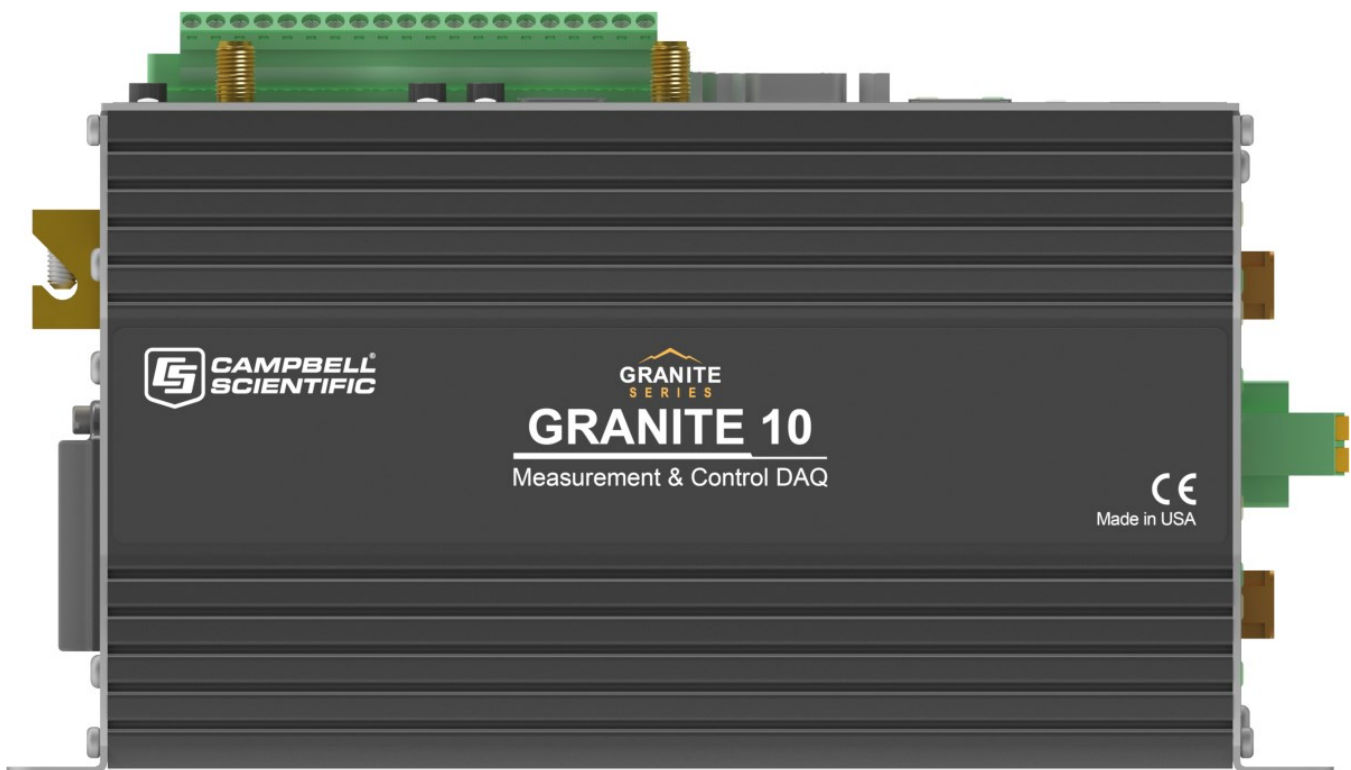




GRANITE 9 and GRANITE 10

Measurement and Control DAQ



Revision: 01/27/2021

Copyright © 2000 – 2021

Campbell Scientific
CSL I.D - 1323



Guarantee

This equipment is guaranteed against defects in materials and workmanship. We will repair or replace products which prove to be defective during the guarantee period as detailed on your invoice, provided they are returned to us prepaid. The guarantee will not apply to:

- Equipment which has been modified or altered in any way without the written permission of Campbell Scientific
- Batteries
- Any product which has been subjected to misuse, neglect, acts of God or damage in transit.

Campbell Scientific will return guaranteed equipment by surface carrier prepaid. Campbell Scientific will not reimburse the claimant for costs incurred in removing and/or reinstalling equipment. This guarantee and the Company's obligation thereunder is in lieu of all other guarantees, expressed or implied, including those of suitability and fitness for a particular purpose. Campbell Scientific is not liable for consequential damage.

Please inform us before returning equipment and obtain a Repair Reference Number whether the repair is under guarantee or not. Please state the faults as clearly as possible, and if the product is out of the guarantee period it should be accompanied by a purchase order. Quotations for repairs can be given on request. It is the policy of Campbell Scientific to protect the health of its employees and provide a safe working environment, in support of this policy a "Declaration of Hazardous Material and Decontamination" form will be issued for completion.

When returning equipment, the Repair Reference Number must be clearly marked on the outside of the package. Complete the "Declaration of Hazardous Material and Decontamination" form and ensure a completed copy is returned with your goods. Please note your Repair may not be processed if you do not include a copy of this form and Campbell Scientific Ltd reserves the right to return goods at the customers' expense.

Note that goods sent air freight are subject to Customs clearance fees which Campbell Scientific will charge to customers. In many cases, these charges are greater than the cost of the repair.



Campbell Scientific Ltd,
80 Hathern Road,
Shepshed, Loughborough, LE12 9GX, UK
Tel: +44 (0) 1509 601141
Fax: +44 (0) 1509 270924
Email: support@campbellsci.co.uk
www.campbellsci.co.uk

About this manual

Please note that this manual was originally produced by Campbell Scientific Inc. primarily for the North American market. Some spellings, weights and measures may reflect this origin.

Some useful conversion factors:

Area: 1 in ² (square inch) = 645 mm ²	Mass: 1 oz. (ounce) = 28.35 g 1 lb (pound weight) = 0.454 kg
Length: 1 in. (inch) = 25.4 mm 1 ft (foot) = 304.8 mm 1 yard = 0.914 m 1 mile = 1.609 km	Pressure: 1 psi (lb/in ²) = 68.95 mb Volume: 1 UK pint = 568.3 ml 1 UK gallon = 4.546 litres 1 US gallon = 3.785 litres

In addition, while most of the information in the manual is correct for all countries, certain information is specific to the North American market and so may not be applicable to European users.

Differences include the U.S standard external power supply details where some information (for example the AC transformer input voltage) will not be applicable for British/European use. *Please note, however, that when a power supply adapter is ordered it will be suitable for use in your country.*

Reference to some radio transmitters, digital cell phones and aerials may also not be applicable according to your locality.

Some brackets, shields and enclosure options, including wiring, are not sold as standard items in the European market; in some cases alternatives are offered. Details of the alternatives will be covered in separate manuals.

Part numbers prefixed with a “#” symbol are special order parts for use with non-EU variants or for special installations. Please quote the full part number with the # when ordering.

Recycling information



At the end of this product's life it should not be put in commercial or domestic refuse but sent for recycling. Any batteries contained within the product or used during the products life should be removed from the product and also be sent to an appropriate recycling facility.

Campbell Scientific Ltd can advise on the recycling of the equipment and in some cases arrange collection and the correct disposal of it, although charges may apply for some items or territories.

For further advice or support, please contact Campbell Scientific Ltd, or your local agent.



Campbell Scientific Ltd, 80 Hathern Road, Shepshed, Loughborough, LE12 9GX,
UK Tel: +44 (0) 1509 601141 Fax: +44 (0) 1509 270924
Email: support@campbellsci.co.uk
www.campbellsci.co.uk

Safety

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.eu or by telephoning +44(0) 1509 828 888 (UK). 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.

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.

Table of Contents

1. GRANITE 9/10 data acquisition system components	1
1.1 The GRANITE 9/10 data acquisition system	2
1.1.1 Overview	2
1.1.2 Operations	3
1.1.3 Programs	3
1.2 Sensors	3
2. Wiring panel and terminal functions	5
2.1 Power input	9
2.1.1 Powering a data logger with a vehicle	9
2.1.2 Power LED indicator	10
2.2 Power output	10
2.3 Grounds	11
2.4 Communications ports	12
2.4.1 USB device port	12
2.4.2 USB host port	13
2.4.3 Ethernet port	13
2.4.4 C terminals for communications	13
2.4.4.1 SDI-12 ports	13
2.4.4.2 RS-232, RS-422, RS-485, TTL, and LVTTTL ports	14
2.4.4.3 SDM ports	14
2.4.5 CS I/O port	14
2.4.6 CPI/RS-232 port	15
2.4.7 EPI port	16
2.4.8 CAN port (GRANITE 10 only)	17
2.5 Programmable logic control	18
3. Setting up the GRANITE 9/10	20
3.1 Setting up communications with the data logger	20
3.1.1 USB or RS-232 communications	21
3.1.2 Virtual Ethernet over USB (RNDIS)	22
3.1.3 Ethernet communications option	23
3.1.3.1 Configuring data logger Ethernet settings	23

3.1.3.2 Ethernet LEDs	24
3.1.3.3 Setting up Ethernet communications between the data logger and computer	25
3.1.4 Wi-Fi communications	26
3.1.4.1 Configuring the data logger to host a Wi-Fi network	26
3.1.4.2 Connecting your computer to the data logger over Wi-Fi	27
3.1.4.3 Setting up Wi-Fi communications between the data logger and the data logger support software	27
3.1.4.4 Configuring data loggers to join a Wi-Fi network	28
3.1.4.5 Wi-Fi mode button	29
3.1.4.6 Wi-Fi LED indicator	29
3.2 Testing communications with EZSetup	30
3.3 Making the software connection	31
3.4 Programming quickstart using Short Cut	31
3.5 Sending a program to the data logger	35
4. Working with data	36
4.1 Default data tables	36
4.2 Collecting data	37
4.2.1 Collecting data using LoggerNet	37
4.2.2 Collecting data using RTDAQ	37
4.2.3 Collecting data using an FTP client	38
4.3 Viewing historic data	39
4.4 Data types and formats	39
4.4.1 Variables	40
4.4.2 Constants	41
4.4.3 Data storage	42
4.5 About data tables	43
4.5.1 Table definitions	43
4.5.1.1 Header rows	43
4.5.1.2 Data records	45
4.6 Creating data tables in a program	46
5. Data memory	48
5.1 Data tables	48
5.2 Memory allocation	48
5.3 SRAM	49
5.3.1 USB drive	50

5.4 DDR-SDRAM	51
5.5 SRAM vs. DDR-SDRAM	51
5.6 SSD - Hard Drive	51
5.7 Flash memory	52
5.7.1 eMMC NAND flash memory	52
5.7.2 NOR flash memory	52
5.7.3 CPU drive	52
5.8 MicroSD (CRD: drive)	52
5.8.1 Formatting microSD cards	53
5.8.2 MicroSD card precautions	54
5.8.3 Act LED indicator	54
5.9 USB Host (USB: drive)	54
5.9.1 USB Host precautions	55
5.9.2 Act LED indicator	55
5.9.3 Formatting drives 32 GB or larger	55
6. Measurements	56
6.1 Pulse measurements	56
6.1.1 High-frequency measurements	57
6.1.1.1 C terminals	58
6.1.2 Switch-closure and open-collector measurements	58
6.1.2.1 C Terminals	58
6.1.3 Edge timing and edge counting	59
6.1.3.1 Single edge timing	59
6.1.3.2 Multiple edge counting	59
6.1.3.3 Timer input NAN conditions	59
6.1.4 Quadrature measurements	60
6.1.5 Pulse measurement tips	61
6.1.5.1 Input filters and signal attenuation	61
6.1.5.2 Pulse count resolution	62
6.2 Sequential and pipeline processing modes	62
6.2.1 Sequential mode	62
6.2.2 Pipeline mode	63
6.2.3 Slow Sequences	63
7. Communications protocols	65
7.1 General serial communications	66

7.1.1 RS-232	68
7.1.2 RS-485	69
7.1.3 RS-422	70
7.1.4 TTL	71
7.1.5 LVTTTL	71
7.1.6 TTL-Inverted	71
7.1.7 LVTTTL-Inverted	72
7.2 CPI	72
7.3 EPI	73
7.4 CAN (GRANITE 10 only)	73
7.4.1 CRBasic instructions	74
7.4.1.1 CANPortOpen() / CANFDPortOpen	74
7.4.1.2 CANRead() / CANFDRead()	77
7.4.1.3 CANFilterRead() / CANFDFilterRead()	78
7.4.1.4 CANWrite() / CANFDWrite()	78
7.4.2 DBC Signal Support	78
7.4.3 J1979 Legislated PIDS Example	81
7.5 Modbus communications	81
7.5.1 About Modbus	82
7.5.2 Modbus protocols	83
7.5.3 Understanding Modbus Terminology	84
7.5.4 Connecting Modbus devices	84
7.5.5 Modbus master-slave protocol	84
7.5.6 About Modbus programming	85
7.5.6.1 Endianness	85
7.5.6.2 Function codes	86
7.5.7 Modbus information storage	87
7.5.7.1 Registers	87
7.5.7.2 Coils	87
7.5.7.3 Data Types	88
Unsigned 16-bit integer	88
Signed 16-bit integer	88
Signed 32-bit integer	89
Unsigned 32-bit integer	89
32-Bit floating point	89
7.5.8 Modbus tips and troubleshooting	89
7.5.8.1 Error codes	89

Result code -01: illegal function	89
Result code -02: illegal data address	90
Result code -11: COM port error	90
7.6 Internet communications	90
7.6.1 IP address	91
7.6.2 HTTPS server	91
7.6.3 FTP server	91
7.7 DNP3 communications	93
7.8 Serial peripheral interface (SPI) and I2C	93
7.9 PakBus communications	93
7.10 SDI-12 communications	94
7.10.1 SDI-12 transparent mode	95
7.10.1.1 Watch command (sniffer mode)	96
7.10.1.2 SDI-12 transparent mode commands	97
7.10.2 SDI-12 programmed mode/recorder mode	97
7.10.3 Programming the data logger to act as an SDI-12 sensor	98
7.10.4 SDI-12 power considerations	98
8. GRANITE 9/10 maintenance	100
8.1 Data logger calibration	100
8.2 Data logger security	101
8.2.1 TLS	102
8.2.2 Security codes	103
8.2.3 Creating a .csipasswd file	104
8.2.3.1 Command syntax	105
8.3 Data logger enclosures	105
8.3.1 Mounting in an enclosure	106
8.4 Internal battery	107
8.4.1 Replacing the internal battery	108
8.5 Electrostatic discharge and lightning protection	109
8.6 Power budgeting	111
8.7 Updating the operating system	111
8.7.1 Sending an operating system to a local data logger	112
8.7.2 Sending an operating system to a remote data logger	113
8.8 File management via powerup.ini	114
8.8.1 Syntax	115
8.8.2 Example powerup.ini files	116

9. Tips and troubleshooting	118
9.1 Checking station status	119
9.1.1 Viewing station status	119
9.1.2 Watchdog errors	120
9.1.3 Results for last program compiled	120
9.1.4 Skipped scans	121
9.1.5 Skipped records	121
9.1.6 Variable out of bounds	121
9.1.7 Battery voltage	121
9.2 Understanding NAN and INF occurrences	121
9.3 Timekeeping	122
9.3.1 Clock best practices	122
9.3.2 GPS	123
9.3.3 Time stamps	123
9.3.4 Avoiding time skew	124
9.4 CRBasic program errors	124
9.4.1 Program does not compile	125
9.4.2 Program compiles but does not run correctly	125
9.5 Resetting the data logger	126
9.5.1 Processor reset	126
9.5.2 Program send reset	126
9.5.3 Manual data table reset	127
9.5.4 Formatting drives	127
9.5.5 Full memory reset	127
9.6 Troubleshooting power supplies	127
9.6.1 SDI-12 transparent mode	128
9.6.1.1 Watch command (sniffer mode)	129
9.6.1.2 SDI-12 transparent mode commands	130
9.7 Ground loops	130
9.7.1 Common causes	131
9.7.2 Detrimental effects	131
9.7.3 Severing a ground loop	133
9.8 Field calibration	133
9.9 File system error codes	134
9.10 File name and resource errors	135
9.11 Background calibration errors	135

10. Information tables and settings (advanced)	136
10.1 DataTableInfo table system information	137
10.1.1 DataFillDays	137
10.1.2 DataRecordSize	137
10.1.3 DataTableName	137
10.1.4 RecNum	137
10.1.5 SecsPerRecord	138
10.1.6 SkippedRecord	138
10.1.7 TimeStamp	138
10.2 Status table system information	138
10.2.1 Battery	138
10.2.2 BuffDepth	138
10.2.3 CardStatus	138
10.2.4 CommsMemFree	139
10.2.5 CompileResults	139
10.2.6 ErrorCalib	139
10.2.7 FullMemReset	139
10.2.8 LastSystemScan	139
10.2.9 LithiumBattery	139
10.2.10 Low12VCount	139
10.2.11 MaxBuffDepth	139
10.2.12 MaxProcTime	140
10.2.13 MaxSystemProcTime	140
10.2.14 MeasureOps	140
10.2.15 MeasureTime	140
10.2.16 MemoryFree	140
10.2.17 MemorySize	140
10.2.18 Messages	140
10.2.19 OSDate	141
10.2.20 OSSignature	141
10.2.21 OSVersion	141
10.2.22 PakBusRoutes	141
10.2.23 CPUtemp	141
10.2.24 PortConfig	141
10.2.25 PortStatus	141
10.2.26 ProcessTime	142
10.2.27 ProgErrors	142

10.2.28 ProgName	142
10.2.29 ProgSignature	142
10.2.30 RecNum	142
10.2.31 RevBoard	142
10.2.32 RunSignature	142
10.2.33 SerialNumber	143
10.2.34 SkippedScan	143
10.2.35 SkippedSystemScan	143
10.2.36 StartTime	143
10.2.37 StartUpCode	143
10.2.38 StationName	143
10.2.39 SW12Volts	143
10.2.40 SystemProcTime	144
10.2.41 TimeStamp	144
10.2.42 VarOutOfBound	144
10.2.43 WatchdogErrors	144
10.2.44 WiFiUpdateReq	144
10.3 CPIStatus system information	144
10.3.1 BusLoad	145
10.3.2 ModuleReportCount	145
10.3.3 ActiveModules	145
10.3.4 BuffErr (buffer error)	145
10.3.5 RxErrMax	145
10.3.6 TxErrMax	145
10.3.7 FrameErr (frame errors)	145
10.3.8 ModuleInfo array	146
10.4 Settings	146
10.4.1 Baudrate	146
10.4.2 Beacon	147
10.4.3 CentralRouters	147
10.4.4 CommsMemAlloc	147
10.4.5 DNS	147
10.4.6 EthernetInfo	148
10.4.7 EthernetPower	148
10.4.8 FilesManager	148
10.4.9 FTPEnabled	148
10.4.10 FTPPassword	148

10.4.11 FTPPort	148
10.4.12 FTPUserName	148
10.4.13 HTTPEnabled	149
10.4.14 HTTPPort	149
10.4.15 HTTPSEnabled	149
10.4.16 HTTPSPort	149
10.4.17 IncludeFile	149
10.4.18 IPAddressEth	149
10.4.19 IPGateway	149
10.4.20 IPMaskEth	150
10.4.21 IPTrace	150
10.4.22 IPTraceCode	150
10.4.23 IPTraceComport	150
10.4.24 IsRouter	150
10.4.25 MaxPacketSize	151
10.4.26 Neighbours	151
10.4.27 PakBusAddress	151
10.4.28 PakBusEncryptionKey	151
10.4.29 PakBusNodes	151
10.4.30 PakBusPort	151
10.4.31 PakBusTCPClients	152
10.4.32 PakBusTCPEnabled	152
10.4.33 PakBusTCPPassword	152
10.4.34 PingEnabled	152
10.4.35 pppDial	152
10.4.36 pppDialResponse	152
10.4.37 pppInfo	153
10.4.38 pppInterface	153
10.4.39 pppIPAddr	153
10.4.40 pppPassword	153
10.4.41 pppUsername	153
10.4.42 RouteFilters	153
10.4.43 RS232Power	154
10.4.44 Security(1), Security(2), Security(3)	154
10.4.45 ServicesEnabled	154
10.4.46 TCPClientConnections	154
10.4.47 TCPPort	154

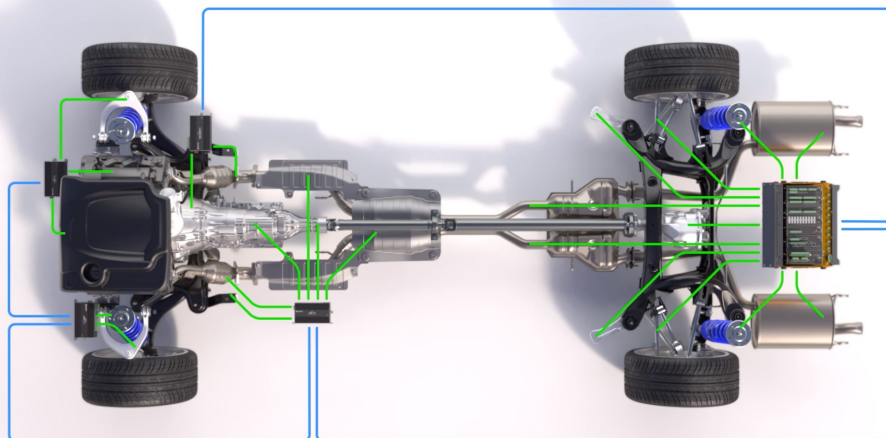
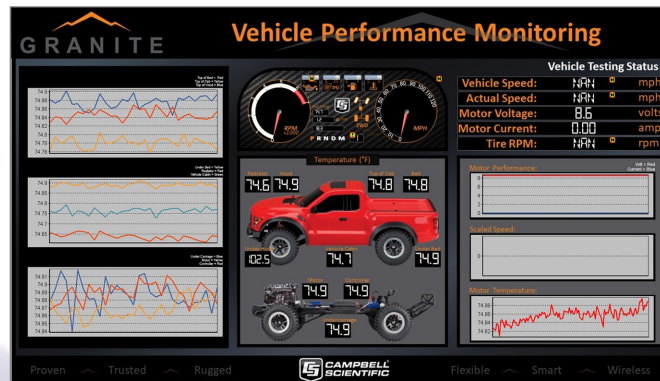
10.4.48 TelnetEnabled	154
10.4.49 TLSConnections (Max TLS Server Connections)	154
10.4.50 TLSPassword	155
10.4.51 TLSStatus	155
10.4.52 UDPBroadcastFilter	155
10.4.53 UTCOffset	155
10.4.54 Verify	155
10.4.55 Wi-Fi settings	155
10.4.55.1 IPAddressWiFi	156
10.4.55.2 IPGatewayWiFi	156
10.4.55.3 IPMaskWiFi	156
10.4.55.4 WiFiChannel	156
10.4.55.5 WiFiConfig	157
10.4.55.6 WiFiEAPMethod	157
10.4.55.7 WiFiEAPPassword	157
10.4.55.8 WiFiEAPUser	157
10.4.55.9 Networks	157
10.4.55.10 WiFiEnable	158
10.4.55.11 WiFiFwdCode (Forward Code)	158
10.4.55.12 WiFiPassword	158
10.4.55.13 WiFiPowerMode	158
10.4.55.14 WiFiSSID (Network Name)	158
10.4.55.15 WiFiStatus	159
10.4.55.16 WiFiTxPowerLevel	159
10.4.55.17 WLANDomainName	159
11. GRANITE 9/10 Specifications	160
11.1 System specifications	160
11.2 Physical specifications	162
11.3 Power requirements	162
11.4 Power output specifications	163
11.4.1 System power out limits (when powered with 12 VDC)	163
11.4.2 12 V and SW12 power output terminals	163
11.4.3 5 V fixed output	163
11.4.4 C as power output	164
11.4.5 CS I/O pin 1	164
11.4.6 CS I/O pin 8	164

11.5 Pulse measurement specifications	164
11.5.1 Switch closure input	164
11.5.2 High-frequency input	165
11.5.3 Low-level AC input	165
11.6 Digital input/output specifications	165
11.6.1 Switch closure input	166
11.6.2 High-frequency input	166
11.6.3 Edge timing	166
11.6.4 Edge counting	167
11.6.5 Quadrature input	167
11.6.6 Pulse-width modulation	167
11.6.7 Maximum time between counter or timer instructions	167
11.7 Communications specifications	167
11.7.1 Wi-Fi specifications	168
11.8 Standards compliance specifications	169
Appendix A. Glossary	170

1. GRANITE 9/10 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](#) (p. 3) for more information).

- **Data logger** - The data logger 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 data logger, usually to a computer, by one or more methods using data logger support software. Most communications options are bi-directional, which allows programs and settings to be sent to the data logger. For more information, see [Sending a program to the data logger](#) (p. 35).
- **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 data logger is adept at programmable logic control. See [Programmable logic control](#) (p. 18) for more information.
- **Measurement and Control Peripherals** - Sometimes, system requirements exceed the capacity of the data logger. The excess can usually be handled by addition of input and output expansion modules.
- **Campbell Distributed Module (CDM)** - CDMs increase measurement capability can be centrally located or distributed throughout the network. Modules are controlled and synchronized by a single GRANITE 9/10. GRANITE Measurement Modules are one type of CDM.

