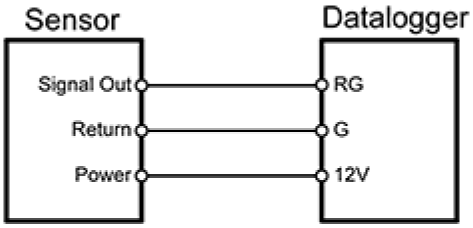
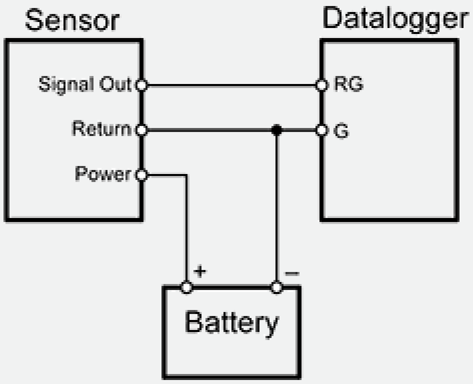
> NOTE:
> This information applies to CR6 dataloggers with serial numbers 7502 and greater. These dataloggers have two blue stripes on the label.

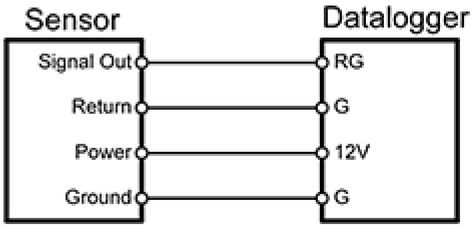
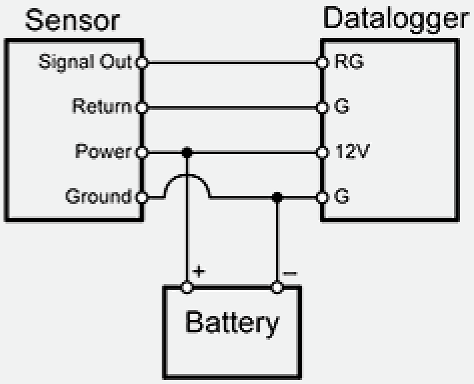
RG terminals can be configured to make analog current measurements using the `CurrentSE()` instruction. When configured to measure current, terminals each have an internal resistance of 101 Ω in the current measurement loop. The return path of the sensor must be connected directly to the **G** terminal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



6.2.1 Example Current-Loop Measurement Connections

The following table shows example schematics for connecting typical current sensors and devices. See also Current-loop measurement specifications (p. 226).

Sensor Type	Connection Example
2-wire transmitter using datalogger power	
2-wire transmitter using external power	 Sensor Datalogger Signal Out RG Power G + – Battery
3-wire transmitter using datalogger power	 Sensor Datalogger Signal Out RG Return G Power 12V
3-wire transmitter using external power	 Sensor Datalogger Signal Out RG Return G Power + – Battery

Sensor Type	Connection Example
4-wire transmitter using datalogger power	 Sensor Datalogger Signal Out Return Power Ground RG G 12V G
4-wire transmitter using external power	 Sensor Datalogger Signal Out Return Power Ground RG G 12V G + – Battery

6.3 Resistance measurements

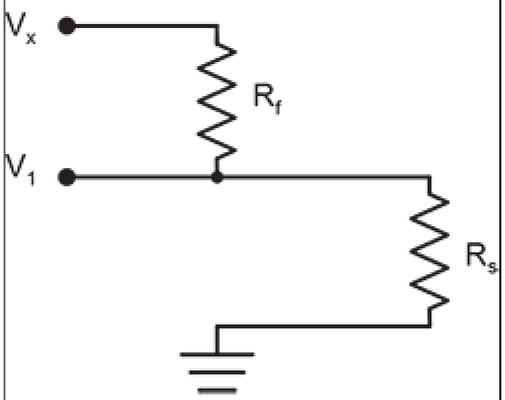
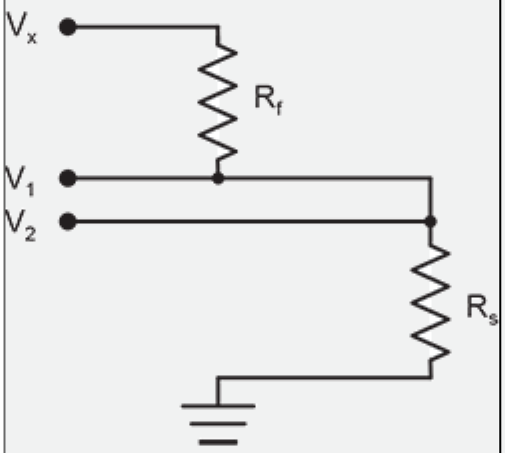
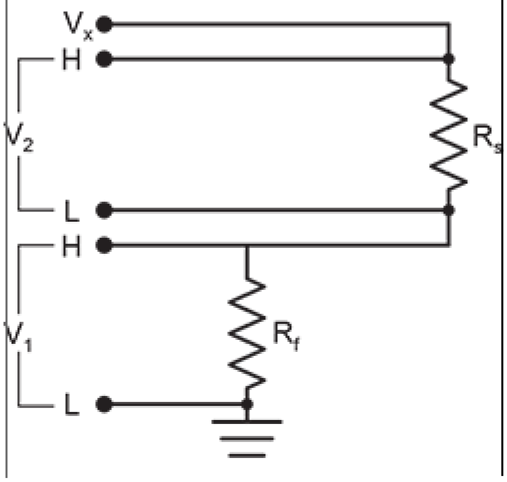
Bridge resistance is determined by measuring the difference between a known voltage or current applied to the excitation (input) of a resistor bridge and the voltage measured on the output arm. The datalogger supplies a precise voltage or current excitation via **U** terminals. Resulting voltage is measured on analog input terminals configured for single-ended or differential input. The result of the measurement is a ratio of excitation voltage or current and measured voltages.

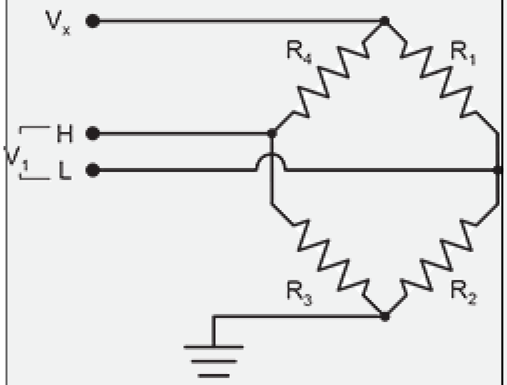
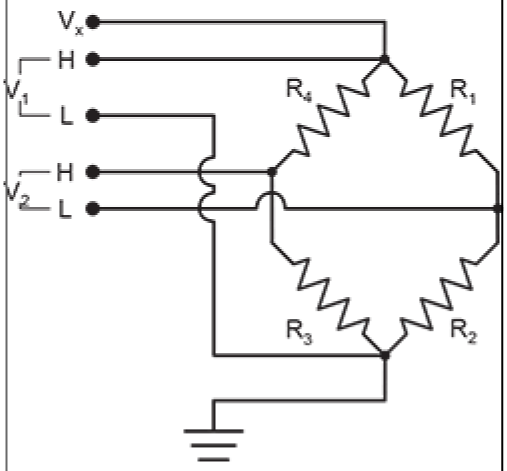
See also Resistance measurements specifications (p. 224).

6.3.1 Resistance measurements with voltage excitation

CRBasic instructions for measuring resistance with voltage excitation 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
Half Bridge[1]  	CRBasic Instruction: `BrHalf()` Fundamental Relationship: X = result w/mult = 1, offset = 0 $X = \frac{V_1}{V_x} = \frac{R_s}{R_x+R_f}$	$R_s = R_f \frac{X}{1-X}$ $R_f = \frac{R_s(1-X)}{X}$
Three Wire Half Bridge[1,2]  	CRBasic Instruction: `BrHalf3W()` Fundamental Relationship: X = result w/mult = 1, offset = 0 $X = \frac{2V_2-V_1}{V_x-V_1} = \frac{R_s}{R_f}$	$R_s = R_f X$ $R_f = \frac{R_s}{X}$
Four Wire Half Bridge[1,2]  V_x H V_2 R_s L H V_1 R_f L	CRBasic Instruction: `BrHalf4W()` Fundamental Relationship: X = result w/mult = 1, offset = 0 $X = \frac{V_2}{V_1} = \frac{R_s}{R_f}$	$R_f = \frac{R_s}{X}$ $R_s = R_f X$

Resistive-Bridge Type and Circuit Diagram	CRBasic Instruction and Fundamental Relationship	Relational Formulas
Full Bridge[1,2]  	CRBasic Instruction: `BrFull()` Fundamental Relationship: X = result w/mult = 1, offset = 0 $X = 1000 \frac{V_1}{V_x} = 1000 \left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$	These relationships apply to `BrFull()` and `BrFull6W()` $R_1 = \frac{R_2(1-X_1)}{X_1}$ $R_2 = \frac{R_1X_1}{1-X_1}$ where $X_1 = \frac{-X}{1000} + \frac{R_3}{R_3+R_4}$ $R_3 = \frac{R_4X_2}{1-X_2}$ $R_4 = \frac{R_3(1-X_2)}{X_2}$ where $X_2 = \frac{-X}{1000} + \frac{R_2}{R_1+R_2}$
Six Wire Full Bridge[1]  	CRBasic Instruction: `BrFull6W()` Fundamental Relationship: X = result w/mult = 1, offset = 0 $X = 1000 \frac{V_2}{V_1} = 1000 \left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$	

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

[2] Campbell Scientific offers terminal input modules to facilitate this measurement.

Offset voltage compensation applies to bridge measurements. In addition to `RevDiff` and `MeasOff` parameters discussed in Minimizing offset voltages (p. 139), 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 may be reduced when using `RevDiff`, `MeasOff`, and `RevEx`. When more than one measurement per sensor is necessary, such as occurs with the `BrHalf3W()`, `BrHalf4W()`, and `BrFull6W()` 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 the measurements (X) must then be processed further to obtain the resistance value, which requires additional program execution time.

CRBasic Example 1: : 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 R3
Public R_4 = 1000 'Resistance of fixed resistor R4
'Main Program
BeginProg
  Scan(500,mSec,1,0)
    'Full Bridge Measurement:
    BrFull(X,1,mV200,U1,U3,1,2500,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
```

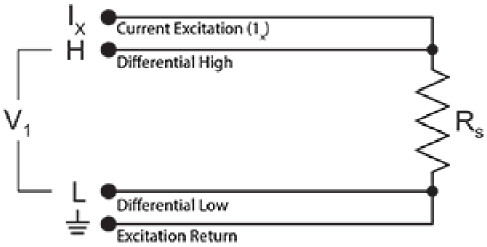
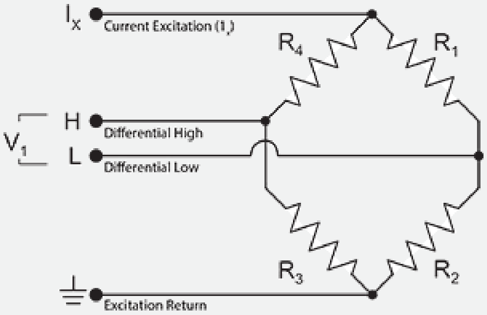
6.3.2 Resistance measurements with current excitation

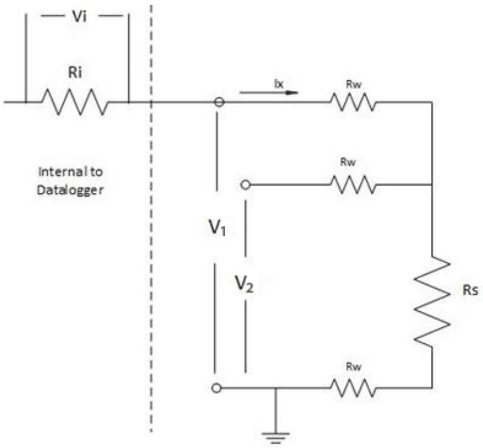
U terminals can be configured to supply precise current excitation for use with resistive bridges. Resistance can be measured by supplying a precise current and measuring the return voltage. The datalogger supplies a precise current from terminals configured for current excitation. Return voltage is measured on **U** terminals configured for single-ended or differential analog input.

U terminals can be configured as current-output terminals under program control for making resistance measurements. Use the terminal adjacent to the current source terminal as the current return. CRBasic instructions that control current excitation include:

- `Resistance()` - Applies an excitation current to a circuit and measures the resistance. The maximum excitation current is ±2500 µA.
- `Resistance3W()` - Makes a three-wire resistance measurement. The maximum excitation current is ±2500 µA.

Resistive Bridge Circuits with Current Excitation: (use the ground associated with the current excitation terminal)

Resistive-Bridge Type and Circuit Diagram	CRBasic Instruction and Fundamental Relationship	Relational Formulas
Four Wire  I_x Current Excitation (I_x) H Differential High V_1 R_s L Differential Low Excitation Return	CRBasic Instruction: `Resistance()` Fundamental Relationship[1]: $X = \frac{V}{I_x} = R_s$	
Four Wire Full Bridge  I_x Current Excitation (I_x) R_4 R_1 H Differential High V_1 L Differential Low R_3 R_2 Excitation Return	CRBasic Instruction: `Resistance()` Fundamental Relationship[1]: $X = \frac{V_1}{I_x} = R_{bridge}\left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$ $= \frac{R_3(R_1+R_2) - R_2(R_3+R_4)}{R_1+R_2+R_3+R_4}$	$R_1 = \frac{-R_2R_4 - X(R_2+R_3+R_4)}{X - R_3}$ $R_2 = \frac{-R_1R_3 - X(R_1+R_3+R_4)}{X - R_4}$ $R_3 = \frac{-R_2R_4 - X(R_1+R_2+R_4)}{X - R_1}$ $R_4 = \frac{-R_1R_3 - X(R_1+R_2+R_3)}{X - R_2}$

Resistive-Bridge Type and Circuit Diagram	CRBasic Instruction and Fundamental Relationship	Relational Formulas
Three Wire  Vi Ri Ix Rw Rw Internal to Datalogger V1 V2 Rs Rw	CRBasic Instruction: `Resistance3W()` Fundamental Relationship[2]: $Rs = (2*V_2-V_1)Ri/Vi$	

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

[2] Where Ri is the precision internal resistor value that is saved as part of the factory calibration procedure and Rs is the sense resistance.

6.3.3 Strain measurements

A principal use of the four-wire full bridge is the measurement of strain gages in structural stress analysis. `StrainCalc()` calculates microstrain (με) from the formula for the specific bridge configuration used. All strain gages supported by `StrainCalc()` use the full-bridge schematic. 'Quarter-bridge', 'half-bridge' and 'full-bridge' refer to the number of active elements in the 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.

Table 6-1: StrainCalc() configuration codes

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: $\mu\epsilon = \frac{-4 * 10^6 V_r}{GF[(1 + v) - 2V_r(v - 1)]}$
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]: $\mu\varepsilon = \frac{-10^6 V_r}{GF}$

Table 6-1: StrainCalc() configuration codes

BrConfig Code	Configuration
5	Full-bridge strain gage. Half the bridge has two gages parallel to +ε and -ε, and the other half to +νε and -νε 1: $\mu\varepsilon = \dfrac{-2 * 10^6 V_r}{GF(v + 1)}$
6	Full-bridge strain gage. Half the bridge has two gages parallel to +ε and -νε , and the other half to -νε and +ε[1]: $\mu\varepsilon = \dfrac{-2 * 10^6 V_r}{GF[(v + 1) - V_r(v - 1)]}$

[1] Where

- 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 • V1 / Vx) when multiplier = 1 and offset = 0.
- "zero": gage offset to establish an arbitrary zero.

6.3.4 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 (p. 128).

6.3.5 Accuracy for 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 CR6 do not include error introduced by the sensor, or by the transmission of the sensor signal to the datalogger.

For accuracy specifications of ratiometric resistance measurements, see Resistance measurements specifications (p. 224). Voltage measurement is variable V_1 or V_2 in Resistance measurements (p. 66). 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:
  - Bridge-resistor errors
  - Sensor noise
  - Measurement noise

6.4 Period-averaging measurements

Period-averaging measurements are used to measure the period or frequency of a signal. For these measurements, the datalogger uses 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 (p. 75).

U 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 (approximately ±10 mV) and limited resolution (approximately 1.2 mV).

See also Period-averaging measurements specifications (p. 223).

> **TIP:**
> Both pulse count and period-average measurements are used to measure frequency output sensors. However, their measurement methods are different. Pulse count measurements use dedicated hardware - pulse count accumulators, which are always monitoring the input signal, even when the datalogger is between program scans. In contrast, period-average measurements use program instructions that only monitor the input signal during a program scan. Consequently, pulse count scans can occur less frequently than period-average scans. Pulse counters may be more susceptible to low-frequency noise because they are always "listening", whereas period-averaging measurements may filter the noise by reason of being "asleep" most of the time.
>
> Pulse count measurements are not appropriate for sensors that are powered off between scans, whereas period-average measurements work well since they can be placed in the scan to execute only when the sensor is powered and transmitting the signal.

6.5 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. Both pulse count and period-average measurements are used to measure frequency-output sensors. For more information, see Period-averaging measurements (p. 74).

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

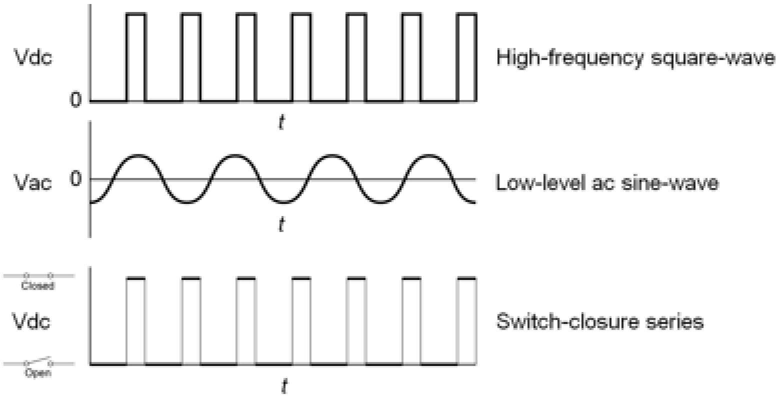


Table 6-2: Pulse input terminals and the input types they can measure

Input Type	Pulse Input Terminal	Data Option
High-frequency	**C** (all) **U** (all)	• Counts • Frequency • Running average of frequency
Low-level ac	**U** (even numbered terminals)	
Switch-closure	**P1** **P2** **C** (all) **U** (all)	

Using the `PulseCount()` instruction, **U** and **C** terminals are configurable for pulse input to measure counts or frequency. Maximum input frequency is dependent on input voltage (see Pulse measurements specifications on page 226 for more information). If pulse input voltages exceed the maximum voltage, third-party external-signal conditioners should be employed. Do not measure voltages greater than 20 V.

U 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. For tips on working with pulse measurements, see Pulse measurement tips (p. 81).

For more information, see Pulse measurements specifications (p. 226).

6.5.1 Low-level ac measurements

Low-level ac (alternating current or sine-wave) signals can be measured on **U** 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, such as the Campbell Scientific **LLAC4**, are available for converting low-level ac signals to square-wave signals measurable by **C** terminals.

For more information, see Pulse measurements specifications (p. 226).

6.5.2 High-frequency measurements

High-frequency (square-wave) signals can be measured on **U** or **C** terminals. Common sensors that output high-frequency pulses 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 `TimerInput()` instruction. See the CRBasic help for more information.

The datalogger has built-in pull-up and pull-down resistors for different pulse measurements which can be accessed using the `PulseCount()` instruction. Note that pull down options are usually used for sensors that source their own power.

6.5.2.1 U terminals

- CRBasic instruction: `PulseCount()`

See Pulse measurements specifications (p. 226) for more information.

6.5.2.2 C terminals

- CRBasic instructions: PulseCount(), TimerInput()

6.5.3 Switch-closure and open-collector measurements

Switch-closure and open-collector (also called current-sinking) signals can be measured on **U** 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 an open-collector signal include:

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

The datalogger has built-in pull-up and pull-down resistors for different pulse measurements which can be accessed using the PulseCount() instruction. Note that pull down options are usually used for sensors that source their own power.

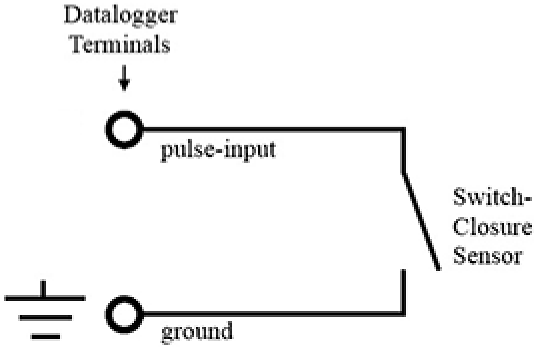
Data output options include counts, frequency (Hz), and running average.

6.5.3.1 U or C Terminals

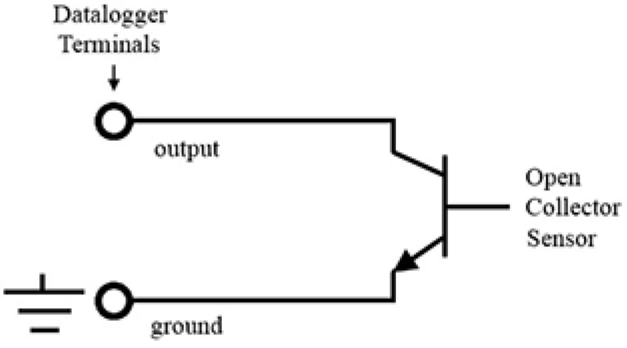
An internal 100 kΩ pull-up resistor pulls an input to 5 Vdc with the switch open, whereas a switch-closure to ground pulls the input to 0 V.

- CRBasic instruction: PulseCount()

Switch Closure on U or C Terminal



Open Collector on U or C Terminal



6.5.3.2 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 measurements specifications (p. 226).

6.5.4 Edge timing and edge counting

Edge time, period, and counts can be measured on **U** or **C** terminals. Feedback control using pulse-width modulation (PWM) is an example of an edge timing application.

6.5.4.1 Single edge timing

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

CRBasic instructions:

- `TimerInput()` – available on **C1**-**C4** and **U1**-**U12**
- `PulseCount()` – available on **C1**-**C4** and **U1**-**U12**
- `PeriodAvg()` – available on **U1**-**U12**

6.5.4.2 Multiple edge counting

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

- CRBasic instruction: `TimerInput()`- available on **C1**-**C4** and **U1**-**U12**

6.5.4.3 Timer input 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

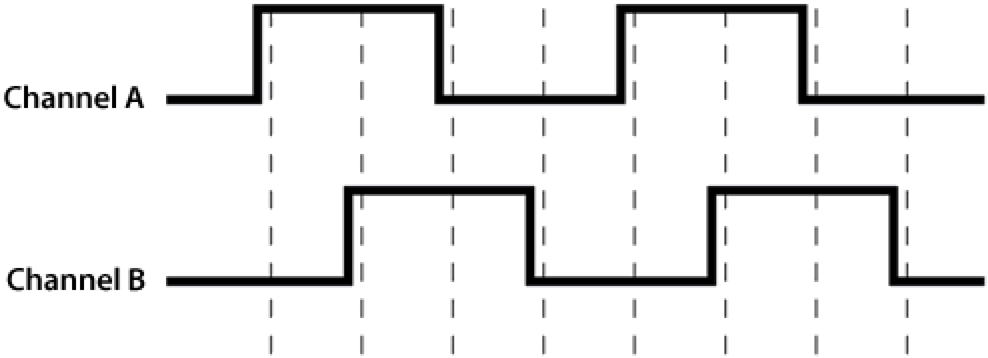
For more information, see:

- Pulse measurements specifications (p. 226)
- Digital input/output specifications (p. 228)
- Period-averaging measurements specifications (p. 223)

6.5.5 Quadrature measurements

The Quadrature() instruction is used to measure shaft or rotary encoders. A shaft encoder outputs a signal to represent the angular position or motion of the shaft. Each encoder will have two output signals, an A line and a B line. As the shaft rotates the A and B lines will generate digital pulses that can be read, or counted, by the datalogger.

In the following example, channel A leads channel B, therefore the encoder is determined to be moving in a clockwise direction. If channel B led channel A, it would be determined that the encoder was moving in a counterclockwise direction.



Terminals **U1**-**U12** can be configured as digital pairs to monitor the two channels of an encoder. The Quadrature() instruction can return:

- The accumulated number of counts from channel A and channel B. Count will increase if channel A leads channel B. Count will decrease if channel B leads channel A.
- The net direction.
- Number of counts in the A-leading-B direction.
- Number of counts in the B-leading-A direction.

Counting modes:

- Counting the increase on rising edge of channel A when channel A leads channel B. Counting the decrease on falling edge of channel A when channel B leads channel A.
- Counting the increase at each rising and falling edge of channel A when channel A leads channel B. Counting the decrease at each rising and falling edge of channel A when channel A leads channel B.
- Counting the increase at each rising and falling edge of both channels when channel A leads channel B. Counting the decrease at each rising and falling edge of both channels when channel B leads channel A.

For more information, see Pulse measurements specifications (p. 226).

6.5.6 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 to **C** terminals

Conflicts can occur when companion ports are used for different instructions (TimerInput(), PulseCount(), SDI12Recorder(), WaitDigTrig()). For example, if **C1** is used for the SDI12Recorder() instruction, **C2** cannot be used in the TimerInput(), PulseCount (), or WaitDigTrig() instructions.

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 (p. 76).

6.5.6.1 Input filters and signal attenuation

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 too much, it may not pass the detection thresholds required by the pulse count circuitry. See Pulse measurements specifications (p. 226) for more information. The listed pulse measurement specifications account for attenuation due to input filtering.

6.5.6.2 Pulse count resolution

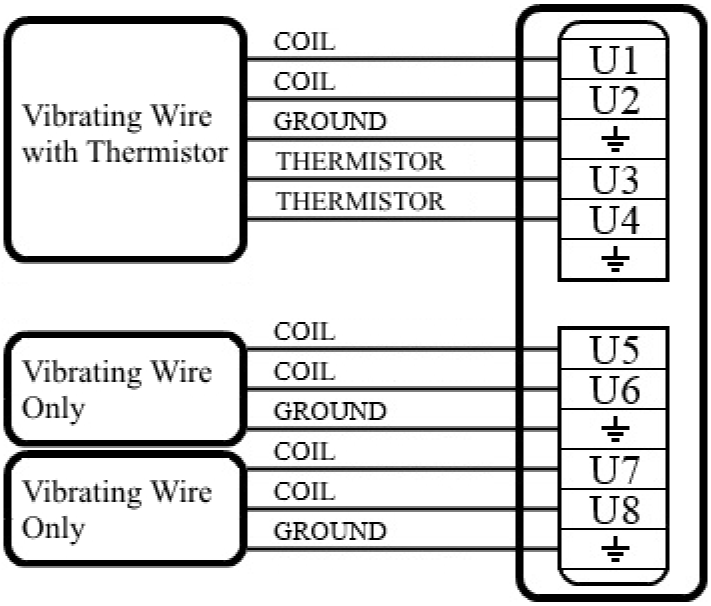
Longer scan intervals result in better resolution. PulseCount() resolution is 1 pulse per scan. On a 1 second scan, the resolution is 1 pulse per second. The resolution on a 10 second scan interval is 1 pulse per 10 seconds, which is 0.1 pulses per second. The resolution on a 100 millisecond interval is 10 pulses per second.

For example, if a flow sensor outputs 4.5 pulses per second and you use a 1 second scan, one scan will have 4 pulses and the next 5 pulses. Scan to scan, the flow number will bounce back and forth. If you did a 10 second scan (or saved a total to a 10 second table), you would get 45 pulses. The total is 45 pulses for every 10 seconds. An average will correctly show 4.5 pulses per second. You wouldn't see the reading bounce on the longer time interval.

6.6 Vibrating wire measurements

The datalogger can measure vibrating wire sensors either directly, or 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.

The following image provides some examples for connecting vibrating wire sensors to the CR6. You can use the Short Cut software to create a program and display a wiring diagram for most types of vibrating wire sensors. Access the vibrating wire measurement in Short Cut through the **Generic Measurements** sensors folder. Short Cut also has measurements for specific sensor models in the **Geotechnical & Structural** and **Water > Level & Flow** folders.



6.6.1 VSPECT®

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.

The datalogger uses an audio ADC to capture vibrating wire signals on **U** terminals. Noise frequencies may be sourced from harmonics of the natural frequency, electronic noise, or harmonics of the electronic noise. For example, 50 Hz or 60 Hz noise from ac mains grid power and associated harmonics are common noise sources. Noise frequencies may also originate from mechanical obstruction of the taut wire, such as may be caused by a loose wire or when the wire vibration is physically changed by sensor movement. VSPECT makes possible the separation of the natural-resonant frequency from these other frequencies.

> NOTE:
> The FFT algorithm requires time for computation. Compile or download errors will occur if the CRBasic program does not allow two seconds for the measurement of each sensor.

6.6.1.1 VSPECT diagnostics

The following diagnostics indicate the condition of a vibrating wire sensor:

- Decay ratio
- Signal-to-noise ratio
- Low signal strength amplitude warning
- Invalid voltage-supply warning

Decay ratio

“Decay” is the dampening of the wire over time. The decay ratio is calculated as:

Decay Ratio = **Ending Amplitude / Beginning Amplitude**

Some sensors will decay very rapidly. A good practice is to characterize sensor decay and amplitude when a sensor is new; so, the health of the sensor can be monitored over time.

Signal-to-noise ratio

The signal-to-noise ratio is calculated as:

Signal-to-Noise Ratio = **Response Amplitude / Noise Amplitude**

Low signal strength amplitude warning

When the response amplitude is measured as less than 0.01 mV RMS, the **Resonant Frequency** value reports NAN indicating that low signal strength amplitudes have occurred. The 0.01 mV threshold can be modified in the `VibratingWire()` instruction.

Invalid voltage supply warning

A **Resonant Frequency** value of **-555555** is an error code indicating an invalid voltage supply in the hardware of the datalogger. This requires factory repair, see Maintaining your datalogger (p. 105).

6.6.2 Improving vibrating wire measurement quality

The following may improve measurement quality:

- Match frequency ranges to expected frequencies.
- Reject noise.
- Minimize resonant decay.
- Prevent spectral leakage.

6.6.2.1 Matching measurement ranges to expected frequencies

Measurements are best when the frequency ranges of the swept excitation and of the response analysis match the range of resonant frequencies expected from the sensor. The swept and analysis ranges for specific sensors are determined using the tools in Device Configuration Utility on the **VW Diagnostics** tab. Once determined, the ranges are then programmed into the CRBasic program by adjusting the `BeginFreq` and `EndFreq` parameters in the `VibratingWire()` instruction.

6.6.2.2 Rejecting noise

More accurate readings can be obtained when the sensor is swept over narrower-frequency ranges. A narrow-frequency measurement reduces noise and yields a greater signal-to-noise ratio than a wide measurement. Sensors with frequency ranges below 450 Hz should work well even in the presence of 50 or 60 Hz noise; however, they should be characterized.

> NOTE:
> Check the manufacturer specifications for the sensor frequency and excitation ranges to help determine the swept frequency range.

6.6.2.3 Minimizing resonant decay

A narrow-swept range ensures minimal decay of the resonant response prior to measurement. Gage response starts to decay as soon as the frequency sweep moves past the resonant frequency. Observing this decay is difficult because the swept frequency overwhelms the resonant-frequency response while excitation is still active. Because wider excitation sweeps take longer, the resonant response decays for a longer time before the datalogger can measure it. The resulting resonant amplitude is smaller.

6.6.2.4 Preventing spectral leakage

Matching the swept excitation to the expected resonant frequency range prevents spectral leakage from complicating the spectral analysis. Vibrating wire sensors are usually optimized for a single resonant frequency to overwhelm harmonic and sub-harmonic responses; so, spectral leakage usually has little impact. Measurements of poorly constructed vibrating wire gages that may have large harmonic and sub-harmonic responses are more susceptible to spectral leakage.

6.7 Sequential and pipeline processing modes

The datalogger has two processing modes: sequential mode and pipeline mode. In sequential mode, datalogger tasks run more or less in sequence. In pipeline mode, datalogger tasks run more or less in parallel. Mode information is included in a message returned by the datalogger, which is displayed by software when the program is sent and compiled, and it is found in the **Status Table**, **CompileResults** field. The CRBasic Editor pre-compiler returns a similar message.

The default mode of operation is pipeline mode. However, when the datalogger program is compiled, the datalogger analyzes the program instructions and automatically determines which mode to use. The datalogger can be forced to run in either mode by placing the `PipeLineMode` or `SequentialMode` instruction at the beginning of the program (before the `BeginProg` instruction).

For additional information, visit the Campbell Scientific blog article, "Understanding CRBasic Program Compile Modes: Sequential and Pipeline."

6.7.1 Sequential mode

Sequential mode executes instructions in the sequence in which they are written in the program. After a measurement is made, the result is converted to a value determined by processing arguments that are included in the measurement instruction, and then program execution proceeds to the next instruction. This line-by-line execution allows writing conditional measurements into the program.

> NOTE:
> The exact time at which measurements are made in sequential mode may vary if other measurements or processing are made conditionally, if there is heavy communications activity, or if other interrupts occur (such as accessing a Campbell Scientific memory card).

6.7.2 Pipeline mode

Pipeline mode handles measurement, most digital, and processing tasks separately, and, in many cases, simultaneously. Measurements are scheduled to execute at exact times and with the highest priority, resulting in more precise timing of measurements, and usually more efficient processing and power consumption.

In pipeline mode, it will take less time for the datalogger to execute each scan of the program. However, because processing can lag behind measurements, there could be instances, such as when turning on a sensor using the `SW12()` instruction, that the sensor might not be on at the correct time to make the measurement.

Pipeline scheduling requires that the program be written such that measurements are executed every scan. Because multiple tasks are taking place at the same time, the sequence in which the instructions are executed may not be in the order in which they appear in the program. Therefore, conditional measurements are not allowed in pipeline mode. Because of the precise execution of measurement instructions, processing in the current scan (including updating public variables and data storage) is delayed until all measurements are complete. Some processing, such as transferring variables to control instructions, like `PortSet()` and `ExciteV()`, may not be completed until the next scan.

When a condition is true for a task to start, it is put in a queue. Because all tasks are given the same priority, the task is put at the back of the queue. Every 1 ms (or faster if a new task is triggered) the task currently running is paused and put at the back of the queue, and the next task in the queue begins running. In this way, all tasks are given equal processing time by the datalogger.

6.7.3 Slow Sequences

Priority of a slow sequence (`SlowSequence`)in the datalogger will vary, depending upon whether the datalogger is executing its program in pipeline mode or sequential mode. With the important exception of measurements, when running in pipeline mode all sequences in the program have the same priority. When running in sequential mode, the main scan has the highest priority for measurements, followed by background calibration (which is automatically run in a slow sequence), then the first slow sequence, the second slow sequence, and so on. The effects of this priority are negligible; however, since, once the tasks begin running, each task is allotted a 10 msec time slice, after which, the next task in the queue runs for 10 msec. The datalogger cycles through the queue until all instructions for all sequences are complete.

7. Communications

Dataloggers communicate with datalogger support software, other Campbell Scientific dataloggers, and other hardware and software 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 communications module. See also Communications specifications (p. 230).

Some communications 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. 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 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 a video.

7.1 General serial communications

The datalogger supports two-way serial communications. These communications 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 Communications ports (p. 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 protocol used by the device with which you will be communicating. 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.

7.2 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 SCADA networks. Dataloggers can communicate using Modbus on all available communication ports. The datalogger supports RTU and ASCII communication modes on RS-232 and RS-485 connections.

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:

- About Modbus (p. 89)
- 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 can be much simpler than from some other serial sensors. Because Modbus uses a common bus and addresses each node, field instruments are effectively multiplexed to a datalogger without additional hardware.

When doing Modbus communications over RS-232, the datalogger, through Device Configuration Utility or the **Settings** editor, can be set to keep communication ports open and awake, but at higher power usage. Set **RS-232Power** to **Always on**. Otherwise, the datalogger goes into sleep mode after 40 seconds of communications 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.

More information on Modbus can be found at:

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

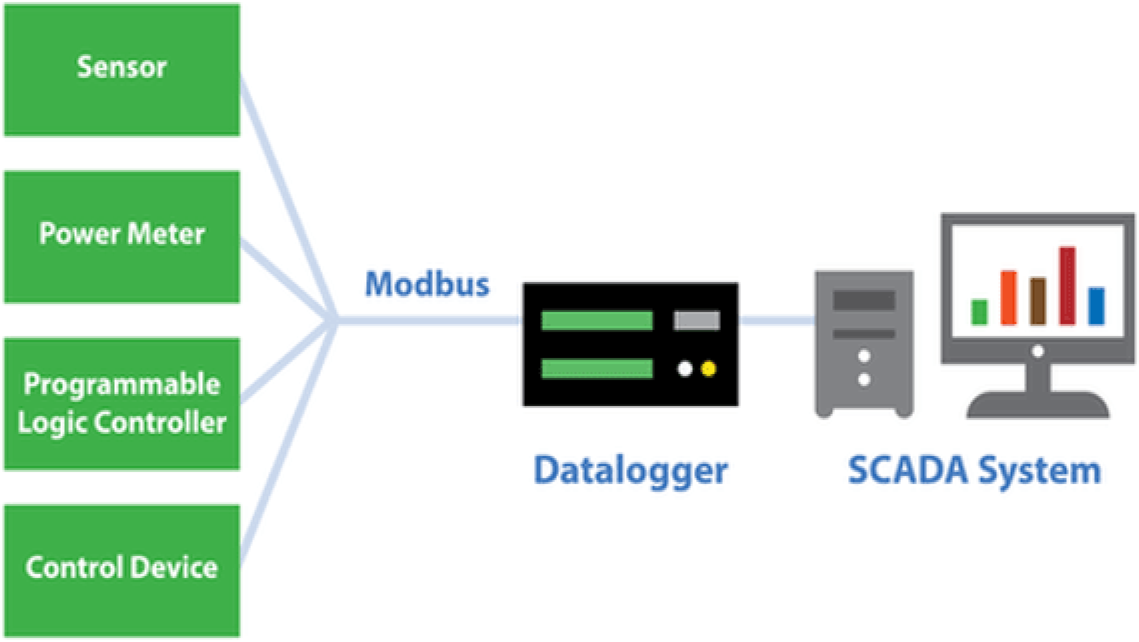
7.2.1 About Modbus

Modbus is a communications protocol that enables communications among many devices connected to the same network. Modbus is often used in supervisory control and data acquisition (SCADA) systems to connect remote terminal units (RTUs) with a supervisory computer - allowing them to relay measurement data, device status, control commands, and configuration information.

The popularity of Modbus has grown because it is freely available and because its messaging structure is independent of the type of physical interface or connection that is used. Modbus can coexist with other types of connections on the same physical interface at the same time. You can operate the protocol over several data links and physical layers.

Modbus is supported by many industrial devices, including those offered by Campbell Scientific. Not only can intelligent devices such as microcontrollers and programmable logic controllers (PLCs) communicate using Modbus, but many intelligent sensors have a Modbus interface that enables them to send their data to host systems. Examples of using Modbus with Campbell Scientific dataloggers include:

- Interfacing dataloggers and Modbus-enabled sensors.
- Sending and retrieving data between dataloggers and other industrial devices.
- Delivering environmental data to SCADA systems.
- Integrating Modbus data into PakBus networks, or PakBus data into Modbus networks.



7.2.2 Modbus protocols

There are three standard variants of Modbus protocols:

- **Modbus RTU** — Modbus RTU is the most common implementation available for Modbus. Used in serial communications, data is transmitted in a binary format. The RTU format follows the commands/data with a cyclic redundancy check checksum.

> NOTE:
> The Modbus RTU protocol standard does not allow a delay between characters of 1.5 times or

> more the length of time normally required to receive a character. This is analogous to “pizza” being understood, and “piz za” being gibberish. It's important to note that communications hardware used for Modbus RTU, such as radios, must transfer data as entire packets without injecting delays in the middle of Modbus messages.

- **Modbus ASCII** — Used in serial communications, data is transmitted as an ASCII representation of the hexadecimal values. Timing requirements are loosened, and a simpler longitudinal redundancy check checksum is used.
- **Modbus TCP/IP or Modbus TCP** — Used for communications over TCP/IP networks. The TCP/IP format does not require a checksum calculation, as lower layers already provide checksum protection. The packet structure is similar to RTU, but utilizes a different header. Devices labeled as Modbus gateways will convert from Modbus TCP to Modbus RTU.

Campbell Scientific dataloggers support Modbus RTU, Modbus ASCII, and Modbus TCP protocols. If the connection is over IP, Campbell Scientific dataloggers always use Modbus TCP. Modbus slave functionality over other comports use RTU. When acting as a master, the datalogger can be switched between ASCII and RTU protocols using an option in the `ModbusMaster()` instruction.

7.2.3 Understanding Modbus Terminology

Many of the object types are named from using Modbus in driving relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or a contact.

Information is stored in the slave device in up to four different tables. Two tables store on/off discrete values (coils) and two store numerical values (registers). The coils and registers each have a read-only table and read/write table.

7.2.4 Connecting Modbus devices

Dataloggers can communicate with Modbus on all available communication ports. Consideration should be given to proper surge protection of any cabled connection. Between systems of significantly different ground potential, optical isolation may be appropriate. For additional information on grounds, see Grounds (p. 11).

The common serial interface used for Modbus RTU connections is RS-485 half-duplex, or two-wire RS-485. This connection uses one differential pair for data, and another wire for a signal ground. When twisted pair cable is used, the signal can travel long distances. Resistors are often used to reduce noise. Bias resistors are used to give a clean default state on the signal lines. For long cable lengths, termination resistors, which are usually 120 ohms, are needed to stop data corruption due to reflections. Signal grounds are terminated to earth ground with resistors to prevent ground loops, but allow a common mode signal. The resistors to ground are usually integral to the equipment. The resistive ground is labeled as **RG** on Campbell Scientific equipment.

7.2.5 About Modbus communications

Modbus is a master-slave protocol. The device requesting the information is called the Modbus master, and the devices supplying information are Modbus slaves. In a standard Modbus network, there is one master and up to 247 slaves. A master does not have a Modbus address. However, each Modbus slave on a shared network has a unique address from 1 to 247.

A single Modbus master device initiates commands (requests for information), sending them to one or more Modbus slave devices on the same network. Only the Modbus master can initiate communications. Modbus slaves, in turn, remain silent, communicating only when responding to requests from the Modbus master.

Every message from the master will begin with the slave address, followed by the function code, function parameters, and a checksum. The slave will respond with a message beginning with its address, followed by the function code, data, and a checksum. The amount of data in the packet will vary, depending on the command sent to the slave. Slave devices only process one command at a time. So, the master needs to wait for a response, or timeout before sending the next command.

A broadcast address is specified to allow simultaneous communications with all slaves. Because response time of slave devices is not specified by the standard, and device manufacturers also rarely specify a maximum response time, broadcast features are rarely used. When implementing a system, timeouts in the master will need to be adjusted to account for the observed response time of the slaves.

Campbell Scientific dataloggers can be programmed to be a Modbus master or Modbus slave - or even both at the same time! This proves particularly helpful when your datalogger is a part of two wider area networks. In one it uses Modbus to query data (as a master) from localized sensors or other data sources, and then in the other, it serves that data up (as a slave) to another Modbus master.

7.2.6 About Modbus programming

Modbus capability of the datalogger must be enabled through configuration or programming. See the CRBasic Editor help for detailed information on program structure, syntax, and each instruction available to the datalogger.

CRBasic Modbus instructions include:

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

- `MoveBytes()`

7.2.6.1 Endianness

Endianness refers to the sequential order in which bytes are arranged into larger numerical values when stored in memory. Words may be represented in big-endian or little-endian format, depending on whether bits or bytes or other components are ordered from the big end (most significant bit) or the little end (least significant bit).

In big-endian format, the byte containing the most significant bit is stored first, then the following bytes are stored in decreasing significance order, with the byte containing the least significant bit stored last. Little-endian format reverses this order: the sequence stores the least significant byte first and the most significant byte last. Endinness is used in some Modbus programming so it is important to note that the CR6 is a big-endian instrument.

7.2.6.2 Function codes

A function code tells the slave which storage entity to access and whether to read from or write to that entity. Different devices support different functions (consult the device documentation for support information). The most commonly used functions (codes 01, 02, 03, 04, 05, 15, and 16) are supported by Campbell Scientific dataloggers.

Most users only require the read- register functions. Holding registers are read with function code **03**. Input registers are read with function code **04**. This can be confusing, because holding registers are usually listed with an offset of 40,000 and input registers with an offset of 30,000. Don't mix up the function codes. Double check the register type in the device documentation.

Function Code	Action	Entity
01 (01 hex)	Read	Discrete Output Coils
05 (05 hex)	Write single	Discrete Output Coil
15 (0F hex)	Write multiple	Discrete Output Coils
02 (02 hex)	Read	Discrete Input
04 (04 hex)	Read	Input Registers
03 (03 hex)	Read	Holding Registers
06 (06 hex)	Write single	Holding Register
16 (10 hex)	Write multiple	Holding Registers

The write-register functions will only work on holding registers. Function **06** only changes one 16-bit register, whereas function 16, changes multiple registers. Note, when writing registers, the

`Variable` parameter for the `ModbusMaster()` instruction refers to a source, not a destination.

7.2.7 Modbus information storage

With the Modbus protocol, most of the data values you want to transmit or receive are stored in registers. Information is stored in the slave device in four different entities. Two store on/off discrete values (coils) and two store numerical values (registers). The four entities include:

- Coils – 1-bit registers, used to control discrete outputs (including Boolean values), read/write.
- Discrete Input – 1-bit registers, used as inputs, read only.
- Input Registers – 16-bit registers, used as inputs, read only.
- Holding Registers – 16-bit registers; used for inputs, output, configuration data, or any requirement for “holding” data; read/write.

7.2.7.1 Registers

In a 16-bit memory location, a 4-byte value takes up two registers. The Modbus protocol always refers to data registers with a starting address number, and a length to indicate how many registers to transfer.

Campbell Scientific uses 1-based numbering (a common convention for numbering registers in equipment) in the `ModbusMaster()` instruction. With 1-based numbering, the first data location is referred to as register number 1. Some equipment uses 0-based numbering (check the equipment documentation). With 0-based numbering, the first register is referred to as 0.

Reading register numbers can be complicated by the fact that register numbers are often written with an offset added. Input registers are written with an offset of 30000. So, the first input register is written as 30001, with 1-based numbering. Holding registers are numbered with an offset of 40000. You must remove the offset before writing the number as the `Start` parameter of `ModbusMaster()`.

There are rare instances when equipment is designed with the registers mapped including the offset. That means 40001 in the documentation is really register number 40001. Those are rare instances, and the equipment is deviating from standards. If 1 or 2 don’t work for the Start parameter, try 40001 and 40002.

7.2.7.2 Coils

Discrete digital I/O channels in Modbus are referred to as coils. The term coil has its roots in digital outputs operating solenoid coils in an industrial environment. Coils may be read only or read/write. A read only coil would be a digital input. A read/write coil is used as an output. Coils are read and manipulated with their own function codes, apart from the registers. Many modern devices do not utilize coils at all.

When working with coils, the datalogger requires Boolean variables. When reading coils, each Boolean in an array will hold the state of one coil. A value of **True** will set the coil, a value of **False** will unset the coil.

7.2.7.3 Data Types

Modbus does not restrict what data types may be contained within holding and input registers. Equipment manufacturers need to indicate what binary data types they are using to store data. Registers are 16-bit, so 32-bit data types use 2 registers each. Some devices combine more registers together to support longer data types like strings. The `ModbusMaster()` instruction has a `ModbusOption` parameter that supports several different data types.

When data types use more than 1 register per value, the register order within the data value is important. Some devices will swap the high and low bytes between registers. You can compensate for this by selecting the appropriate `ModbusOption`.

Byte order is also important when communicating data over Modbus. Big Endian byte order is the reverse of Little Endian byte order. It may not always be apparent which a device uses. If you receive garbled data, try reversing the byte order. Reversing byte order is done using the `MoveBytes()` instruction. There is an example in CRBasic help for reversing the bytes order of a 32-bit variable.

After properly reading in a value from a Modbus device, you might have to convert the value to proper engineering units. With integer data types, it is common to have the value transmitted in hundredths or thousandths.

Unsigned 16-bit integer

The most basic data type used with Modbus is unsigned 16-bit integers. It is the original Modbus data type with 1 register per value. On the datalogger, declare your destination variable as type **Long**. A **Long** is a 32-bit signed integer that contains the value received. Select the appropriate `ModbusOption` to avoid post-processing.

Signed 16-bit integer

Signed 16-bit integers use 1 register per value. On the datalogger, declare your destination variable as type **Long**. A **Long** is a 32-bit signed integer that contains the value received . Select the appropriate `ModbusOption` to avoid post-processing.

Signed 32-bit integer

Signed 32-bit integers require two registers per value. This data type corresponds to the native **Long** variable type in Campbell dataloggers. Declare your variables as type **Long** before using them as the Variable parameter in `ModbusMaster()`. Select the appropriate `ModbusOption` to avoid post-processing.

Unsigned 32-bit integer

Unsigned 32-bit integers require two registers per value. Declare your variables as type **Long** before using them as the `Variable` parameter in `ModbusMaster()`. The **Long** data type is a signed integer, and does not have a range equal to that of an unsigned integer. If the integer value exceeds 2,147,483,647 it will display incorrectly as a negative number. If the value does not exceed that number, there are no issues with a variable of type **Long** holding it.

32-Bit floating point

32-bit floating point values use 2 registers each. This is the default FLOAT data type in Campbell Scientific dataloggers . Select the appropriate `ModbusOption` to avoid post-processing.

7.2.8 Modbus tips and troubleshooting

Most of the difficulties with Modbus communications arise from deviations from the standards, which are not enforced within Modbus. Whether you are connecting via Modbus to a solar inverter, power meter, or flow meter, the information provided here can help you overcome the challenges, and successfully gather data into a Campbell datalogger. 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

7.2.8.1 Error codes

Modbus defines several error codes, which are reported back to a master from a slave. `ModbusMaster()` displays these codes as a negative number. A positive result code indicates no response was received.

Result code -01: illegal function

The illegal function error is reported back by a Modbus slave when either it does not support the function at all, or does not support that function code on the requested registers. Different devices support different functions (consult the device documentation). If the function code is supported, make sure you are not trying to write to a register labeled as read-only. It is common for devices to have holding registers where read-only and read/write registers are mapped next to each other.

An uncommon cause for the **-01** result is a device with an incomplete implementation of Modbus. Some devices do not fully implement parsing Modbus commands. Instead, they are hardcoded to respond to certain Modbus messages. The result is that the device will report an error when you try selectively polling registers. Try requesting all of the registers together.

Result code -02: illegal data address

The illegal data address error occurs if the slave rejects the combination of starting register and length used. One possibility, is a mistake in your program on the starting register number. Refer to the earlier section about register number and consult the device documentation for support information. Also, too long of a length can trigger this error. The `ModbusMaster()` instruction uses length as the number of values to poll. With 32-bit data types, it requests twice as many registers as the length.

An uncommon cause for the -02 result is a device with an incomplete implementation of Modbus. Some devices do not fully implement parsing Modbus commands. Instead, they are hard coded to respond to certain Modbus messages. The result is that the device will report an error when you try selectively polling registers. Try requesting all of the registers together.

Result code -11: COM port error

Result code -11 occurs when the datalogger is unable to open the COM port specified. For serial connections, this error may indicate an invalid COM port number. For Modbus TCP, it indicates a failed socket connection.

If you have a failed socket connection for Modbus TCP, check your `TCPOpen()` instruction. The socket returned from `TCPOpen()` should be a number less than 99. Provided the datalogger has a working network connection, further troubleshooting can be done with a computer running Modbus software. Connect the computer to the same network and attempt to open a Modbus TCP connection to the problem slave device. Once you resolve the connection between the computer and the slave device, the connection from the datalogger should work.

7.3 Internet communications

See the Communications specifications (p. 230) 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, PakBus communications over TCP/IP are possible. These functions include the following:

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

Datalogger callback to LoggerNet and datalogger-to-datalogger communications are also possible over TCP/IP. For details and example programs see the CRBasic help.

7.3.1 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 CR6 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.

7.3.2 HTTPS

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

7.4 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 communications 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

7.5 PakBus communications

PakBus is a Campbell Scientific communications protocol. By using signed data packets, PakBus increases the number of communication and networking options available to the datalogger. The datalogger allows PakBus communications on all available communications ports. For additional information, see The Many Possibilities of PakBus Networking.

Advantages of PakBus include:

- Simultaneous communications between the datalogger and other devices.
- Peer-to-peer communications - 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 assigned 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 **Settings Editor**, or **PakBus Graph** software.

CRBasic PakBus instructions include:

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

7.6 SDI-12 communications

SDI-12 is a 1200 baud communications protocol that supports many smart sensors, probes and devices. The datalogger supports SDI-12 communications through two modes — transparent mode and programmed mode (see SDI-12 ports (p. 14) for wiring terminal information). Conflicts can occur when companion ports are used for different instructions (`TimerInput()`, `PulseCount()`, `SDI12Recorder()`, `WaitDigTrig()`). For example, if **C1** is used for the `SDI12Recorder()` instruction, **C2** cannot be used in the `TimerInput()`, `PulseCount()`, or `WaitDigTrig()` instructions.

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 (p. 100) for more information.

Programmed mode automates much of the SDI-12 protocol and provides for data recording. See SDI-12 programmed mode/recorder mode (p. 102) for more information.

CRBasic SDI-12 instructions include:

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

The datalogger uses SDI-12 version 1.4.

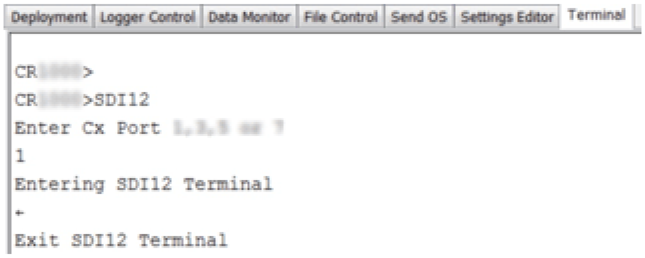
7.6.1 SDI-12 transparent mode

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 communications 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 communicating with the datalogger through a terminal emulator program such as through Device Configuration Utility or other datalogger support software. Keyboard displays cannot be used. For how-to instructions for communicating directly with an SDI-12 sensor using a terminal emulator, watch this video.

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