1.1 The GRANITE 9/10 data acquisition system

The GRANITE 9/10 data logger provides fast communications, low power requirements, and built-in host and devices USB in a compact size. It includes digital I/O (C) terminals, which allow connection to virtually any digital or smart sensor. The GRANITE 9/10 in conjunction with GRANITE Measurement Modules can be collocated in a chassis or distributed over distances of thousands of feet and all behave as a single unit for the purposes of programming, synchronization, and data collection and storage.

1.1.1 Overview

The GRANITE 9/10 data logger is the main part of a data acquisition system (see [GRANITE 9/10 data acquisition system components](#) (p. 1) for more information). It has a central-processing unit (CPU), digital measurement inputs, analogue 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 GRANITE 9/10 can simultaneously provide measurement and communications functions. Low power consumption allows the data logger to operate for extended time on a battery recharged with a solar panel, eliminating the need for ac power. The GRANITE 9/10 temporarily suspends operations when primary power drops below 9.6 V, reducing the possibility of inaccurate measurements.

1.1.2 Operations

The GRANITE 9/10 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.

Distributed measurements are one of the hallmarks of the GRANITE series. All modules are interconnected using CAT5e Ethernet cable. This makes running cables inexpensive and familiar. One of the advantage of distributed measurements is to take the GRANITE 9/10 to the sensors and shorten the sensor cables. This replaces many long sensor cables with a single inexpensive data cable. It also reduces the distance for signals to travel therefore reducing opportunities for corruption of the signals by noise.

1.1.3 Programs

A program directs the data logger on how and when sensors are measured, calculations are made, data is stored, and devices are controlled. The application program for the GRANITE 9/10 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 GRANITE 9/10 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 GRANITE 9/10 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.

GRANITE measurement modules allow flexibility both in measurement type and channel count. Most electronic sensors, regardless of manufacturer, will interface with a measurement module. The GRANITE data acquisition system 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.eu/support to assist in measuring many sensor types.

- Pulse
 - High frequency
 - Switch-closure
 - Low-level ac
 - Quadrature
- Vibrating wire
- Smart sensors
 - SDI-12
 - RS-232
 - Modbus
 - DNP3
 - TCP/IP
 - RS-422
 - RS-485

2. Wiring panel and terminal functions

The GRANITE 9/10 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 data logger functions so studying it is a good way to get acquainted with the data logger. Functions of the terminals are broken down into the following categories:

- Pulse counting
- Communications
- Digital I/O
- Power input
- Power output
- Power ground

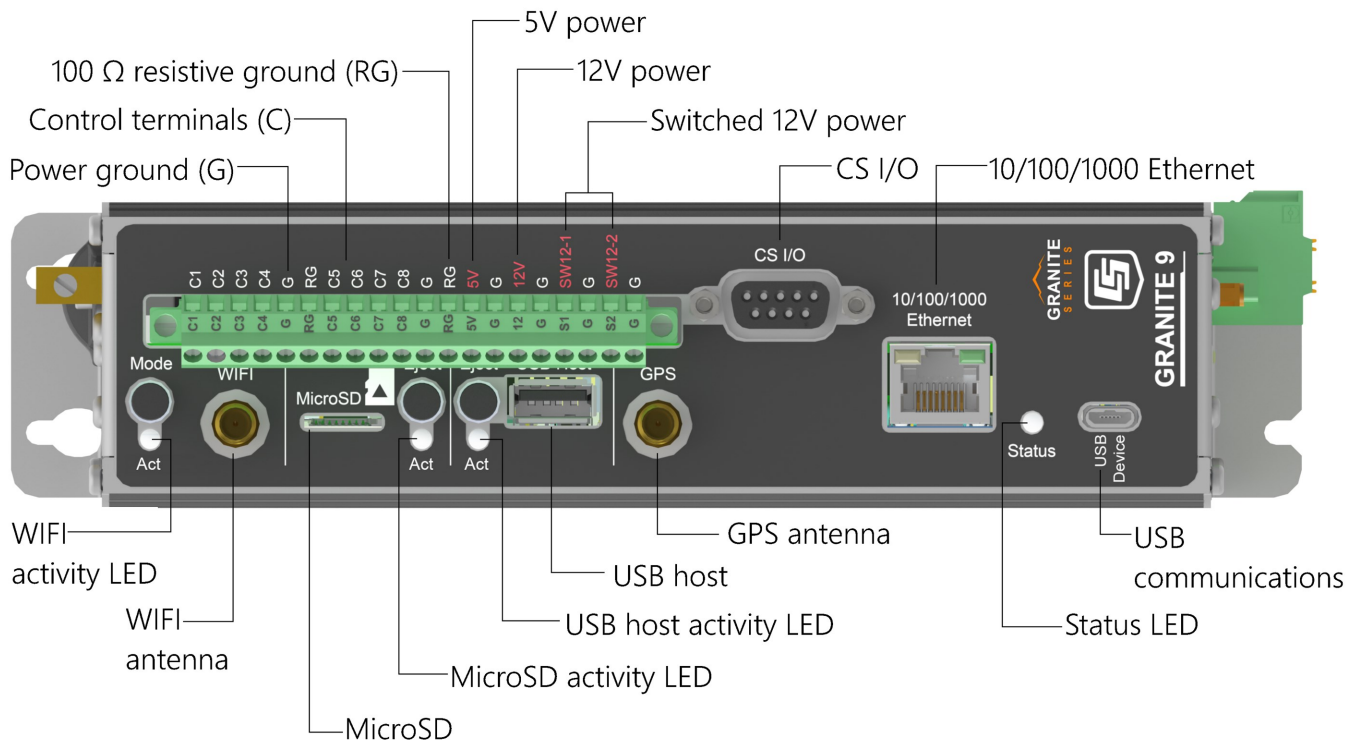


FIGURE 2-1. GRANITE 9 Wiring panel

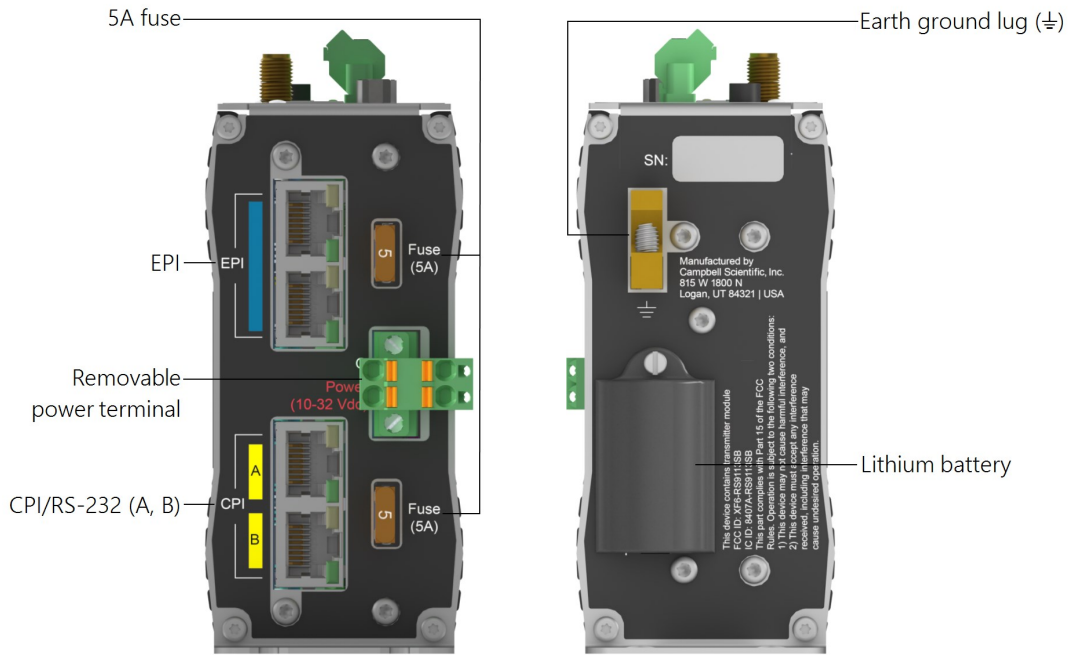


FIGURE 2-2. GRANITE 9

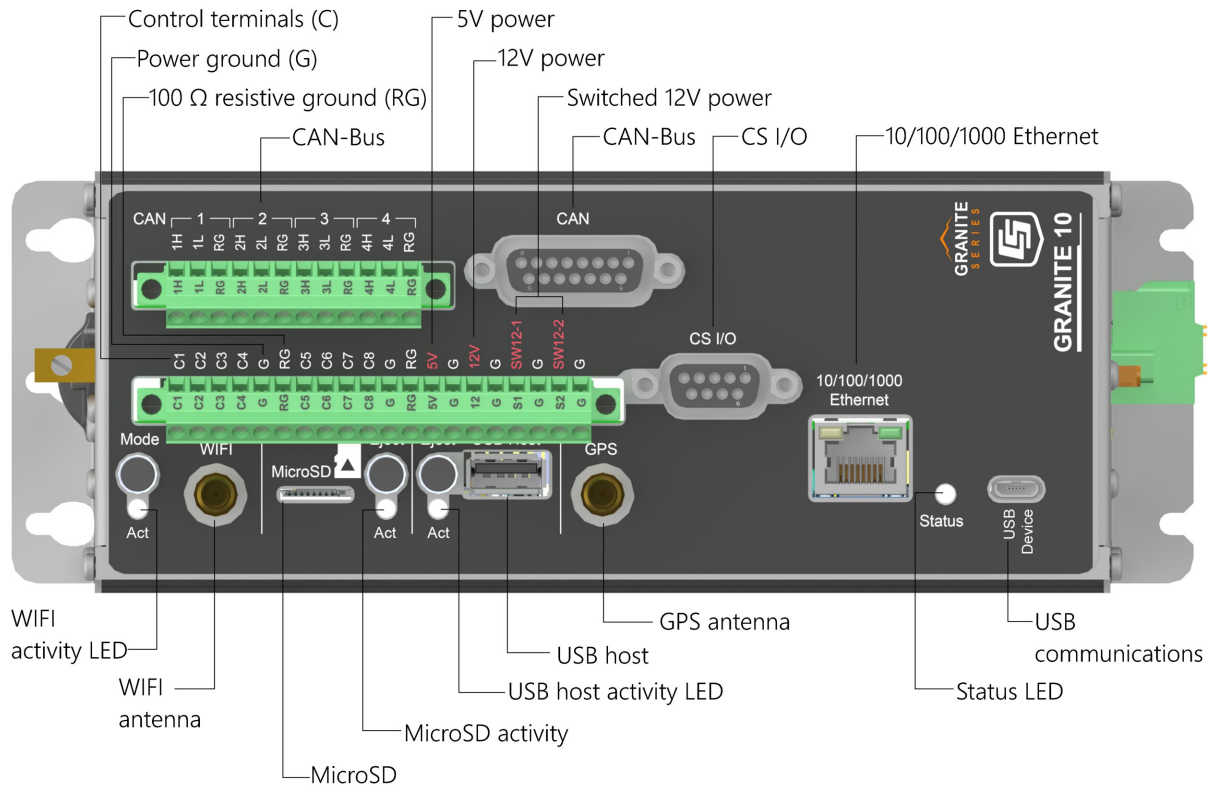


FIGURE 2-3. GRANITE 10 Wiring panel

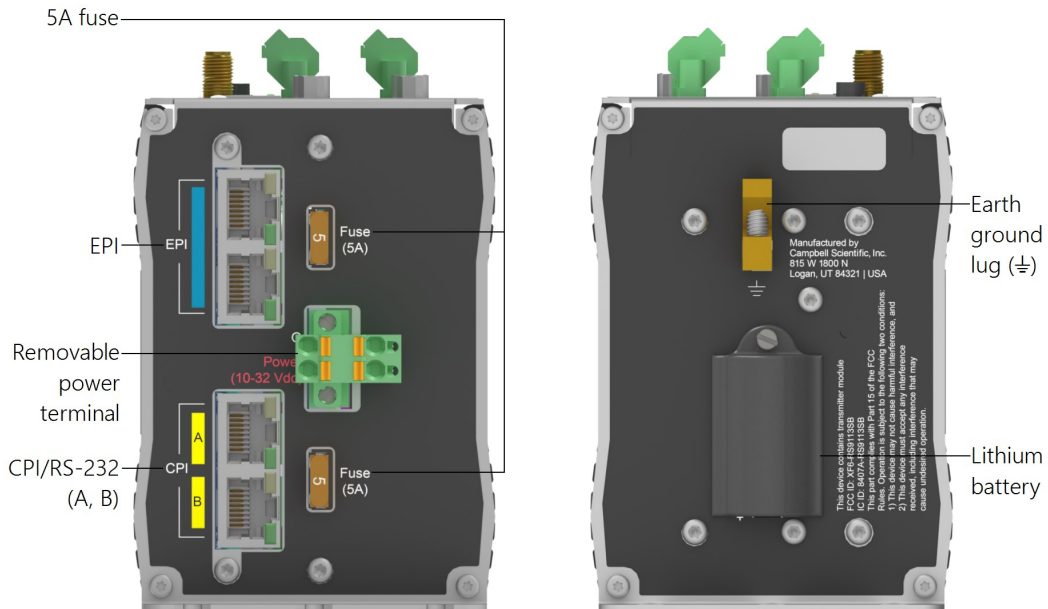


FIGURE 2-4. GRANITE 10

	C1-C8
Switch-Closure	✓
High Frequency	✓
Quadrature	

NOTE:

Conflicts can occur when a control port pair is used for different instructions ([TimerInput\(\)](#), [PulseCount\(\)](#), [SDI12Recorder\(\)](#), [WaitDigTrig\(\)](#)). For example, if C1 is used for [SDI12Recorder\(\)](#), C2 cannot be used for [TimerInput\(\)](#), [PulseCount\(\)](#), or [WaitDigTrig\(\)](#).

	C1-C8	12V	SW12-1	SW12-2	5V
3.3 VDC	✓				
5 VDC	✓				✓
+POWER IN up to 12 VDC		✓	✓	✓	

C terminals have limited drive capacity. Voltage levels are configured in pairs.

	C1	C2	C3	C4	C5	C6	C7	C8	RS-232/ CPI	GRANITE 10 only H/L/RG (1-4)
SDI-12	✓		✓				✓			
GPS Time Sync	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx		
TTL 0-5 V	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx		
LVTTL 0-3.3 V	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx		
RS-232	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	✓	
RS-485 (Half Duplex)	A-	B+	A-	B+	A-	B+	A-	B+		
RS-485 (Full Duplex)	Tx-	Tx+	Rx-	Rx+	Tx-	Tx+	Rx-	Rx+		
I2C	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA		
SPI	MOSI	SCLK	MISO		MOSI	SCLK	MISO			
SDM	Data	Clk	Enabl							
CPI/ CDM									✓	
CAN bus										✓

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

2.1 Power input

The data logger 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 data logger operates with external power connected to the green **POWER IN** port on the side of the wiring panel (see [Wiring Panel and Terminal Functions](#)). The positive power wire connects to the **Power (10-32 Vdc)** terminal. The negative wire connects to **G**. The power terminals are internally protected against polarity reversal and high voltage transients.

The primary power source, which is often a transformer, power converter, or solar panel, connects to the charging regulator, as does a nominal 12 VDC sealed rechargeable battery. A third connection connects the charging regulator to the **12V** and **G** terminals of the **POWER IN** port. UPS (uninterruptible power supply) is often the best power source for long-term installations. If external alkaline power is used, the alkaline battery pack is connected directly to the **POWER IN** port. External UPS consists of a primary-power source, a charging regulator external to the data logger, and an external battery.

WARNING:

Sustained input voltages in excess of those listed in the , 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. [Troubleshooting power supplies](#) (p. 127)

The GRANITE 9/10 cannot run solely from **USB** power.

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. For information about the internal battery, see [Internal battery](#) (p. 107).

2.1.1 Powering a data logger with a vehicle

If a data logger 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 data logger operation. This may cause the data logger 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 data logger is powered, the Status LED will turn on according to power and program states:

- **Off:** No power, no program running.
- **Green flash every 1/2 second:** Program running
- **Solid yellow:** Fault
- **Solid red:** Boot code is active

2.2 Power output

The data logger can be used as a power source for communications devices, sensors and peripherals. Take precautions to prevent damage to these external devices due to 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:** regulated 12 VDC. The 12 VDC supply is regulated to within 10% of 12 VDC as long as the main power supply for the data logger does not drop below the minimum supply voltage. See [Power requirements](#) (p. 162).
- **5V:** regulated 5 VDC. The 5 VDC supply is regulated to within a few millivolts of 5 VDC as long as the main power supply for the data logger does not drop below the minimum supply voltage. It is intended to power sensors or devices requiring a 5 VDC power supply. It is not intended as an excitation source for bridge measurements. Current output is shared with the CS I/O port; so, the total current must be within the current limit.
- **SW12:** program-controlled, regulated switched 12 VDC terminals. Voltage on a **SW12** terminal will be approximately the same as the **12V** terminal. Each SW12 terminal has an independent current limit. CRBasic instruction **SW12 (C)** controls the **SW12** terminal. See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.
- **CS I/O port:** used to communicate with and often supply power to Campbell Scientific peripheral devices.

- **C** terminals: can be set low or high as output terminals . With limited drive capacity, digital output terminals are normally used to operate external relay-driver circuits. See also [Digital input/output specifications](#) (p. 165).

See also [Power output specifications](#) (p. 163).

2.3 Grounds

Proper grounding lends stability and protection to a data acquisition system. Grounding the data logger 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 data logger grounds:

- **Power Ground (G)** - return for 3.3 V, 5 V, 12 V, **C** terminals configured for control, and digital sensors.
 - 6 common terminals
- **Resistive Ground (RG)** - used for decoupling ground on RS-485 and CAN bus (GRANITE 10 only) signals. Includes 100 Ω resistance to ground.
 - 6 common terminals, GRANITE 10
 - 2 common terminals, GRANITE 9
- **Earth Ground Lug (⚡)** - connection point for heavy-gauge earth-ground wire. A good earth connection is necessary to secure the ground potential of the data logger 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 data logger and sensors by providing a low-resistance path around the system to a point of low potential. Campbell Scientific recommends that all data loggers be earth grounded. All components of the system (data loggers, 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:

- [Ground loops](#) (p. 130)

2.4 Communications ports

The data logger 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 data loggers
- GRANITE Measurement Modules
- Vehicles using CAN bus (GRANITE 10 only)

Campbell Scientific data logger communications ports include:

- CS I/O
- CPI/RS-232
- EPI
- CAN bus (GRANITE 10 only)
- USB Device
- USB Host
- Ethernet
- C terminals

2.4.1 USB device port

One USB device port supports communicating with a computer through data logger support software or through virtual Ethernet (RNDIS), and provides 5 VDC power to the data logger (powering through the USB port has limitations - details are available in the specifications). The

data logger USB device 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 USB host port

USB host provides portable data storage on a mass storage device (MSD). A single USB thumb drive can be inserted into the drive and will show up as a drive (USB:) in file related operations. Measurement data is stored on USB: as discrete files by using the **TableFile()** instruction. Files on USB can be collected by inserting the thumb drive into a computer and copying the files.

USB: can be used in the **TableFile()** instruction and all file access related instructions in CRBasic. Because of data-reliability concerns in non-industrial rated drives, this drive is not intended for long term unattended data storage. USB: is not affected by program recompilation or formatting of other drives.

2.4.3 Ethernet port

The RJ45 **10/100/1000 Ethernet** port is used for IP communications.

2.4.4 C terminals for communications

C terminals are configurable for the following communications types:

- SDI-12
- RS-232
- RS-422
- RS-485
- TTL (0 to 5 V)
- LVTTTL (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 data logger specifications.

2.4.4.1 SDI-12 ports

SDI-12 is a 1200 baud protocol that supports many smart sensors. **C1**, **C3**, **C5**, and **C7** can 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. 94).

2.4.4.2 RS-232, RS-422, RS-485, TTL, and LVTTTL ports

RS-232, RS-422, RS-485, TTL, and LVTTTL communications are typically used for the following:

- Reading sensors with serial output
- Creating a multi-drop network
- Communications with other data loggers or devices over long cables

Configure **C** terminals as serial ports using *Device Configuration Utility* or by using the [SerialOpen\(\)](#) CRBasic instruction. **C** terminals are configured in pairs for TTL, LVTTTL, RS-232, and half-duplex RS-422 and RS-485 communications. For full-duplex RS-422 and RS-485, four **C** terminals are required. See also [Communications protocols](#) (p. 65).

NOTE:

RS-232 ports are not isolated.

2.4.4.3 SDM ports

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 **C5**, **C6**, and **C7** can be configured together to be used as the SDM port by using the [SDMBeginPort\(\)](#) instruction.

See also [Communications specifications](#) (p. 167).

2.4.5 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. 167).

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.

Pin Number	Function	Input (I) Output (O)	Description
3	RING	I	Ring: raised by a peripheral to put the GRANITE 9/10 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 GRANITE 9/10 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 data logger 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.

2.4.6 CPI/RS-232 port

The data logger includes one RJ45 module jack labeled RS-232/CPI. CPI is a proprietary interface for communications between Campbell Scientific data loggers and Campbell Distributed Modules (CDMs) such as the GRANITE-Series peripheral devices and smart sensors. 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. Some GRANITE devices may require more power to operate in general than do SDM devices. Consult the manuals for GRANITE modules for more information.

NOTE:

CPI/RS-232 port is not isolated.

CPI port power levels are controlled automatically by the GRANITE 9/10:

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

When used with a Campbell Scientific RJ45-to-DB9 converter cable, the **CPI/RS-232** 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.

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

2.4.7 EPI port

The data logger includes two RJ45 module jacks labeled EPI. Ethernet Peripheral Interface (EPI) is a proprietary interface for communications between Campbell Scientific data loggers and Campbell Distributed Modules (CDMs) such as the GRANITE-Series peripheral devices and smart sensors. EPI expands the functionality or channel count of the GRANITE 9/10. This communications connection satisfies the tight timing requirements imposed on independent GRANITE Measurement Modules that are working together as part of a measurement and control system.

The underlying communications of EPI are built using TCP/IP. More specifically, the IEEE 1588 protocol is implemented at the lowest hardware levels for synchronization of device clocks across the entire network. This accomplishes tighter device synchronization and 100-times the data

throughput of CPI. The additional power, cost, and complexity are warranted for fast sampling applications.

2.4.8 CAN port (GRANITE 10 only)

The CAN (Controller Area Network) physical layer is a differential signal that is generally a twisted pair. The GRANITE 10 has 4 general purpose CAN ports, CAN 2.0 up to 1 Mbps, or CAN FD up to 5 Mbps. They can be accessed either by the screw terminals, or the 15-pin connector; they are electrically connected. Wire the differential signal (H and L) into the screw terminals, or use a custom cable and the 15-pin connector.