```
CR[redacted]>
CR[redacted]>SDI12
Enter Cx Port [redacted]
1
Entering SDI12 Terminal
←
Exit SDI12 Terminal
```

1. Press **Enter** until the datalogger responds with the prompt `CR6>`.
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 `CR6>` 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.

7.6.1.1 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. SDI-12 sensors accept addresses 0 through 9, a through z, and A through Z.
- 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).

7.6.2 SDI-12 programmed mode/recorder mode

The datalogger can be programmed to read SDI-12 sensors or act as an SDI-12 sensor itself. The `SDI12Recorder()` instruction automates sending commands and recording responses. With this instruction, the commands to poll sensors and retrieve data is 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.

7.6.3 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 to copy data from the datalogger to other Campbell Scientific dataloggers over a single data-wire interface (terminal configured for SDI-12 to terminal configured for SDI-12), or to copy 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.

When acting as a sensor, the 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.

7.6.4 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.

Table 7-1: Example power use 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

8. 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 instrumentation enclosure to provide protection, and control humidity. Desiccant 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 Tips and troubleshooting on page 118).

Also, consider checking, or posting your question to, the Campbell Scientific user forum http://www.campbellsci.com/forum. Our web site https://www.campbellsci.com has additional manuals (with example programs), FAQs, specifications and compatibility information for all of our products.

Video tutorials https://www.campbellsci.com/videos and blog articles https://www.campbellsci.com/blog are also useful troubleshooting resources.

If calibration or repair is needed, the procedure shown on: https://www.campbellsci.com/repair should be followed when sending the product.

8.1 Datalogger calibration

Campbell Scientific recommends factory recalibration every three years. During calibration, all the input terminals, peripheral and communications ports, operating system, and memory areas 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. To request recalibration for a product, see www.campbellsci.com/repair.

It is recommended that you maintain a level of calibration appropriate to the datalogger application. 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 (p. 106).

You can download and print calibration certificates for many products you have purchased by logging in to the Campbell Scientific website and going to: https://www.campbellsci.com/calcerts.

> NOTE:
> Note, you will need your product's serial number to access its certificate.

Watch an instructional video.

8.1.1 About background calibration

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

Background calibration will be disabled automatically when the scan rate is too fast for the background calibration measurements to occur in addition to the measurements in the program. The `Calibrate()` instruction can be used to override or disable background calibration. Disable 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 when background calibration is disabled.

8.2 Datalogger security

Datalogger security concerns include:

- Collection of sensitive data
- Operation of critical systems
- Networks that are accessible to 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 (p. 115) for more information.
- Disabling unused services and securing those that are used. This includes disabling HTTP, HTTPS, FTP, Telnet, and Ping network services (**Device Configuration Utility** > **Deployment** > **Network Services** tab). These services can be used to discover your datalogger on an IP network.

> NOTE:
> In more recent operating systems, FTP, Telnet, and Ping services are disabled by default.

- 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.
- Disabling FTP or setting an FTP username and password in Device Configuration Utility.
- Setting a PakBus encryption (AES-128) key in Device Configuration Utility. This forces PakBus data to be encrypted during transmission.
- Disabling HTTP/HTTPS or creating a `.csipasswd` file to secure HTTP/HTTPS (see Creating a .csipasswd file on page 109 for more information).
- Enabling HTTPS and disabling HTTP. To prevent data collection via the web interface, both HTTP and HTTPS must be disabled.
- Tracking Operating System, Run, and Program signatures.
- Encrypting program files if they contain sensitive information (see CRBasic help `FileEncrypt()` instruction or use the CRBasic Editor **File** menu, **Save and Encrypt** option).
- Hiding program files for extra protection (see CRBasic help `FileManage()` instruction).
- Monitoring your datalogger for changes by tracking program and operating system signatures, as well as CPU, USR, and CRD file contents.
- Securing the physical datalogger and power supply under lock and key.

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.

8.2.1 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 all 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. Security codes are held in the datalogger Settings Editor.

The preferred methods of enabling security include the following:

- Device Configuration Utility: Security codes are set on the **Deployment**> **Datalogger** tab.
- Network Planner: Security codes can be set as dataloggers are added to the network.

Alternatively, in CRBasic the `SetSecurity()` instruction can be used. It is only executed at program compile time. This is not recommended because 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 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**.

Table 8-1: Functions affected by security codes

Function	Security code 1 set	Security code 2 set	Security code 3 set
Datalogger program	Cannot change or retrieve		All communications prohibited
Settings editor and Status table	Writable variables cannot be changed		
Setting clock	unrestricted	Cannot change or set	
Public table	unrestricted	Writable variables cannot be changed	
Collecting data	unrestricted	unrestricted	

See Security (p. 149) for more information.

For additional information on datalogger security, see:

- 4 Ways to Make your Data More Secure
- Available Security Measures for Internet-Connected Dataloggers
- How to Use Datalogger Security Codes
- How Can Data be Made More Secure on a CRBasic PakBus Datalogger

8.2.2 Creating a .csipasswd file

The datalogger employs a security code scheme that includes three levels of security (see Datalogger security on page 106 for more information). This scheme can be used to limit access to a datalogger that is publicly available. However, the security codes are visible in Device Configuration Utility. 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`. Read/write access to the web interface requires a `.csipasswd` file. The web interface provides access to real-time and stored datalogger data. For more information on the web interface, watch an instructional video.

When a file named `.csipasswd` is stored on the datalogger CPU drive, basic access authentication is enabled in the datalogger and read/write access to the web interface can be defined. Multiple user accounts with differing levels of access can be defined for one datalogger. Four levels of access are available:

- **None**: Disable a user account.
- **Read Only**: Data collection is unrestricted. Clock and writable variables cannot be changed. Programs cannot be viewed, stopped, deleted, or retrieved.
- **Read/Write**: Data collection is unrestricted. Clock and writable variables can be changed. Programs cannot be viewed, stopped, deleted, or retrieved.
- **All**: Data collection is unrestricted. Clock and writable variables can be changed. Programs can be viewed, stopped, deleted and retrieved.

> NOTE:
> All levels of access allow data collection.

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 the **Edit .csipasswd File** button.

3. Define user accounts and access levels.
4. Click **Apply**. The `.csipasswd` file is automatically saved to the datalogger CPU drive.

When a `.csipasswd` file is used, the 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 prompt the web client to display a username and 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 previously, 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, 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 username and password employed by the basic access authentication will eliminate the need for entering the numeric security code.

8.2.2.1 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).

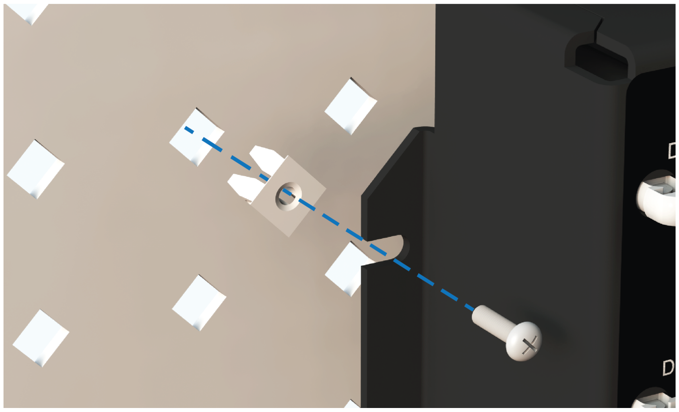
8.3 Datalogger enclosures

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 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 datalogger.
2. Holding the datalogger to the backplate, screw the screws into the nylon anchors.



8.4 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 (p. 105).

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.

- 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.

See Power requirements (p. 214) 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.

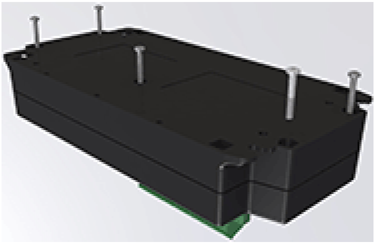
NOTE:
The **Status** field **Battery** value and the destination variable from the `Battery()` instruction (often called `batt_volt`) in the **Public** table reference the external battery voltage.

8.4.1 Replacing the internal battery

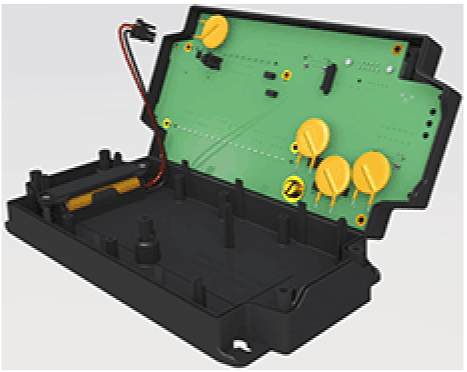
It is recommended that you send the datalogger in for scheduled calibration, which includes internal battery replacement (see Datalogger calibration on page 105).

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

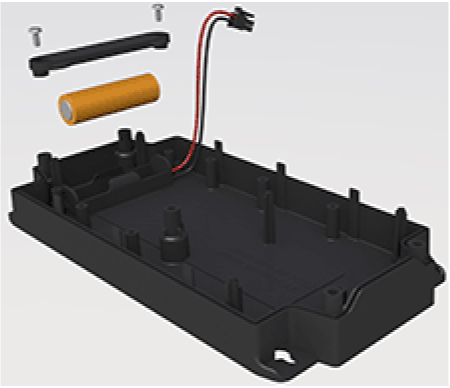
1. Remove the screws from the back panel.



2. Pull the bottom and top of the datalogger apart.
3. Disconnect the battery connector.



4. Remove and replace the battery.

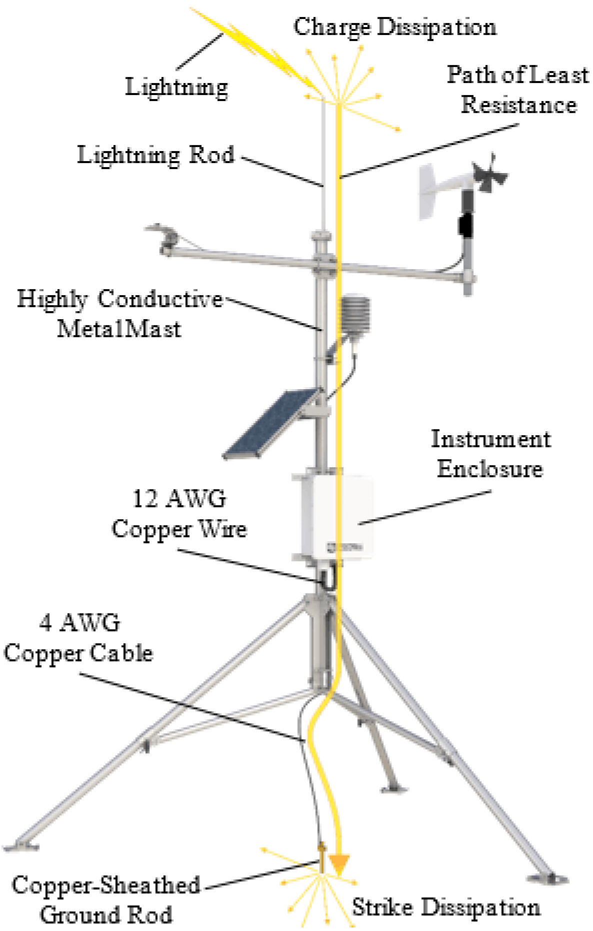


5. Reassemble the datalogger.

8.5 Electrostatic discharge and lightning protection

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

Electrostatic discharge (ESD) can originate from several sources, the most common and destructive are 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 consists of 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:



All critical inputs and outputs on the datalogger are ESD protected to 75 V. To be effective, the earth ground lug must be properly connected to earth (chassis) ground.

Communications ports are another path for transients. You should provide communications 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 earth (chassis) ground.

For detailed information on grounding, see Grounds (p. 11).

8.6 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 estimated by dividing the battery capacity (ampere hours) by the average system current drain (amperes).

For more information see:

- **Application Note** - Power Supplies
- Power Budget Spreadsheet
- **Video Tutorial** - Power Budgeting

See also:

- Power input (p. 8)
- Power output (p. 10)
- Power requirements (p. 214)
- Power output specifications (p. 217)

8.7 Updating the operating system

Campbell Scientific posts operating system (OS) updates at www.campbellsci.com/downloads when they become available. It is recommended that before deploying instruments, you check operating system versions and update them 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.

WARNING:
Because sending an OS resets datalogger memory and resets all settings on the datalogger to factory defaults, data loss will certainly occur. Depending on several factors, the datalogger may also become incapacitated for a time.

TIP:
It is recommended that you retrieve data from the datalogger and back up your programs and settings before updating your OS. To collect data using LoggerNet, connect to your datalogger and click **Collect Now**. To backup your datalogger, connect to it in Device Configuration Utility, click the **Backup** menu and select **Backup Datalogger**.

8.7.1 Sending an operating system to a local datalogger

Send an OS using Device Configuration Utility. This method requires a direct connection between your datalogger and computer.

1. Download the latest Operating System at www.campbellsci.com/downloads.
2. Locate the .exe download and double-click to run the file. This will extract the .obj OS file to the **C:\Campbellsci\Lib\OperatingSystems** folder.
3. Supply power to the datalogger. If connecting via USB for the first time, you must first install USB drivers by using Device Configuration Utility (select your datalogger, then on the main page, click **Install USB Driver**). Alternately, you can install the USB drivers using EZ Setup. A USB connection supplies 5 V power (as well as a communication link), which is adequate for setup, but a 12 V battery will be needed for field deployment.
4. Physically connect your datalogger to your computer using a USB cable, then open Device Configuration Utility and select your datalogger.
5. Select the communications port used to communicate with the datalogger from the **COM Port** list (you do not need to click **Connect**).
6. Click the **Send OS** tab. At the bottom of the window, click **Start**.
7. On the **Avoid Conflicts with the Local Server** window, click **OK**.
8. Navigate to the **C:\Campbellsci\Lib\OperatingSystems** folder.
9. Ensure **Datalogger Operating System Files (*.obj)** is selected in the **Files of type** list, select the new OS .obj file, and click **Open** to update the OS on the datalogger.

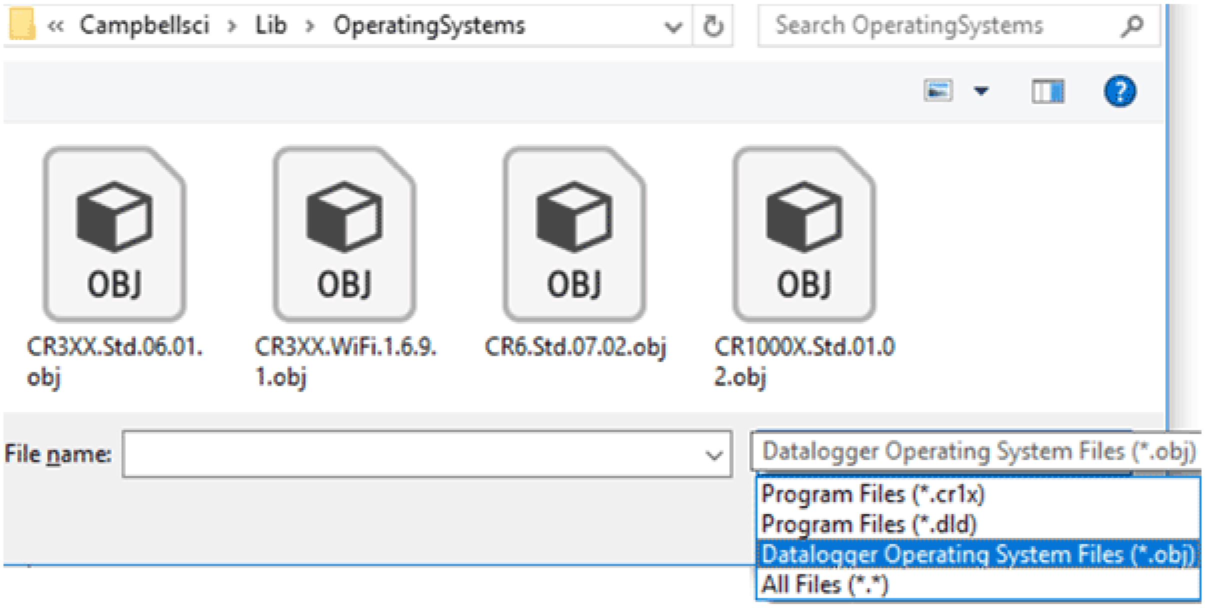
Watch a video: Sending an OS to a Local Datalogger.

8.7.2 Sending an operating system to a remote datalogger

> NOTE:
> This information applies to CR6 dataloggers with serial numbers 7502 and newer. These dataloggers have two blue stripes on the label. For CR6 dataloggers with serial numbers 7501 and older, see: Video | Sending an OS to a Remote Datalogger.

If you have a datalogger that is already deployed, you can update the OS over a telecommunications link by sending the OS to the datalogger as a program. In most instances, sending an OS 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.

1. Download the latest Operating System at www.campbellsci.com/downloads.
2. Locate the .exe download and double-click to run the file. This will extract the .obj OS file to the **C:\Campbellsci\Lib\OperatingSystems** folder.
3. Using datalogger support software, connect to your datalogger.
   - LoggerNet users, select **Main** and click **Connect** on the LoggerNet toolbar, select the datalogger from the **Stations** list, then click **Connect**.
   - PC200W and PC400 users, select the datalogger from the list and click **Connect**.
4. Select **File Control** at the top of the Connect window.
5. Click **Send** at the top of the File Control window.
6. Navigate to the **C:\Campbellsci\Lib\OperatingSystems** folder.
7. Ensure **Datalogger Operating System Files (*.obj)** is selected in the files of type list, select the new OS .obj file, and click **Open** to update the OS on the datalogger.



Note the following precautions when sending as a program:

- 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.

9. Tips and troubleshooting

Start with these basic procedures if a system is not operating properly.

1. Using a voltmeter, check the voltage of the primary power source at the **CHG** and **BAT** terminals on the face of the datalogger, it should be 10 to 18 Vdc. If connecting to a power source via the **CHG** terminals, voltage measured should be 16 to 32 Vdc.
2. Check wires and cables for the following:
   - Incorrect wiring connections. Make sure each sensor and device are wired to the terminals 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 in. To troubleshoot, create a simpler 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 125 for more information).

For additional troubleshooting options, see:

Also, consider checking, or posting your question to, the Campbell Scientific user forum http://www.campbellsci.com/forum. Our web site https://www.campbellsci.com has additional manuals (with example programs), FAQs, specifications and compatibility information for all of our products.

Video tutorials https://www.campbellsci.com/videos and blog articles https://www.campbellsci.com/blog are also useful troubleshooting resources.

9.1 Checking station status

View the condition of the datalogger using **Station Status**. Here you 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.

9.1.1 Viewing station status

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

- From LoggerNet: Click **Connect** 🔗, then **Station Status** ☑ to view the **Summary** tab.
- From PC200W and PC400: Select the **Datalogger** menu and **Station Status** ☑ to view the **Summary** tab.

9.1.2 Watchdog errors

Watchdog errors indicate that the datalogger has crashed and reset itself. Experiencing a few watchdog errors is normal. You can reset the Watchdog error counter in the **Station Status** > **Status Table**.

TIP:
Before resetting the counter, make note of the number accumulated and the date.

Watchdog errors could be due to:

- Transient voltage
- Incorrectly wired or malfunctioning sensor
- Poor ground connection on the power supply
- Numerous `PortSet()` instructions back-to-back with no delay
- High-speed serial data on multiple ports with very large data packets or bursts of data

The error "Results for Last Program Compiled: Warning: Watchdog Timer IpTask Triggered" can result from:

- The IP communications on the datalogger got stuck, and the datalogger had to reboot itself to recover. Or communications failures may cause the datalogger to reopen the IP connections more than usual. Check your datalogger operating system version; recent operating system versions have improved stability of IP communications.

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

- Memory corruption
- Operating System problem
- Hardware problem
- IP communications problem

9.1.3 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 (p. 124).

9.1.4 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.

9.1.5 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.

9.1.6 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 out-of-bounds errors at compile time. However, it is not always possible; so, these errors may occur during runtime. Variable-out-of-bounds errors are always caused by programming problems.

9.1.7 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 (p. 8)
- Power requirements (p. 214)

9.2 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 ÷ 0.

NAN is a constant that can be used in expressions. This is shown in the following code snip that sets a CRBasic variable to False when the wind direction is **NAN**:

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

If an output processing instruction encounters a **NAN** in the values being processed, **NAN** will be stored. For example, if one measurement in a data storage interval results in **NAN**, then the average, maximum and minimum will record **NAN**.

> NOTE:
> 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 *variable* = **NAN** in the `DisableVar` parameter to discard **NAN**s from affecting the other good values.

9.3 Timekeeping

Measurement of time is an essential datalogger function. Time measurement with the onboard clock enables the datalogger to run on a precise interval, 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 (p. 213). An internal lithium battery backs the clock when the datalogger is not externally powered (see Internal battery on page 111 for more information).

9.3.1 Clock best practices

When setting the clock with LoggerNet, initiate it manually during a maintenance period when the datalogger is not actively writing to Data Tables. Click the **Set** button in the Clocks field of the LoggerNet Connect Screen.

If you are going to use automated clock check with LoggerNet (clock settings can be found on the LoggerNet Setup Standard View **Clock** tab). 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.

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

9.3.2 Time stamps

A measurement without an accurate time reference often has little meaning. Data collected from dataloggers is 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 datalogger clock. 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 a time-consuming part of the program, 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.

9.3.3 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. A close approximation is:

> time skew = reps * (settling time + integration time + ADC time) + instruction setup time
> where ADC time equals 170 µs, and instruction setup time is 15 µs.
> If reps (repetitions) > 1 (multiple measurements by a single instruction), no setup time is required. If reps = 1 for consecutive voltage instructions, include the setup time for each instruction.

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 of the scan when 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.

9.4 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.

9.4.1 Program does not compile

When a program is compiled, the CRBasic Editor checks the program for syntax errors and other inconsistencies. The results of the check are displayed in a message window at the bottom of the main window. If an error can be traced to a specific line in the program, the line number will be listed before the error. Double-click an error preceded by a line number and that line will be highlighted in the program editing window. Correct programming errors and recompile the program.

Occasionally, the CRBasic Editor compiler states that a program compiles OK; however, the program may not compile in the datalogger itself. This is rare, but reasons may include:

- The datalogger has a different operating system than 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 (p. 115).
- 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 the CPU drive for copies of old programs. 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.
    - Check the USR drive size. If it is too large it may be using memory needed for the program.

  - Ensure a memory card is available when a program is attempting to access the CRD drive.

9.4.2 Program compiles but does not run correctly

If the program compiles but does not run correctly, timing discrepancies may be 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 datalogger, but it can be a useful tool for fine-tuning a program.

9.5 Troubleshooting Radio Communications

If there are intermittent communication problems when connecting via radio, there may be another network in the area causing interference. To help remove the interference, use Device Configuration Utility to change the **Network ID** and **RF Hop Sequence** in all RF407, RF412, and RF422 radios within a network (standalone or included in a datalogger) to another value. Each of these settings must have the same value in all radios and dataloggers within a network. For example, the **Network ID** in all devices could be set to **1726**, and the **RF Hop Sequence** in all devices could be set to **1**. The **Network ID** can be any number between 0 and 32767. The **RF Hop Sequence** can be any number between 0 and 7 in an RF407 or RF412 network; it can be any number between 0 and 9 in an RF422 network.

See also Radio communications (p. 27). For specifications information, see RF radio option specifications (p. 233).

9.6 Resetting the datalogger

A datalogger reset is sometimes referred to as a "memory reset." Backing up the current datalogger configuration before a reset makes it easy 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

9.6.1 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 communications 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 a **Terminal Emulator** program (type REBOOT <Enter>).

9.6.2 Program send reset

Final-data memory is 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**.

> **TIP:**
> It is good practice to always collect data before sending a program to the datalogger.

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 CR6 has the `PreserveVariables()` and `PreserveOneVariable()` instructions.

9.6.3 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.
- CR1000KD Keyboard/Display add-on: **Data** > **Reset Data Tables**.

9.6.4 Formatting drives

CPU, USR, 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 data associated with the program are erased. Drive formatting is performed through the datalogger support software **File Control** > **Format** command.

9.6.5 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** (see FullMemReset (p. 153)). A full memory reset does the following:

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

Full memory reset does not affect the CRD drive directly. Subsequent user program uploads, however, can erase CRD. See Updating the operating system (p. 115) for more information.

9.7 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 maintenance tool kit.) Ensure correct polarity of the connection. If the datalogger powers up and works, troubleshoot the datalogger power supply.

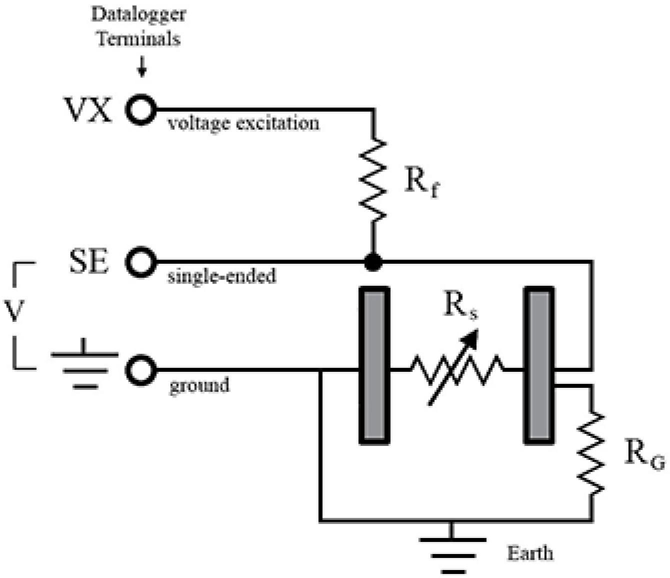
When diagnosing or adjusting power equipment supplied by Campbell Scientific, it is recommended you consider:

- 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. For additional information, see Power budgeting (p. 114).

9.8 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 that 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 (p. 11).

9.9 Improving voltage measurement quality

The following topics discuss methods of generally improving voltage measurements:

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

9.9.1 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 the following noise reduction features 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 (p. 141).

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. Some 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 (p. 11) and Minimizing ground potential differences (p. 131).

9.9.2 Minimizing ground potential differences

Low-level, single-ended voltage measurements (<200 mV) are sensitive to ground potential fluctuation due to changing return currents from **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 (p. 11). To take advantage of this design, observe the following rules:

- Connect grounds associated with **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 wires with large current flow, tying the problem wires into terminals next to terminals configured for excitation and pulse-count should help. Problem wires can also be tied directly to the ground lug to minimize induced single-ended offset voltages.

9.9.2.1 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 will be 1 mV greater than the thermocouple output. With a Type T (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 wire resistance result in different ground potentials at the two instru-

ments. 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 (p. 139).

9.9.3 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 detect an open circuit in the following situations:

- When 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.

9.9.4 Minimizing power-related artifacts

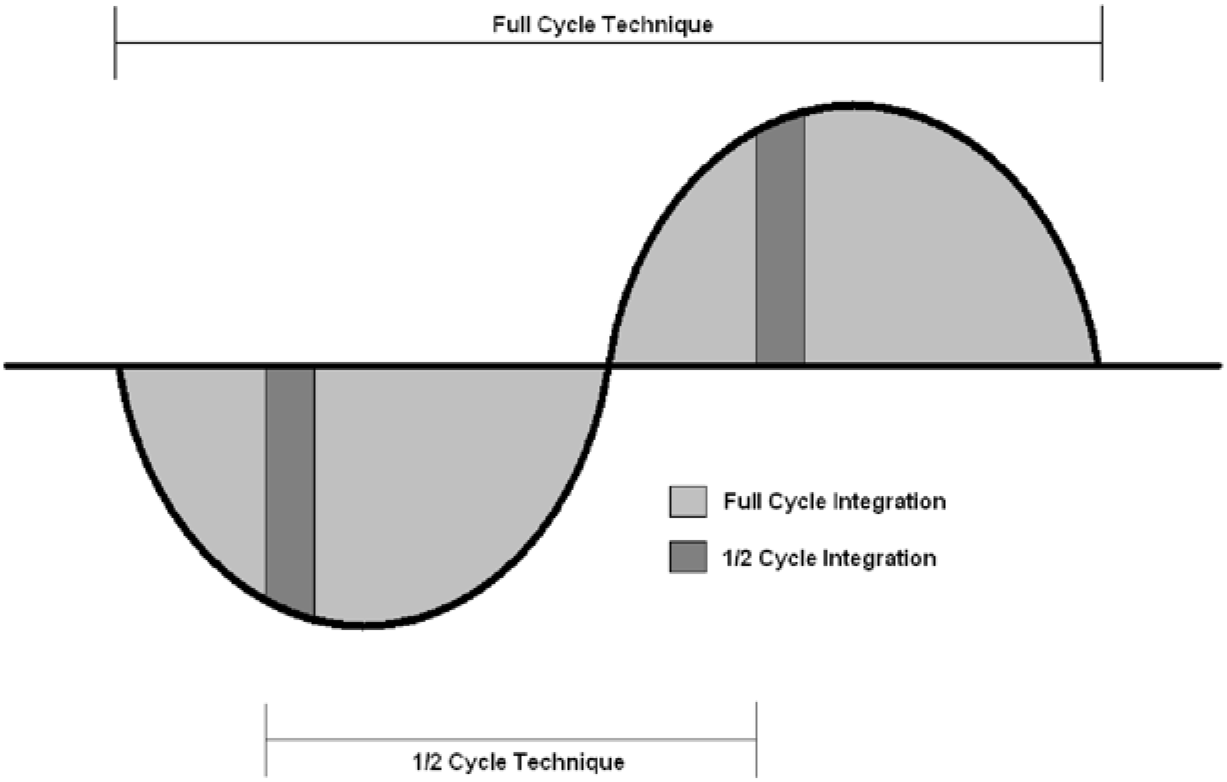
Some Vac-to-Vdc power converters produce switching noise or ac ripple as an artifact of the ac-to-dc rectification process. Excessive switching noise on the output side of a power supply can increase measurement noise, and so increase measurement error. Noise from grid or mains power also may be transmitted 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 50 Hz or 60 Hz first notch frequency (f_{N1}) option for CRBasic analog input measurement instructions often improves rejection of noise sourced from power mains. The CRBasic standard deviation output instruction, `StdDev()`, can be used to evaluate measurement noise.

The datalogger includes adjustable digital filtering, which serves two purposes:

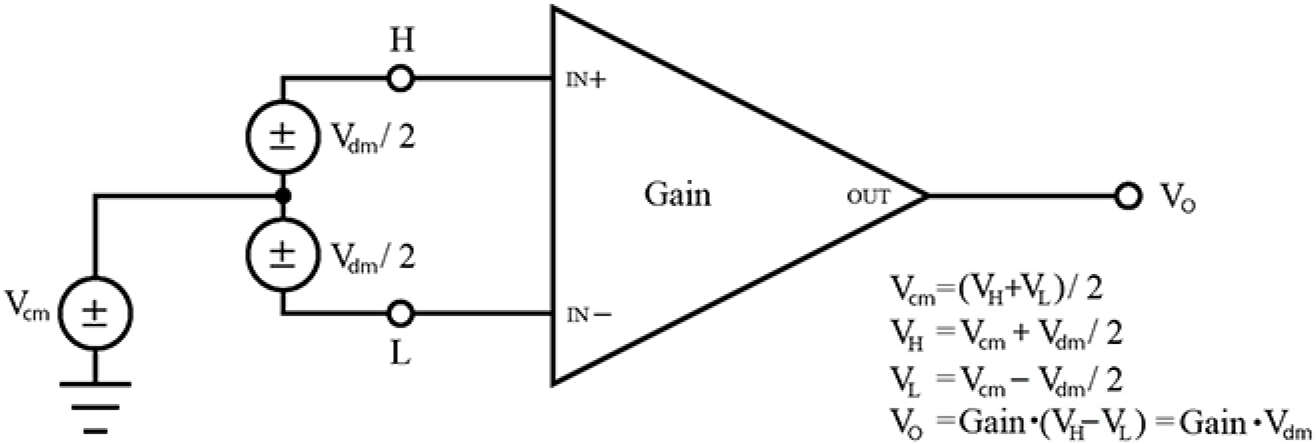
- Arrive as close as possible to the true input signal
- Filter out measurement noise at specific frequencies, the most common being noise at 50 Hz or 60 Hz, which originate from mains-power lines.

Filtering time is inversely proportional to the frequency being filtered.



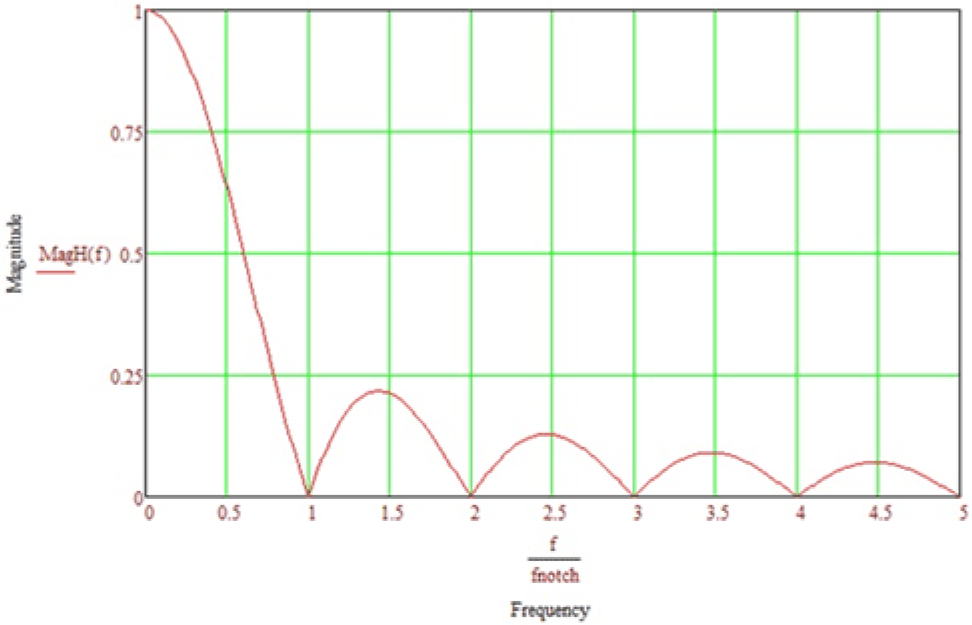
9.9.4.1 Minimizing electronic noise

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.



9.9.5 Filtering to reduce measurement noise

An adjustable filter is applied to analog measurements, reducing signal components at selected frequencies. The following figure shows the filter frequency response. Using the first notch frequency (`fN1`) parameter, users can select the placement of the filter notches. The first notch falls at the specified `fN1`, and subsequent notches fall at integer multiples of `fN1`. Commonly, `fN1` is set at 50 or 60 Hz to filter 50 or 60 Hz signal components, reducing noise from ac power mains.



Filtering comes at the expense of measurement time. The time required for filtering is equal to $1/f_{N1}$. For example, setting `fN1` equal to 50 will require 1/50 sec (20 ms) for filtering. As `fN1` is set to smaller values, random noise in the measurement results decreases, while measurement time increases. The total time required for a single result includes settling + filtering + overhead.

Consult the following technical paper at www.campbellsci.com/app-notes for in-depth treatment of measurement noise: Preventing and Attacking Measurement Noise Problems.

9.9.5.1 CR6 filtering details

The datalogger utilizes a sigma-delta ADC that outputs digitized data at a rate of 93750 samples per second. User-specified filtering is achieved by averaging several samples from the ADC. Recall that averaging the signal over a period of $1/f_{N1}$ seconds will filter signal components at f_{N1} Hz. The final result, then, is the average calculated from $93750/f_{N1}$ samples. For example, if `fN1` is set to 50 Hz, 1875 samples (93750 / 50) are averaged to generate the final filtered result.

The actual f_{N1} may deviate from the user-specified setting since a whole integer number of samples must be averaged. For example, if `fN1` is set to 60 Hz, 1563 samples (93750 / 60 = 1562.5) will be averaged to produce the filtered result. The rounding of 1562.5 to 1563 moves the actual f_{N1} to 93750 / 1563 = 59.98 Hz.

9.9.6 Minimizing settling 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 the `SettlingTime` parameter is set to `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 cable lengths, use the following general practices to minimize settling errors:

- Do not use leads 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.

9.9.6.1 Measuring settling time

Settling time for a particular sensor and cable can be measured with the CR6. 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 2: Measuring Settling Time

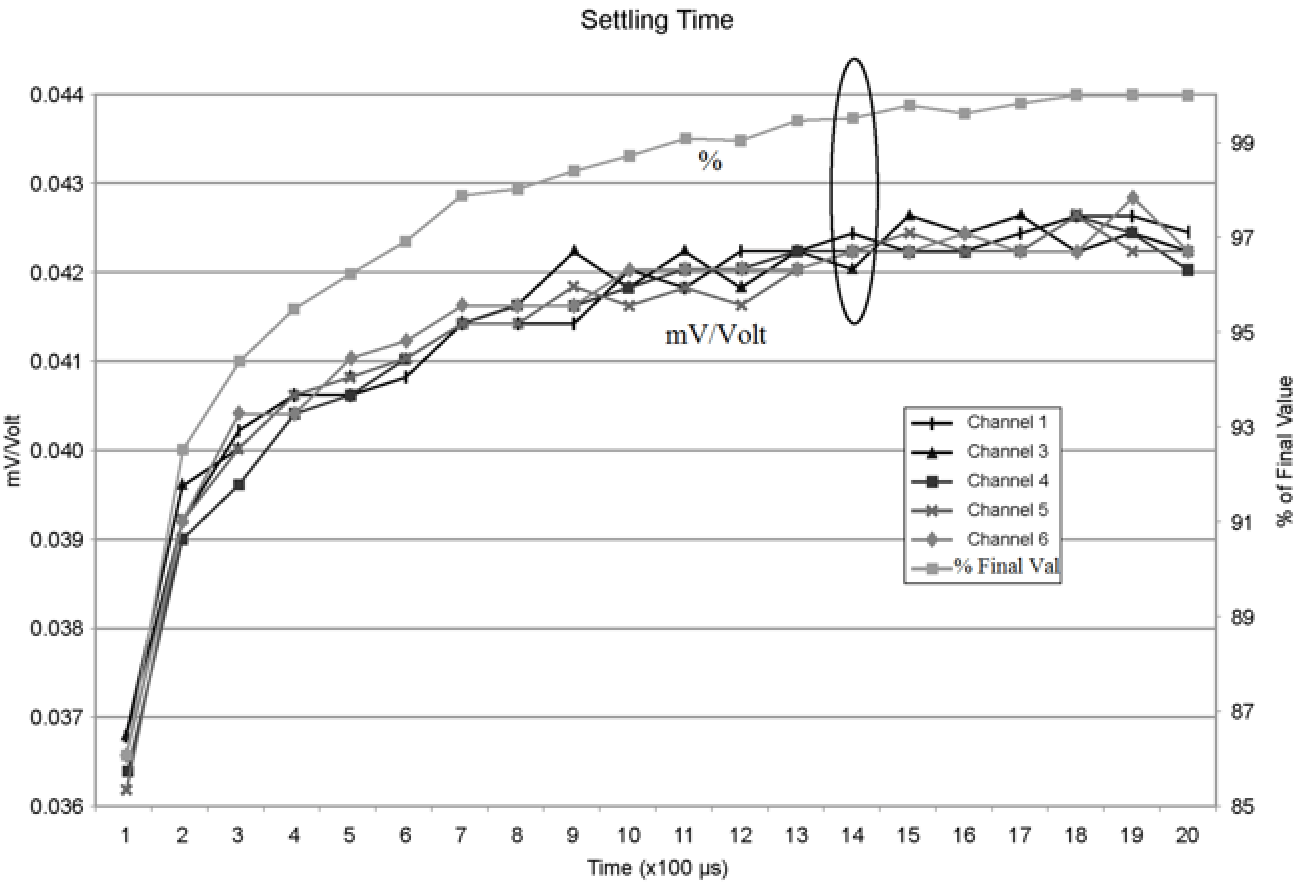
```
'This program example demonstrates the measurement of settling time
'using a single measurement instruction multiple times in succession.
Public PT(20) 'Variable to hold the measurements
DataTable(Settle,True,100)
  Sample(20,PT(),IEEE4)
EndTable
BeginProg
  Scan(1,Sec,3,0)
    BrFull(PT(1), 1,mV200,U1,U11,1,2500,True,True, 100,15000,1.0,0)
    BrFull(PT(2), 1,mV200,U1,U11,1,2500,True,True, 200,15000,1.0,0)
    BrFull(PT(3), 1,mV200,U1,U11,1,2500,True,True, 300,15000,1.0,0)
    BrFull(PT(4), 1,mV200,U1,U11,1,2500,True,True, 400,15000,1.0,0)
    BrFull(PT(5), 1,mV200,U1,U11,1,2500,True,True, 500,15000,1.0,0)
    BrFull(PT(6), 1,mV200,U1,U11,1,2500,True,True, 600,15000,1.0,0)
    BrFull(PT(7), 1,mV200,U1,U11,1,2500,True,True, 700,15000,1.0,0)
    BrFull(PT(8), 1,mV200,U1,U11,1,2500,True,True, 800,15000,1.0,0)
    BrFull(PT(9), 1,mV200,U1,U11,1,2500,True,True, 900,15000,1.0,0)
    BrFull(PT(10),1,mV200,U1,U11,1,2500,True,True,1000,15000,1.0,0)
    BrFull(PT(11),1,mV200,U1,U11,1,2500,True,True,1100,15000,1.0,0)
    BrFull(PT(12),1,mV200,U1,U11,1,2500,True,True,1200,15000,1.0,0)
    BrFull(PT(13),1,mV200,U1,U11,1,2500,True,True,1300,15000,1.0,0)
    BrFull(PT(14),1,mV200,U1,U11,1,2500,True,True,1400,15000,1.0,0)
    BrFull(PT(15),1,mV200,U1,U11,1,2500,True,True,1500,15000,1.0,0)
    BrFull(PT(16),1,mV200,U1,U11,1,2500,True,True,1600,15000,1.0,0)
    BrFull(PT(17),1,mV200,U1,U11,1,2500,True,True,1700,15000,1.0,0)
    BrFull(PT(18),1,mV200,U1,U11,1,2500,True,True,1800,15000,1.0,0)
    BrFull(PT(19),1,mV200,U1,U11,1,2500,True,True,1900,15000,1.0,0)
    BrFull(PT(20),1,mV200,U1,U11,1,2500,True,True,2000,15000,1.0,0)
    CallTable Settle
  NextScan
EndProg
```

The first six measurements are shown in the following table:

Table 9-1: Example data from Measuring Settling Time Program

Timestamp	Record Number	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 and 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 µs is determined to be adequate.



9.9.7 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 an analog voltage measurement. Accuracy is defined as follows:

accuracy = percent-of-reading + offset

where percents-of-reading and offsets are displayed in the Analog measurements specifications (p. 221).

> **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.

9.9.7.1 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: approximately 1050 mV
- CRBasic measurement instruction: `VoltDiff()`
- Programmed input-voltage range (`Range`) **: mV 500 0** (±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

percent-of-reading = 1050 mV • ±0.04%

= ±0.42 mV

and

offset = 5 µV

Therefore,

accuracy = ±(0.42 mV + 5 µV) = ±0.425 mV

9.9.8 Minimizing offset voltages

Voltage offset can be the source of significant error. For example, an offset of 3 µV 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 131 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 datalogger wiring panel, such as another manufacturer's communications modem, or a sensor that requires a lot of power. Currents greater than 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 (⏚) 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 datalogger 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 **U1/U2**. 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 table lists some of the tools available to minimize the effects of offset voltages:

Table 9-2: Offset voltage compensation options

CRBasic Measurement Instruction	Input Reversal (RevDiff=True)	Excitation Reversal (RevExx=True)	Measure Offset During Measurement (MeasOff=True)	Measure Offset During Background Calibration (RevDiff=False) (RevEx=False) (MeasOff=False)
`BrHalf()`		✓		✓
`BrHalf3W()`		✓		✓
`BrHalf4W()`	✓	✓		✓
`BrFull()`	✓	✓		✓
`BrFull6W()`	✓	✓		✓
`Resistance()`	✓	✓		✓
`Resistance3W()`	✓	✓		
`TCDiff()`	✓			✓
`TCSe()`			✓	✓
`VoltDiff()`	✓			✓
`VoltSe()`			✓	✓

9.9.8.1 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 or current 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 mV – (–4.997 mV) = 10.000 mV

10.000 mV / 2 = 5.000 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.

9.9.8.2 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 106 for more information). Single-ended measurement instructions `VoltSE()` and `TCSe()` include the `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()` or `TCSe()` 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()` or `TCSe()` 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.

9.10 Field calibration

Calibration increases accuracy of a measurement device by adjusting its output, or the measurement of its output, to match independently verified quantities. Adjusting sensor output directly is preferred, but not always possible or practical. By adding the `FieldCal()` or `FieldCalStrain()` instruction to a CRBasic program, measurements of a linear sensor can be adjusted by modifying the programmed multiplier and offset applied to the measurement, without modifying or recompiling the CRBasic program. For more information, see the CRBasic help.

9.11 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 (via an add-on peripheral). 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

9.12 File name and resource errors

The maximum file name size 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.

9.13 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 lose 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.

9.14 Information tables and settings (advanced)

Information tables and settings consist of fields, settings, and system information essential to setup, programming, and debugging of many advanced CR6 systems. In many cases, the info tables and settings keyword can be used to pull that field into a running CRBasic program. There are several locations where this system information and settings are stored or changed:

- **Status table**: The **Status** table is an automatically created data table. View the **Status** table by connecting the datalogger to your computer (see Connecting the datalogger to a computer on page 41 for more information) **Station Status** ☑, then clicking the **Status Table** tab.
- **DataTableInfo table**: The **DataTableInfo** table is automatically created when a program produces other data tables. View the **DataTableInfo** table by connecting the datalogger to your computer (see Connecting the datalogger to a computer on page 41 for more information)
  - PC200W and PC400 users, click the **Monitor Data** tab and add the **DataTableInfo** to display it.
  - LoggerNet users, select **DataTableInfo** from the **Table Monitor** list.
- **Device Configuration Utility Settings**: Access settings, using Device Configuration Utility. Clicking on a setting in Device Configuration Utility also provides information about that setting.
- **Terminal Mode**: A list of setting field names is also available from the datalogger's terminal mode (from **Device Configuration Utility**, click the **Terminal** tab) using command "**F**".
- Setting values may be accessed programmatically using `Tablename.Fieldname` syntax. For example: `*Variable* = Settings.Fieldname`.

Communications and processor bandwidth are consumed when generating the **Status** and other information tables. If datalogger is very tight on processing time, as may occur in very fast, long, or complex operations, retrieving these tables repeatedly may cause skipped scans.

Settings that affect memory usage force the datalogger program to recompile, which may cause loss of data. Before changing settings, it is a good practice to collect your data (see Collecting data on page 46 for more information). Examples of settings that force the datalogger program to recompile:

- IP address
- IP default gateway
- Subnet mask
- PPP interface
- PPP dial string
- PPP dial response
- Baud rate change on control ports
- Maximum number of TLS server connections
- USR drive size
- PakBus encryption key
- PakBus/TCP server port
- HTTP service port
- FTP service port
- PakBus/TCP service port
- PakBus/TCP client connections
- Communications allocation

9.14.1 Information tables directories

Use the following links to help you navigate through the Information Tables and Settings system:

- Find the PakBus address of the datalogger (p. 146)
- Communications (p. 147)
- Data (p. 149)
- OS and hardware versions (p. 149)
- Power monitoring (p. 149)
- Security (p. 149)
- Signatures (p. 149)

9.14.1.1 Frequently used

Action	Keywords
Find the PakBus address of the datalogger	PakBusAddress
See messages pertaining to compilation of the CRBasic program running in the datalogger	CompileResults

Action	Keywords
Programming Errors	ProgErrors ProgSignature SkippedScan StartUpCode VarOutOfBound
Data tables	DataFillDays SkippedRecord
Memory	MemoryFree MemorySize
Datalogger auto-resets	WatchdogErrors
Operating system	OSDate OSSignature OSVersion
Power	Battery LithiumBattery Low12VCount

9.14.1.2 Communications

For detailed information on communication protocols, see Communications (p. 87).

General communications

Baudrate CommsMemAlloc CommsMemFree	RS232Handshaking RS232Power RS232Timeout

PakBus communications

Beacon CentralRouters IsRouter MaxPacketSize Neighbors	PakBusAddress PakBusEncryptionKey PakBusPort PakBusRoutes	PakBusTCPClients PakBusTCPPassword RouteFilters Verify

TCP_IP communications

- CSIOnetEnable
- DNS
- EthernetPower
- FTPEnabled
- FTPPassword
- FTPPort
- FTPUserName
- HTTPEnabled
- HTTPPort
- IPAddressCSIO
- IPAddressEth
- IPGateway
- IPGatewayCSIO
- IPMaskCSIO
- IPMaskEth
- IPTraceCode
- IPTraceComport
- PingEnabled
- TelnetEnabled
- UDPBroadcastFilter

RF407-Series radio communications

- RadioChanMask
- RadioEnable
- RadioHopSeq
- RadioMAC
- RadioModel
- RadioModuleVer
- RadioNetID
- RadioProtocol
- RadioPwrMode
- RadioRetries
- RadioRSSI
- RadioRSSIAddr
- RadioStats
- RadioTxPwr

RF451 radio communications

- RadioCarrier
- RadioDataRate
- RadioDiag
- RadioEnable
- RadioFirmwareVer
- RadioFreqKey
- RadioFreqRepeat
- RadioFreqZone
- RadioHopSize
- RadioHopVersion
- RadioLowPwr
- RadioMaxPacket
- RadioMinPacket
- RadioMMSync
- RadioModOS
- RadioModuleVer
- RadioNetID
- RadioOpMode
- RadioPacketRepeat
- RadioRepeaters
- RadioRetryOdds
- RadioRetryTimeout
- RadioRxSubID
- RadioSlaveRepeat
- RadioSlaveRetry
- RadioTxPwr
- RadioTxRate
- RadioTxSubID

Wi-Fi communications

- WiFiChannel
- WiFiConfig
- WiFiEapMethod
- WiFiEapPassword
- WiFiEapUser
- WiFiEnable
- WiFiPassword
- WiFiPowerMode
- WiFiSSID
- WiFiStatus
- WiFiTxPowerLevel
- WiFiNetworks

9.14.1.3 Background calibration

See About background calibration (p. 106) for more information.

CalGain CalOffset	ErrorCalib LastSystemScan MaxSystemProcTime	SkippedSystemScan SystemProcTime

9.14.1.4 Data

DataFillDays DataRecordSize	DataTableName SecsPerRecord	SkippedRecord

9.14.1.5 OS and hardware versions

OSDate OSSignature	OSVersion RevBoard	SerialNumber

9.14.1.6 Power monitoring

See Power output (p. 10) for more information.

Battery LithiumBattery	Low12VCount

9.14.1.7 Security

See Datalogger security (p. 106) for more information.

PakBusTCPPassword PakBusEncryptionKey Security

9.14.1.8 Signatures

OSSignature	ProgSignature	RunSignature

9.14.2 Information tables and settings descriptions

Information tables and settings consist of fields, settings, and system information essential to setup, programming, and debugging of many advanced CR6 systems. There are several locations where this system information and settings are stored or changed.

- DataTableInfo Table System Information and Settings
- Status Table Information and Settings

- Device Configuration Utility Settings

9.14.2.1 DataTableInfo table system information and settings

The **DataTableInfo** table is automatically created when a program produces other data tables. View the **DataTableInfo** table by connecting the datalogger to your computer (see Connecting the datalogger to a computer on page 41 for more information)

- PC200W and PC400 users, click the **Monitor Data** tab and add the **DataTableInfo** to display it.
- LoggerNet users, select **DataTableInfo** from the **Table Monitor** list.

Keyword	Information and Location
DataFillDays	Reports the time required to fill a data table. Each table has its own entry in a two-dimensional array. First dimension is for on-board memory. Second dimension is for card memory. • Numeric data type • Read only
DataRecordSize	Reports the number of records allocated to a data table. • Numeric data type • Read only
DataTableName	Reports the names of data tables. Array elements are in the order the data tables are declared in the CRBasic program. • String data type • Read only
RecNum	Record number is incremented when any one of the DataTableInfo fields change, for example **SkippedRecord**. • Long data type • Read only
SecsPerRecord	Reports the data output interval for a data table. • Numeric data type • Read only

Keyword	Information and Location
SkippedRecord	Reports how many times records have been skipped in a data table. Array elements are in the order that data tables are declared in the CRBasic program. Enter **0** to reset. • Numeric data type
TimeStamp	Scan time that a record was generated • NSEC data type • Read only

9.14.2.2 Status Table system information and settings

The **Status** table is an automatically created data table and most of the settings are read only. View the **Status** table by connecting the datalogger to your computer (see Connecting the datalogger to a computer on page 41 for more information)

Keyword	Information and Location
Battery	Voltage (Vdc) of the battery powering the system. Updates when viewing the **Status** table or via program code. • Numeric data type • Read only
BuffDepth	Shows the current pipeline mode processing buffer depth, which indicates how far the processing task is currently behind the measurement task. Updated at the conclusion of scan processing, prior to waiting for the next scan. • Numeric data type • Read only
CalGain	Array of floating-point values reporting calibration gain (mV) for each integration / range combination. • Numeric data type • Read only
CalOffSet	Displays the offset calibration factor for the different voltage ranges. • Numeric data type • Read only

Keyword	Information and Location
CalVolts	Array of floating-point values reporting a factory calibrated correction factor for the different voltage ranges. • Numeric data type • Read only
CardStatus	Contains a string with the most recent status information for the removable memory card. • String data type • Read only
ChargeInput	Voltage on the **CHG** terminals. Updates when background calibration executes. • Numeric data type • Read only
ChargeState	State of the **CHG** terminals. Responses include Regulator Fault, No Charge, Current Limited, Float Charging, Low Charge Input. Updates when background calibration executes (see Power Input for more information). • String data type • Read only
CommsMemFree	Memory allocations for communications. Numbers outside of parentheses reflect current memory allocation. Numbers inside parentheses reflect the lowest memory size reached. • Numeric data type • Read only
CompileResults	Contains error messages generated at compilation or during runtime. Updated after compile. Also appended to at run time for run time errors such as variable out of bounds. • String data type • Read only

Keyword	Information and Location
ErrorCalib	Number of erroneous calibration values measured. Erroneous values are discarded. Updated at startup. • Numeric data type • Read only
FullMemReset	Enter **98765** to start a full-memory reset. • Numeric data type
IxResistor	Factory-calibrated correction factor (ohms) applied to resistance measurements. Normal value is near 1000 ohms. Updates at start up. • Float data type • Read only
LastSystemScan	Reports the time of the of the last auto (background) calibration, which runs in a hidden slow-sequence type scan. See MaxSystemProcTime, SkippedSystemScan, and SystemProcTime. • Numeric data type • Read only
LithiumBattery	Voltage of the internal lithium battery. Updated at CR6 power up. For battery information, see Internal battery (p. 111). • Numeric data type • Read only
Low12VCount	Counts the number of times the primary CR6 supply voltage drops below ≈9.0 Vdc. Updates with each **Status** table update. Range = 0 to 99. Reset by entering 0. Incremented prior to scan (slow or fast) with measurements if the internal hardware signal is asserted. • Numeric data type • Read only
MaxBuffDepth	Maximum number of buffers the CR6 will use to process lagged measurements. • Numeric data type

Keyword	Information and Location
MaxProcTime	Maximum time (µs) required to run through processing for the current scan. Value is reset when the scan exits. Enter **0** to reset. Updated at the conclusion of scan processing, prior to waiting for the next scan. • Numeric data type
MaxSystemProcTime	Maximum time (µs) required to process the auto (background) calibration, which runs in a hidden slow-sequence type scan. Displays **0** until a background calibration runs. Enter **0** to reset. • Numeric data type • Read only
MeasureOps	Reports the number of task-sequencer opcodes required to do all measurements. Calculated at compile time. Includes operation codes for calibration (compile time), auto (background) calibration (system), and slow sequences. Assumes all measurement instructions run each scan. Updated after compile and before running. • Numeric data type • Read only
MeasureTime	Reports the time (µs) needed to make measurements in the current scan. Calculated at compile time. Includes integration and settling time. In pipeline mode, processing occurs concurrent with this time so the sum of **MeasureTime** and **ProcessTime** is not equal to the required scan time. Assumes all measurement instructions will run each scan. Updated when a main scan begins. • Numeric data type • Read only
MemoryFree	Unallocated final storage memory on the CPU (bytes). All free memory may not be available for data tables. As memory is allocated and freed, holes of unallocated memory, which are unusable for final-storage memory, may be created. Updated after compile completes. • Numeric data type • Read only