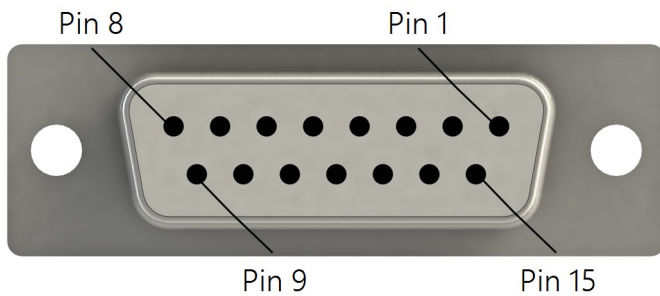


Table 2-7: CAN bus pinout

Pin Number	Description
1	CAN1 H
2	CAN1 L
3	RG1
4	CAN2 H
5	CAN2 L
6	RG2
7	CAN3 H
8	CAN3 L
9	RG3
10	CAN4 H
11	CAN4 L
12	RG4
13	Not used

Pin Number	Description
14	Not used
15	Not used

2.5 Programmable logic control

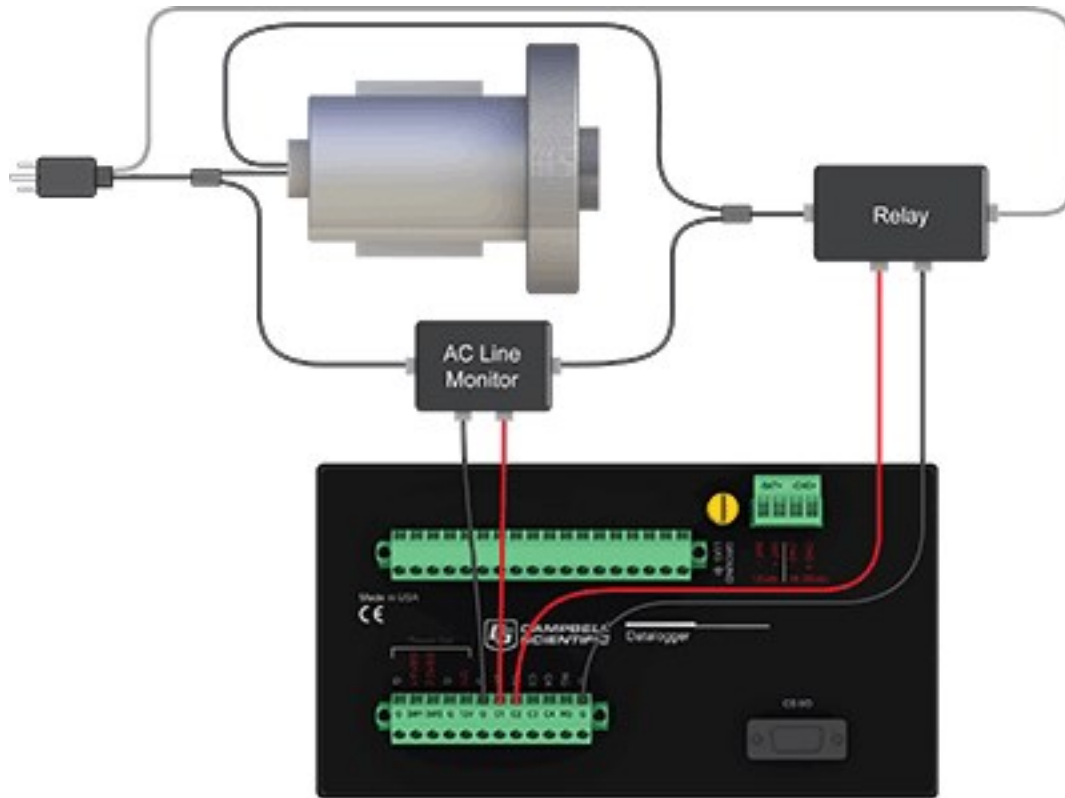
The data logger can control instruments and devices such as:

- 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**, or **SW12** terminals. *Short Cut* has provisions for simple on/off control. Control modules and relay drivers are available to expand and augment data logger control capacity.

- **C** terminals are selectable as binary inputs, control outputs, or communication ports. These terminals can be set low (0 VDC) or high (5 VDC) using the `PortSet()` or `WriteIO()` instructions. See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>. 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** 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 (see the *CRBasic* help for more information).

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



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 data logger 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 GRANITE 9/10

The basic steps for setting up your data logger to take measurements and store data are included in the following sections:

3.1 Setting up communications with the data logger	20
3.2 Testing communications with EZSetup	30
3.3 Making the software connection	31
3.4 Programming quickstart using Short Cut	31
3.5 Sending a program to the data logger	35

3.1 Setting up communications with the data logger

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

The default settings for the data logger 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 data logger support software.


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


3.1.1 USB or RS-232 communications	21
3.1.2 Virtual Ethernet over USB (RNDIS)	22
3.1.3 Ethernet communications option	23
3.1.4 Wi-Fi communications	26



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 data logger networks, use Network Planner. For more information on using the

Network Planner, watch a video at <https://www.campbellsci.eu/videos/loggernet-software-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 data logger. Because these connections do not require configuration (like an IP address), you need only set up the communications between your computer and the data logger. Use the following instructions or watch the Quickstart videos at <https://www.campbellsci.eu/videos> .


Follow these steps to get started. These settings can be revisited using the data logger support software **Edit Datalogger Setup** option .

1. Using data logger support software, launch the EZSetup Wizard.
 - **LoggerNet** users, click **Setup** , click the **View** menu to ensure you are in the **EZ (Simplified)** view, then click **Add Datalogger**.
 - RTDAQ users, click **Add Datalogger** .
2. Click **Next**.
3. Select your data logger from the list, type a name for your data logger (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 GRANITE 9/10 via USB, click **Install USB Driver**, select your data logger, click **Install**, and follow the prompts to install the USB drivers.
6. Plug the data logger into your computer using a USB or RS-232 cable and provide 12V power. If using RS-232, a CPI/RS-232 RJ45 to DB9 cable is required to connect to the computer.
7. From the **COM Port** list, select the COM port used for your data logger.
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 data logger. The default PakBus address is **1**. A USB connection does not require a baud rate selection. RS-232 connections default to 115200 baud.
10. Set an **Extra Response Time** if you have a difficult or marginal connection and you want the data logger support software to wait a certain amount of time before returning a communication failure error.

11. **LoggerNet** users can set a **Max Time On-Line** to limit the amount of time the data logger remains connected. When the data logger 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 data logger.
12. Click **Next**.
13. By default, the data logger 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 [Data logger security](#) (p. 101) for more information.
14. Click **Next**.
15. Review the **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 data logger clock, and send a program to the data logger. See [Testing communications with EZSetup](#) (p. 30) for more information.

3.1.2 Virtual Ethernet over USB (RNDIS)

GRANITE 9/10 data loggers support RNDIS (virtual Ethernet over USB). This allows the data logger to communicate via TCP/IP over USB. Watch a video  <https://www.campbellsci.eu/videos/ethernet-over-usb> or use the following instructions.



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

NOTE:


Ensure the data logger 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 data logger to your computer using a USB cable, then in *Device Configuration Utility* select your data logger.
3. Retrieve your IP Address. On the bottom, left side of the screen, select **IP** as the Connection Type, then click the browse button next to the **Server Address** box. Note the IP Address (default is **192.168.66.1**). If you have multiple data loggers in your network, more than one data logger may be returned. Ensure you select the correct data logger by verifying the data logger serial number or station name (if assigned).
4. A virtual IP address can be used to connect to the data logger using *Device Configuration Utility* or other computer software, or to view the data logger internal web page in a browser. To view the web page, open a browser and enter linktodevice.eu or the IP address you retrieved in the previous step (for example, **192.168.66.1**) into the address bar. To secure your data logger from others who have access to your network, we recommend that you set security. For more information, see [Data logger security](#) (p. 101).

NOTE:


Ethernet over USB (RNDIS) is considered a direct communications connection. Therefore, it is a trusted connection and `csipasswd` does not apply.

3.1.3 Ethernet communications option

The GRANITE 9/10 offers a 10/100 Ethernet connection. Use *Device Configuration Utility* to enter the data logger IP Address, Subnet Mask, and IP Gateway address. After this, use the EZSetup Wizard to set up communications with the data logger. If you already have the data logger IP information, you can skip these steps and go directly to [Setting up Ethernet communications between the data logger and computer](#) (p. 25). Watch a video  <https://www.campbellsci.eu/videos/datalogger-ethernet-configuration> or use the following instructions.

3.1.3.1 Configuring data logger Ethernet settings

1. Supply power to the data logger. If connecting via USB for the first time, you must first install USB drivers by using *Device Configuration Utility* (select your data logger, then on the main page, click **Install USB Driver**). Alternately, you can install the USB drivers using EZ Setup.
2. Connect an Ethernet cable to the **10/100 Ethernet** port on the data logger. 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. 24).


3. Using data logger support software (*LoggerNet* or RTDAQ), open *Device Configuration Utility* .
4. Select the **GRANITE 9/10** data logger from the list
5. Select the port assigned to the data logger from the **Communication Port** list. If connecting via Ethernet, select **Use IP Connection**.
6. By default, this data logger 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 [Data logger security](#) (p. 101) 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 data logger. If there is no Ethernet connection, the data logger 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 data logger 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 data logger 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 data logger 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 data logger and computer

Once you have configured the Ethernet settings or obtained the IP information for your data logger, you can set up communications between your computer and the data logger over Ethernet. Watch a video  <https://www.campbellsci.eu/videos/ezsetup-ethernet-connection> or use the following instructions.

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


1. Using data logger support software, open **EZSetup**.
 - **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 **Add Datalogger**.
 - RTDAQ users, click **Add Datalogger** .
2. Click **Next**.
3. Select the **GRANITE 9/10** from the list, enter a name for your station (for example, a site or project name), **Next**.
4. Select the **IP Port** connection type and click **Next**.
5. Type the data logger IP address followed by a colon, then the port number of the data logger in the **Internet IP Address** box (these were set up through the [Ethernet communications option](#) (p. 23)) 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 data logger. The default PakBus address is **1**.
 - Set an **Extra Response Time** if you want the data logger support software to wait a certain amount of time before returning a communications failure error.
 - **LoggerNet** users can set a **Max Time On-Line** to limit the amount of time the data logger remains connected. When the data logger 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 data logger. **Next**.
7. By default, the data logger 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 [Data logger security](#) (p. 101). **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 data logger clock, and sending a program to the data logger. See [Testing communications with EZSetup](#) (p. 30) for more information.

3.1.4 Wi-Fi communications

By default, the GRANITE 9/10 is configured to host a Wi-Fi network. The LoggerLink mobile app for iOS and Android can be used to connect with a GRANITE 9/10. Up to eight devices can connect to a network created by a GRANITE 9/10. 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. 167).

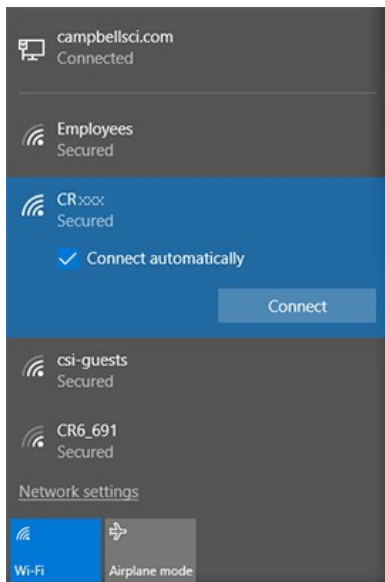
3.1.4.1 Configuring the data logger to host a Wi-Fi network

By default, the GRANITE 9/10 is configured to host a Wi-Fi network. If the settings have changed, you can follow these instructions to reconfigure it.

1. Ensure your GRANITE 9/10 is connected to an antenna and power.
2. Using *Device Configuration Utility*, connect to the data logger.
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 data logger over Wi-Fi

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



2. Select the Wi-Fi-network hosted by the data logger. The default name is **GRANITE 9/10** followed by the serial number of the data logger. 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 data logger and the data logger support software

1. Using **LoggerNet** 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** .
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 data logger IP address created when the GRANITE 9/10 creates a network.
4. Click **Next**.
5. The PakBus address must match the hardware settings for your data logger. The default PakBus address is **1**.

- Set an **Extra Response Time** if you want the data logger 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 data logger remains connected. When the data logger 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 data logger.
6. Click **Next**.
 7. By default, the data logger 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 [Data logger security](#) (p. 101) 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 data logger clock, and send a program to the data logger. See [Testing communications with EZSetup](#) (p. 30) for more information.

3.1.4.4 Configuring data loggers to join a Wi-Fi network

By default, the GRANITE 9/10 is configured to host a Wi-Fi network. To set it up to join a network:

1. Ensure your GRANITE 9/10 is connected to an antenna and power.
2. Using *Device Configuration Utility*, connect to the data logger.
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. To join a hidden network, manually enter its SSID.
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 data logger 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 data logger was not able to obtain an address from the DHCP server. Contact your network administrator for help.

8. Apply your changes.
9. For each data logger you want to connect to network, you must follow the instruction in [Setting up Wi-Fi communications between the data logger and the data logger support software](#) (p. 27), using the IP address used to configure that data logger (step 7 in this instruction).

3.1.4.5 Wi-Fi mode button

Configure the Wi-Fi mode button using *Device Configuration Utility* software.

Disable button - When this configuration is selected, pressing the button will have no effect on the operation of the device. The Wi-Fi network will continue to work as configured.

Temporarily enable Wi-Fi - When this configuration is selected, the normally disabled Wi-Fi network will be activated temporarily when the button is pressed.

Temporarily create a network - When this configuration is selected, the device will temporarily create a network when the button is pressed. If the Wi-Fi configuration is set to "**Join a Network**" then the temporarily created network will be an open network with the name GRANITE 9/10_ [Serial Number]. If the Wi-Fi configuration is set to "**Create a Network**" then the configured Wi-Fi network will normally be disabled, and it will be activated temporarily when the button is pressed.

NOTE:

When the Wi-Fi configuration is set to "**Create a Network**" the device behaviour is the same for both button configurations.

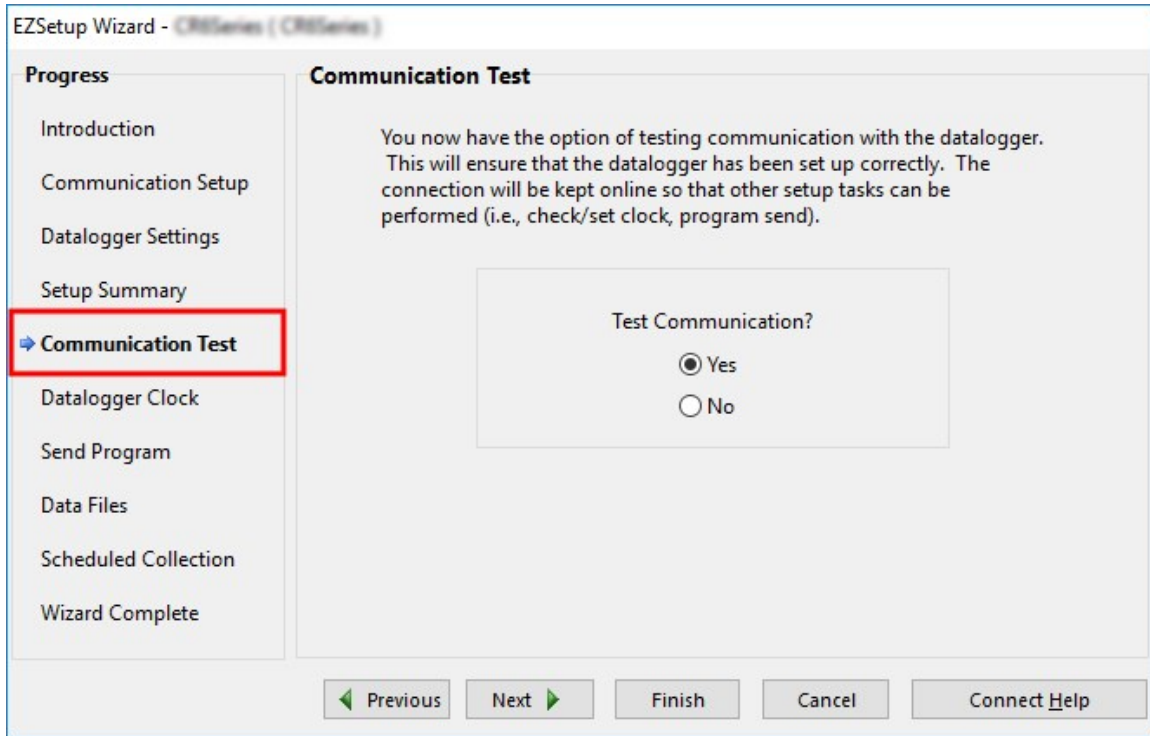
3.1.4.6 Wi-Fi LED indicator

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

- **Off:** Insufficient power, Wi-Fi disabled, or data logger 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.


3.2 Testing communications with EZSetup

1. Using data logger support software EZ Setup, access the **Communication Test** window. This window is accessed during EZ Setup (see [USB or RS-232 communications](#) (p. 21) for more information). Alternatively, you can double-click a data logger 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 data logger 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 data logger and the computer.
 - The **Adjusted Server Date/Time** displays the current reading of the clock for the computer or server running your data logger support software. If the **Datalogger Date/Time** and **Adjusted Server Date/Time** don't match, you can set the data logger 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 data logger clock. This offset will allow you to set

the clock for a data logger 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 data logger ships with a default **GettingStarted** program. If the data logger does not have a program, you can choose to send one by clicking **Select and Send Program**. Click **Next**.
6. *LoggerNet* only - Use the following instructions or watch the [Scheduled/Automatic Data Collection video](#) 
 - The **Datalogger Table Output Files** window displays the data tables available to be collected from the data logger and the output file name. By default, all data tables set up in the data logger 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 data logger on the **Collection Interval** entered. When the **Base Date** and **Time** are in the past, scheduled collection will begin immediately after finishing the EZSetup wizard. Click **Next** twice.
7. Click **Finish**.

3.3 Making the software connection

Once you have configured your hardware connection (see [Setting up communications with the data logger](#) (p. 20), your data logger and computer can communicate. You'll use the **Connect** screen to send a program, set the clock, view real-time data, and manually collect data.

- *LoggerNet* users, select **Main** and **Connect**  on the *LoggerNet* toolbar, select the data logger from the **Stations** list, then **Connect** .
- RTDAQ users, select the data logger from the list and click **Connect** .

To disconnect, click **Disconnect** .

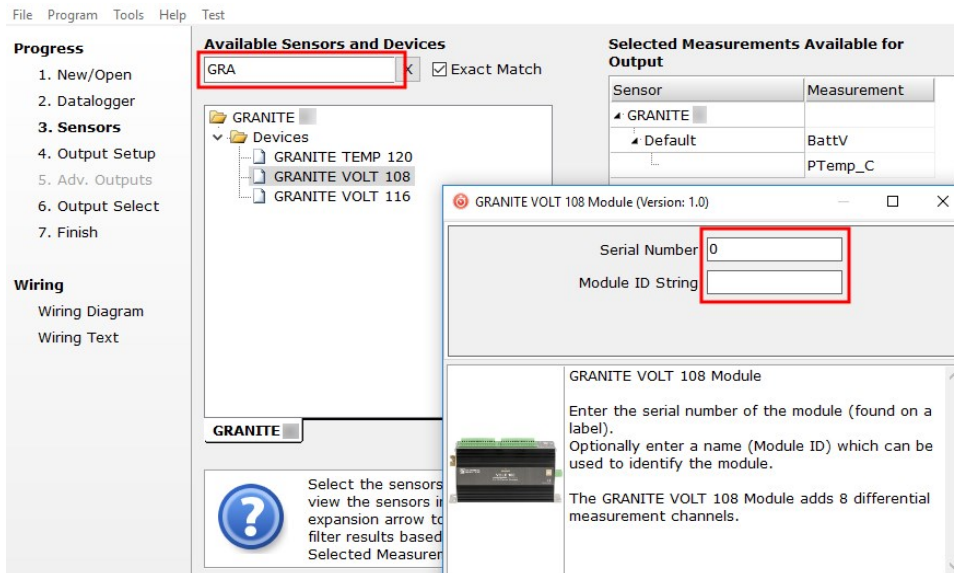
For more information see the [Connect Window Tutorial](#) .

3.4 Programming quickstart using Short Cut

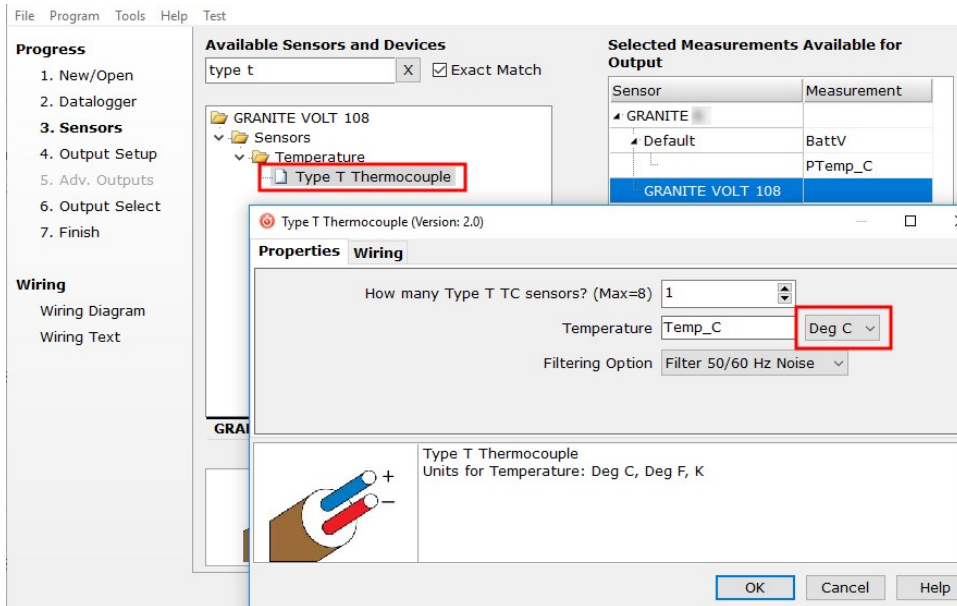
Short Cut is an easy way to program the GRANITE data acquisition system to measure a sensor and assign wiring terminals. Short Cut is available as a download from <https://www.campbellsci.eu/shortcut>. It is also included in installations of LoggerNet and RTDAQ.

The following procedure shows using Short Cut to program the data logger to measure a type-T thermocouple on a VOLT 100 series.

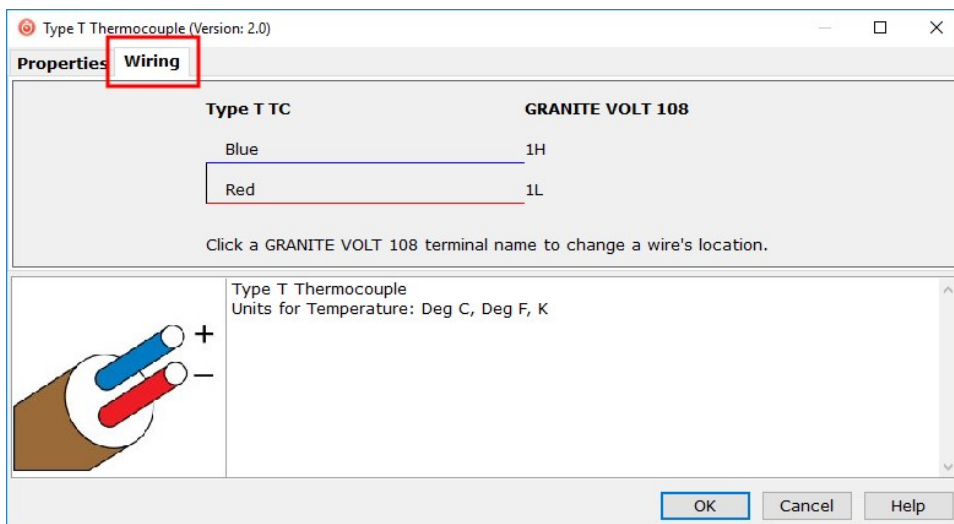
1. Open Short Cut and click **Create New Program**.
2. Double-click your **GRANITE** data logger.
3. In the **Available Sensors and Devices** box, start typing GRANITE. You can also locate it in the **Devices** folder. Double click the GRANITE Measurement Module you are working with. Type the serial number located on the VOLT 100 series label. Optionally, type a name in the **Module ID String** if you want the module to have an ID. This is useful when multiple VOLT 100 series modules are connected to the GRANITE data logger.