Keyword	Information and Location
MemorySize	Total final storage memory size (bytes) in the CR6. Updated at startup. - Numeric data type - Read only
Messages	Contains a string of manually entered messages. - String data type
OSDate	Release date of the operating system in the format mmddyyyy. Updated at startup. - String data type - Read only
OSSignature	Signature of the operating system. - Numeric data type - Read only
OSVersion	Version of the operating system in the CR6. Updated at OS startup. - String data type - Read only
PakBusRoutes	Lists routes or router neighbors known to the datalogger at the time the setting was read. Each route is represented by four components separated by commas and enclosed in parentheses: (port, via neighbor address, pakbus address, response time in ms). Updates when routes are added or deleted. - String data type - Read only
PanelTemp	Current processor board temperature (°C). Updates when viewing the **Status** table or via program code. - Float data type - Read only

Keyword	Information and Location
PortConfig	Provides information on the configuration settings (input, output, SDM, SDI-12, COM port) for **C** or **U** terminals in numeric order of terminals. Default = **Input**. Updates when the port configuration changes. • String data type • Read only
PortStatus	States of **C** terminals configured for control. On/high (**True**) or off/low (**False**). Array elements in numeric order of **C** terminals. Default = **False**. Updates when state changes. • Boolean data type
PowerSource	Primary source of datalogger power. Updates when background calibration executes. • String data type • Read only
ProcessTime	Processing time (µs) of the last scan. Time is measured from the end of the **EndScan** instruction (after the measurement event is set) to the beginning of the **EndScan** (before the wait for the measurement event begins) for the subsequent scan. Calculated on-the-fly. Updated at the conclusion of scan processing, prior to waiting for the next scan. • Numeric data type • Read only
ProgErrors	Number of compile or runtime errors for the running program. Updated after compile. • Numeric data type • Read only
ProgName	Name of current (running) program; updates at startup. • String data type • Read only

Keyword	Information and Location
ProgSignature	Signature of the running CRBasic program including comments. Does not change with operating-system changes. Updates after compiling the program. - Numeric data type - Read only
RecNum	Record number increments only when the record is requested by support software. Range = 0 to 2^{32}. - Long data type - Read only
RevBoard	Electronics board revision in the form **xxx.yyy**, where **xxx** = hardware revision number; **yyy** = clock chip software revision. Stored in flash memory. Updated at startup. - String data type - Read only
RunSignature	Signature of the running binary (compiled) program. Value is independent of comments or non-functional changes. Often changes with operating-system changes. Updates after compiling and before running the program. - Numeric data type - Read only
SerialNumber	CR6 serial number assigned by the factory when the datalogger was calibrated. Stored in flash memory. Updated at startup. - Numeric data type - Read only
SkippedScan	Number of skipped program scans (see Checking station status on page 119 for more information) that have occurred while running the CRBasic program. Does not include scans intentionally skipped as may occur with the use of **ExitScan** and **Do** / **Loop** instructions. Updated when they occur. - Numeric data type

Keyword	Information and Location
SkippedSystemScan	Number of scans skipped in the background calibration. Enter **0** to reset. See LastSystemScan, MaxSystemProcTime, and SystemProcTime. - Numeric data type
StartTime	Time (date and time) the CRBasic program started. Updates at beginning of program compile. - NSEC data type - Read only
StartUpCode	Indicates how the running program was compiled. Updated at startup. 65 = Run on powerup is running and normal powerup occurred. - Numeric data type - Read only
StationName	Station name stored in flash memory. This is not the same name as that is entered into your datalogger support software. This station name can be sampled into a data table, but it is not the name that appears in data file headers. Updated at startup or when the name is changed. This value is read-only if the datalogger is currently running a program with CardOut() instructions. - String data type
SW12Volts	Status of switched, 12 Vdc terminal. **True** = on. Updates when the state changes. - Boolean data type
SystemProcTime	Time (µs) required to process auto (background) calibration. Default is a large number until background calibration runs. - Float data type - Read only
TimeStamp	Scan-time that a record was generated. - NSEC data type - Read only

Keyword	Information and Location
VarOutOfBound	Number of attempts to write to an array outside of the declared size. The write does not occur. Indicates a CRBasic program error. If an array is used in a loop or expression, the pre-compiler and compiler do not check to see if an array is accessed out-of-bounds (i.e., accessing an array with a variable index such as **arr(index) = arr(index–1)**, where **index** is a variable). Updated at run time when the error occurs. Enter **0** to reset. • Numeric data type
WatchdogErrors	Number of watchdog errors that have occurred while running this program. Resets automatically when a new program is compiled. Enter **0** to reset. Updated at startup and at occurrence. • Numeric data type

9.14.2.3 Device Configuration Utility settings

Access settings, using Device Configuration Utility. Clicking on a setting in Device Configuration Utility also provides information about that setting. See also:

- RF407-Series Radio Settings
- RF451 Radio Settings
- Wi-Fi Settings

Keyword	Information and Location
Baudrate	This setting governs the baud rate that the datalogger will use for a given port in order to support serial communications. For some ports (COM), this setting also controls whether the port will be enabled for serial communications. Some ports (RS-232 and CS I/O ME) support auto-baud synchronization while the other ports support only fixed baud. With auto-baud synchronization, the datalogger will attempt to match the baud rate to the rate used by another device based upon the receipt of serial framing errors and invalid packets. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Com Ports Settings | Baud Rate**
Beacon	This setting, in units of seconds, governs the rate at which the datalogger will broadcast PakBus messages on the associated port in order to discover any new PakBus neighboring nodes. If this setting value is set to a value of 0 or 65,535, the datalogger will not broadcast beacon messages on this port. This setting will also govern the default verification interval if the value of the Verify() setting for the associated port is zero. If the value of this setting is non-zero, and the value of the Verify setting is zero, the effective verify interval will be calculated as 2.5 times the value for this setting. If both the value of this setting and the value of the Verify setting is zero, the effective verify interval will be 300 seconds (five minutes). • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Com Ports Settings | Beacon Interval** • **Settings Editor** tab in Device Configuration Utility: **Advanced | Beacon Interval Internet**

Keyword	Information and Location
CentralRouters	This setting specifies a list of PakBus addresses for routers that are able to work as Central Routers. By specifying a non-empty list for this setting, the datalogger will be configured as a Branch Router meaning that it will not be required to keep track of neighbors of any routers except those in its own branch. Configured in this fashion, the datalogger will ignore any neighbor lists received from addresses in the central routers setting and will forward any messages that it receives to the nearest default router if it does not have the destination address for those messages in its routing table. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | Central Routers**
CommsMemAlloc	Replaces **PakBusNodes**. Controls the amount of memory allocated for PakBus routing and communications in general. Increase the value of this setting if you require more memory dedicated to communications. Increase this value if the datalogger will be used for routing a large number of PakBus nodes (>50). Increase this value if your datalogger is dropping connections during short periods of high TCP/IP traffic. This setting will effect the values reported in CommsMemFree. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | Communication Allocation**

Keyword	Information and Location
ConfigCom*x*	Specifies the configuration for a datalogger control port as it relates to serial communications. It is significant only when the associated port baud rate setting is set to something other than **Disabled**. This setting denotes the physical layer properties used for communications. It does not indicate the port's current configuration as it relates to standard or inverted logic. Options include: • **RS-232**: Configures the port as RS-232 with standard voltage levels. • **TTL**: The port is configured to use TTL, 0 to 5V voltage levels. By default, the port will use inverted logic levels. Use the **SerialOpen** CRBasic command to configure this port for standard TTL logic levels. • **LVTTL**: The port is configured to use Low Voltage TTL (LVTTL), 0 to 3.3V voltage levels. By default, the port will use inverted logic levels. Use the SerialOpen() CRBasic command to configure this port for standard TTL logic levels. • **RS-485 Half-Duplex PakBus**: The port is configured as RS-485 half-duplex (two wire) and uses the PakBus/MDROP protocol. This allows reliable PakBus peer-to-peer networking of multiple devices including the MD485 and NL100 using the RS-485 interface. • **RS-485 Half-Duplex Transparent**: The port is configured as RS-485 half-duplex (two wire). This setting is most commonly used when communicating with other non-PakBus RS-485 devices. Use this setting when communicating with devices such as Modbus RTUs or third-party serial sensors with RS-485 interfaces. • **RS-485 Full Duplex Transparent**: The port is configured as RS-485 full-duplex (four wire). In this configuration, four adjacent control ports will be required. Where to find: • **Settings Editor** tab in Device Configuration Utility: **Com Ports Settings | COMx | Configuration**

Keyword	Information and Location
CSIOxnetEnable	Controls whether the CS I/O IP #1 or #2 TCP/IP interface should be enabled. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **CS I/O IP | Interface Enabled #1** or **2**
CSIOInfo	Reports the IP address, network mask, and default gateway for each of the datalogger's active network interfaces. If DHCP is used for the interface, this setting will report the value that was configured by the DHCP server. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **CS I/O | CS I/O Network Addresses**
DisableLithium	Controls whether the datalogger will maintain its real time clock and battery backed memory when it loses power. Setting this value to one will cause the datalogger clock to lose time on power loss. If this value is set to one, the datalogger will not maintain its program or data after it powers down. This value is useful when the datalogger needs to be stored as it will prolong the shelf life of the lithium battery almost indefinitely. If this value is set to one, the datalogger will set it to zero when it powers up. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | Disable Lithium Battery**

Keyword	Information and Location
DeleteCardFiles OnMismatch	Controls the behavior of the datalogger when it restarts with a different program and it detects that data files created by the `CardOut()` are present but do not match the new program. If this value is set to one, the datalogger will delete these files so that new files can be stored. If set to a value of zero, the datalogger will retain the existing files and prevent any data from being appended to these files. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | Delete CardOut Data Files if CardOut Data Table Mismatch**
DNS	This setting specifies the addresses of up to two domain name servers that the datalogger can use to resolve domain names to IP addresses. Note that if DHCP is used to resolve IP information, the addresses obtained via DHCP will be appended to this list. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Ethernet, CS I/O IP, PPP, Wi-Fi | DNS Servers**
EthernetInfo	Reports the IP address, network mask, and default gateway for each of the datalogger's active network interfaces. If DHCP is used for the interface, this setting will report the value that was configured by the DHCP server. • String data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **Ethernet | Network Addresses**

Keyword	Information and Location
EthernetPower	This setting specifies how the datalogger controls power to its Ethernet interface. This setting provides a means of reducing the datalogger power consumption while Ethernet is not connected. Always on, 1 Minute, or Disable. Where to find: • **Settings Editor** tab in Device Configuration Utility: **Ethernet | Ethernet Power**
FilesManager	This setting controls how the datalogger will handle incoming files with specific extensions from various sources. There can be up to four specifications. Each specification has three required fields: **PakBus Address**, **File Name**, and **Count**. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | Files Manager**
FTPEnabled	Set to **1** if to enable FTP service. Default is **0**. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | FTP Enabled**
FTPPassword	Specifies the password that is used to log in to the FTP server. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | FTP Password**

Keyword	Information and Location
FTPPort	Configures the TCP port on which the FTP service is offered. The default value is usually sufficient unless a different value needs to be specified to accommodate port mapping rules in a network address translation firewall. Default = **21**. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | FTP Service Port**
FTPUserName	Specifies the user name that is used to log in to the FTP server. An empty string (the default) inactivates the FTP server. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | FTP User Name**
HTTPEnabled	Specifies additions to the HTTP header in the web service response. It can include multiple lines. Set to **1** to enable HTTP (web server) service or **0** to disable it. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | HTTP Enabled**
HTTPHeader	Specifies additions to the HTTP header in the web service response. It can include multiple lines. Example: `Access-Control-Allow-Origin: *` • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | HTTP Header**

Keyword	Information and Location
HTTPPort	Configures the TCP port on which the HTTP (web server) service is offered. Generally, the default value is sufficient unless a different value needs to be specified to accommodate port-mapping rules in a network-address translation firewall. Default = 80. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | HTTP Service Port**
HTTPSEnabled	Set to 1 to enable the HTTPS (secure web server) service. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | HTTPS Enabled**
HTTPSPort	Configures the TCP port on which the HTTPS (encrypted web server) service is offered. Generally, the default value is sufficient unless a different value needs to be specified to accommodate port mapping rules in a network address translation firewall. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | HTTPS Service Port**

Keyword	Information and Location
IncludeFile	This setting specifies the name of a file to be implicitly included at the end of the current CRBasic program or can be run as the default program. In order to work as an include file, the file referenced by this setting cannot contain a `BeginProg()` statement or define any variable names or tables that are defined in the main program file. This setting must specify both the name of the file to run as well as on the device (CPU:, USR:, or CRD:) on which the file is located. The extension of the file must also be valid for a datalogger program (.dld, .cr6). See also File management via powerup.ini (p. 58). • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | Include File Name**
IPAddressCSIO	Arrays that specify the IP addresses of the internet interfaces that use the CS I/O bridge protocol. If a value is specified as zero (the default), the datalogger will use DHCP to configure the IP address, network mask, and default gateway for that interface. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **CS I/O IP | IP Address**
IPAddressEth	Specifies the IP address used by the Ethernet interface. If this value is specified as "0.0.0.0" (the default), the datalogger will use DHCP to configure the effective value for this setting as well as the `Ethernet Default Gateway` and `Ethernet Subnet Mask` settings. This setting is the equivalent to the `IPAddressEth` status table variable. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Ethernet | IP Address**

Keyword	Information and Location
IPAddressWiFi	Specifies the IP address for the Wi-Fi Interface. If specified as zero, the address, net mask, and gateway will be configured automatically using DHCP. This setting is made available only if the WiFi module is in the CR6. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | IP Address**
IPGateway	Specifies the IP address of the network gateway on the same subnet as the Ethernet interface. If the value of the Ethernet IP Address setting is set to "0.0.0.0" (the default), the datalogger will configure the effective value of this setting using DHCP. This setting is the equivalent to the IPGateway status table variable. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Ethernet | Gateway**
IPGatewayCSIO	These settings specify the IP addresses of the router on the subnet to which the first or second CS I/O bridge internet interface is connected. The datalogger will forward all non-local IP packets to this address when it has no other route. If the CS I/O IP Address setting is set to a value of "0.0.0.0", the datalogger will configure the effective value of this setting using DHCP. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **CS I/O IP | Gateway**

Keyword	Information and Location
IPGatewayWiFi	Specifies the address of the IP router to which the datalogger will forward all non-local IP packets for which it has no route. This setting is made available only if the WiFi module is in the CR6. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | IP Gateway**
IPMaskCSIO	These settings specify the subnet masks for the CS I/O bridge mode internet interface. If the corresponding CS I/O Address setting is set to a value of "0.0.0.0", the datalogger will configure the effective value of this setting using DHCP. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **CS I/O IP | Subnet Mask**
IPMaskEth	Specifies the subnet mask for the Ethernet interface. If the value of the Ethernet IP Address setting is set to "0.0.0.0" (the default), the datalogger will configure the effective value of this setting using DHCP. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Ethernet | Subnet Mask**
IPMaskEthWiFi	Specifies the subnet mask for the WiFi interface. This setting is made available only if the WiFi module is in the CR6. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **WiFi | Subnet Mask**
IPTrace	Discontinued; aliased to IPTraceComport

Keyword	Information and Location
IPTraceCode	Controls what type of information is sent on the port specified by **IPTraceComport** and via Telnet. Each bit in this integer represents a certain aspect of tracing that can be turned on or off. Values for particular bits are described in the Device Configuration Utility. Default = **0**, no messages generated. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | IP Trace Code**
IPTraceComport	Specifies the port (if any) on which TCP/IP trace information is sent. Information type is controlled by **IPTraceCode**. Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | IP Trace COM Port**
IsRouter	This setting controls whether the datalogger is configured as a router or as a leaf node. If the value of this setting is **true**, the datalogger will be configured to act as a PakBus router. That is, it will be able to forward PakBus packets from one port to another. To perform its routing duties, a datalogger configured as a router will maintain its own list of neighbors and send this list to other routers in the PakBus network. It will also obtain and receive neighbor lists from other routers. If the value of this setting is **false**, the datalogger will be configured to act as a leaf node. In this configuration, the datalogger will not be able to forward packets from one port to another and it will not maintain a list of neighbors. Under this configuration, the datalogger can still communicate with other dataloggers and wireless sensors. It cannot, however, be used as a means of reaching those other dataloggers. The default value is false. - Boolean data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | Is Router**

Keyword	Information and Location
MaxPacketSize	Specifies the maximum number of bytes per data collection packet. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | Max Packet Size**
Neighbors	This setting specifies, for a given port, the explicit list of PakBus node addresses that the datalogger will accept as neighbors. If the list is empty (the default value) any node will be accepted as a neighbor. This setting will not affect the acceptance of a neighbor if that neighbor's address is greater than 3999. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Com Ports Settings | Neighbors Allowed**
NTPServer	This setting specifies an NTP Server to be queried (once per day) to adjust the datalogger clock. This setting uses the UTC Offset setting. If UTC Offset setting is not set, it is assumed to be 0. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | NTP Server**

Keyword	Information and Location
PakBusAddress	This setting specifies the PakBus address for this device. Valid values are in the range 1 to 4094. The value for this setting must be chosen such that the address of the device will be unique in the scope of the datalogger network. Duplication of PakBus addresses can lead to failures and unpredictable behavior in the PakBus network. When a device has an allowed neighbor list for a port, any device that has an address greater than or equal to 4000 will be allowed to connect to that device regardless of the allowed neighbor list. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Datalogger | PakBus Address**
PakBusEncryptionKey	This setting specifies text that will be used to generate the key for encrypting PakBus messages sent or received by this datalogger. If this value is specified as an empty string, the datalogger will not use PakBus encryption. If this value is specified as a non-empty string, however, the datalogger will not respond to any PakBus message unless that message has been encrypted. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Datalogger | PakBus Encryption Key**
PakBusNodes	Discontinued; aliased to CommsMemAlloc
PakBusPort	This setting specifies the TCP service port for PakBus communications with the datalogger. Unless firewall issues exist, this setting probably does not need to be changed from its default value. Default 6785. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | PakBus/TCP Service Port**

Keyword	Information and Location
PakBusTCPClients	This setting specifies outgoing PakBus/TCP connections that the datalogger should maintain. Up to four addresses can be specified. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Network Services | PakBus/TCP Client Connections**
PakBusTCPEnabled	By default, PakBus TCP communications are enabled. To disable PakBus TCP communications, set the PakBusPort setting to **65535**. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Network Services | PakBus/TCP Service Port**
PakBusTCPPassword	This setting specifies a password that, if not empty, will make the datalogger authenticate any incoming or outgoing PakBus/TCP connection. This type of authentication is similar to that used by CRAM-MD5. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Datalogger | PakBus/TCP Password**
PingEnabled	Set to one to enable the ICMP ping service. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Network Services | Ping Enabled**

Keyword	Information and Location
PCAP	PCAP is a packet capture (PCAP) file of network packet data (network traffic) that can be opened by Wireshark. This setting specifies the network interface, file name, and maximum size of the PCAP file. For example: • "usr:debug.pcap" saves the file to the USR drive with the file type .pcap. • ".ring." found in name will create new files once the file size has been reached. "crd:debug.ring.pcap" creates crd:debug001.pcap, crd:debug002.pcap... • If a number follows .ring. then only that number of files will be saved, with the oldest deleted. For example: "usr:debug.ring.3.pcap" will save three files. If **All Networks** is selected as the Network Interface and PPP/Cell is active, then seperate files will be opened for the PPP/Cell network with "ppp." prefixed on the file name. Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | PCAP Network Interface**, **PCAP File Name**, **PCAP File Size**

Keyword	Information and Location
pppDial	Specifies the dial string that would follow the ATD command (#777 for the Redwing CDMA). Alternatively, this value can specify a list of AT commands where each command is separated by a semi-colon (;). When specified in this fashion, the datalogger will transmit the string up to the semicolon, transmit a carriage return to the modem, and wait for two seconds before proceeding with the rest of the dial string (or up to the next semicolon). If multiple semicolons are specified in succession, the datalogger will add a delay of one second for each additional semicolon. If a value of PPP is specified for this setting, will configure the datalogger to act as a PPP client without any modem dialing. Finally, an empty string (the default) will configure the datalogger to listen for incoming PPP connections also without any modem dialing. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **PPP | PPP Dial**
pppDialResponse	Specifies the response expected after dialing a modem before a PPP connection can be established. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **PPP | PPP Dial Response**

Keyword	Information and Location
pppInfo	Reports the IP address, network mask, and default gateway for each of the datalogger's active network interfaces. If DHCP is used for the interface, this setting will report the value that was configured by the DHCP server. • String data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **PPP | PPP Network Addresses**
pppInterface	This setting controls which datalogger port PPP service will be configured to use. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **PPP | PPP Interface**
pppIPAddr	Specifies the IP address that will be used for the PPP interface if that interface is active (the PPP Interface setting needs to be set to something other than Inactive). • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **PPP | IP Address**
pppPassword	Specifies the password that will be used for PPP connections when the value of **PPP Interface** is set to something other than **Inactive**. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **PPP | Password**

Keyword	Information and Location
pppUsername	Specifies the user name that is used to log in to the PPP server. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **PPP | User Name**
Radio Settings	For dataloggers with integrated RF modules, see Information tables and settings descriptions (p. 149) and RF451 radio settings (p. 189).
RouteFilters	This setting configures the datalogger to restrict routing or processing of some PakBus message types so that a "state changing" message can only be processed or forwarded by this datalogger if the source address of that message is in one of the source ranges and the destination address of that message is in the corresponding destination range. If no ranges are specified (the default), the datalogger will not apply any routing restrictions. "State changing" message types include set variable, table reset, file control send file, set settings, and revert settings. If a message is encoded using PakBus encryption, the router will forward that message regardless of its content. If, however, the routes filter setting is active in the destination node and the unencrypted message is of a state changing type, the route filter will be applied by that end node. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | Route Filters**
RS232Handshaking	If non-zero, hardware handshaking is active on the RS-232 port. This setting specifies the maximum packet size sent between checking for CTS. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | RS232 Hardware Handshaking Buffer Size**

Keyword	Information and Location
RS232Power	Controls whether the RS-232 port will remain active even when communications are not taking place. Note that if **RS232Handshaking** is enabled (handshaking buffer size is non-zero), that this setting must be set to **Yes**. - Boolean data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | RS232 Always On**
RS232Timeout	RS-232 hardware handshaking timeout. Specifies the time (tens of ms) that the CR6 will wait between packets if CTS is not asserted. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | RS232 Hardware Handshaking Timeout**

Security(1)
Security(2)
Security(3)
	An array of three security codes. A value of zero for a given level will grant access to that level's privileges for any given security code. For more information, see Datalogger security (p. 106).
	- Numeric data type
	Where to find:
	- **Settings Editor** tab in Device Configuration Utility: **Datalogger | Security Level 1**, **2**, or **3**

ServicesEnabled	Discontinued; replaced by/aliased to HTTPEnabled, PingEnabled, TelnetEnabled.
TCPClientConnections	Discontinued; replaced by / aliased to PakBusTCPClients.

Keyword	Information and Location
TCP_MSS	The maximum TCP segment size. This value represents the maximum TCP payload size. It is used to limit TCP packet size. A maximum TCP transmission unit (MTU) can be calculated by adding the IP Header size (20 bytes), the TCP Header size (20 bytes), and the payload size. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | Ping Enabled**
TCPPort	Discontinued; replaced by / aliased to PakBusPort.
TelnetEnabled	Enables (**1**) or disables (**0**) the Telnet service. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Network Services | Telnet Enabled**
TLSConnections	This setting controls the number of concurrent TLS (secure or encrypted) client socket connections that the datalogger will be capable of handling at any given time. This will affect FTPS and HTTPS services. This count will be increased by the number of **DNP()** instructions in the datalogger's program. This setting will control the amount of RAM that the datalogger will use for TLS connections. For every connection, approximately 20KBytes of RAM will be required. This will affect the amount of memory available for program and data storage. Changing this setting will force the datalogger to recompile its program so that it can reallocate memory • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **TLS | Max TLS Server Connections**

Keyword	Information and Location
TLSPassword	This setting specifies the password that will be used to decrypt the TLS Private Key setting. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **TLS | Private Key Password** (setup available over USB or RS-232 only)
TLSStatus	Reports the current status of the datalogger TLS network stack. • String data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **TLS | TLS Status**
UDPBroadcastFilter	Set to one if all broadcast IP packets should be filtered from IP interfaces. Do not set this if you use the IP discovery feature of the device configuration utility or of LoggerLink. If this is set, the datalogger will fail to respond to the broadcast requests. Default = **0**. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | IP Broadcast Filtered**

Keyword	Information and Location
USBEnumerate	Controls the behavior of the datalogger when its USB connector is plugged into the computer. If set to a value of one, the datalogger will use its own serial number for identification in the USB enumeration. If set to a value of zero (the default), the datalogger will use a fixed serial number in the USB enumeration. This behavior controls whether the computer will allocate a new virtual serial port for the datalogger USB connection or will use a previously allocated (but not currently used) virtual serial port. Default = **0**. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | USB Enumerate**
USRDriveFree	Provides information on the available bytes for the USR drive. Where to find: - Device Configuration Utility **File Control** tab
USRDriveSize	Specifies the size in bytes allocated for the USR: ram disk drive. This memory is allocated from the memory that the datalogger would normally use to store its compiled program or RAM based data tables. If this setting is too large, some programs may not be able to compile on the datalogger. Setting the USR: Drive Size setting will force the datalogger to recompile its program and may result in the loss of data. This setting controls the amount of memory set aside for the USR: size and is only indirectly related to the amount of storage within that file system. The amount of space available for storing files is always going to be less than this value because of the overhead of file system structures. - Numeric data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Advanced | USR: Drive Size**

Keyword	Information and Location
UTCOffset	Specifies the offset, in seconds, of the datalogger's clock from Coordinated Universal Time (UTC, or GMT). For example, if the clock is set to Mountain Standard Time in the U.S. (-7 Hours offset from UTC) then this setting should be -25200 (-7*3600). This setting is used by the **NTP Server** setting as well as **EmailSend()** and **HTTP()**, which require Universal Time in their headers. This setting will also be adjusted by the Daylight Savings functions if they adjust the clock. If a value of **-1** is supplied for this setting, no UTC offset will be applied. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Advanced | UTC Offset**
Verify	This setting specifies the interval, in units of seconds, that will be reported as the link verification interval in the PakBus hello transaction messages. It will indirectly govern the rate at which the datalogger will attempt to start a hello transaction with a neighbor if no other communications have taken place within the interval. • Numeric data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **ComPorts Settings | Verify Interval** • **Settings Editor** tab in Device Configuration Utility: **Advanced | Verify Interval Internet**
Wi-Fi Settings	For dataloggers with integrated WIFI modules, see Wi-Fi settings (p. 208).

Keyword	Information and location
RadioChanMask	The channel mask allows channels to be selectively enabled or disabled. This allows you to avoid using frequencies that experience unacceptable levels of RF interference. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | Radio Channel Mask**
RadioEnable	Global control for the internal radio module. Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | Radio Enable**
RadioHopSeq	Specifies the radio channel hop sequence. This setting must match in all radios in the same RF network. This setting can also be used to prevent radios in one RF network from listening to transmissions of another. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | RF Hop Sequence**
RadioMAC	Radio serial number. • String data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | Radio MAC Address**
RadioModel	Reports the model of the internal radio module. • String data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | Radio Model**

Keyword	Information and location
RadioModuleVer	Radio hardware version. - Long data type - Read only Where to find: - **Settings Editor** tab in Device Configuration Utility: **Radio | Radio Hardware Version**
RadioNetID	The RadioNetID specifies the identifier for the RF network. The radio will ignore any packets received that do not use this network identifier - therefore, all radios in the network must use the same value. Valid entries are between **0** and **32767**. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Radio | Network ID**
RadioProtocol	Specifies the protocol mode that will be used by the radio. **Transparent**: Provides a transparent link with no interpretation of the data packet. This mode is most commonly used with array based dataloggers, and it must be used when communicating with other transparent devices such as the RF407/412/422, CR300-RF407/CR300-RF412/CR300-RF422, and CR6-RF407. This mode is also used for non PakBus protocols like Modbus. When used this way, **Retry Level** must be set to **None**. **PakBus Aware**: This is the most commonly used protocol setting for PakBus networks. The radio will automatically inherit an RF identifier equal to the PakBus address of the device to which it is serially attached. In this mode, the radio will be capable of performing RF level retries and acknowledgments and provide a more reliable link than **Transparent** mode used for broadcast messaging. You do not need to manually set a unique RF Radio Address or a unique PakBus Address. This device will not appear in PakBus Graph. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Radio | Protocol**

Keyword	Information and location
RadioPwrMode	This setting governs the duty cycle that the radio will use for powering its receiver circuit. As such, it governs the amortized current drain for the radio. This setting should be set the same for all radios in the same network. Power Modes include: **Always On**: The radio is always on and does not transmit a wakeup header. **.5 Second**: The radio wakes every 0.5 seconds for a 100 msec interval to listen for RF activity. It will transmit a 700 msec wakeup header with the first transmission following a period of RF inactivity. **1 Second**: The radio wakes every 1 second for a 100 msec interval to listen for RF activity. It will transmit a 1200 msec wakeup header with the first transmission following a period of RF inactivity. **4 Second**: The radio wakes every 4 seconds for a 100 msec interval to listen for RF activity. It will transmit a 4200 msec wakeup header with the first transmission following a period of RF inactivity. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | Power Mode**

Keyword	Information and location
RadioRetries	Specifies the level to which the radio should retry to deliver an unacknowledged RF packet transmission. When an RF packet fails to be acknowledged by the destination, the radio will resend the packet again. A receiving radio responds to the sending radio with an ACK packet for every radio packet that it receives that is addressed to it and has a valid CRC. Retry levels and counts: • **None** - 0 • **Low** - 2 • **Medium** - 4 • **High** - 6 Set **Retry Level** to **None** when the **Protocol** setting is set to **Transparent** for the purpose of communicating with other RF407 series radios. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | Retry Level**
RadioRSSI	Indicates the signal strength of the last packet received by this radio. The units of the RSSI are dBm; -40 is a stronger signal than -70. Because the received signal strength can vary due to multipath, interference, or other environmental effects; this setting may not give a true indication of communication performance or range. However, received signal strength can be useful for activities such as: • Determining the optimal direction to aim a Yagi antenna. • Determining the effects of antenna height and location. • Trying alternate (reflective) paths. • Seeing the effect of vegetation and weather over time. • Long data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | RFSignalLevel** (first number)

Keyword	Information and location
RadioRSSIAddr	Indicates the PakBus address of the **RadioRSSI** signal radio. - Long data type - Read only Where to find: - **Settings Editor** tab in Device Configuration Utility: **Radio | RFSignalLevel** (second number)
RadioStats	Provides the performance statistics for the integrated radio for the datalogger. The datalogger will maintain a radio routing table for each PakBus neighbor accessed using the integrated radio and this setting is generated from that table. The fields reported for this setting are as follows: - **PakBus Address**: Specifies the PakBus address of the neighbor reached through an integrated radio link. - **Sent Packets**: Reports the number of radio packets that have been transmitted to the PakBus neighbor using the integrated radio link. - **Received Packets**: Reports the number of radio packets that have been received from the PakBus neighbor using the integrated radio link. - **Packet Retries**: Reports the number of radio packet transmissions to the PakBus neighbor using the integrated radio link that had to be retransmitted by the radio module. - **Packet Failures**: Reports the number of radio packet transmissions to the PakBus neighbor that were never acknowledged. - String data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Radio | Radio Performance Statistics**

Keyword	Information and location
RadioTxPwr	Specifies the power level at which the RF module transmits. Levels are approximate. It is very important that the TX power level selected and the gain of the attached antenna do not exceed the maximum allowed ERP permitted by local laws. These rules vary from region to region. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Radio | Radio TX Power Level**

RF451 radio settings

Access RF451 radio settings, using Device Configuration Utility. Clicking on a setting in Device Configuration Utility also provides information about that setting. These settings are available for RF451 dataloggers.

Keyword	Information and Location
RadioCarrier	Indicates whether or not a carrier is detected. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Carrier Detect**
RadioDataRate	FreeWave transceivers have two settings for the RF Data Rate: - **High RF Data Rate**: Should be used when the transceivers are close together and data throughput needs to be optimized. Setting 2 must also be used when the full throughput of 115.2KBaud is necessary. - **Normal RF Data Rate**: Should be used when the transceivers are farther away and a solid data link is preferred over data throughput. Note: In MultiPoint networks, the RF Data Rate must be set identically in all transceivers. Any transceiver with an RF Data Rate different from the Master will not establish a link. In Point to Point networks the Masters settings take precedence over the Slave. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | RF Data Rate**

Keyword	Information and Location
RadioDiag	Provides diagnostics data to be viewed at the Master in parallel with application data. The diagnostic program MUST be run from the Master transceiver. Diagnostics require the following: 1. **RadioRepeaters** set to **1** and **RadioDiag** set to (**1** to **128**) in the Master. 2. A second computer or serial connection to run the diagnostics software. 3. A diagnostics cable. 4. Diagnostics software. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Diagnostics**
RadioEnable	Global control for the internal radio module. Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | RF451 Radio Enable**
RadioFirmwareVer	Radio firmware version. - Long data type - Read only Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Radio Firmware Version**

Keyword	Information and Location
RadioFreqKey	Determines the frequency hopping sequence of the transceiver. There are fifteen choices available (0-14) which represent fifteen unique pseudo-random hop patterns. This setting allows you to minimize RF interference with other FreeWave transceivers operating in the same RF area. The Frequency Key setting should be the same for all radios in the entire network. The exception to this is if the **RadioFreqRepeat** setting is used. If this is used, the Repeater Frequency Key would be different from the Master radio, and downstream radios intended to connect to the Repeater would have the same Frequency Key setting as the Repeater. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Frequency Key**
RadioFreqRepeat	This must be set when you want a Repeater to use a Frequency Key other than that of the Master. This is a setting that is only used by Repeaters. The conditions where this is useful is when there are parallel Repeaters in a network, and you want to force communications through a particular Repeater. When this setting is used, the Repeater will receive on the Frequency Key of the upstream Master (or Repeater), and transmit on its Frequency Key setting (which typically is set to a different value than the Masters). The default setting of Use Master Frequency Key (box unused) causes the Repeater to transmit on the Masters Frequency Key. When this setting is not used the Frequency Key setting should match that of the Master or of the Repeater acting as the Master for that transceiver. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Repeater Frequency**

Keyword	Information and Location
RadioFreqZone	Divides the available band into smaller bands of 7 or 8 channels. A value of 1 enables the band. - Frequency Zone 1: 902.2464 to 903.8592 MHz - Frequency Zone 2: 904.0896 to 905.4720 MHz - Frequency Zone 3: 905.7024 to 907.0848 MHz - Frequency Zone 4: 907.3152 to 908.6976 MHz - Frequency Zone 5: 908.9280 to 910.3104 MHz - Frequency Zone 6: 910.5408 to 911.9232 MHz - Frequency Zone 7: 912.1536 to 913.5360 MHz - Frequency Zone 8: 913.7664 to 915.1488 MHz - Frequency Zone 9: 915.3792 to 916.7616 MHz - Frequency Zone 10: 916.9920 to 918.6048 MHz - Frequency Zone 11: 918.8352 to 920.2176 MHz - Frequency Zone 12: 920.4480 to 921.8304 MHz - Frequency Zone 13: 922.0608 to 923.4432 MHz - Frequency Zone 14: 923.6736 to 925.0560 MHz - Frequency Zone 15: 925.2864 to 926.6688 MHz - Frequency Zone 16: 926.8992 to 927.8208 MHz - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Frequency Zone (1-16)**
RadioHopSize	Defines how many separate channels will be used by the network. Range is **50** to **112**. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Hop Table Size**

Keyword	**Information and Location**
RadioHopVersion	This setting allows the user to choose the portion of the band in which the transceiver will operate. • **0** = Standard, Full 902-928 MHz • **1** = Australia, 915-928 MHz • **2** = Table 2, 903.744-926.3232 MHz • **3** = Table 3, 916-920 MHz • **4** = New Zealand, 921-928 MHz • **5** = Notch, Uses 902-928 MHz with center frequencies of 911-919 MHz notched out • **6 = Brazil, 902-915 MHz** Note: Per radio manufacturer instructions, do not use Frequency Key **14** (E) when using setting **1** (Australia), **3**, or **4** (New Zealand). Refer to the FreeWave MM2-LV-T manual for more information. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Hop Table Version**

Keyword	Information and Location
RadioLowPwr	This setting allows a MultiPoint Slave to consume less power. When set to **2** through **31**, the transceiver will sleep between slots. For example, at a setting of **2** the transceiver sleeps 1 out of 2 slots; at a setting of **3** the transceiver sleeps 2 out of 3 slots... The actual current draw depends on many factors. A low number reduces latency and a high number reduces current consumption. An optimum setting (balancing latency and power savings) is **2** or **3**. The most significant benefit comes when changing from **0** to **2**. To ensure rigorous communications, make sure the **RadioPacketRepeat** setting is set on the Master Radio (of the network) to a value at least as large as this value. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Low Power Mode**
RadioMaxPacket	Determines the maximum number of bytes in the packets. Throughput can be enhanced when packet sizes are optimized. In Point-to-Point mode, the Max and Min Packet Settings will not have material impact on throughput unless 115.2 KBaud is desired. However, this may have an impact on latency. For example, if small amounts of data is sent and large packet sizes are selected, there would be a certain amount of time "wasted" between each packet. The default settings for Max packet size, Min packet size and RF Data Rate are **8**, **9**, and **Normal RF Data Rate**, respectively. In MultiPoint networks, the Max Packet Size and Min Packet Size must be set identically in all transceivers. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Max Packet Size**