4. In the **Available Sensors and Devices** box, type Type T. You can also locate the thermocouple in the **Sensors > Temperature** folder. Double click **Type T Thermocouple**. Type the number of type T thermocouples connected to the VOLT 100 series. The temperature defaults to degree C. This can be changed by clicking the **Temperature** units box and selecting one of the other options.

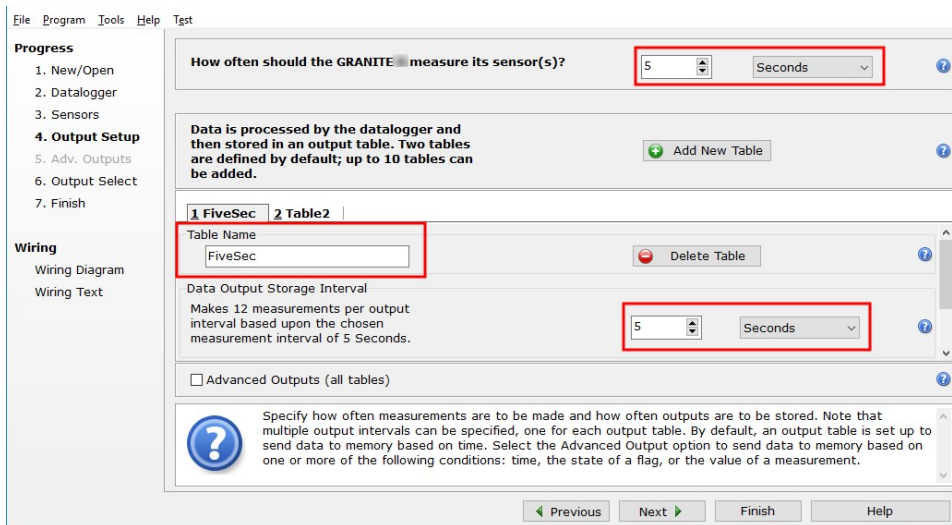


5. Click on the **Wiring** tab to see how the sensor is to be wired to the VOLT 100 series. Click **OK** after wiring the thermocouple.

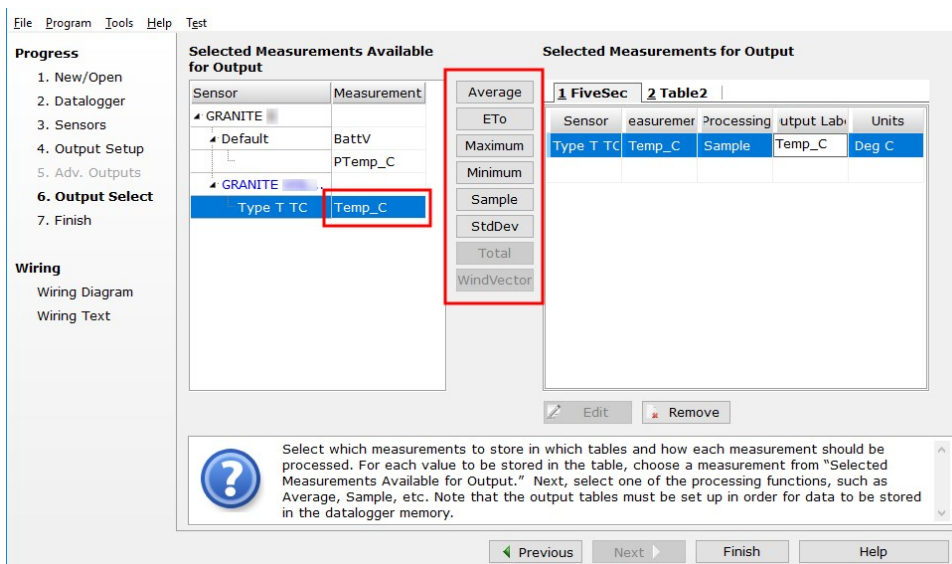


6. Repeat steps four and five for other sensors you want to measure. Click **Next**.

- In **Output Setup**, type the scan rate, a meaningful table name, and the **Data Output Storage Interval**.



- Select the measurement and its associated output option.



- Click **Finish** and save the program. Send the program just created to the GRANITE data logger if it is connected to the computer.
- If the thermocouple is connected to the VOLT 100 series, check the output of the thermocouple in the data logger support software data display in LoggerNet or RTDAQ to make sure it is making reasonable measurements.

3.5 Sending a program to the data logger

TIP:

It is good practice is to always retrieve data from the data logger before sending a program; otherwise, data may be lost. See [Collecting data](#) (p. 37) for detailed instruction.

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 the [Quickstart part 3 video](#) .

1. Connect the data logger to your computer (see [Making the software connection](#) (p. 31) for more information).
2. Using your data logger 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 program, select it, and click **Open**. For example: navigate to **C:\Campbellsci\SCWin** and select **MyTemperature.CRB**.
4. Confirm that you would like to proceed and erase all data tables saved on the data logger. The program will send and compile.
5. Review the **Compile Results** window for errors, messages and warnings.
6. Click **Details**, select the **Table Fill Times** tab. Ensure that the times shown are expected for your application. Click **OK**.

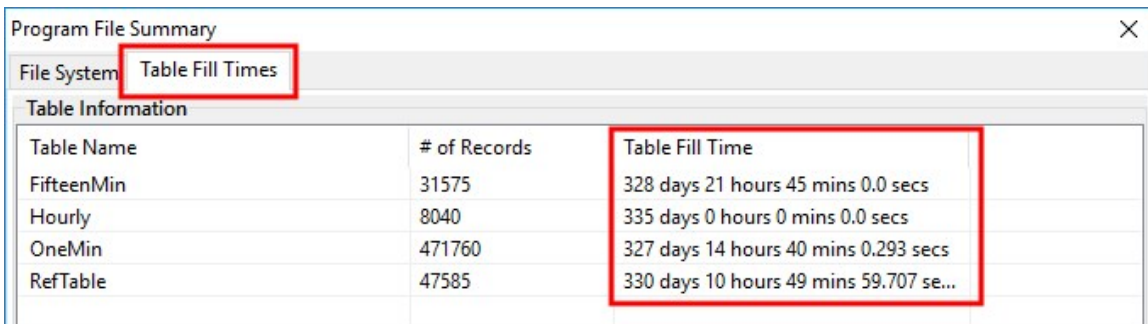


Table Name	# of Records	Table Fill Time
FifteenMin	31575	328 days 21 hours 45 mins 0.0 secs
Hourly	8040	335 days 0 hours 0 mins 0.0 secs
OneMin	471760	327 days 14 hours 40 mins 0.293 secs
RefTable	47585	330 days 10 hours 49 mins 59.707 se...

After sending a program, it is a good idea to monitor the Public Table to make sure sensors are taking good measurements. See [Working with data](#) (p. 36) for more information.

4. Working with data



4.1 Default data tables

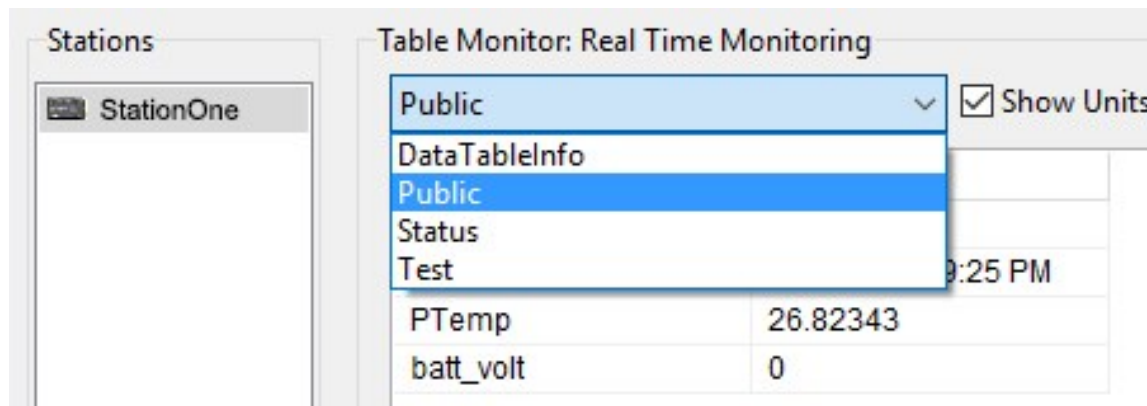
By default, the data logger includes three tables: **Public**, **Status**, and **DataTableInfo**. Each of these tables only contains the most recent measurements and information.


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

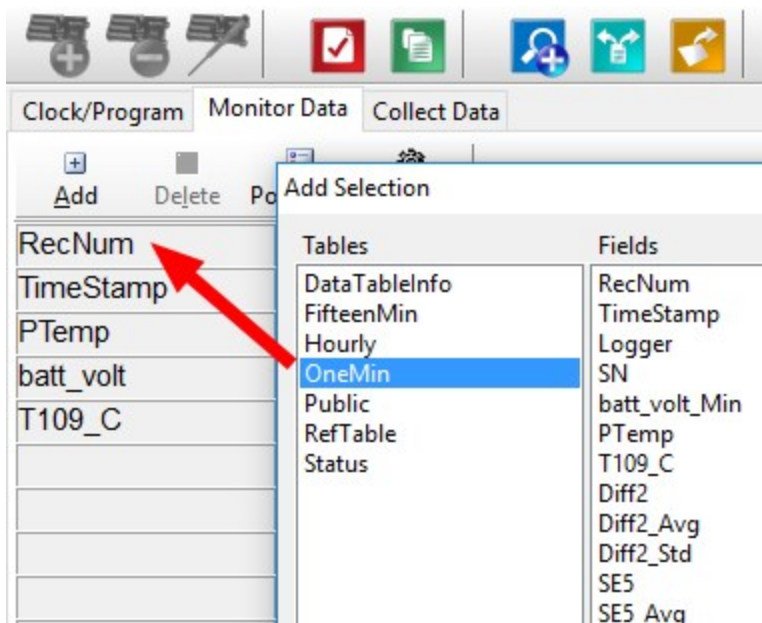
For information on collecting your data, see [Collecting data](#) (p. 37).

Use these instructions or follow the [Connect Window tutorial](#)  to monitor real-time data.


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






RTDAQ users, click **Connect** , then **Monitor Data**. When this tab is first opened for a data logger, 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. Click **Start** to begin monitoring data values. The name of this button then changes to **Stop**, and it can be clicked to stop monitoring.




4.2 Collecting data

The data logger writes to data tables based on intervals and conditions set in the CRBasic program (see [Creating data tables in a program](#) (p. 46) for more information). After the program has been running for enough time to generate data records, data may be collected by using data logger support software. During data collection, data is copied to the computer and still remains on the data logger. Collections may be done manually, or automatically through scheduled collections set in *LoggerNet Setup*. Use these instructions or follow the [Collect Data Tutorial](#) .

4.2.1 Collecting data using *LoggerNet*

1. From the *LoggerNet* toolbar, click **Main** and **Connect** , select the data logger from the **Stations** list, then **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. 39)

4.2.2 Collecting data using RTDAQ

1. Click **Connect**  on the main RTDAQ window.
2. Go to the **Collect Data** tab.

3. Leave the default **Collection Options**:
 - **Collect Mode** should be **Data Since Last Collection**
 - **File Mode** should be **Append to End of File**.
 - **File Format** should be **ASCII Table Data (TOA5)**
4. Select the check boxes for tables in the **Table Collection** list to indicate which tables will be collected.
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. 39)

4.2.3 Collecting data using an FTP client

FTP is an efficient method to retrieve large files from the SSD. Data files created by [TableFileO](#) can be up to 2GB.


1. [Ethernet communications option](#) (p. 23)
2. Once connected, go to the **Deployment > Network Services** tab.
3. Select the check box for **FTP Enabled**.
4. Enter an **FTP User Name** and **FTP Password**, and **Confirm FTP Password**.

The screenshot shows the 'Network Services' configuration window. The 'Deployment' tab is selected at the top. Within the 'Network Services' sub-tab, the 'FTP Enabled' checkbox is checked and highlighted with a red box. Below it, the 'FTP User Name', 'FTP Password', and 'Confirm FTP Password' fields are visible, with the password fields masked with dots. Other options like 'HTTP Enabled', 'HTTPS Enabled', 'Telnet Enabled', and 'Ping (ICMP) Enabled' are also present, along with a 'PakBus/TCP Port' dropdown set to 6785.




5. Write down the data logger IP address. Find it on the **Deployment > Ethernet** tab.

6. Use an FTP Client, such as FileZilla, to copy the files to your computer. You will need the data logger IP address, FTP User Name and FTP Password.
7. Use **View Pro** to view the data, see [Viewing historic data](#) (p. 39)

4.3 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 the [View Data Tutorial](#) .

Once the data logger has had enough time to store multiple records, you should collect and review the data.

1. To view the most recent data, connect the data logger to your computer and collect your data (see [Collecting data](#) (p. 37) for more information).
2. Open View Pro:
 - **LoggerNet** users click **Data** then **View Pro**  on the **LoggerNet** toolbar.
 - RTDAQ 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 data logger software application]`). For example: navigate to the `C:\Campbellsci\LoggerNet` folder and select **OneMin.dat**.

4.4 Data types and formats

Data takes different formats as it is created and manipulated in the data logger, as it is displayed through software, and as it is retrieved to a computer file. It is important to understand the different data types, formats and ranges, and where they are used.

Data type	Description	Range	Resolution	Where used
Float	IEEE four-byte floating point	+/-1.8 *10 ⁻³⁸ to +/-3.4 *10 ³⁸	24 bits (about 7 digits)	variables
Long	four-byte signed integer	-2,147,483,648 to +2,147,483,647	1 bit	variables, output
Boolean	four-byte signed integer	-1, 0	True (-1) or False (0)	variables, sample output
String	ASCII String			variables, sample output

Table 4-1: Data types, ranges and resolutions

Data type	Description	Range	Resolution	Where used
IEEE4	IEEE four-byte floating point	+/-1.8 *10 ⁻³⁸ to +/-3.4 *10 ³⁸	24 bits (about 7 digits)	internal calculations, output
IEEE8	IEEE eight-byte floating point	+/-2.23 *10 ⁻³⁰⁸ to +/-1.8 *10 ³⁰⁸	53 bits (about 16 digits)	internal calculations, output
FP2	Campbell Scientific two-byte floating point	-7999 to +7999	13 bits (about 4 digits)	output
NSEC	eight-byte time stamp		nanoseconds	variables, output

4.4.1 Variables

In CRBasic, the declaration of variables (via the **DIM** or the **PUBLIC** statement) allows an optional type descriptor **As** that specifies the data type. The data types are **Float**, **Long**, **Boolean**, and **String**. The default type is **Float**.

Example variables declared with optional data types

```
Public PTemp As Float, Batt_volt
Public Counter As Long
Public SiteName As String * 24
```

As Float specifies the default data type. If no data type is explicitly specified with the **As** statement, then **Float** is assumed. Measurement variables are stored and calculations are performed internally in IEEE 4 byte floating point with some operations calculated in double precision. A good rule of thumb is that resolution will be better than 1 in the seventh digit.

As Long specifies the variable as a 32 bit integer. There are two possible reasons a user would do this: (1) speed, since the GRANITE 9/10 Operating System can do math on integers faster than with **Floats**, and (2) resolution, since the **Long** has 31 bits compared to the 24 bits in the **Float**. A good application of the **As Long** declaration is a counter that is expected to get very large.

As Boolean specifies the variable as a 4 byte Boolean. Boolean variables are typically used for flags and to represent conditions or hardware that have only 2 states (e.g., On/Off, High/Low). A Boolean variable uses the same 32 bit long integer format as a **Long** but can set to only one of two values: True, which is represented as -1, and false, which is represented with 0. When a **Float** or **Long** integer is converted to a **Boolean**, zero is False (0), any non-zero value will set the Boolean to True (-1). The Boolean data type allows application software to display it as an On/Off, True/False, Red/Blue, etc.

The GRANITE 9/10 uses `-1` rather than some other non-zero number because the **AND** and **OR** operators are the same for logical statements and binary bitwise comparisons. The number `-1` is expressed in binary with all bits equal to 1, the number `0` has all bits equal to 0. When `-1` is anded with any other number the result is the other number, ensuring that if the other number is non-zero (true), the result will be non-zero.

As String * size specifies the variable as a string of ASCII characters, NULL terminated, with an optional size specifying the maximum number of characters in the string. A string is convenient in handling serial sensors, dial strings, text messages, etc. When size is not specified, a default of 24 characters will be used (23 usable bytes and 1 terminating byte).

As a special case, a string can be declared **As String * 1**. This allows the efficient storage of a single character. The string will take up 4 bytes in memory and when stored in a data table, but it will hold only one character.

Structures (**StructureType/EndStructureType**) are an advanced technique used to organize variables and display data in a structured manner. They can significantly shorten program code, especially for instructions that output an array of values, such as **AW2000**, **GPSO**, and **SDI12Recorder()**. For example, a single **StructureType** may be used to organize and display data for multiple vibrating wire sensors or many SDI-12 sensors without creating aliases for each sensor. See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

4.4.2 Constants

The **Const** declaration is used to assign a name that can be used in place of a value in the data logger CRBasic program. Once a value is assigned to a constant, each time the value is needed in the program, the programmer can type in the constant name instead of the value itself. The use of the **Const** declaration can make the program easier to follow, easier to modify, and more secure against unintended changes. Unlike variables, constants cannot be changed while the program is running.

Constants must be defined before they are used in the program. Constants can be defined in a **ConstTable/EndConstTable** construct allowing them to be changed using the keyboard display, the **C** command in terminal mode, or via a custom menu.

Constants can also be typed. For example: **Const** A as Long = 9999, and **Const** B as String = "MyString". Valid data types for constants are: **Long**, **Float**, **Double**, and **String**. Other data types return a compile error.

When the CRBasic program compiles, the compiler determines the type of the constant (**Long**, **Float**, **Double**, or **String**) from the expression. This data type is communicated to the

software. The software formats or restricts the input based on the data type communicated to it by the data logger.

You can declare a constant with or without specifying a data type. If a data type is not specified, the compiler determines the data type from the expression. For example: `Const A = 9999` will use the `Long` data type. `Const A = 9999.0` will use the `Float` data type.

4.4.3 Data storage

Data can be stored in either IEEE4 or FP2 formats. The format is selected in the program instruction that outputs the data, i.e. minimum, maximum, etc.

While `Float` (IEEE 4 byte floating point) is used for variables and internal calculations, `FP2` is adequate for most stored data. Campbell Scientific 2 byte floating point (`FP2`) provides 3 or 4 significant digits of resolution, and requires half the memory space as `IEEE4` (2 bytes per value vs 4).

Zero	Minimum magnitude	Maximum Magnitude
0.000	±0.001	±7999.

The resolution of `FP2` is reduced to 3 significant digits when the first (left most) digit is 8 or greater. Thus, it may be necessary to use `IEEE4` output or an offset to maintain the desired resolution of a measurement. For example, if water level is to be measured and output to the nearest 0.01 foot, the level must be less than 80 feet for `FP2` output to display the 0.01 foot increment. If the water level is expected to range from 50 to 90 feet the data could either be output in `IEEE4` or could be offset by 20 feet (transforming the range to 30 to 70 feet).

Absolute value	Decimal location
0 – 7.999	X.XXX
8 – 79.99	XX.XX
80 – 799.9	XXX.X
800 – 7999.	XXXX.

NOTE:

`String` and `Boolean` variables can be output with the `Sample()` instruction. Results of Sampling a `Boolean` variable will be either -1 or 0 in the collected Data Table. A Boolean displays in the `Numeric Monitor` Public and Data Tables as `true` or `false`.

4.5 About data tables

A data table is essentially a file that resides in data logger memory (for information on data table storage, see [Data memory](#) (p. 48)). 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 data logger support software (see [Collecting data](#) (p. 37)).


Table 4-4: Example data				
TOA5, MyStation, GRANITE 9/10, 1142, GRANITE 9/10.Std.01, CPU:MyTemperature..CRB, 1958, OneMin				
TIMESTAMP	RECORD	BattV_Avg	PTemp_C_Avg	Temp_C_Avg
TS	RN	Volts	Deg C	Deg C
		Avg	Avg	Avg
2019-03-08 14:24:00	0	13.68	21.84	20.71
2019-03-08 14:25:00	1	13.65	21.84	20.63
2019-03-08 14:26:00	2	13.66	21.84	20.63
2019-03-08 14:27:00	3	13.58	21.85	20.62
2019-03-08 14:28:00	4	13.64	21.85	20.52
2019-03-08 14:29:00	5	13.65	21.85	20.64

4.5.1 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 data logger 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.

4.5.1.1 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.
- **GRANITE 9/10** - Data logger model.
- **1142** - Data logger serial number.
- **GRANITE 9/10.Std.01** - Data logger OS version.
- **CPU:MyTemperature.CRB** - Data logger program name. Changed by sending a new program (see [Sending a program to the data logger](#) (p. 35) for more information).
- **1958** - Data logger program signature. Changed by revising a program or sending a new program (see [Sending a program to the data logger](#) (p. 35) for more information).
- **OneMin** - Table name as declared in the running program (see [Creating data tables in a program](#) (p. 46) 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. 45).

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 data logger 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. 43).

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 data logger 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.

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

4.5.1.2 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. 43), **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 2019-03-08 00:00:00 will contain the total rainfall beginning at 2019-03-07 00:00:01 and ending at 2019-03-08 00:00:00.