Keyword	Information and Location
RadioMinPacket	Determines the minimum number of bytes in the packets. Throughput can be enhanced when packet sizes are optimized. In Point-to-Point mode, the Max and Min Packet Settings will not have material impact on throughput unless 115.2 KBaud is desired. However, this may have an impact on latency. For example, if small amounts of data is sent and large packet sizes are selected, there would be a certain amount of time "wasted" between each packet. The default settings for Max packet size, Min packet size and RF Data Rate are **8**, **9**, and **Normal RF Data Rate**, respectively. In MultiPoint networks, the Max Packet Size and Min Packet Size must be set identically in all transceivers. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Min Packet Size**
RadioMMSync	This setting is reserved for applications, in both Point-to-Point and MultiPoint modes, with concentrations of Master units where it is necessary to reduce interference between the Masters. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | MultiMaster Sync**
RadioModOS	Radio firmware version. This setting is available in dataloggers with integrated RF modules. • String data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | RF451 iBus Module OS**

Keyword	Information and Location
RadioModuleVer	Radio hardware version. - Long data type - Read only Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Freewave Radio Module Firmware Version**
RadioNetID	The RadioNetID allows MultiPoint networks to be established without using the Call Book. To enable the Network ID the value must be set between **0** and **4095** (excluding 255). The setting of **255** enables the Call Book (not used for MultiPoint Operation Modes). All radios in the MultiPoint Network need to have the same Network ID. Since MultiPoint networks do not use serial numbers, MultiPoint Masters and Repeaters may be replaced without reprogramming all of the Slaves in the network. Slaves will link with the first Master or Repeater that it hears that has a matching NetWork ID. The Network ID function should be used in conjunction with the SubNet ID feature (If necessary). - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Network ID**

Keyword	Information and Location
RadioOpMode	Designates the method FreeWave transceivers use to communicate with each other. FreeWave transceivers operate in a Master to Slave configuration. Before the transceivers can operate together, they must be set up to properly communicate. In a Point-to-Point configuration, Master or Slave Mode may be used on either end of the communications link without performance degradation. When setting up the transceiver, remember that a number of parameters are controlled by the settings in the Master. Also, radio network diagnostics can only be accessed at the Master radio. Therefore, we suggest you deploy the Master on the communications end where it will be easier to access. For a datalogger PakBus Network, the MultiPoint Radio modes should be used. **Point-to-MultiPoint Master**: Designates the transceiver as a Master in MultiPoint mode. This mode allows one Master transceiver to simultaneously be in communication with numerous Slaves and Repeaters. A Point-to-MultiPoint Master communicates only with other transceivers designated as Point-to-MultiPoint Slaves or Point-to-MultiPoint Repeaters. **Point-to-MultiPoint Slave**: Designates the transceiver as a Slave in MultiPoint mode. This mode allows the Slave to communicate with a MultiPoint Master. The Slave may communicate with its Master through one or more Repeaters. **Point-to-MultiPoint Repeater**: Allows the transceiver to operate as a Repeater in a MultiPoint network. **Point-to-MultiPoint Slave/Repeater**: Allows the transceiver to operate as a Repeater and a Slave in a MultiPoint network. The radio will repeat packets sent across the network as well as use the Active Interface. Choosing this setting effectively sets the operation mode to MultiPoint Repeater and sets the Slave/Repeater mode. - Long data type Where to find:

Keyword	Information and Location
	- **Settings Editor** tab in Device Configuration Utility: **RF451 | Radio Operation Mode**

Keyword	Information and Location
RadioPacketRepeat	In a Point-to-MultiPoint network, Slaves do not acknowledge transmissions from the Master. If Slaves in a large network did acknowledge all data transmissions, the Master would soon become overwhelmed with acknowledgments from the Slaves. Without acknowledgements, there is not 100% confidence every Slave has received every packet. To address this issue, the user may modify the **RadioPacketRepeat** setting, assigning a value between **0** (the packet is transmitted once) to **9** (the packet is transmitted 10 times). For networks with solid RF links, this setting should be set to a low value such as **1** or **2**. If a network has some weak or marginal links it should be set with higher values. If a Slave receives a good packet from a Master more than once, it will discard the repeated packets. Similarly, once a MultiPoint Repeater receives a good packet from the Master, it will discard any further repeated packets. In turn, the Repeater will send the packet out to the next Repeater or Slaves the number of times corresponding to its own **RadioPacketRepeat** setting. Increasing the **RadioPacketRepeat** setting increases the probability of a packet getting through, but it also increases latency in the network because each packet from the Master or Repeater is being sent multiple times. It is important to find the optimal mix between network robustness, throughput, and latency. In general, a setting of **2** to **3** will work well for most well designed networks. MASTER PACKET REPEAT IN MULTIPOINT NETWORKS WITH REPEATERS: The **RadioPacketRepeat** setting must also be set in MultiPoint Repeaters since a Repeater will appear as a Master to a Slave. Therefore, the Repeater will send the packet out the number of times corresponding to its own **RadioPacketRepeat** setting. If this setting is set improperly the reliability of the overall network may be diminished. For example, if a Master's **RadioPacketRepeat** is set to **3**, the link between the Master and Repeater should be robust. If the Repeater's **RadioPacketRepeat** is set to **0**, this could cause marginal communications between the Repeater and the Slaves. The Slaves communicating through

Keyword	Information and Location
	this Repeater will only receive the initial packet from the Master with no repeats. Therefore, if the packet is not received on the first try, the Slave will not respond as expected. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Master Packet Repeat**
RadioRepeaters	In a MultiPoint network it is critical to transmission timing to configure this setting correctly. Check this box if there are any Repeaters in the network. This setting should be set to the same value in all transceivers in a MultiPoint network. Note, this option should be checked when running Diagnostics from the Master. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Repeaters Used**

Keyword	Information and Location
RadioRetryOdds	While packets transmitted from the Master to the Slaves in a MultiPoint network are not acknowledged, packets transmitted from Slaves to the Master are acknowledged. It is possible that more than one Slave will attempt to transmit to the Master at the same time. Therefore, it is important that a protocol exists to resolve contention for the Master between Slaves. This is addressed through **RadioSlaveRetry** and **RadioRetryOdds**. Once the Slave has unsuccessfully attempted to transmit the packet the number of times specified in **RadioSlaveRetry**, it will attempt to transmit to the Master on a random basis. **RadioRetryOdds** determines the probability that the Slave will attempt to retransmit the packet to the Master; a low setting will assign low odds to the Slave attempting to transmit. Conversely, a high setting will assign higher odds. For example, this setting might be used when considering two different Slaves in a MultiPoint network, one with a strong RF link, and the other with a weak RF link to the Master. Assigning a higher **RadioRetryOdds** to the Slave with the weaker link gives it a better chance of competing with the closer Slave(s) for the Master's attention. When **RadioRetryOdds** = **0**, after the Slave has exhausted the number of retries set in **RadioSlaveRetry** and still not gained the Master’s attention, the Slave’s data buffer will be purged, and no further attempts will be made. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Retry Odds**

Keyword	Information and Location
RadioRetryTimeout	Setting **RadioRetryTimeout** in a Slave or Repeater sets the delay the unit will wait before dropping the connection to a Master or Repeater in MultiPoint mode. This setting is useful when a MultiPoint network has a roving Master or Slave(s). As the link gets weaker, a lower setting will allow a poor link to break in search of a stronger one. The factory default is set at the maximum of **255**. With a setting of **255**, the Master will allow a Slave or Repeater to stay connected as long as 1 packet in 255 is successfully received at the Master. The minimum setting is **8**. This allows a Slave or Repeater to drop a connection if less than 1 in 8 consecutive packets is successfully received from the Master. Setting **RadioRetryTimeout** to **20** is recommended in areas where several FreeWave networks exist. This setting will allow Slaves and Repeaters to drop the connection if the link becomes too weak, while at the same time preventing errant disconnects due to interference from neighboring networks. While intended primarily for MultiPoint networks, the **RadioRetryTimeout** setting may also be modified in Point-to-Point networks. However, the value in Point-to-Point mode should not be set to less than **151**. • Long data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Retry Timeout**

Keyword	Information and Location
RadioRxSubID	In a MultiPoint Network (with Subnet IDs disabled), a Slave or Repeater will connect with the first Repeater or Master that it hears with the same Network ID. There are scenarios, however, where communications need to be forced to follow a specific path. The Subnet ID is particularly helpful to force two Repeaters in the same network to operate in series rather than in parallel; or, if desired, to force Slaves to communicate to a specific Repeater for load balancing purposes. There are two components to the Subnet ID: 1. **Receive Subnet ID (RX ID)**: This setting identifies which transceiver a Repeater or Slave will listen to. The **RX ID** has no effect on a radio configured as Master and should be left at the default value of 15. 2. **Transmit Subnet ID (TX ID)**: This setting identifies the ID on which this device transmits, and in turn which devices will listen to it. The TX ID Subnet ID setting is only relevant for MultiPoint Masters or Repeaters. **Default Setting**:The default (disable) setting for both RX ID and TX ID is 15 (0xF), which is a visual way to indicate that the device is the final in the line of communication and does not use a subnet ID. A Multipoint Slave with a Subnet ID of 15,15 (0xF,0xF) does not roam from one Repeater or network to the next, it only links to a Master or Repeater that has either a TX ID setting of 0 or 15. • For the Master, the default setting causes the Master to actually transmit on 0, and it will receive any Subnet ID. • For Slaves, the default setting (15,15) disables the use of Subnet IDs; they will connect to the first Master or Repeater that has the same Network ID as the Slave. • If Subnet IDs are to be used, the downstream radios (slave or repeater) that need to connect directly to the Master will need their RX ID set to same value as the Master radio TX ID. If the Master radio is set to its default value of 15 the slave or repeaters that need to connect directly to the

Keyword	Information and Location
	Master must have a RX ID setting of 0. - Long data type - Read only Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Receive SubNet ID**
RadioSlaveRepeat	The Slave/Repeater mode allows a transceiver in a MultiPoint network to switch between Slave and Repeater functions. When in this mode, a transceiver will repeat any packets sent across the network as well as utilize the wired port. To operate a transceiver as a MultiPoint Slave/Repeater, the operation mode must be set to **MultiPoint Repeater**and Slave/Repeater set to **Enable Slave/Repeater**. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Slave/Repeater**
RadioSlaveRetry	This setting defines how many times (**0** to **9**) the Slave will attempt to retransmit a packet to the Master before beginning to use a back-off algorithm (defined by the **RadioRetryOdds** setting). Slave retries will stop when an acknowledgement is received from the Master. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **RF451 | Max Slave Retry**

Keyword	Information and Location
RadioTxPwr	Specifies the power level at which the RF module transmits. It is important to note that the FCC specifies a maximum EIRP (Effective Isotropic Radiated Power) of 36 dBm. `EIRP = (Transmitter Power) + (Antenna Gain) - (Cable Losses) [all in dB or dBm]` Since the RF451 maximum power is 30 dBm, a 6 dB (or lower) gain antenna can be used with any Transmit Power setting. If higher gain antennas are used, the cable loss will need to be determined, and the Transmit Power adjusted so as not to exceed the FCC limit (36 dBm). Note that lower transmit power can be used (to conserve battery power) if the required range allows it. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | RF Transmit Power**
RadioTxRate	There are two options for the **RadioRxRate**, **0** and **1**. For normal operation of the transceiver, **RadioRxRate** = **1**. **RadioRxRate** = **0** is useful to qualitatively gauge signal strength in Point to Point mode. When **RadioRxRate** = **0**, the transceivers will transmit back and forth continuously regardless if they have any actual data. In Point-to-Point operation, **RadioRxRate** = **0** should be used only as a diagnostic tool and not for normal operation. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **RF451 | Xmit Rate**

Keyword	Information and Location
RadioTxSubID	In a MultiPoint Network (with Subnet IDs disabled), a Slave or Repeater will connect with the first Repeater or Master that it hears with the same Network ID. There are scenarios, however, where communications need to be forced to follow a specific path. The Subnet ID is particularly helpful to force two Repeaters in the same network to operate in series rather than in parallel; or, if desired, to force Slaves to communicate to a specific Repeater for load balancing purposes. There are two components to the Subnet ID: 1. **Receive Subnet ID (RX ID)**: This setting identifies which transceiver a Repeater or Slave will listen to. The **RX ID** has no effect on a radio configured as Master and should be left at the default value of 15. 2. **Transmit Subnet ID (TX ID)**: This setting identifies the ID on which this device transmits, and in turn which devices will listen to it. The TX ID Subnet ID setting is only relevant for MultiPoint Masters or Repeaters. **Default Setting**:The default (disable) setting for both RX ID and TX ID is 15 (0xF), which is a visual way to indicate that the device is the final in the line of communication and does not use a subnet ID. A Multipoint Slave with a Subnet ID of 15,15 (0xF,0xF) does not roam from one Repeater or network to the next, it only links to a Master or Repeater that has either a TX ID setting of 0 or 15. • For the Master, the default setting causes the Master to actually transmit on 0, and it will receive any Subnet ID. • For Slaves, the default setting (15,15) disables the use of Subnet IDs; they will connect to the first Master or Repeater that has the same Network ID as the Slave. • If Subnet IDs are to be used, the downstream radios (slave or repeater) that need to connect directly to the Master will need their RX ID set to same value as the Master radio TX ID. If the Master radio is set to its default value of 15 the slave or repeaters that need to connect directly to the

Keyword	Information and Location
	Master must have a RX ID setting of 0. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Radio | Transmit SubNet ID**

Wi-Fi settings

Access Wi-Fi settings, using Device Configuration Utility. Clicking on a setting in Device Configuration Utility also provides information about that setting. These settings are available for dataloggers with integrated WIFI modules.

Keyword	Information and location
WiFiChannel	This setting is only applicable when the device is configured to create a network (**WiFiConfig**). It then specifies in which channel the network should be created. If **Auto** is selected, the device will select to operate on a channel that has minimal interference from other networks detected in the area. When manually selecting a channel, it should be noted that two Wi-Fi networks operating on the same channel will interfere with each other and will have to compete for bandwidth. The center frequencies of adjacent channels are 5 MHz apart and the bandwidth of each channel is 20 MHz which means that adjacent channels overlap. To completely avoid interference there must be a spacing of at least 5 channels between each Wi-Fi network. It is therefore recommended to use channels 1, 6, and 11. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Channel**
WiFiConfig	Configure the WiFi network. Disable, join a network, or create a network. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Configuration**

Keyword	Information and location
WiFiEapMethod	The EAP Method must be chosen to match the EAP method being used by the Enterprise Security network. The inner EAP Methods supported are MSCHAPv2, MSCHAP, CHAP, and PAP. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | EAP Method**
WiFiEapPassword	If joining an Enterprise Security-enabled network then this is where the password is entered. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | EAP Password**
WiFiEapUser	If joining an Enterprise Security-enabled network then this is where the user name is entered. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | EAP User**
WiFiEnable	Set to enable or disable the WiFi service. By default, WiFi is enabled. To disable, set the **Configuration** option to **Disable**. **0** Disabled, **1** or <>**0** Enabled. • Boolean data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Configuration**

Keyword	Information and location
WiFiPassword	If joining a WPA or WPA2 security enabled network then this is where the passphrase is entered. If joining a WEP security enabled network then this is where the WEP key is entered. If creating a network and a password is supplied, the network will be created using WPA2 encryption. The password must be at least 8 characters. If a password is not supplied, an open (unencrypted) network will be created. When joining a network the device supports 64-bit WEP and 128-bit WEP. For 64-bit WEP enter a 40 bit key in the form of 5 ASCII characters or 10 hexadecimal digits (0-9, A-F). For 128-bit WEP enter a 104 bit key in the form of 13 ASCII characters or 26 hexadecimal digits (0-9, A-F). • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Password**
WiFiPowerMode	This setting controls the power saving mode of the device. Regardless of the **Power Mode** setting, the device enables power-save mode when communications are not active. **Power Mode** determines how the device acts when communications are ongoing. This setting only applies when the device is configured to **Join a Network** using the **WiFiConfig** option. • Long data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Power Mode**

Keyword	Information and location
WiFiSSID	The Network Name (SSID) is the name that identifies a wireless network (31 character maximum). The SSID differentiates one wireless network from another, so all devices attempting to connect to the same network must use the same SSID. If the device is configured to 'Join a Network', then enter the SSID of the network to join here. If no SSID is specified, the device will join the first open network it finds. If the device is configured to **Create a Network** using the **WiFiConfig** option, then the SSID entered here will be the SSID of the network created. • String data type Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Network Name (SSID)**
WiFiStatus	Specifies the current status of the Wi-fi module. • String data type • Read only Where to find: • **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Wi-Fi Status**

Keyword	Information and location
WiFiTxPowerLevel	This fixes the transmit power level of the Wi-fi module. This value can be set as follows: Low (7 +/- 1 dBm), Medium (10 +/- 1 dBm), High (15 +/- 2 dBm). The value of this setting does not affect power consumption. - Long data type Where to find: - **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Tx Power Level**
WiFiNetworks	Lists the networks available in the area. Information listed for each network is shown as {SSID, RSSI / Signal Strength, Channel, Security}. Sometimes areas are covered by multiple access points configured with the same network name (SSID). In that case multiple unique access points possessing the same network name (SSID) may be listed here. - String data type - Read only Where to find: - **Settings Editor** tab in Device Configuration Utility: **Wi-Fi | Wireless Networks in Area**

10. 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 is recommended every three years. Critical specifications and system configuration should be confirmed with Campbell Scientific before purchase.

Specifications

10.1 System specifications

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

Memory (see Data memory on page 53 for more information):

- 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

NOTE:
CR6 dataloggers with serial numbers 7502 and greater have the following memory specifications:

CPU Drive / Programs: 1 MB flash
USR Drive / Data: 4 MB SRAM (battery backed)

Program Execution: 1 ms to 1 day

Real-Time Clock:

- Battery backed while external power is disconnected
- **Resolution**: 1 ms
- **Accuracy**: ±3 min. per year, optional correction (±10 µs) using GPS, PakBus, or NTP

Wiring Panel Temperature: A thermistor, located on the processing board, is measured when reporting wiring panel temperature.

10.2 Physical specifications

Dimensions: 21.0 x 10.2 x 5.7 cm (8.3 x 4.0 x 2.2 in); additional clearance required for cables and wires. For CAD files, see CR6 Images and CAD 2D Drawings.

Weight/Mass:

CR6: 0.42 kg (0.92 lb)

CR6-WIFI: 0.50 kg (1.10 lb)

CR6-RF451: 0.52 kg (1.15 lb)

CR6-RF407/412/422/427: 0.51 kg (1.13 lb)

10.3 Power requirements

Protection: Power inputs are protected against surge, over-voltage, over-current, and reverse power.

CHG **Terminals**:

- **Voltage Input**: 16 to 32 Vdc
- **Input Current Limit**: 1.2 A @ 20 °C maximum
- Input voltage must be at least 0.3 V higher than the voltage required to charge the battery; 40 Vdc sustained voltage limit without damage. Transient voltage suppressor (TVS) diodes at the **BAT** and **CHG** terminals clamp transients to 19 to 21 V and 19 to 40 V respectively. Sustained input voltages in excess of 19 V or 40 V respectively can damage the TVS diodes.

Battery Input:

- **BAT** terminal
- 10 – 18 Vdc

- 19 Vdc sustained voltage limit without damage; transient voltage protected
- 2.5 A max current at 12 Vdc at 20° C; trickle charges the battery

External Batteries:

- Float charge on **BAT** terminal
- 12 Vdc
- Valve-regulated, lead-acid (VRLA)
- 2 to 24 Ah battery typical

USB Power: Functions that will be active with USB 5 Vdc include sending programs, adjusting datalogger settings, and making some measurements. If USB is the only power source, then the **CS I/O** port and the **12V** and **SW12** terminals will not be operational. 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. See also Internal battery (p. 111).

Average Current Drain (assumes 12 Vdc on **BAT** terminals — add 2 mA if using **CHG** terminals):

- Idle: <1 mA
- Active 1 Hz Scan: 3 mA
- Active 20 Hz Scan: 67 mA
- Serial (RS-232/RS-485): Active + 25 mA
- Ethernet Power Requirements:
  - Ethernet 1 Minute: Active + 1 mA
  - Ethernet Idle: Active + 4 mA
  - 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.

Wi-Fi Additional Current Contribution at 12 Vdc:

- Client Mode: 7 mA idle, 70 mA communicating
- Access Point Mode: 62 mA idle, 70 mA communicating
- Sleep (using IPNetPower or DevConfig setting to disable): <1 mA

RF Average Additional Current Contribution at 12 Vdc:

- Transmit
    - **RF407**, **RF412**, and **RF427**: < 80 mA
    - **RF422**: 20 mA
    - **RF451**: 650 mA, maximum
- Idle On
    - **RF407**, **RF412**, and **RF427**:12 mA
    - **RF422**: 9.5 mA
    - **RF451**: 15 mA, maximum
- Idle 0.5 sec Power Mode
    - **RF407**, **RF412**, and **RF427**:4 mA
    - **RF422**: 3.5 mA
    - **RF451**: NA
- Idle 1 sec Power Mode
    - **RF407**, **RF412**, and **RF427**:3 mA
    - **RF422**: 2.5 mA
    - **RF451**: NA
- Idle 4 sec Power Mode
    - **RF407**, **RF412**, and **RF427**:1.5 mA
    - **RF422**: 1.5 mA
    - **RF451**: NA

10.4 Ground specifications

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

- 6 common terminals

Power Ground (G) - return for **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) - one resistance-to-ground input 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 terminal is ±16 V. See also Current-loop measurement specifications (p. 226).

> **NOTE:**
> Resistance to ground input for non-isolated 0-20 mA and 4-20 mA current loop measurements is available in CR6 dataloggers with serial numbers 7502 and newer.

10.5 Power output specifications

System Power Out Limits (when powered with 12 Vdc):

Temperature (°C)	Current Limit[1] (A)
–40°	3.88
0°	2.98
20°	2.50
50°	1.80
70°	1.35
85°	1.00

[1] Limited by self-resetting thermal fuse

12V and SW12V 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. Voltage output is 0.3 Vdc less than voltage at **BAT** terminals. The **12V** terminal is limited to the current shown in the previous table.

  SW12-1 and **SW12-2** can be independently set under program control. Each SW12 terminal has a hold current limited to the values in the following table.

  Table 10-1: SW12 current limits

Temperature (°C)	Current Limit [1] (mA)
–40°	1660
0°	1290
20°	1100

Table 10-1: SW12 current limits

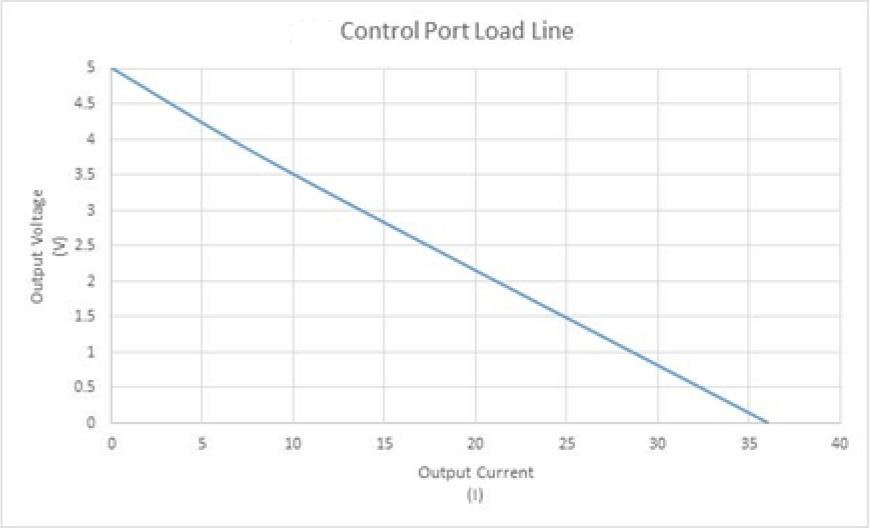
Temperature (°C)	Current Limit [1] (mA)
50°	830
70°	640
85°	500

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

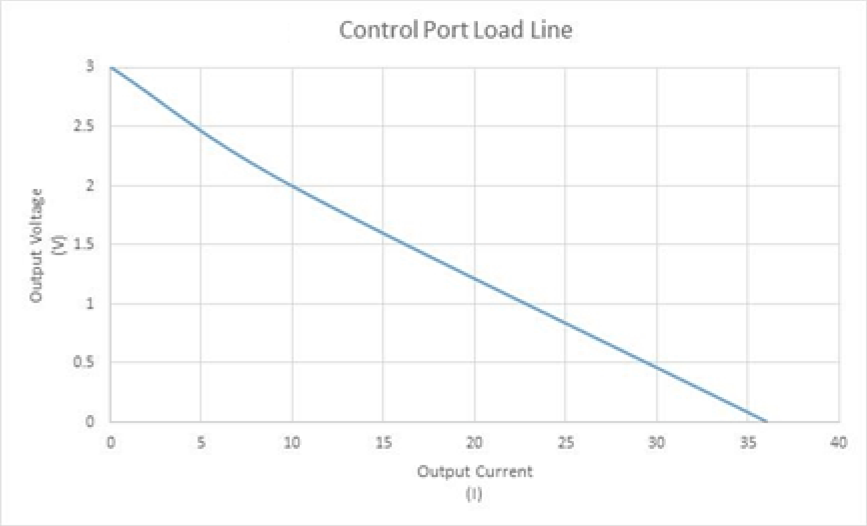
U and C as Power Output:

Values 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 if 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. It is expressed as: $V_o = 4.9\text{ V} - (R_o \cdot I_o)$, where V_o is the drive limit, and I_o is the current required by the external device.

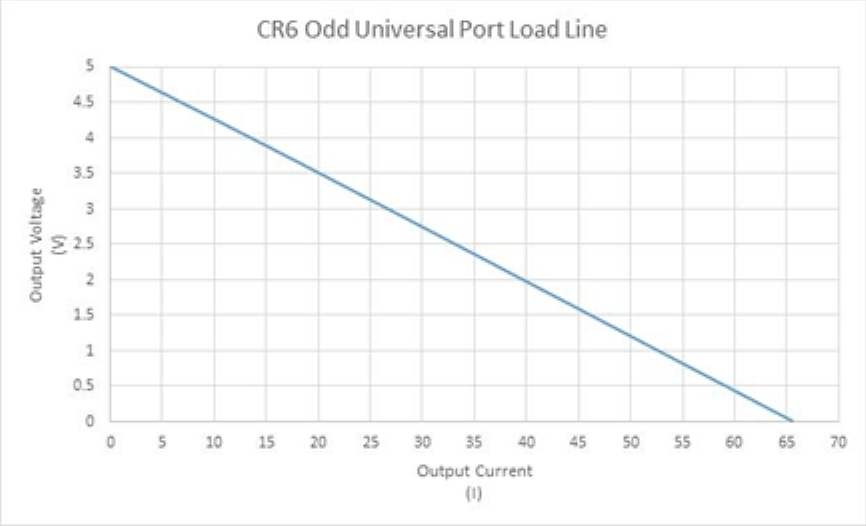
- **C** Terminals:
  - **Output Resistance (R_o)**: 150 Ω
  - **5 V Logic Level Drive Capacity**: 10 mA @ 3.5 Vdc; $V_{out} = 5\text{ V} - (I_{out} \times 150\ \Omega)$



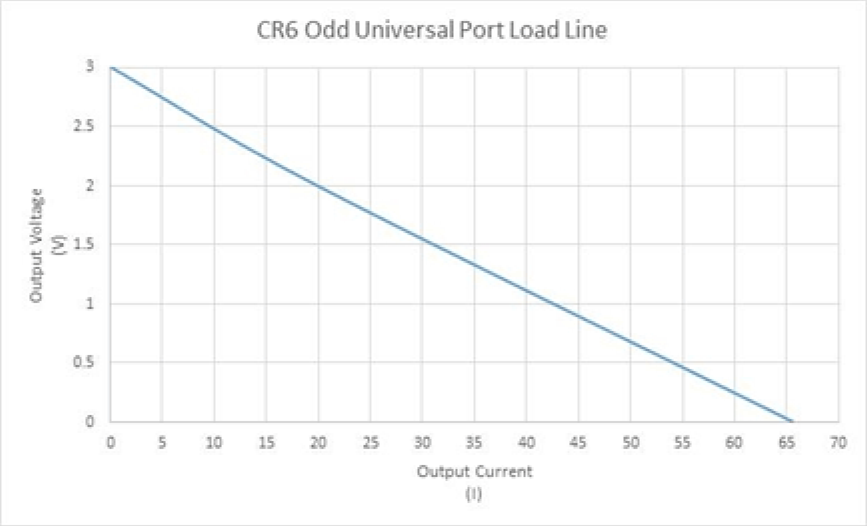
- **3.3 V Logic Level Drive Capacity**: 10 mA @ 1.8 Vdc; V_{out} = 3.3 V - (I_{out} × 150 Ω)



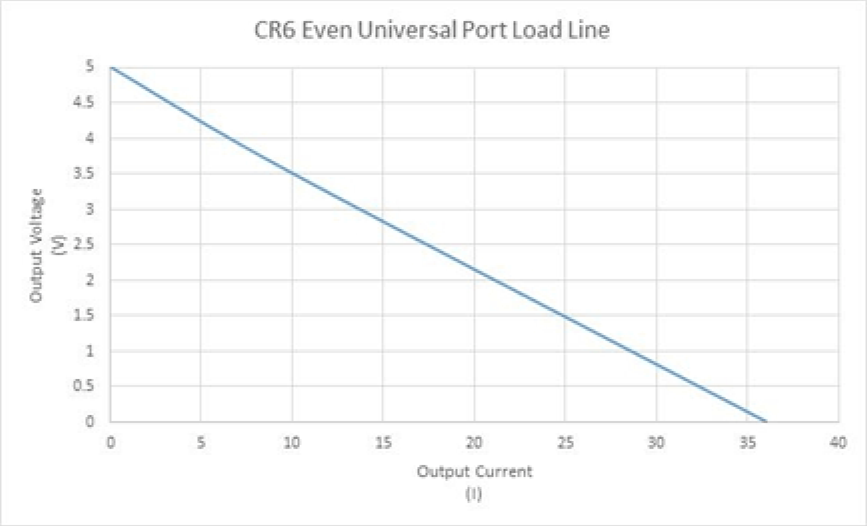
- U1, U3, U5, U7, U9, U11:
  - **Voltage Excitation Max Current @ ±2500 mV**: ±25 mA
  - **Current Excitation Max Current**: ±2500 µA
  - **5 V Logic Level Output Resistance (R_o)**: 75 Ω
  - **5 V Logic Level Max Current @ 3.5V**: 20 mA
  - **3.3 V Logic Level Output Resistance (R_o)**: 73 Ω
  - **3.3 V Logic Level Max Current @1.85V**: 20 mA
  - **5 V Logic Level Drive Capacity**:



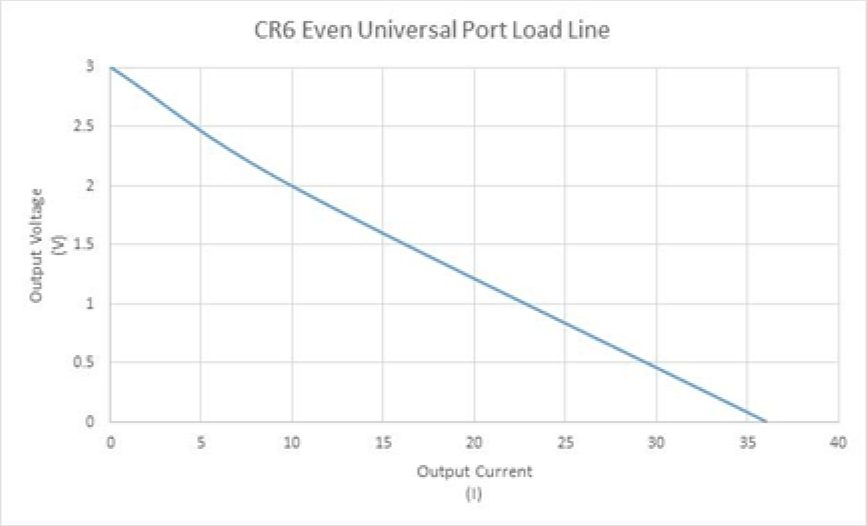
  - **3.3 V Logic Level Drive Capacity**:



- U2, U4, U6, U8, U10, U12:
  - **Voltage Excitation Max Current @ ±2500 mV**: ±25 mA
  - **Current Excitation Max Current**: ±2500 µA
  - **5 V Logic Level Output Resistance (R_o)**: 150 Ω
  - **5 V Logic Level Max Current @3.5V**: 10mA
  - **3.3 V Logic Level Output Resistance (R_o)**: 145 Ω
  - **3.3 V Logic Level Max Current @1.85V**: 10mA
  - **5 V Logic Level Drive Capacity**:



  - **3.3 V Logic Level Drive Capacity**:



- **CS I/O Pin 1**:
  - **5 V Logic Level Max Current**: 200 mA

10.6 Analog measurements specifications

12 universal inputs individually configurable for voltage, thermocouple, ratiometric, static vibrating wire, and period average measurements, using a 24-bit ADC. One channel at a time is measured in numeric succession.

10.6.1 Voltage measurements

Terminals: U1 - U12

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 @ 25 °C: ±2 nA typical

Filter First Notch Frequency (f_{N1}) Range: 5 Hz to 93 kHz (user specified)

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	20.0	19	30.0	18
	±1000	4.0	19	5.5	18
	±200	1.6	18	1.8	17
50/60[4]	±5000	1.2	23	5.0	20
	±1000	0.24	23	1.1	20
	±200	0.10	22	0.24	20
5	±5000	0.60	24	4.9	20
	±1000	0.12	24	1.0	20
	±200	0.05	23	0.22	20

[1] Valid frequencies are 0.5 Hz to 31.2593 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 corresponds 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	±10	±40
±1000	±5	±2
±200	±2	±6

Multiplexed Measurement Time: (450 µs + settling time + (1/fN1)) • 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	2.8	1.4
60	36	18.1
50	42.07	21.3
5	402.7	201.4

[1] Notch frequency (1/integration time).

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

10.6.2 Period-averaging measurements specifications

Up to 12 analog inputs may be configured for period averaging.

Accuracy: ±(0.01% of measurement + resolution), where resolution is 0.13 µs divided by the 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 * (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

10.6.3 Resistance measurements specifications

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

Accuracy: Assumes input reversal for differential measurements and excitation reversal for excitation voltage <1000 mV and excitation current < 1 mA. 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: ±(0.02% of voltage measurement + offset)
- –40 to 70 °C: ±(0.025% of voltage measurement + offset)
- –55 to 85 °C (XT): ±(0.03% of voltage measurement + offset)

10.6.4 Voltage and current excitation specifications

A 12-bit DAC produces voltage and current excitation. When used for resistance measurement, excitation is active only during measurement.

10.6.4.1 Voltage excitation

Range: ±2500 mV

Voltage Excitation Absolute Accuracy (note that ratiometric accuracy, rather than the absolute accuracy or excitation or analog measurement, determines the accuracy of ratiometric-resistance measurements):

- 0 to 40 °C: ±(0.1% of setting + 1.2 mV)
- –40 to 70 °C: ±(0.1% of setting + 1.5 mV)
- –55 to 85 °C (XT): ±(0.1% of setting + 1.6 mV)

Resolution: 0.6 mV

Maximum Source or Sink Current (exceeding current limits causes voltage output to become unstable. Voltage should stabilize when current is reduced to within stated limits): ±25 mA

10.6.4.2 Current excitation

Range: ±2.5 mA

Current Excitation Absolute Accuracy (note that ratiometric accuracy, rather than the absolute accuracy or excitation or analog measurement, determines the accuracy of ratiometric-resistance measurements):

- 0 to 40 °C: ±(0.11% of setting + 2.0 µA)
- –40 to 70 °C: ±(0.12% of setting + 2.5 µA)
- –55 to 85 °C (XT): ±(0.13% of setting + 3.0 µA)

Resolution: 0.6 µA

Compliance Voltage: ±5 V

10.6.5 Static vibrating wire measurements specifications

Up to 6 static vibrating wire measurements without thermistor measurements, or up to 3 static vibrating wire measurements with thermistor measurements. A **U** terminal pair both excites and measures vibrating wire transducers. Logarithmic sine-wave-frequency excitation is adjustable up to ±6 V (12 V peak-to-peak), programmable from 100 Hz to 6.5 kHz, then followed by frequency domain measurements, one at a time in numeric succession.

Input Resistance: 4.75 kΩ

Measurement Type: Differential voltage

Range: ±200 mV

Accuracy: ±0.013% of reading

Resolution: 0.001 Hz RMS

Measurement Speed: (vibrating wire and thermistor combined): < 1 s

10.6.6 Thermistor measurements specifications

6 **U** terminal pairs can be configured to measure two-wire thermistors directly using an onboard 5 kΩ resistor to complete the bridge. The **U** terminal pair both excites and measures the thermistor.

Input Resistance: 5 kΩ ±0.1%, 10 ppm/°C completion resistor

Measurement Type: Single-ended voltage

- **Range**: ±5000 mV
- **Accuracy**: ±0.25% of reading
- **Resolution**: 0.001 Ω RMS
- **Speed** (vibrating wire and thermistor combined): < 1 s

10.7 Current-loop measurement specifications

The datalogger makes current-loop measurements by measuring across a current-sense resistor associated with the RS-485 resistive ground terminal **RG**.

> NOTE:
> Resistance to ground input for non-isolated 0-20 mA and 4-20 mA current loop measurements is available in CR6 dataloggers with serial numbers 7502 and newer.

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: ±(0.1% of Reading + 100 nA) @ -40 to 70 °C

See also Current-loop measurements (p. 64).

10.8 Pulse measurements specifications

The datalogger can measure switch closure or high-frequency pulse signals on **C** and **U** terminals. Each terminal has its own independent 32-bit counter. Terminals are configured as pairs with options for pull-up or pull-down. Even-numbered **U** terminals can be configured as low-level ac inputs. See also Digital input/output specifications (p. 228).

Pulse Counting Programmable Terminals:

- Switch closure: **C1-C4**, **U1-U12**
- High frequency: **C1-C4**, **U1-U12**
- Low-level ac: **U2**, **U4**, **U6**, **U8**, **U10**, **U12**
  When an even numbered **U** terminal is used for low-level ac pulse counting, its odd numbered mate (for example, **U1** is the mate of **U2**) can be used only for switch-closure or high-frequency pulse counting or in the digital I/O control or state function.

Pulse Event: Transition from logic low to logic high

Maximum Input Voltage: ±20 Vdc

Maximum Counts Per Channel: 2^{32}

Maximum Counts Per Scan: 2^{32}

Input Resistance: 5 kΩ

Logic Levels for Terminal Pair Configuration:

Terminal Pair Configuration	Logic Low	Logic High
5 V	≤ 1.5 V	≥ 3.5 V
3.3 V	≤ 0.8 V	≥ 2.0 V

Accuracy: ±(0.02% of reading + 1/scan)

10.8.1 Switch closure input

Pull-Up Resistance: 100 kΩ

Maximum Input Frequency: 150 Hz

Minimum Switch Closed Time: 5 ms

Minimum Switch Open Time: 6 ms

Maximum Bounce Time: 1 ms open without being counted

Software Debounce Time: 3.3 ms

10.8.2 High-frequency input

Pull-Up Resistance: 100 kΩ

Typical Wave Form: 5 or 3.3 Vdc square wave

Maximum Input Frequency: 1 MHz

10.8.3 Low-level ac input

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 for U Terminals:

- 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

10.8.4 Quadrature input

Terminals: **U1**-**U12** can be configured as digital pairs to monitor the two sensing channels of an encoder.

Maximum Frequency: 2.5 kHz

Resolution: 31.25 µs or 32 kHz

10.9 Digital input/output specifications

16 terminals (**C1**-**C4**, **U1**-**U12**) configurable for digital input and output including status high/low, pulse width modulation, external interrupt, edge timing, 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.

Conflicts can occur when companion terminals are used for different instructions (`TimerInput`, `PulseCount`, `SDI12Recorder`, `WaitDigTrig`). For example, if **C3** is used for the `SDI12Recorder` instruction, **C4** cannot be used in the `TimerInput`, `PulseCount`, or `WaitDigTrig` instructions.

Maximum Input Voltage: ±20 V

Logic Levels and Drive Current:

Terminal Pair Configuration	5 V Source (mA @ 3.5V)	3.3 V Source (mA @ 1.85V)
C1 - C4	10	10
U1, U3, U5, U7, U9, U11	20	20
U2, U4, U6, U8, U10, U12	10	10

Digital I/O Function	C1	C2	C3	C4	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	RS-232/CPI
Switch Closure	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
High-Frequency	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
General I/O	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Pulse-Width Modulation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Timer Input	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Interrupt	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
SDI-12	✓		✓		✓		✓		✓		✓		✓		✓		
GPS Time Sync	Tx	Rx	Tx	Rx	PPS												✓
5 V or 3.3 V TTL or RS-232	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	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	Data	Clk	Enable		Data	Clk	Enable		Data	Clk	Enable		Data	Clk	Enable		
I2C	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	SCL	
SPI	MOSI	SCLK	MISO		MOSI	SCLK	MISO		MOSI	SCLK	MISO		MOSI	SCLK	MISO		
CPI/CDM																	✓

10.9.1 Edge timing

Maximum Input Frequency: ≤ 2.2 kHz

Resolution: 520 ns

10.9.2 Edge counting

Maximum Input Frequency: 250 kHz

10.9.3 Pulse width modulation

Modulation Voltage: Logic high

Maximum Period: 128 s

Resolution:

- **0 to ≤ 5 ms**: 1/12 MHz or 83.33 ns
- **> 5 to ≤ 300 ms**: 187.62 kHz or 5.33 µs
- **> 300 ms**: 1/32 kHz or 31.25 µs

10.10 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 Ethernet communications (p. 20).

Internet Protocols: Ethernet, PPP, CS I/O IP, RNDIS, ICMP/Ping, Auto-IP(APIPA), IPv4, IPv6, UDP, TCP, TLS, DNS, DHCP, SLAAC, SNMPv3, 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: TOA5, TOB1, TOB3, CSV, XML, JSON, binary, encrypted

USB: Micro-B device for computer connectivity

CS I/O: 9-pin D-sub connector 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 CR6 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 CR6 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.

0 – 5 V Serial (**U1** to **U12**, **C1** to **C4**): Eight independent TX/RX pairs

RS-485 (**C1** to **C4**): One full duplex or two half duplex

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

RS-232/CPI 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

SDI-12 (**C1**, **C3**, **U1**, **U3**, **U5**, **U7**, **U9**, **U11**): Eight independent SDI-12 compliant terminals are individually configured and meet SDI-12 Standard v 1.4.

Antenna Connection:

- Wi-Fi
- Cellular
- SS 900 MHz
- 2.4 GHz

Wireless: VHF, UHF, spread spectrum, ELOS

Hardwired: Multi-drop, short haul, RS-232, fiber optic

Satellite: GOES, Argos, Inmarsat Hughes, Irridium

10.10.1 Wi-Fi option specifications

WLAN (Wi-Fi) (CR6-WIFI only)

Maximum Possible Over-the-Air Data Rates: <11 Mbps over 802.11b, <54 Mbps over 802.11g, <72 Mbps over 802.11n

Operating Frequency: 2.4 GHz, 20 MHz bandwidth

Antenna Connector: Reverse Polarity SMA (RPSMA)

Antenna (shipped with datalogger): Unity gain (0 dBd), 1/2 wave whip, omnidirectional. Features an articulating knuckle joint that can be oriented vertically or at right angles

Supported Technologies: 802.11 a/b/g/n, WPA/WPA2-Personal, WPA/WPA2-Enterprise Security, WEP

Client Mode: WPA/WPA2-Personal and Enterprise, WEP

Access Point Mode: WPA2-Personal

WiFi Average Additional Current Contribution @ 12 Vdc:

- **Client Mode**: 7 mA idle, 70 mA communicating
- **Access Point Mode**: 62 mA idle, 70 mA communicating
- **Sleep (using IPNetPower or DevConfig setting to disable)**: <1 mA

Receive Sensitivity: -97 dBm

10.10.2 RF radio option specifications

Antenna Terminal: Reverse Polarity SMA (RPSMA)

Radio Type

- **RF407, RF412**,**RF427**, and **RF451**: Frequency Hopping Spread Spectrum (FHSS)
- **RF422**: SRD860 Radio with Listen before Talk (LBT) and Automatic Frequency Agility (AFA)

Frequency

- **RF407**: 902 to 928 MHz (US, Canada)
- **RF412**: 915 to 928 MHz (Australia, New Zealand)
- **RF422**: 863 to 870 MHz (European Union)
- **RF427**: 902 to 907.5 MHz/915 to 928 MHz (Brazil)
- **RF451**: 902 to 928 MHz

Transmit Power Output (software selectable)

- **RF407** and **RF412**: 5 to 250 mW
- **RF422**: 2 to 25 mW
- **RF427**: 5 to 250 mW
- **RF451**: 10 mW to 1,000 mW

Channel Capacity

- **RF407**: Eight 25-channel hop sequences sharing 64 available channels.
- **RF412**: Eight 25-channel hop sequences sharing 31 available channels.
- **RF422**: Ten 30-channel hop sequences (default), software configurable to meet local regulations; 10 sequences for reducing interference through channel hop.
- **RF427**: Eight 25-channel hop sequences sharing 43 available channels.
- **RF451**: 50 to 112 user-selectable channels for a given network.

Receive Sensitivity

- **RF407**, **RF412**, and **RF427**: –101 dBm
- **RF422**: –106 dBm
- **RF451**:
  - -108 dBm at 115.2 kbps for 10^{-4} BER
  - -103 dBm at 153.6 kbps for 10^{-4} BER

RF Data Rate

- **RF407**, **RF412**, and **RF427**: 200 kbps
- **RF422**: 10 kbps
- **RF451**: 115.2 or 153.6 kbps

For RF additional current contribution specifications, see Power requirements (p. 214).

See also Radio communications (p. 27).

11. 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.

alias

A second name assigned to variable in CRBasic.

allowed neighbor list

In PakBus networking, an allowed neighbor list is a list of neighbors with which a device will communicate. If a device address is entered in an allowed neighbor list, a hello exchange will be initiated with that device. Any device with an address between 1 and 3999 that is not entered in the allowed neighbor list will be filtered from communicating with the device using the list.

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.

APN

Cellular Access Point Name (obtained from your cellular network provider)

argument

Part of a procedure call (or command execution).

array

A group of variables as declared in CRBasic.

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 wire since only small currents are carried when measurements are made.

B

baud rate

The rate at which data is 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 and later) 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 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 communications protocols.

CRBasic Editor

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

CRC

Cyclic Redundancy Check

CRD

An optional memory drive that resides on a memory card.

CS I/O

Campbell Scientific proprietary input/output port. Also, the proprietary serial communications 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 is 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-data memory as the result of a data-output processing instruction. Data points output at the same time make up a record in a data table.

data table

A concept that describes how data is 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 datalogger drives, and in binary or ASCII files that result from collecting final-data 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 wires. The phenomenon measured is proportional to the difference in voltage between the two wires.

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 datalogger, it denotes the pin configuration, gender, and function of an RS-232 port. The RS-232 port on the datalogger 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 datalogger is programmed with 1 second scan interval, but the program completes after only 100 milliseconds, 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.

endian

The sequential order in which bytes are arranged into larger numerical values when stored in memory.

engineering units

Units that explicitly describe phenomena, as opposed to, for example, the datalogger 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

FAT

File Allocation Table - a computer file system architecture and a family of industry-standard file systems utilizing it.

FFT

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

field

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, PC200W, Device Configuration Utility, 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 is 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 is 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 is 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 is 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

In a PakBus network, this is 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 USR:, 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, communications, 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 for a 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 °F = °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 communicates 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 communications 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 • 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 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(), Maximum(), Minimum(), 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-data memory, but the CRBasic program can be written to divert to variable memory 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 developed by Campbell Scientific to facilitate communications between Campbell Scientific devices. Similar in concept to IP (Internet Protocol), PakBus is a packet-switched network protocol with routing capabilities. A registered trademark of Campbell Scientific, Inc.

PakBus Graph

Software that shows the relationship of various nodes in a PakBus network and allows for monitoring and adjustment of some registers in each node.

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 is 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.

PDP

Packet Data Protocol

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 follow 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. 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.

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 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

A 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. Routers maintain a routing table, which is a list of known nodes and routes. A router will only accept and forward packets that are destined for known devices. Routers pass their lists of known neighbors to other routers to build the network routing system.

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.

SCADA

Supervisory Control And Data Acquisition

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 hardwire 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, PakBus Graph, 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 wire 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.

SMS

Short message service. A text messaging service for web and mobile device systems.

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.

SRAM

Static Random-Access Memory

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

See data table.

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 is 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 communications.

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, a solar panel, an ac/ac or ac/dc wall adapter, a charge controller, and a rechargeable battery.

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 µV, 10 mV becomes 5 mV, and so forth. Resistive-bridge circuits are voltage dividers.

volts

SI unit for electrical potential.

VSPECT®

® A registered 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.

X

XML

Extensible markup language.

T

τ

Time constant

12. Index

Campbell Scientific Worldwide Offices

Australia

Location: Garbutt, QLD Australia
Email: *info@campbellsci.com.au*
Website: *www.campbellsci.com.au*

Brazil

Location: São Paulo, SP Brazil
Email: *andread@campbellsci.com.br*
Website: *www.campbellsci.com.br*

Canada

Location: Edmonton, AB Canada
Email: *dataloggers@campbellsci.ca*
Website: *www.campbellsci.ca*

China

Location: Beijing, P. R. China
Email: *info@campbellsci.com.cn*
Website: *www.campbellsci.com.cn*

Costa Rica

Location: San José, Costa Rica
Email: *info@campbellsci.cc*
Website: *www.campbellsci.cc*

France

Location: Antony, France
Email: *info@campbellsci.fr*
Website: *www.campbellsci.fr*

Germany

Location: Bremen, Germany
Email: *info@campbellsci.de*
Website: *www.campbellsci.de*

South Africa

Location: Stellenbosch, South Africa
Email: *sales@csafrica.co.za*
Website: *www.campbellscientific.co.za*

Southeast Asia

Location: Bangkok, Thailand
Email: *info@campbellsci.asia*
Website: *www.campbellsci.asia*

Spain

Location: Barcelona, Spain
Email: *info@campbellsci.es*
Website: *www.campbellsci.es*

UK

Location: Shepshed, Loughborough, UK
Email: *sales@campbellsci.co.uk*
Website: *www.campbellsci.co.uk*

USA

Location: Logan, UT USA
Email: *info@campbellsci.com*
Website: *www.campbellsci.com*

Please visit *www.campbellsci.com/contact* to obtain contact information
for your local US or international representative.