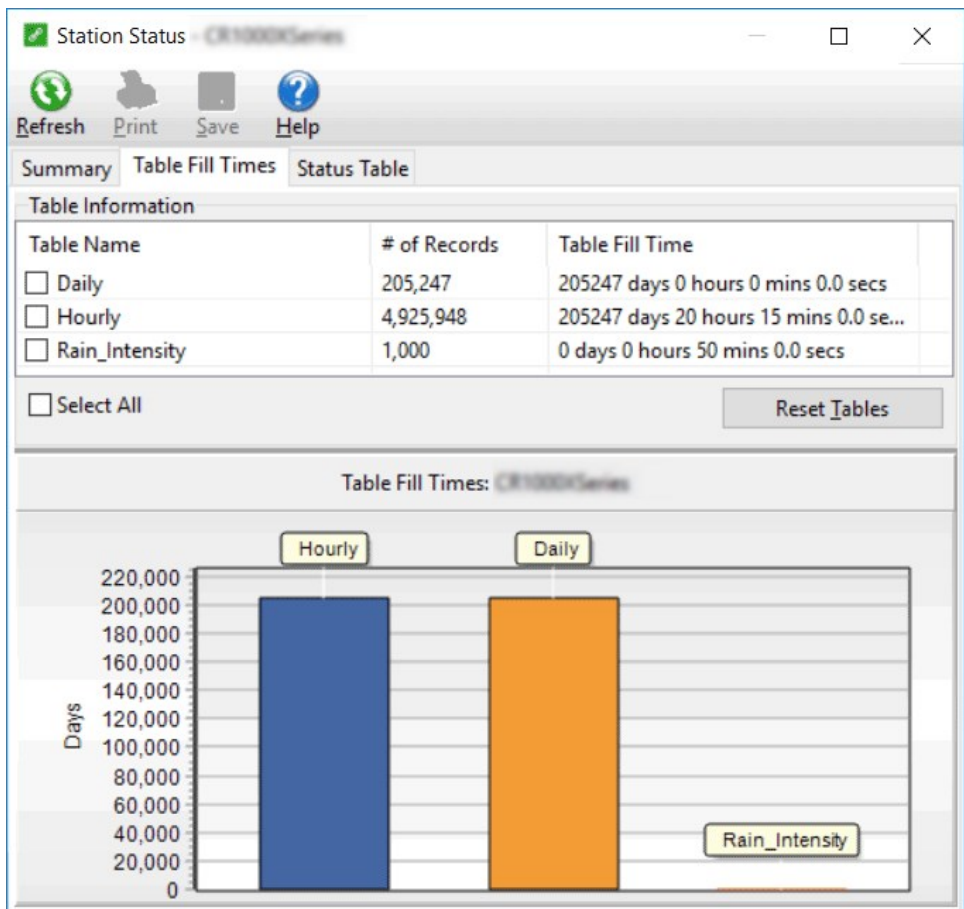
4.6 Creating data tables in a program

Data is stored in tables as directed by the CRBasic program. In *Short Cut*, data tables are created in the **Output** steps (see [Programming quickstart using Short Cut](#) (p. 31)). Data tables are created within the CRBasic data logger 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
```

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

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 data logger to auto-allocate table size. With auto-allocation, the data logger 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 [Data memory](#) (p. 48).



5. Data memory

The data logger includes five types of memory: DDR-SDRAM, SRAM, eMMC NAND Flash, NOR Flash, and a solid-state hard drive (SSD). A memory card slot is also available for an optional microSD card. The USB host port supports mass storage devices (MSD) such as USB flash or thumb drives.

5.1 Data tables

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

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 data logger uses those parameters to assign sectors in a way that maximizes the life of its memory. See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

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 data logger support software.

Data concerning the data logger memory are posted in the **Status** and **DataTableInfo** tables. For additional information on these tables, see [Information tables and settings \(advanced\)](#) (p. 136).

For additional information on data logger memory, visit the Campbell Scientific blog article, "[How to Know when Your Datalogger Memory is Getting Full.](#)"

5.2 Memory allocation

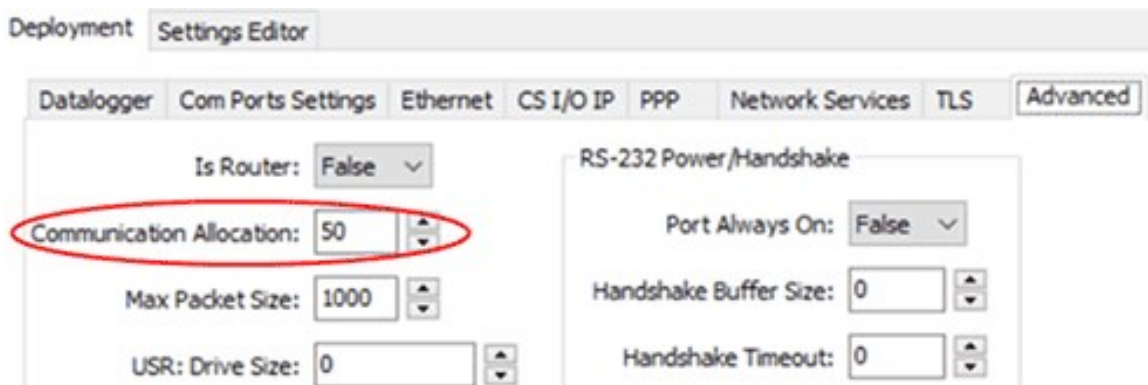
Some data storage tables are allocated in the DDR-SDRAM. Due to its volatile nature, all data storage done on the DDR-SDRAM, must be duplicated on the non-volatile solid-state hard drive. All tables that are designated as auto-allocate, use this DDR-SDRAM/SSD combination for data storage. Tables that specify a fixed number of records are allocated from the SRAM.

5.3 SRAM

SRAM holds communications buffers, some final-data memory, and, if allocated, the USB drive. Data tables that specify a fixed number of records are allocated from SRAM. An internal lithium battery retains this memory when primary power is removed.

The structure of the data logger 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 neighbour 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.
- **Final-Data Memory:** This memory stores some data. Data tables with a fixed number of records 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.



- **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**.

5.3.1 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 data logger 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 data logger 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 2990080 bytes.

WARNING:

Partitioning or changing the size of the USR drive will delete stored data from tables. Collect data first.

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 data logger support software **File Control** or through the **FileManage()** instruction in CRBasic program.

5.4 DDR-SDRAM

This high-speed memory is volatile (it is not maintained through a power cycle or recompile) and has several functions. First, the operating system is copied from the eMMC NAND flash memory into the DDR-SDRAM where code is executed. This memory is also used for compiling the CRBasic program, storing variables and constants, intermediate data processing and running instructions. Some data storage tables are allocated in the DDR-SDRAM as well.

5.5 SRAM vs. DDR-SDRAM

Battery-backed SRAM is used for data tables that are not designated as auto-allocated. Alternatively, because it is very fast and has a large capacity, DDR-SDRAM is well suited to storing high volumes of data at sustained and higher rates. The combination of buffering in the DDR-SDRAM and copying into the SSD is very powerful for high speed data.

However, there is a window of vulnerability with this DDR-SDRAM/SSD combination. Data resides on volatile media until it can be transferred to the non-volatile hard drive. If a power fault or some other failure occurs, all data that is in the DDR-SDRAM prior to being written to the SSD, will be lost. The slower, less data efficient, battery backed SRAM does not suffer from this window of vulnerability, a power loss does not result in compromised data. The ideal use of these different memories is to store slower and/or smaller data sets in the battery backed SRAM and the high speed, high volume data in the DDR-SDRAM and SSD combination.

5.6 SSD - Hard Drive

This non-volatile memory shows up as SSD (Solid State Drive): in the data logger file system. Data tables that are auto-allocated are stored in this memory. It can also be used for all file related operations such as `TableFile()`, and image storage. This drive with its very fast read/write speeds is the preferred location for storing large data sets.

NOTE:

Even though the SSD has 64 GB (or 128 GB for XD) of available memory, individual file size is limited to 2 GB.

Use `Tablefile()` when more than 2 GB of memory is needed for a given data table. See the *CRBasic Editor* help for detailed instruction information and program examples:

<https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

An alternative method to fill the SSD is to use multiple auto-allocated data tables.

TIP:

Use File Control to manually delete files from the SSD.

5.7 Flash memory

5.7.1 eMMC NAND flash memory

The data logger operating system is stored in a separate section of flash memory. To update the operating system, see [Updating the operating system](#) (p. 111).

5.7.2 NOR flash memory

Serial flash memory holds the CPU drive, web page, and data logger 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.

5.7.3 CPU drive

The serial flash memory CPU drive contains data logger 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. 160) for information on data logger memory.

5.8 MicroSD (CRD: drive)

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

Use [CardOut\(\)](#) to create a new DataTable saved on a card. When storing high-frequency data, or when storing data to cards greater than 2 GB, [TableFile\(\)](#) with **Option 64** may be a better method to write final storage data to a card.

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

The CRD: drive uses microSD cards exclusively. 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 data logger operating temperature range.
- Provide faster read/write times.
- Include vibration and shock resistance.
- Have longer life spans (more read/write cycles).

A "card controller error" indicates that the data logger has failed to communicate with the card. It is an error caused by the micro-controller built into the microSD card. If the error repeats itself, try an industrial-grade card.

A maximum of 30 data tables can be created using `CardOut()` 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. 134).

When a new program is compiled that sends data to the card, the data logger 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 data logger 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](#) (p. 114) for more information).

5.8.1 Formatting microSD cards

The data logger 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 data logger formats memory cards as FAT32.

Because of the way the FAT32 card format works, you can avoid long data logger 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 data

logger program to compile faster. This only needs to be done once when the card is formatted. If you have the data logger update the card sector, the first data logger program compile with the card can take up to 10 minutes. After that, compile times will be normal.

5.8.2 MicroSD card precautions

Observe the following precautions when using optional memory cards:

- Before removing a card from the data logger, 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 data logger often erases all data.
- See [System specifications](#) (p. 160) for information on maximum card size.

5.8.3 Act LED indicator

When the data logger is powered and a microSD card installed, the Act (Activity) LED will turn on according to card activity or status:

- **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 data logger 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.

5.9 USB Host (USB: drive)

USB host provides portable data storage on a mass storage device (MSD). A single USB thumb drive can be inserted into the drive and will show up as a drive (USB:) in file related operations. Measurement data is stored on USB: as discrete files by using the **TableFile()** instruction. Files on USB can be collected by inserting the thumb drive into a computer and copying the files.

USB: can be used in the **TableFile()** instruction and all file access related instructions in CRBasic. Because of data-reliability concerns in non-industrial rated drives, this drive is not intended for long term unattended data storage. USB: is not affected by program recompilation or formatting of other drives.

5.9.1 USB Host precautions

Observe the following precautions when using optional MSD:

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

5.9.2 Act LED indicator

When the data logger is powered and a USB MSD installed, the Act (Activity) LED will turn on according to MSD activity or status:

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

5.9.3 Formatting drives 32 GB or larger

Windows does not support creating a FAT32 partition on a 32 GB or greater drive. The work-around is to use a Windows computer to format the drive as NTFS (NT file system). Then use the data logger to format the drive as FAT32.

6. Measurements

6.1 Pulse measurements	56
6.2 Sequential and pipeline processing modes	62

6.1 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 data logger detects the state transition as each wave varies between voltage extremes (high-to-low or low-to-high). Measurements are processed and presented as counts, frequency, or timing data.

The data logger includes terminals that are configurable for pulse input as shown in the following image.

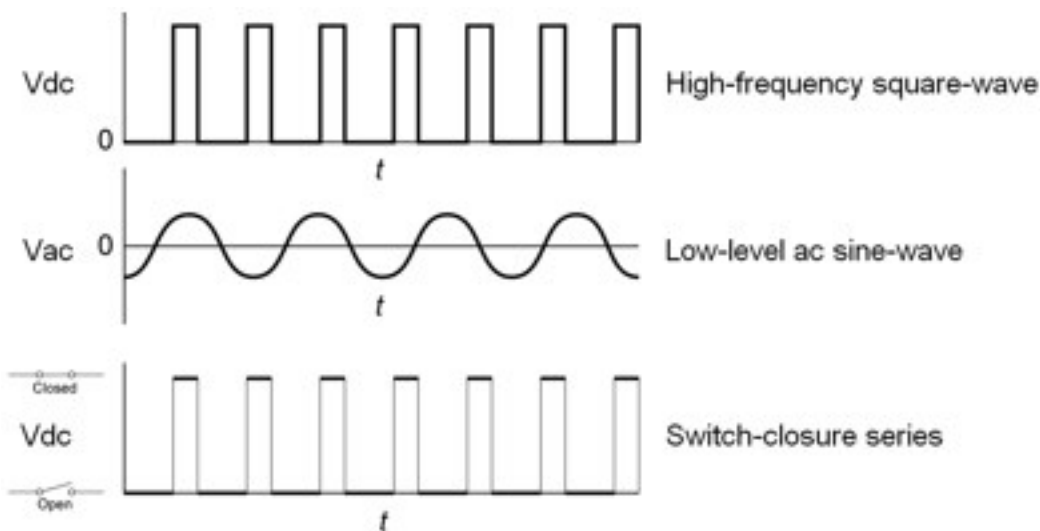


Table 6-1: Pulse input terminals and the input types they can measure

Input Type	Pulse Input Terminal
High-frequency	C (all)
Switch-closure	C (all)

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

NOTE:

Conflicts can occur when a control port pair is used for different instructions (`TimerInput()`, `PulseCount()`, `SDI12Recorder()`, `WaitDigTrig()`). For example, if C1 is used for `SDI12Recorder()`, C2 cannot be used for `TimerInput()`, `PulseCount()`, or `WaitDigTrig()`.

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. 61).

Output can be recorded as counts, frequency or a running average of frequency.

For more information, see [Pulse measurement specifications](#) (p. 164).

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

6.1.1 High-frequency measurements

High-frequency (square-wave) signals can be measured on terminals:

- C

Common sensors that output high-frequency pulses include:

- Photo-chopper anemometers
- Flow meters

Measurement output options include counts, frequency in hertz, and running average.

The data logger 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.

6.1.1.1 C terminals

- CRBasic instructions: `PulseCount()`

See [Pulse measurement specifications](#) (p. 164) for more information.

6.1.2 Switch-closure and open-collector measurements

Switch-closure and open-collector (also called current-sinking) signals can be measured on terminals:

- C

Mechanical switch-closures have a tendency to bounce before solidly closing. Unless filtered, bounces can cause multiple counts per event. The data logger 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 gauges
- Switch-closure anemometers
- Flow meters

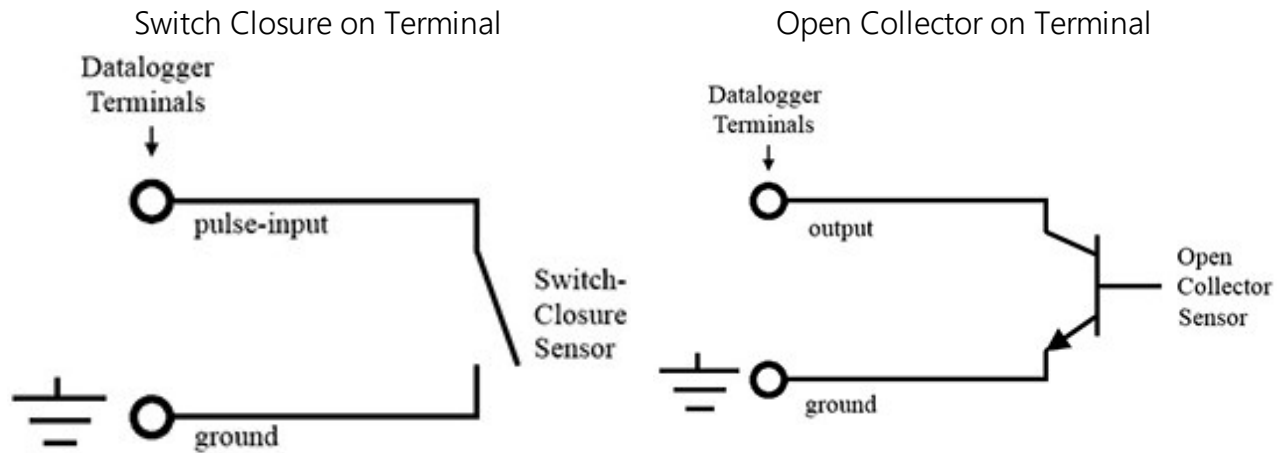
The data logger has built-in pull-up and pull-down resistors for different pulse measurements which can be accessed using the `PulseCount()` instruction. Note that pull down options are usually used for sensors that source their own power.

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

6.1.2.1 C Terminals

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

- CRBasic instruction: `PulseCount()`. See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.



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 measurement specifications](#) (p. 164).

6.1.3 Edge timing and edge counting

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

6.1.3.1 Single edge timing

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

CRBasic instruction: `TimerInput()`

6.1.3.2 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. Measurements can be expressed as a time (μs), frequency (Hz) or period (μs).

- CRBasic instruction: `TimerInput()`

6.1.3.3 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

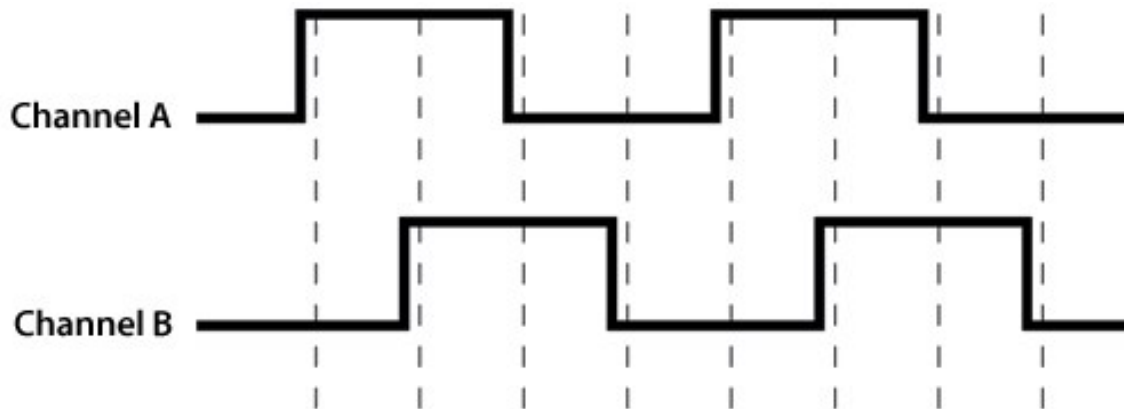
For more information, see:

- [Pulse measurement specifications](#) (p. 164)
- [Digital input/output specifications](#) (p. 165)

6.1.4 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 data logger.

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 **C1-C8** 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 B leads channel A.

- 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 [Digital input/output specifications](#) (p. 165).

6.1.5 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. 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. See the *CRBasic Editor* help for detailed instruction information and program examples:

<https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

Counts are the preferred `PulseCount()` output option when measuring the number of tips from a tipping-bucket rain gauge 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 a control port pair is used for different instructions (`TimerInput()`, `PulseCount()`, `SDI12Recorder()`, `WaitDigTrig()`). For example, if C1 is used for `SDI12Recorder()`, C2 cannot be used for `TimerInput()`, `PulseCount()`, or `WaitDigTrig()`.

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. 57).

6.1.5.1 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 measurement specifications](#) (p. 164) for more information. The listed pulse measurement specifications account for attenuation due to input filtering.

6.1.5.2 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.

6.2 Sequential and pipeline processing modes

The data logger has two processing modes: sequential mode and pipeline mode. In sequential mode, data logger tasks run more or less in sequence. In pipeline mode, data logger tasks run more or less in parallel. Mode information is included in a message returned by the data logger, 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 data logger program is compiled, the data logger analyzes the program instructions and automatically determines which mode to use. The data logger 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.](#)"

6.2.1 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).

6.2.2 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 data logger 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()`, 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 data logger.

6.2.3 Slow Sequences

Priority of a slow sequence (`SlowSequence`) in the data logger will vary, depending upon whether the data logger 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 data logger cycles through the queue until all instructions for all sequences are complete.

7. Communications protocols

Data loggers communicate with data logger support software, other Campbell Scientific data loggers, 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 and CAN FD are natively supported in GRANITE 10 only. See also [Communications specifications](#) (p. 167).

7.1 General serial communications	66
7.2 CPI	72
7.3 EPI	73
7.4 CAN (GRANITE 10 only)	73
7.5 Modbus communications	81
7.6 Internet communications	90
7.7 DNP3 communications	93
7.8 Serial peripheral interface (SPI) and I2C	93
7.9 PakBus communications	93
7.10 SDI-12 communications	94

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 data loggers, 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). See the *CRBasic Editor* help for detailed instruction information and program

examples: <https://help.campbellsci.eu/crbasic/granite10/>,
<https://help.campbellsci.eu/crbasic/granite9/>.

- Set the data logger to be a PakBus router only as needed. When the data logger 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. For more information, see the Device Configuration **Settings Editor Information tables and settings (advanced)** (p. 136).
- 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 more information, see the Device Configuration **Settings Editor Beacon()** and **Verify()** settings.

For information on Designing a PakBus network using the Network Planner tool in *LoggerNet*, watch the following video:



7.1 General serial communications

The data logger 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.

CRBasic instructions for general serial communications include:

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

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

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.

Configure **C** terminals as serial ports using *Device Configuration Utility* or by using the [SerialOpen\(\)](#) CRBasic instruction. **C** terminals are configured in pairs for TTL, LVTTTL, RS-232, and half-duplex RS-422 and RS-485 communications. For full-duplex RS-422 and RS-485, four **C** terminals are required.

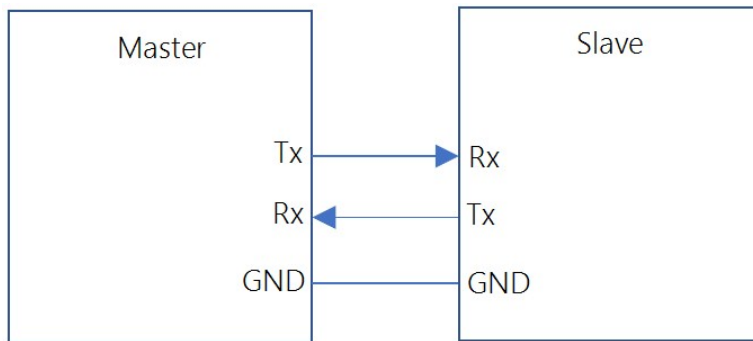


FIGURE 7-1. Single-ended full-duplex communications

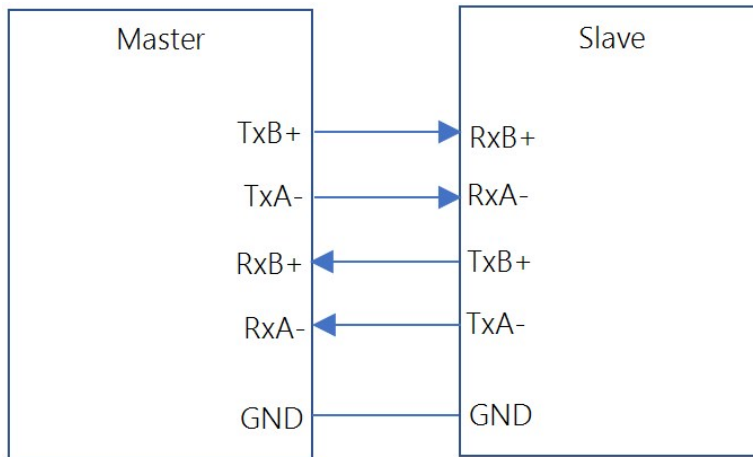


FIGURE 7-2. Differential-pair full-duplex communications

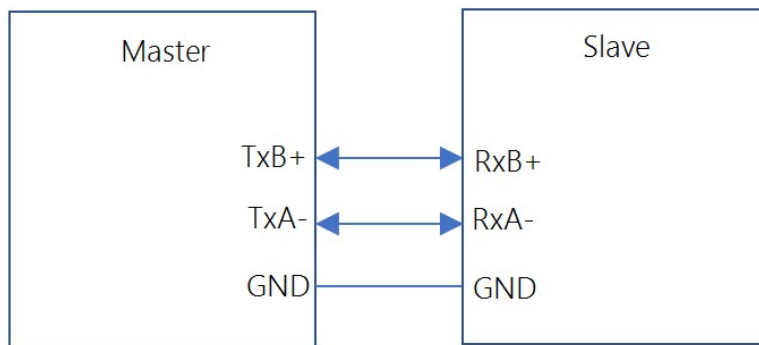


FIGURE 7-3. Differential-pair half-duplex communications

	RS-232	RS-485	RS-422	TTL	LVTTL	TTL-INV	LVTTL-INV
Communications	Point to point	Multi-drop	Single ended Multi-drop	Point to point	Point to point	Point to point	Point to point
Number of devices	1 Transmitter 1 Receiver	32 Transmitters 32 Receivers	1 Transmitter 10 Receivers	1 Tx 1 Rx	1 Tx 1 Rx	1 Tx 1 Rx	1 Tx 1 Rx
Mode type	Full duplex	Full duplex Half duplex	Full duplex Half duplex	Full duplex	Full duplex	Full duplex	Full duplex
Reference	Single wire Tx ref to GND	Differential pair TxB+ referenced to TxA-		Single wire Tx referenced to GND			
Voltage range	-25 V to 25 V	-5 V to 5 V	-6 V to 6 V	0 V to 5 V	0 V to 3.3 V	0 V to 5 V	0 V to 3.3 V
Idle voltage	-25 V to -5 V	-0.2 V to 5 V	-0.2 V to 6 V	5 V	3.3 V	0 V	0 V

7.1.1 RS-232

RS-232 supports point to point communications between one master and one slave device. See [FIGURE 7-1](#) (p. 67). Data bits are sent from master to slave across the transmit (Tx) line with respect to DC ground. The Tx line idle state is between -25 V and -3 V, depending on the transmitter. The transition from negative voltage to above 3 V begins data transmission.

NOTE:

Most RS-232 devices are also compatible communicating with the data logger using TTL-inverted.

NOTE:

The data logger uses about -7 V to represent logic 1, and about 5.8 V to represent logic 0.

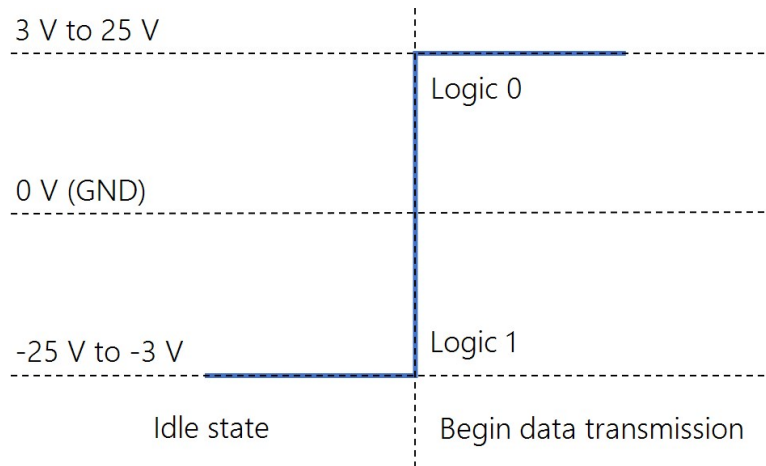


FIGURE 7-4. RS-232 Tx voltage with respect to GND

7.1.2 RS-485

RS-485 supports communications between 32 masters and 32 slaves. See [FIGURE 7-3](#) (p. 68) and [FIGURE 7-2](#) (p. 67). It uses a differential voltage between two lines (A & B) to transmit data. When the voltage of B with respect to A is between -0.2 V and -6 V that is interpreted as logic 0. When the differential voltage in the range of positive 0.2 V to 6 V that is interpreted as logic 1.

NOTE:

The GRANITE 9/10 uses about -1 V to represent logic 0, and about 1 V to represent logic 1.

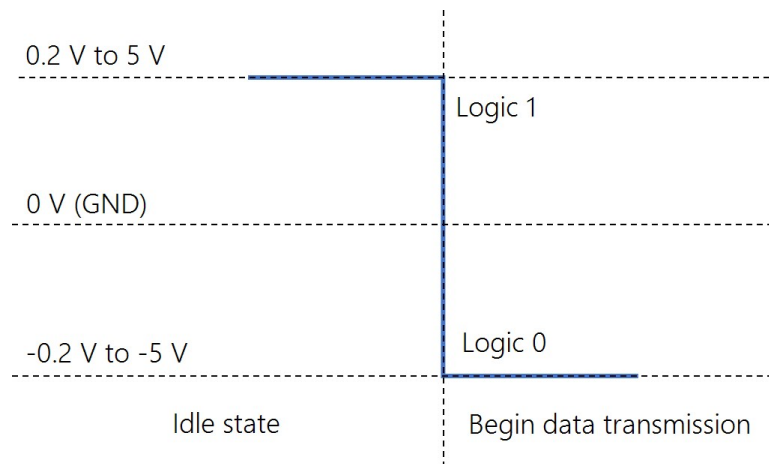


FIGURE 7-5. RS-485 Voltage B with respect to A

7.1.3 RS-422

RS-422 communications protocol is similar to RS-485. The difference is that RS-422 ranges from -6 V to 6 V instead of -5 V to 5 V. Also, RS-422 only supports communications from 1 master to 10 slaves, but not return communications from all 10 slaves. In full-duplex point-to-point (1 master, 1 slave) RS-422 communications, both devices can transmit and receive. Half-duplex can be used in cases where sensors broadcast data to a receiving data logger. See [FIGURE 7-3](#) (p. 68) and [FIGURE 7-2](#) (p. 67).

NOTE:

Use the RS-485 communications type when setting up the data logger for RS-422 communications. Most RS-422 sensors will work with RS-485 protocol.

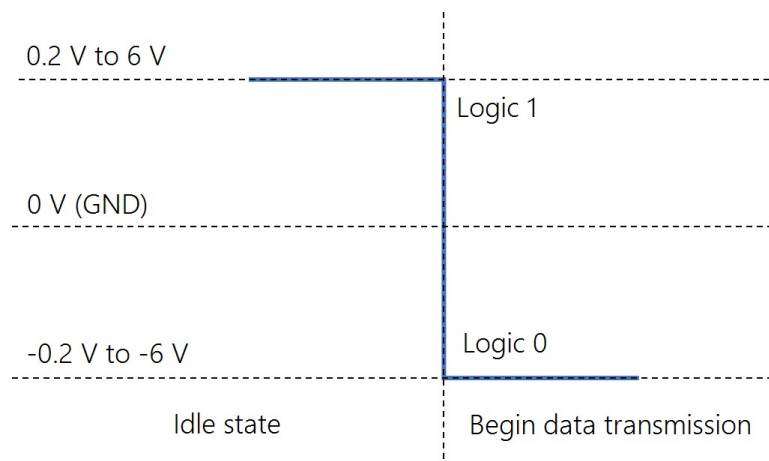


FIGURE 7-6. RS-422 Voltage B with respect to A

7.1.4 TTL

TTL supports point to point communications between one master and one slave device. See [FIGURE 7-1](#) (p. 67). Data bits are sent from master to slave with a voltage between transmit (Tx) and ground. The transmit line idle state is 5 V (logic 1). Data is sent after one clock cycle once the voltage is pulled low (to 0 V).

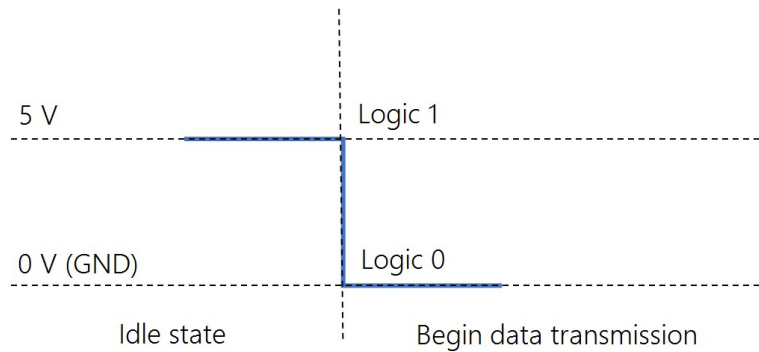


FIGURE 7-7. TTL Tx voltage with respect to GND

7.1.5 LVTTTL

The only difference between low-voltage TTL (LVTTTL) and TTL is that the voltage range is 0 V to 3.3 V. See [FIGURE 7-1](#) (p. 67).

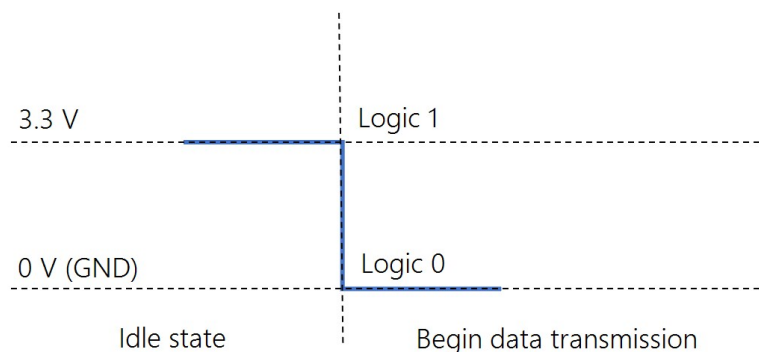


FIGURE 7-8. LVTTTL Tx voltage with respect to GND

7.1.6 TTL-Inverted

The only difference between TTL-inverted and TTL is that the logic is inverted. The idle state for TTL-inverted is 0 V instead of 5 V. See [FIGURE 7-1](#) (p. 67). Data is sent after the voltage is pulled high (to 5 V).

NOTE:

Many RS-232 devices are compatible with this communications protocol.

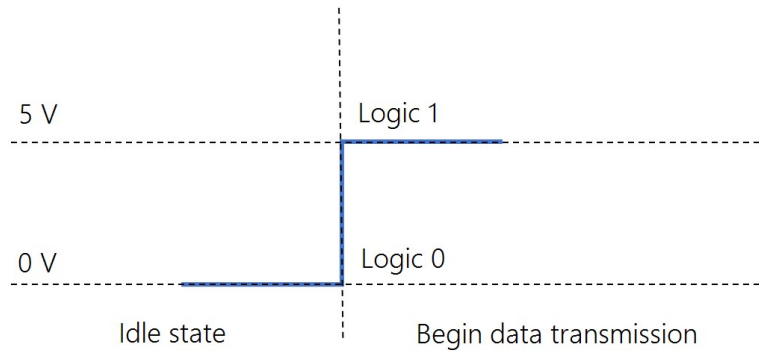


FIGURE 7-9. TTL-inverted Tx voltage with respect to GND

7.1.7 LVTTL-Inverted

The only difference between LVTTL-inverted and TTL-inverted is that the voltage range is 0 V to 3.3 V. See [FIGURE 7-1](#) (p. 67).

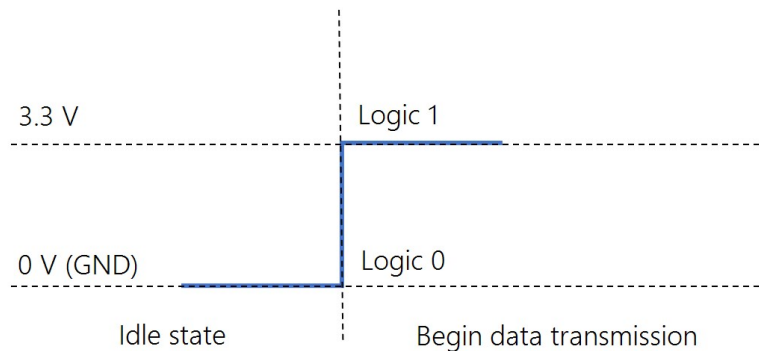


FIGURE 7-10. LVTTL-inverted Tx voltage with respect to GND

7.2 CPI

The data logger includes one RJ45 module jack labeled RS-232/CPI. CPI is a proprietary interface for communications between Campbell Scientific data loggers and Campbell Distributed Modules (CDMs) such as the GRANITE-Series peripheral devices and smart sensors. 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. Some GRANITE devices may require more power to operate in general than do SDM devices. Consult the manuals for GRANITE modules for more information.

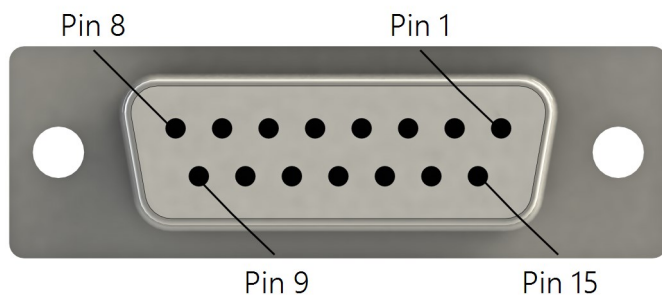
7.3 EPI

The data logger includes two RJ45 module jacks labeled EPI. Ethernet Peripheral Interface (EPI) is a proprietary interface for communications between Campbell Scientific data loggers and Campbell Distributed Modules (CDMs) such as the GRANITE-Series peripheral devices and smart sensors. EPI expands the functionality or channel count of the GRANITE 9/10. This communications connection satisfies the tight timing requirements imposed on independent GRANITE Measurement Modules that are working together as part of a measurement and control system.

The underlying communications of EPI are built using TCP/IP. More specifically, the IEEE 1588 protocol is implemented at the lowest hardware levels for synchronization of device clocks across the entire network. This accomplishes tighter device synchronization and 100-times the data throughput of CPI. The additional power, cost, and complexity are warranted for fast sampling applications.

7.4 CAN (GRANITE 10 only)

The CAN (Controller Area Network) physical layer is a differential signal that is generally a twisted pair. The GRANITE 10 has 4 general purpose CAN ports, CAN 2.0 up to 1 Mbps, or CAN FD up to 5 Mbps. They can be accessed either by the screw terminals, or the 15-pin connector; they are electrically connected. Wire the differential signal (H and L) into the screw terminals, or use a custom cable and the 15-pin connector.



Pin Number	Description
1	CAN1 H
2	CAN1 L
3	RG1
4	CAN2 H

Pin Number	Description
5	CAN2 L
6	RG2
7	CAN3 H
8	CAN3 L
9	RG3
10	CAN4 H
11	CAN4 L
12	RG4
13	Not used
14	Not used
15	Not used

7.4.1 CRBasic instructions

In the GRANITE 10 program, CAN functionality is accessed with the `CANPortOpen()`, `CANRead()`, `CANFilterRead()`, `CANWrite()`, `CANFDPortOpen()`, `CANFDRead()`, `CANFDFilterRead()`, and `CANFDWrite()` instructions. Both *Short Cut* and CRBasic support DBC files for vehicle CAN integration. If a protocol other than DBC is needed, use `CANRead()` and `CANWrite()` to implement it. See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

7.4.1.1 CANPortOpen() / CANFDPortOpen

These instruction configure the bit rate and other physical parameters of the specified CAN port. The instruction only needs to be executed once per port, and should located after `BeginProg()`, and before `Scan()`.

`CANPortOpen()` has the following parameters:

<code>CANPort</code>	Specifies which of the 4 CAN ports to use.
<code>BitRate</code>	Specifies the bit rate in Kbps that is used for communication on the CAN port.
<code>SilentMode</code>	Specifies whether messages can be received without

acknowledgment.

Termination	Specifies if the CAN-H and CAN-L signal are terminated with a 120 Ohm resistor.
Prescale (Optional)	Valid ranges are 2 to 4096. The CAN peripheral is driven by a 50 MHz clock. The prescale scales this clock down to the desired time segment. For example: a 4 in this parameter will use a 12.5 MHz (80 ns period) clock as the fundamental bus time segment (TimeQuanta).
TimeSeg1 (Optional)	The number of TimeQuanta before the sample point.
TimeSeg2 (Optional)	The number of TimeQuanta after sample point.
SJW (Optional)	(Synchronization Jump Width) – The maximum number of TimeQuanta that a bit time can be changed by one resynchronization. Valid values are 1-4.

CANFDPortOpen() has the following parameters:

CANPort	Specifies which of the 4 CAN ports to use.
SlowBitRate	An integer that specifies the bit rate (Kbps) used for communications on the CAN FD port at the slower CAN FD bit rate.
FastBitRate	An integer that specifies the bit rate (Kbps) used for communications on the CAN FD port at the faster CAN FD bit rate.
Bosh/IsoMode	Specifies which standard to use: Bosch 1.0 CAN FD (released 2012), or ISO 11898-1 CAN FD (released 2014).
SilentMode	Specifies whether messages can be received without acknowledgment.
RestrictMode	Specifies whether the CAN FD port should use the Restricted Mode outlined in Section 3.3.2 of the CAN FD specification. In this mode, a CAN FD node is able to transmit and receive DATA FRAMES and REMOTE FRAMES and it gives ACKNOWLEDGE to valid frames, but it does not send ACTIVE ERROR FRAMES or OVERLOAD FRAMES. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of BUS IDLE condition to resynchronize itself to the CAN communications.

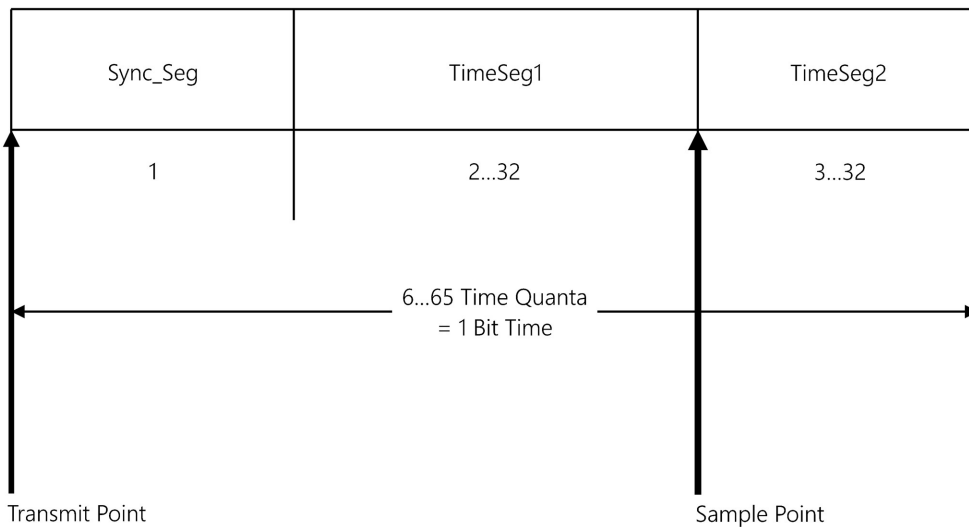
Termination	Specifies if the CAN-H and CAN-L signal are terminated with a 120 Ohm resistor.
(Optional)	These parameters only apply when SlowBitRate or FastBitRate is 0.
SlowPrescale FastPrescale	Valid ranges are 2 to 513 in ISO mode and 2 to 4097 in Bosch mode. The CAN peripheral is driven by a 160 MHz clock. The prescale scales this clock down to the desired time segment. For example: a 4 in this parameter will use a 40 MHz (25 ns period) clock as the fundamental bus time segment (TimeQuanta).
SlowTimeSeg1 FastTimeSeg1	The number of TimeQuanta before the sample point. Valid ranges are 1 to 256 in ISO mode, and 1 to 64 in Bosch Mode.
SlowTimeSeg2 FastTimeSeg2	The number of TimeQuanta after sample point. Valid ranges are 2 to 257 in ISO mode, and 2 to 65 in Bosch Mode.
SlowSJW FastSJW	(Synchronization Jump Width) – The maximum number of TimeQuanta that a bit time can be changed by one resynchronization. Valid values are 1 to 120 in ISO mode and 1 to 16 in Bosch Mode..

For convenience, several bitrates have been predefined. The predefined bitrates have a sample point of 85-88% in accordance with the CAN-in-Automation user group recommendations. The predefined baud rates use an SJW setting of 1. If a custom sample point, baud rate, or SJW is needed, set the baud rate parameter to 0 and use the optional parameters (**Prescale**, **TimeSeg1**, **TimeSeg2**, and **SJW**) to define the bitrate. When developing a user-specified baud rate, the following formulas are used:

$$\text{Total Quanta} = (1.0 + \text{timea} + \text{timeb})$$

$$\text{samplepoint} = (100 * (\text{timea} + 1.0)) / \text{Total Quanta}$$

$$\text{baud} = (50,000,000 / (\text{Total Quanta} * \text{Prescale}))$$



The CRBasic pre-compiler will do these and report the baud rate and sample point found from the entered parameters.

```

1  |'GRANITE 10 DataLogger
2  BeginProg
3  CANPortOpen (1,0,0,0,2,21,3,1)
4  Scan (1,Sec,0,0)
5    'Enter other instructions
6  NextScan
7  EndProg
8

```

[Version]C:\Campbellsci\Lib\Compilers\Granite10Comp.exe VERSION:GRANITE10.1 DATE:09/05/2019
 NoName0.CRB -- Compiled in PipelineMode.
 ProgSignature = 36999
 line 3: Warning: User CAN parameters give a 1000.00 Kbits/s bit rate and 88.00% Sample Point

7.4.1.2 CANRead() / CANFDRead()

CANRead() and **CANFDRead()** are used to read a message on a CAN port and write it to a CRBasic variable. **CANPortOpen()** must be declared before **CANRead()** or **CANFDRead()** is allowed. **CANRead()** / **CANFDRead()** can be used to read a signal as defined in a DBC file. It can also be used in conjunction with **CANWrite()** for other CAN protocols. **CANRead()** and **CANFDRead()** have the same parameters.

Syntax:

CANRead (Dest, Reqs, RunOption, RunOptTimeout, CANPort, CanID, EXTID, MessageSize, StartBit, ReadSize, ByteOrder, ValType, MULT, Offset)

CANFDRead (Dest, Reqs, RunOption, RunOptTimeout, CANPort, CanID, EXTID, MessageSize, StartBit, ReadSize, ByteOrder, ValType, MULT, Offset)

7.4.1.3 CANFilterRead() / CANFDFilterRead()

[CANFilterRead\(\)](#) and [CANFDFilterRead\(\)](#) are used to filter messages based on ID and when the filter is met, read a message on a CAN port and write it to a CRBasic variable.

[CANPortOpen\(\)](#) must be declared before [CANFilterRead\(\)](#) / [CANFDFilterRead\(\)](#) is allowed.

[CANFilterRead\(\)](#) and [CANFDFilterRead\(\)](#) have the same parameters.

Syntax:

```
CANFilterRead ( Dest, Reps, RunOption, RunOptTimeOut, CANPort, CanID, CANIDMask,
CANIDFilter, EXTID, MessageSize, StartBit, ReadSize, ByteOrder, ValType,
MULT,Offset)
```

```
CANFDFilterRead ( Dest, Reps, RunOption, RunOptTimeOut, CANPort, CanID,
CANIDMask, CANIDFilter, EXTID, MessageSize, StartBit, ReadSize, ByteOrder,
ValType, MULT,Offset)
```

7.4.1.4 CANWrite() / CANFDWrite()

[CANWrite\(\)](#) and [CANFDWrite\(\)](#) are used to write a message from a CRBasic variable to be transmitted onto a CAN port. [CANPortOpen\(\)](#) must be declared before a [CANWrite\(\)](#) / [CANFDWrite\(\)](#) is allowed. [CANWrite\(\)](#) / [CANFDWrite\(\)](#) can be used to write a signal as defined in a DBC file. [CANFDWrite\(\)](#) has an additional BRS (bit rate switch) parameter.

Syntax:

```
CANWrite ( OutVar, Reps, PartialFrame, CANPort, CANID, EXTID, MessageSize,
StartBit, WriteSize, ByteOrder, ValType)
```

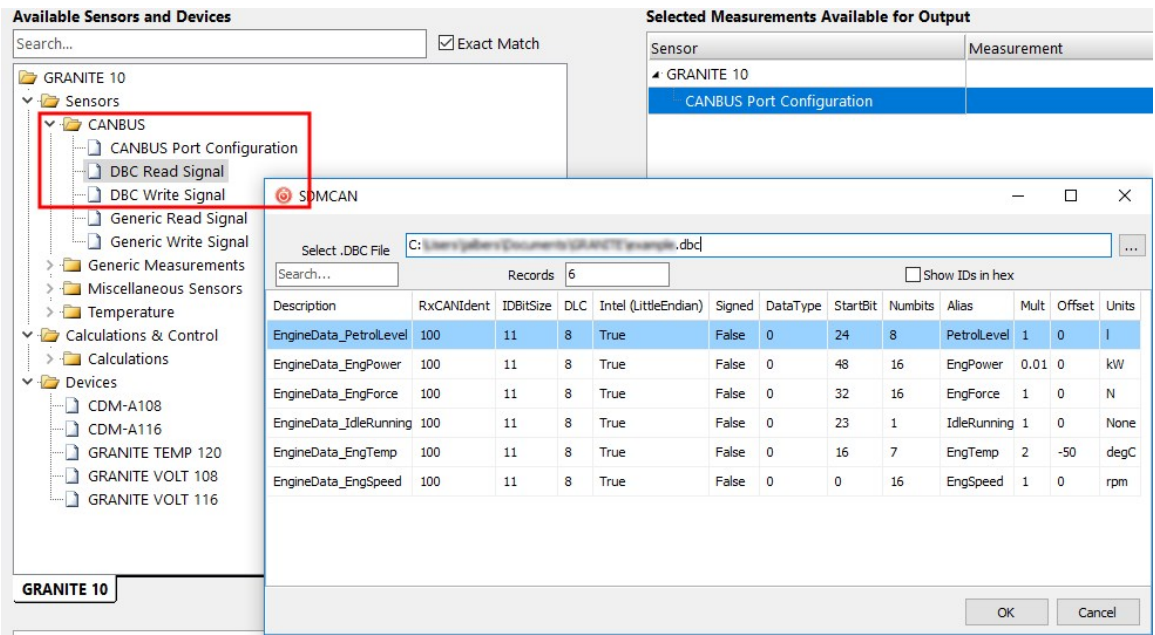
```
CANFDWrite( OutVar, Reps, PartialFrame, CANPort, BRS, CANID, EXTID, MessageSize,
StartBit, WriteSize, ByteOrder, ValType)
```

A few notes on [CANRead\(\)](#) / [CANFDRead\(\)](#), [CANFilterRead\(\)](#) / [CANFDFilterRead\(\)](#) and [CANWrite\(\)](#) / [CANFDWrite\(\)](#): All of the type conversion is handled by the data logger. CRBasic variables defined as FLOAT will be transmitted as integers using [CANWrite\(\)](#) (assuming ValType is unsigned or signed integer). The GRANITE 10 automatically casts the FLOAT variable to an integer and transmits it in the CAN frame as specified. When a variable is defined as a FLOAT and then read as an integer from a CAN frame, the GRANITE 10 will cast the integer, apply the multiplier and offset specified, and store it as a FLOAT.

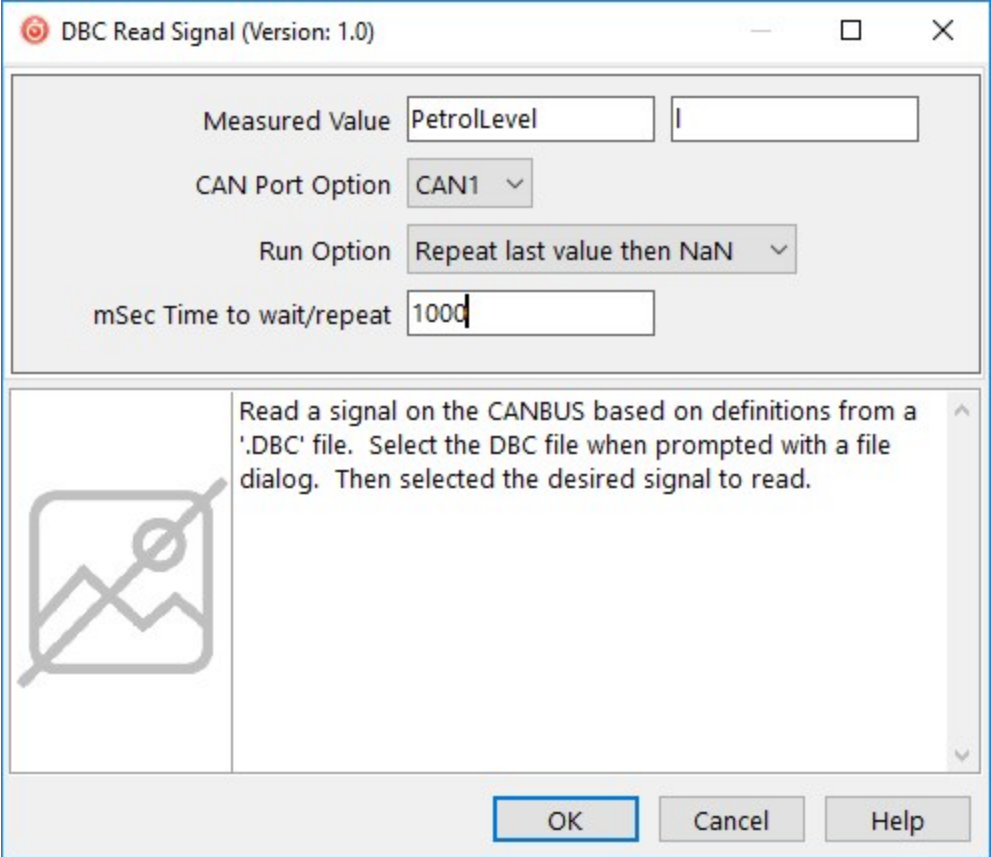
7.4.2 DBC Signal Support

Both *Short Cut* and the *CRBasic Editor* have support for DBC signals.

In *Short Cut*, after adding a **CAN Port Configuration**, add a **DBC Read Signal** or a **DBC Write Signal**. Then you will be prompted to open a DBC file.

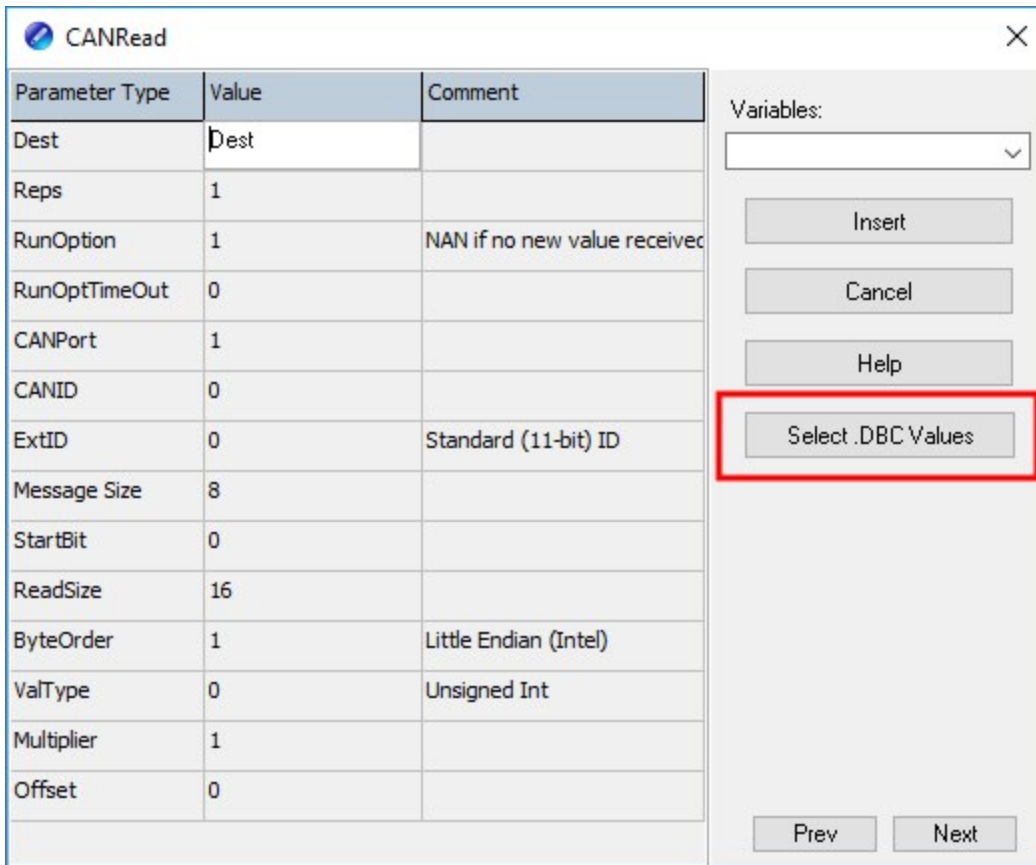


After opening a DBC file, select the signals in that DBC file. Click **OK**.

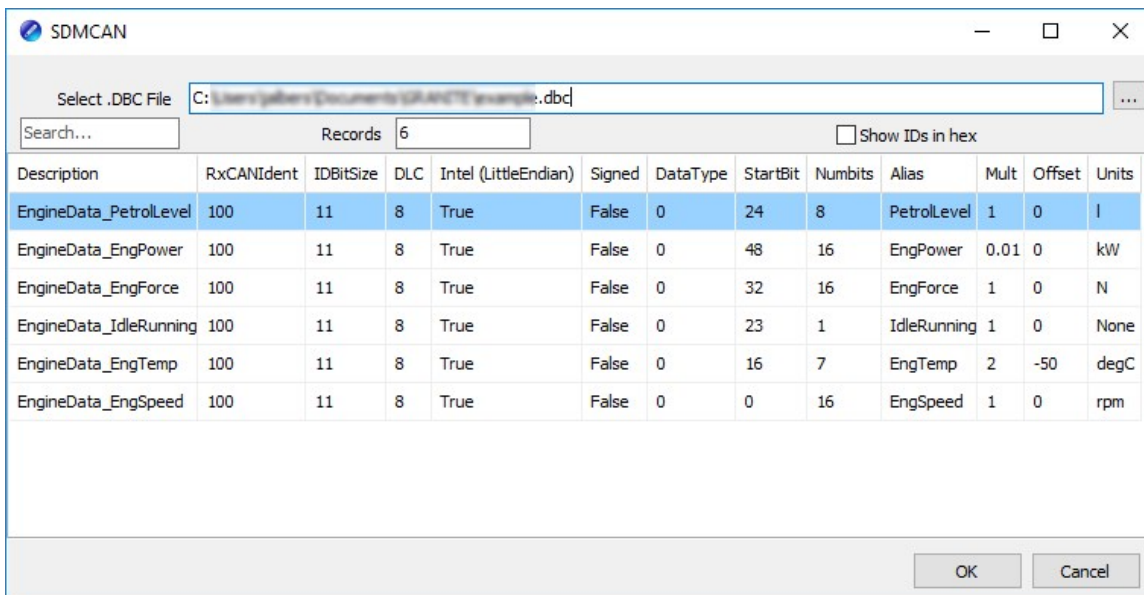


Click **OK** again and the program will be set up to read that signal into the variable specified. **DBC Write Signal** is similar.

To access the same information from CRBasic, click **Select .DBC Values** in the **CANWrite()** and **CANRead()** parameter entry box.



Click **OK** to automatically enter the required parameters.



7.4.3 J1979 Legislated PIDS Example

See <https://www.campbellsci.eu/downloads/granite10-example-can-j1979> for an example program to query an ECU (electronic control unit) for the legislated PID (Parameter ID) values once a second. `CANPortOpen()` configures CAN 1 for this example. `CANWrite()` is used to request each PID, and `CANRead()` is used to interrupt the response and store the requested values in CRBasic Variables. This example program can be modified to read other manufacturer specific PIDs.

7.5 Modbus communications

The data logger 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. Data loggers can communicate using Modbus on all available communication ports. The data logger supports RTU and ASCII communication modes on RS-232 and RS-485 connections.

CRBasic Modbus instructions include:

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

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

For additional information on Modbus, see:

- [About Modbus \(p. 82\)](#)
- [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 data logger 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 data logger without additional hardware.

When doing Modbus communications over RS-232, the data logger, 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 data logger 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 data logger; the second packet is

received as data. This would make a Modbus master fail to poll the data logger, 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

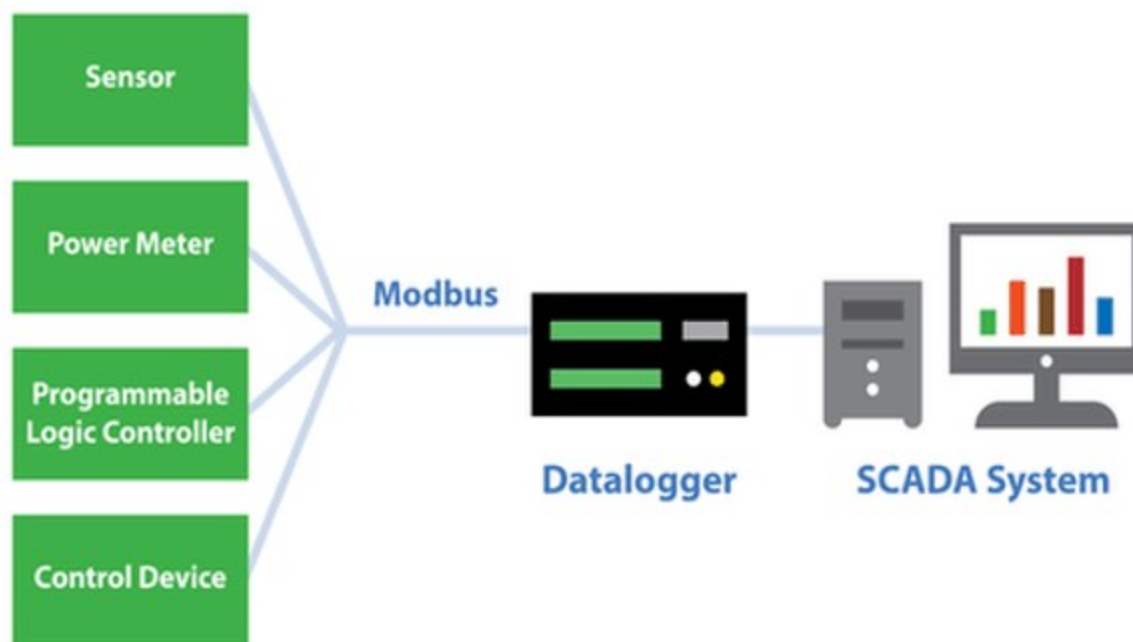
7.5.1 About Modbus

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 data loggers include:

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



7.5.2 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 uses a different header. Devices labeled as Modbus gateways will convert from Modbus TCP to Modbus RTU.

Campbell Scientific data loggers support Modbus RTU, Modbus ASCII, and Modbus TCP protocols. If the connection is over IP, Campbell Scientific data loggers always use Modbus TCP. Modbus slave functionality over other comports use RTU. When acting as a master, the data logger can be switched between ASCII and RTU protocols using an option in the **ModbusMaster()** instruction. See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

7.5.3 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.

7.5.4 Connecting Modbus devices

data loggers 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.

7.5.5 Modbus master-slave protocol

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 data loggers can be programmed to be a Modbus master or Modbus slave - or even both at the same time! This proves particularly helpful when your data logger 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.

7.5.6 About Modbus programming

Modbus capability of the data logger 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 data logger.

CRBasic Modbus instructions include:

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

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

7.5.6.1 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. Endianness is used in some Modbus programming so it is important to note that the GRANITE 9/10 is a big-endian instrument.

7.5.6.2 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 data loggers.

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.

7.5.7 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.

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

7.5.7.1 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.

7.5.7.2 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 use coils at all.

When working with coils, the data logger 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.

7.5.7.3 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 data logger, 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 data logger, 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 data loggers. 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 data loggers. Select the appropriate **ModbusOption** to avoid post-processing.

7.5.8 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 data logger. 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

7.5.8.1 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 data logger 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 data logger 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 data logger should work.

7.6 Internet communications

See the [Communications specifications](#) (p. 167) for a list of the internet protocols supported by the data logger. The most up-to-date information on implementing these protocols is contained in *CRBasic Editor* help.

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 data logger clock
- Collecting data
- Displaying the current record in a data table

Data logger callback to *LoggerNet* and data logger-to-data logger communications are also possible over TCP/IP. For details and example programs see the CRBasic help.

7.6.1 IP address

When connected to a server with a list of IP addresses available for assignment, the data logger 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 GRANITE 9/10 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 data logger instructions.

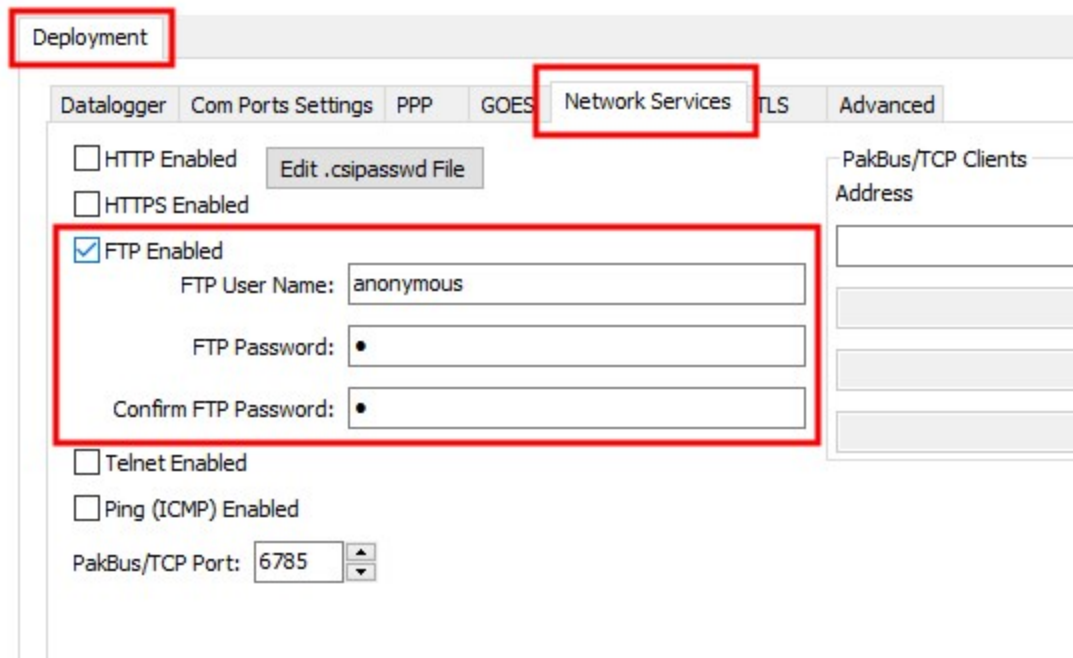
7.6.2 HTTPS server

Use *Device Configuration Utility* to configure the data logger to act as an HTTPS server.

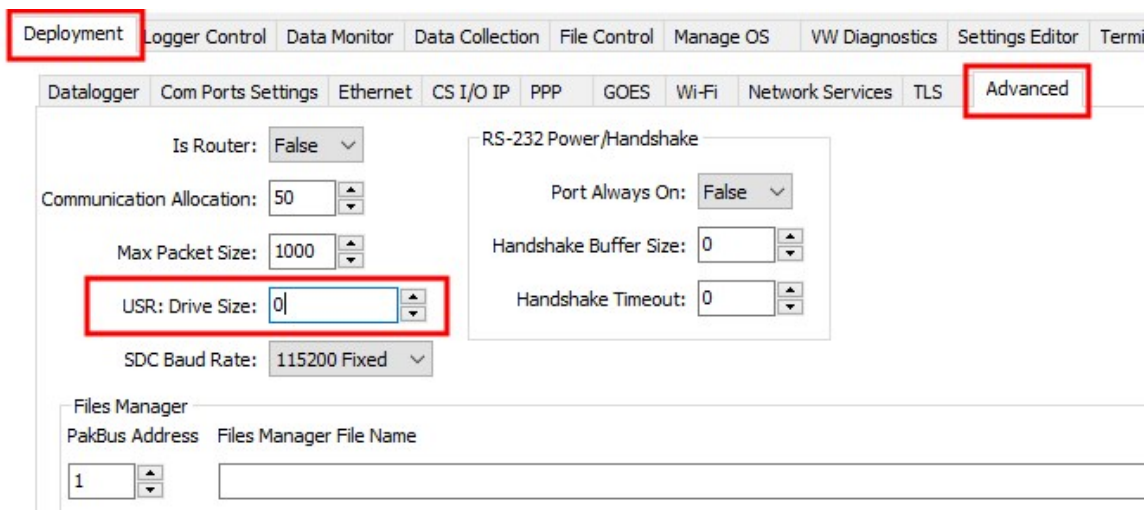
7.6.3 FTP server

An FTP server facilitates file transfers. Use *Device Configuration Utility* to configure the data logger to act as an FTP server. This is useful when receiving and storing images from an Ethernet enabled device such as a camera.

Select [FTPEnabled](#) (p. 148) and assign a **User Name** and **Password**.



Allocate memory where the received files will be stored. Often this is on the USB drive. [Data memory](#) (p. 48)



WARNING:

Partitioning or changing the size of the USB drive will delete stored data from tables. Collect data first.

Specify the memory drive in the path when putting or getting files. For example, to put a file named *image.jpg* on the USB drive, use a command similar to `put image.jpg /USB/image.jpg`.

NOTE:

Use `FTPClient()` to send files to a remote server. This is different than setting up the data logger to act as an FTP server. See the [FTP streaming technical paper](#) for more information.

7.7 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 data logger allows DNP3 communications on all available communications ports. CRBasic DNP3 instructions include:

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

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

When `DNPUpdate()` is used to set up the data logger as a remote (slave) device, up to three DNP3 masters are supported.

For additional information on DNP3 see:

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

7.8 Serial peripheral interface (SPI) and I2C

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

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

7.9 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 data logger.

The data logger allows PakBus communications on all available communications ports. For additional information, see [The Many Possibilities of PakBus Networking](#) blog article.

Advantages of PakBus include:

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

In a PakBus network, each data logger is assigned a unique address. The default PakBus address in most devices is 1. To communicate with the data logger, the data logger support software must know the data logger PakBus address. The PakBus address is changed using *Device Configuration Utility*, data logger *Settings Editor*, or *PakBus Graph* software.

7.10 SDI-12 communications

SDI-12 is a 1200 baud communications protocol that supports many smart sensors, probes and devices. The data logger supports SDI-12 communications through two modes — transparent mode and programmed mode (see [SDI-12 ports](#) (p. 13) for wiring terminal information).

Conflicts can occur when a control port pair is used for different instructions ([TimerInput\(\)](#), [PulseCount\(\)](#), [SDI12Recorder\(\)](#), [WaitDigTrig\(\)](#)). For example, if C1 is used for [SDI12Recorder\(\)](#), C2 cannot be used for [TimerInput\(\)](#), [PulseCount\(\)](#), or [WaitDigTrig\(\)](#).

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. 128) 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. 97) for more information.

CRBasic SDI-12 instructions include:

- [SDI12Recorder\(\)](#)
- [SDI12SensorSetup\(\)](#)
- [SDI12SensorResponse\(\)](#)

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

The data logger uses SDI-12 version 1.4.

7.10.1 SDI-12 transparent mode

All SDI-12 probes have just three wires—a signal, ground, and 12 V power line. They are connected to the data logger according to the following table.

Wire function	Data logger connection
Shield	\perp (analogue ground)
Power	12V
Power ground	G

System operators can manually interrogate and enter settings in probes, connected to the data logger, 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 data logger programs may not execute. Data logger security may need to be unlocked before transparent mode can be activated.

Transparent mode is entered while the computer is communicating with the data logger through a terminal emulator program such as through *Device Configuration Utility* or other data logger 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 data logger support software terminal emulator:

```
Deployment | Logger Control | Data Monitor | File Control | Send OS | Settings Editor | Terminal
CR: >
CR: >SDI12
Enter Cx Port 1,2,3 or ?
1
Entering SDI12 Terminal
+
Exit SDI12 Terminal
```

1. Press **Enter** until the data logger responds with the prompt **GRANITE 9/10>**.
2. Type **SDI12** at the prompt and press **Enter**.
3. In response, the query **Select SDI12 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.

7.10.1.1 Watch command (sniffer mode)

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



1. Enter the transparent mode as described previously.
2. Press **Enter** until a **GRANITE 9/10>** prompt appears.
3. Type **W** and then press **Enter**.
4. In response, the query **Select SDI12 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**.
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.

7.10.1.2 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).

7.10.2 SDI-12 programmed mode/recorder mode

The data logger 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 data logger 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 **aD!** 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 data logger 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 data logger issues the **aM!** and **aD!** commands with proper elapsed time between the two. The data logger 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.

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

7.10.3 Programming the data logger to act as an SDI-12 sensor

The `SDI12SensorSetup()` / `SDI12SensorResponse()` instruction pair programs the data logger to behave as an SDI-12 sensor. A common use of this feature is to copy data from the data logger to other Campbell Scientific data loggers 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.

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

When programmed as an SDI-12 sensor, the data logger will respond to SDI-12 commands **M**, **MC**, **C**, **CC**, **R**, **RC**, **V**, **?**, and **I**.

When acting as a sensor, the data logger can be assigned only one SDI-12 address per SDI-12 port. For example, a data logger 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.

7.10.4 SDI-12 power considerations

When a command is sent by the data logger to an SDI-12 probe, all probes on the same SDI-12 port will wake up. However, only the probe addressed by the data logger 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 7-4: 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. GRANITE 9/10 maintenance

Protect the data logger from humidity and moisture. When humidity levels reach the dewpoint, condensation occurs, and damage to data logger 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 data logger to Campbell Scientific for calibration or repair, consult first with Campbell Scientific. If the data logger is malfunctioning, be prepared to perform some troubleshooting procedures (see [Tips and troubleshooting](#) (p. 118)).

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

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

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

8.1 Data logger calibration	100
8.2 Data logger security	101
8.3 Data logger enclosures	105
8.4 Internal battery	107
8.5 Electrostatic discharge and lightning protection	109
8.6 Power budgeting	111
8.7 Updating the operating system	111
8.8 File management via powerup.ini	114

8.1 Data logger 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 data logger is checked to ensure that all

hardware operates within published specifications before it is returned. To request recalibration for a product, see <https://www.campbellsci.eu/repair>.

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

- The importance of the measurements
- How long the data logger will be used
- The operating environment
- How the data logger will be handled

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.eu/calcerts>.

NOTE:

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

Watch an instructional [video](#).



8.2 Data logger security

Data logger security concerns include:

- Collection of sensitive data
- Operation of critical systems
- Networks that are accessible to many individuals

Some options to secure your data logger from mistakes or tampering include:

- Sending the latest operating system to the data logger. See [Updating the operating system](#) (p. 111) 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 data logger on an IP network.

NOTE:

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](#) (p. 104) for more information).
- Enabling HTTPS and disabling HTTP. To prevent data collection via the web interface, both HTTP and HTTPS must be disabled.
- Using a public/private key pair for SFTP authentication.
- 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 data logger for changes by tracking program and operating system signatures, as well as CPU, USR, and CRD file contents.
- Securing the physical data logger and power supply under lock and key.

WARNING:

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

8.2.1 TLS

Transport Layer Security (TLS) is an internet communications security protocol. TLS settings are necessary for **server** applications, not for client applications.

Example server application instructions include:

- HTTPS server
- [DNP3\(\)](#)

Example client application instructions include:

- [HTTPGet\(\)](#), [HTTPPut\(\)](#) and [HTTPPost\(\)](#)
- [EmailRelay\(\)](#)

- [EmailSend\(\)](#) and [EmailRecv\(\)](#)
- [FTPClient\(\)](#)

Use the *Device Configuration Utility* to enable and set up TLS. See **Deployment > Datalogger > TLS** tab.

8.2.2 Security codes

The data logger 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 data logger or to the communications hardware can, with only minimal trouble, overcome the five-digit security codes. Security codes are held in the data logger 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 data loggers 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**.

Function	Security code 1 set	Security code 2 set	Security code 3 set
data logger 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\(1\)](#), [Security\(2\)](#), [Security\(3\)](#) (p. 154) for the related fields in the Settings Editor.

For additional information on data logger 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](#)

8.2.3 Creating a .csipasswd file

The data logger employs a security code scheme that includes three levels of security (see [Data logger security](#) (p. 101) for more information). This scheme can be used to limit access to a data logger 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 data logger data. For more information on the web interface, watch an instructional [video](#).



NOTE:

Ethernet over USB (RNDIS) is considered a direct communications connection. Therefore, it is a trusted connection and `csipasswd` does not apply.

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

When a **.csipasswd** file is used, the PakBus/TCP Password security setting is not used when accessing the data logger 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 data logger web server is attempted without the appropriate security level, the data logger will prompt the web client to display a username and password request dialog. If an invalid username or password is entered, the data logger 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 data logger, then that numeric security code must be entered to access the data logger. If a **.csipasswd** file is on the data logger, the username and password employed by the basic access authentication will eliminate the need for entering the numeric security code.

8.2.3.1 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 Data logger enclosures

The data logger and most of its peripherals must be protected from moisture and humidity. Moisture in the electronics will seriously damage the data logger. In most cases, protection from moisture is easily accomplished by placing the data logger 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.

See also [Physical specifications](#) (p. 162).

8.3.1 Mounting in an enclosure

GRANITE data loggers and measurement module brackets have mounting holes through which small screws are inserted into nylon anchors in an enclosure backplate.

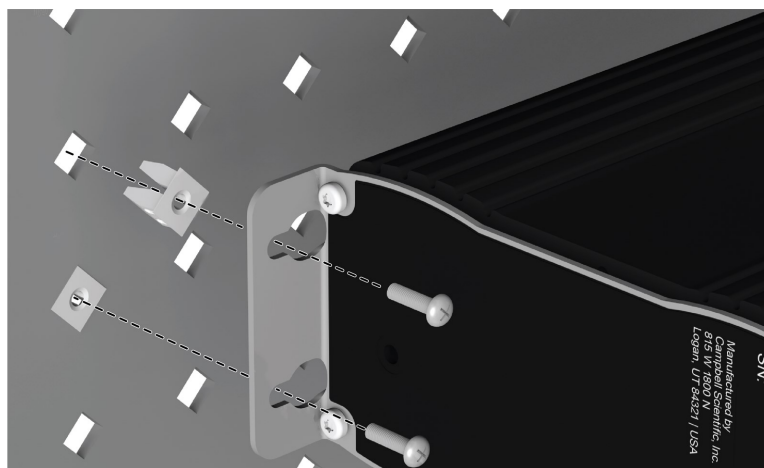


FIGURE 8-1. Enclosure backplate mounting

If mounting to a DIN rail, use the GRANITE-series DIN-Rail Kit as shown in the following images.

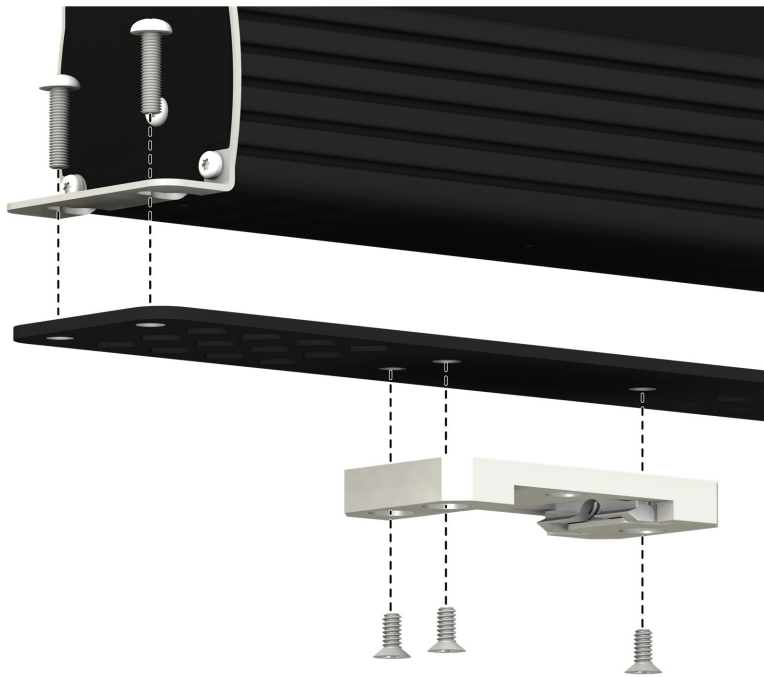


FIGURE 8-2. GRANITE DIN-Rail Mounting Kit



FIGURE 8-3. DIN rail mounting

8.4 Internal battery

The lithium battery powers the internal clock and SRAM when the data logger is not powered. This voltage is displayed in the LithiumBattery (see [Information tables and settings \(advanced\)](#) (p. 136)) field in the **Status** table. Replace the battery when voltage is approximately 2.7 VDC. The internal lithium battery life is extended when the data logger is installed with an external power source. If the data logger 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 [Data logger calibration](#) (p. 100).

When the lithium battery is removed (or is depleted and primary power to the data logger 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.

A replacement lithium battery can be purchased from Campbell Scientific or another supplier.

- 1/2AA, 1.2 Ah, 3.6 VDC (Tadiran L5902S) for battery-backed memory and clock. 5-year life with no external power source.

See [Power requirements](#) (p. 162) 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 data logger in for scheduled calibration, which includes internal battery replacement (see [Data logger calibration](#) (p. 100)).

WARNING:

Any damage made to the data logger during user replacement of the internal battery is not covered under warranty.

1. Remove external battery cover.



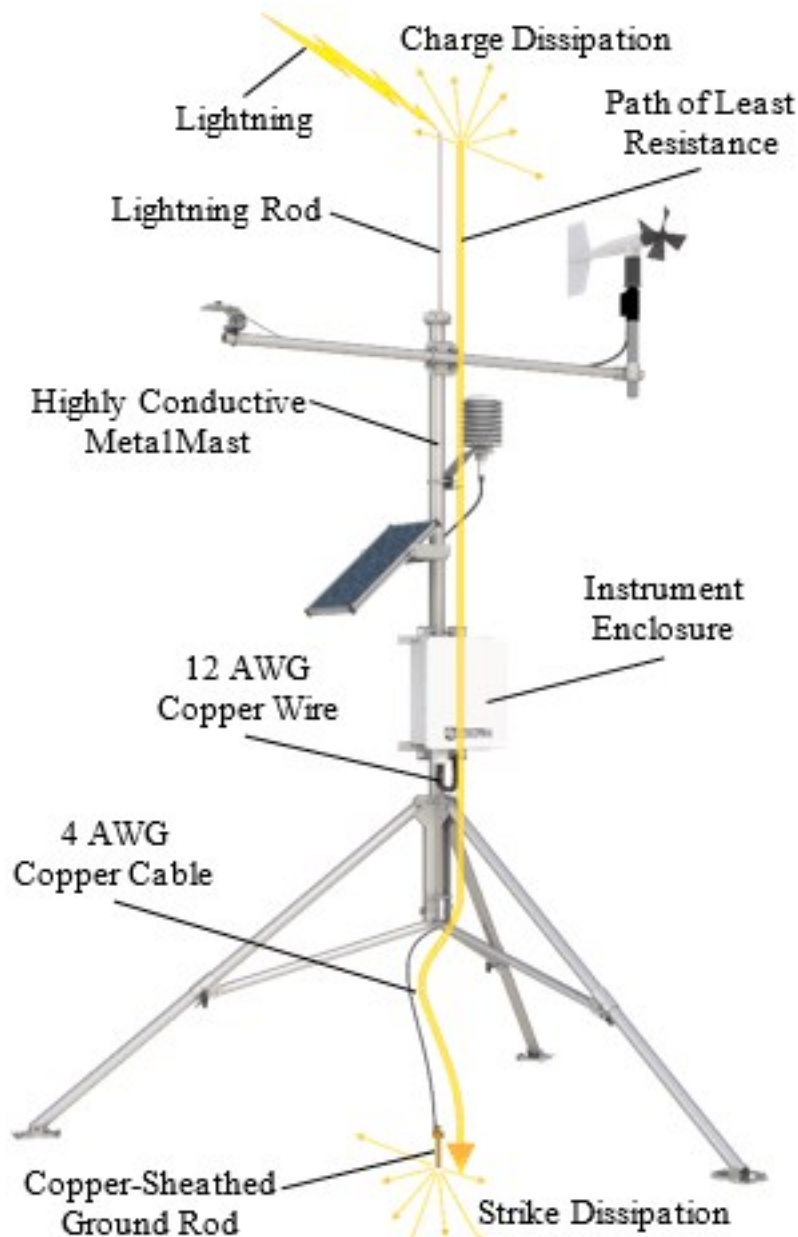
2. Remove the lithium battery by gently prying it out with a small flat-bladed screwdriver.
3. Install new battery.
4. Replace external battery cover.

8.5 Electrostatic discharge and lightning protection

WARNING:

Lightning strikes may damage or destroy the data logger, 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 data logger. This system, however, is not infallible. The following image displays a typical application of the system:



All critical inputs and outputs on the data logger are ESD protected. 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 data logger 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:

- [Power Budget Spreadsheet](#)
- [Video Tutorial - Power Budgeting](#)



See also:

- [Power input](#) (p. 9)
- [Power output](#) (p. 10)
- [Power requirements](#) (p. 162)
- [Power output specifications](#) (p. 163)


8.7 Updating the operating system

Campbell Scientific posts operating system (OS) updates at <https://www.campbellsci.eu/downloads> when they become available. It is recommended that before deploying instruments, you check operating system versions and update them as needed. The data logger 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 data logger memory and resets all settings on the data logger to factory defaults, data loss will certainly occur. Depending on several factors, the data logger may also become incapacitated until the new OS is programmed into memory.

TIP:

It is recommended that you retrieve data from the data logger and back up your programs and settings before updating your OS. To collect data using *LoggerNet*, connect to your data logger and click **Collect Now** . To backup your data logger, 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 data logger

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





1. Download the latest Operating System at <https://www.campbellsci.eu/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 data logger. If connecting via USB for the first time, you must first install USB drivers by using *Device Configuration Utility* (select your data logger, then on the main page, click **Install USB Driver**). Alternately, you can install the USB drivers using EZ Setup.
4. Physically connect your data logger to your computer using a USB cable, then open *Device Configuration Utility* and select your data logger.
5. Select the communications port used to communicate with the data logger 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 data logger.

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

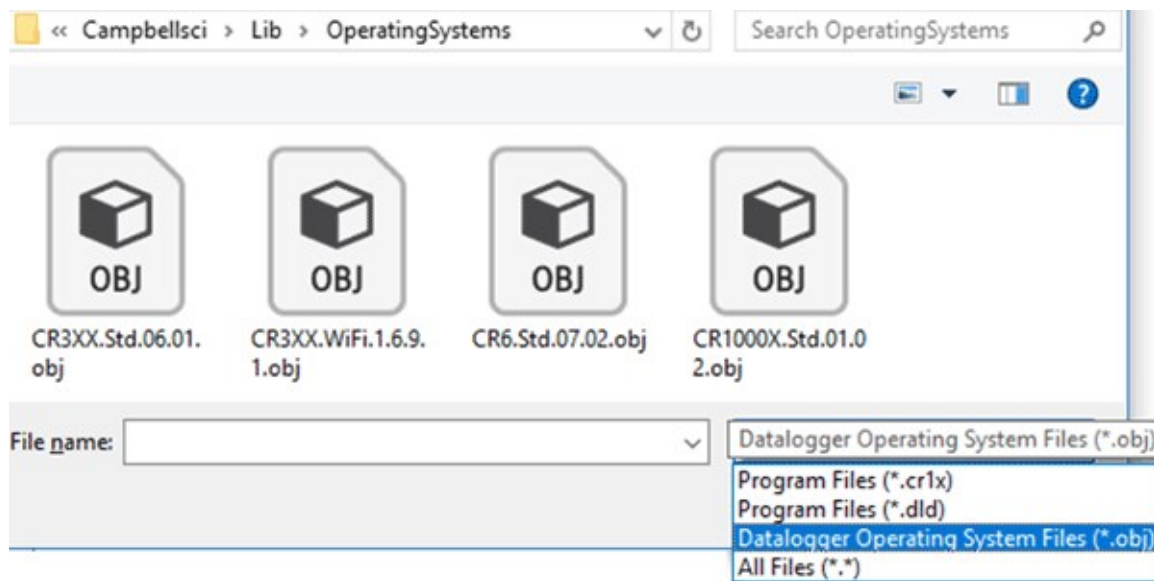


8.7.2 Sending an operating system to a remote data logger

If you have a data logger that is already deployed, you can update the OS over a telecommunications link by sending the OS to the data logger 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 data logger settings, even settings critical to supporting the telecommunication link.

1. Download the latest Operating System at <https://www.campbellsci.eu/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 data logger support software, connect to your data logger.
 - **LoggerNet** users, select **Main** and click **Connect**  on the **LoggerNet** toolbar, select the data logger from the **Stations** list, then 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 data logger.



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.

8.8 File management via powerup.ini

Another way to upload a program, install a data logger 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 data logger using the data logger support software **File Control > Send** command. With the memory card or SC115 connected, or with the **powerup.ini** file saved in the data logger memory, a power cycle to the data logger begins the process chosen in the **powerup.ini** file.

When uploading a program, the following rules determine what data logger 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.

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

Table 8-2: 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.

Table 8-2: Powerup.ini commands

Command	Action	Details
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 data logger operating system.
13	Run always, erase data	Copies a program to a drive and sets the program to both Run Now and Run on Power Up . Data on a memory card from the previously running program will be erased.
14	Run now, erase data	Copies a program to a drive and sets the program to Run Now . Data on a memory card from the previously running program will be erased.
15	Move file	Moves a file, such as an Include or program support file, to the specified drive.

8.8.2 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 data logger 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.CRB,cpu:
```

Example: Run Program on Power Up

```
'Copy program file pwrap.CRB from the external drive to CPU:
'File will run only when the data logger is powered-up later.
2,pwrap.CRB,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,GRANITE9.Std.01.obj
9,GRANITE10.Std.01.obj

Example: Run Program from SC115 Flash Memory Drive

'A program file is carried on an SC115 Flash Memory drive.
'Do not copy program file from SC115
'Run program always, erase data.
13,toobigforcpu.CRB,cs9:

Example: Always Run Program, Erase Data

13,pwrup_1.CRB,cpu:

Example: Run Program Now and Erase Data Now

14,run.CRB,cpu:

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 data logger, 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 data logger. Sometimes the easiest way to resolve a problem is by resetting the data logger (see [Resetting the data logger](#) (p. 126) for more information).

For additional troubleshooting options, see:

9.1 Checking station status	119
9.2 Understanding NAN and INF occurrences	121
9.3 Timekeeping	122
9.4 CRBasic program errors	124
9.5 Resetting the data logger	126

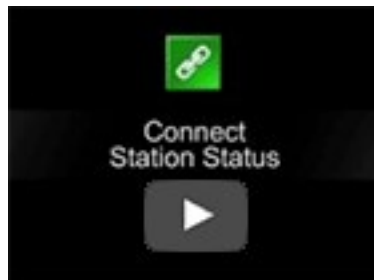
9.6 Troubleshooting power supplies	127
9.7 Ground loops	130
9.8 Field calibration	133
9.9 File system error codes	134
9.10 File name and resource errors	135
9.11 Background calibration errors	135

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

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

9.1 Checking station status

View the condition of the data logger using **Station Status**. Here you see the operating system version of the data logger, 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 data logger support software, access the **Station Status** to view the condition of the data logger.

- From *LoggerNet*: Click **Connect** , then **Station Status**  to view the **Summary** tab.

9.1.2 Watchdog errors

Watchdog errors indicate that the data logger has crashed and reset itself. Experiencing occasional 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 data logger got stuck, and the data logger had to reboot itself to recover. Or communications failures may cause the data logger to reopen the IP connections more than usual. Check your data logger 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.eu/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 data logger 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 data logger 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 data logger attempts to catch out-of-bounds errors at compile time. However, it is not always possible; when these errors occur during runtime the variable-out-of-bounds field increments. 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. 9)
- [Power requirements](#) (p. 162)

9.2 Understanding NAN and INF occurrences

NAN (not a number) and INF (infinite) are data words indicating an exceptional occurrence in data logger function or processing. **INF** indicates that the program has encountered an undefined arithmetic expression, such as $0 \div 0$. **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.

NAN is a constant that can be used in expressions. This is shown in the following example code 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**. However, 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. The following example code discards **NAN** WindSpeed measurements from the Minimum output:

```
Minimum (1, WindSpeed, FP2, WindSpeed=NAN, False)
```

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 -2147483648.

9.3 Timekeeping

Measurement of time is an essential data logger function. Time measurement with the onboard clock enables the data logger 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. 160). An internal lithium battery backs the clock when the data logger is not externally powered (see [Internal battery](#) (p. 107)).

9.3.1 Clock best practices

When setting the clock with *LoggerNet*, initiate it manually during a maintenance period when the data logger is not actively writing to Data Tables. Click **Set** 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 data logger, 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 data logger and subtracts it from any adjustment that it might make.

9.3.2 GPS

When the **GPS()** instruction is present in the CRBasic program, the system time is synchronized to the GPS PPS signal. GPS settings are configured automatically and the GPS output streams can be monitored in the program.

For more information, refer to the CRBasic help and specifications at [GPS](#): (p. 161)

9.3.3 Time stamps

A measurement without an accurate time reference often has little meaning. Data collected from data loggers 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 data logger 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.

9.3.4 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:

$$\text{time skew} = \text{reps} * (\text{settling time} + \text{integration time} + \text{ADC time}) + \text{instruction setup time}$$

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.

Applications that are very sensitive to time skew should consider using measurement modules such as GRANITE SPECTRUM series. These provide dedicated sampling hardware for every channel and very tight time synchronization.

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 data logger such that the time stamp reflects the system time rather than the scan time using the **DataTime()** instruction. See the *CRBasic Editor* help for detailed instruction information and program examples:
<https://help.campbellsci.eu/crbasic/granite10/>,
<https://help.campbellsci.eu/crbasic/granite9/>.

9.4 CRBasic program errors

Analyze data soon after deployment to ensure the data logger 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 data logger itself. This is rare, but reasons may include:

- The data logger 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.eu. When run, the executable file updates the computer compiler. To update the data logger operating system, see [Updating the operating system](#) (p. 111).
- The program has large memory requirements for data tables or variables and the data logger 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 data logger 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 USB 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 data logger, but it can be a useful tool for fine-tuning a program. For more information, see [Information tables and settings \(advanced\)](#) (p. 136).

See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

9.5 Resetting the data logger

A data logger reset is sometimes referred to as a "memory reset." Backing up the current data logger configuration before a reset makes it easy to revert to the old settings. To back up the data logger configuration, connect to the data logger using Device Configuration Utility, and click **Backup > Back Up Datalogger**. To restore a configuration after the data logger has been reset, connect and click **Backup > Restore Datalogger**.

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

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

9.5.1 Processor reset

To reset the processor, simply power cycle the data logger. 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 data logger is remote, a power cycle can be mimicked in a **Terminal Emulator** program (type REBOOT <Enter>).

9.5.2 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 is to always retrieve data from the data logger before sending a program; otherwise, data may be lost. See [Collecting data](#) (p. 37) for detailed instruction.

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, consider [PreserveVariables\(\)](#).

9.5.3 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**.

9.5.4 Formatting drives

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

9.5.5 Full memory reset

Full memory reset occurs when an operating system is sent to the data logger using *Device Configuration Utility* or when entering **98765** in the **Status** table field **FullMemReset** (see [Information tables and settings \(advanced\)](#) (p. 136)). 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. 111) for more information.

9.6 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 data logger 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 data logger powers up and works, troubleshoot the data logger 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. 111).

9.6.1 SDI-12 transparent mode

All SDI-12 probes have just three wires—a signal, ground, and 12 V power line. They are connected to the data logger according to the following table.

Wire function	Data logger connection
Shield	\perp (analogue ground)
Power	12V
Power ground	G

System operators can manually interrogate and enter settings in probes, connected to the data logger, 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 data logger programs may not execute. Data logger security may need to be unlocked before transparent mode can be activated.

Transparent mode is entered while the computer is communicating with the data logger through a terminal emulator program such as through *Device Configuration Utility* or other data logger 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 data logger support software terminal emulator:

```
Deployment | Logger Control | Data Monitor | File Control | Send OS | Settings Editor | Terminal
CR: >
CR: >SDI12
Enter Cx Port 1,2,3 as ?
1
Entering SDI12 Terminal
+
Exit SDI12 Terminal
```

1. Press **Enter** until the data logger responds with the prompt **GRANITE 9/10>**.
2. Type **SDI12** at the prompt and press **Enter**.
3. In response, the query **Select SDI12 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.

9.6.1.1 Watch command (sniffer mode)

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



1. Enter the transparent mode as described previously.
2. Press **Enter** until a **GRANITE 9/10>** prompt appears.
3. Type **W** and then press **Enter**.

4. In response, the query **Select SDI12 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**.
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.

9.6.1.2 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).

9.7 Ground loops

A ground loop is a condition in an electrical system that contains multiple conductive paths for the flow of electrical current between two nodes. Multiple paths are usually associated with the ground or 0 V-potential point of the circuit. Ground loops can result in signal noise, communications errors, or a damaging flow of ground current on long cables. Most often, ground loops do not have drastic negative effects and may be unavoidable. Special cases exist where additional grounding helps shield noise from sensitive signals; however, in these cases, multiple ground conductors are usually run tightly in parallel without conductive shielding material placed between the parallel grounds. If possible, ground loops should be avoided. When problems arise in a system, ground loops may be the source of the problems.

See also [Grounds](#) (p. 11).

9.7.1 Common causes

Some of the common causes of ground loops include the following:

- The drain wire of a shielded cable is connected to the local ground at both ends, and the ground is already being carried by a conductor inside the cable. In this case, two wires, one on either side of the cable shield, are connected to the ground nodes at both ends of the cable.
- A long cable connects the grounds of two electrical devices, and the mounting structure or grounding rod also directly connects the grounds of each device to the local earth ground. The two paths, in this case, are the connecting cable and earth itself.
- When electrical devices are connected to a common metal chassis such as an instrument tower, the structure can create a ground path in parallel to the ground wires in sensor cables running over the structure.
- Conductors connected to ground are found in most cables that connect to a data logger. These include sensors cables, communication cables, and power cables. Any time one of these cables connects to the same two endpoints as another cable, a ground loop is formed.

9.7.2 Detrimental effects

The harm from a ground loop can be seen in different ways. One consideration is the electromagnetically induced effect. This will manifest as an AC noise or pulse. As seen in [FIGURE 9-1](#) (p. 131) the parallel conductive paths form an electrical loop that acts as an antenna to pick up electromagnetic energy.

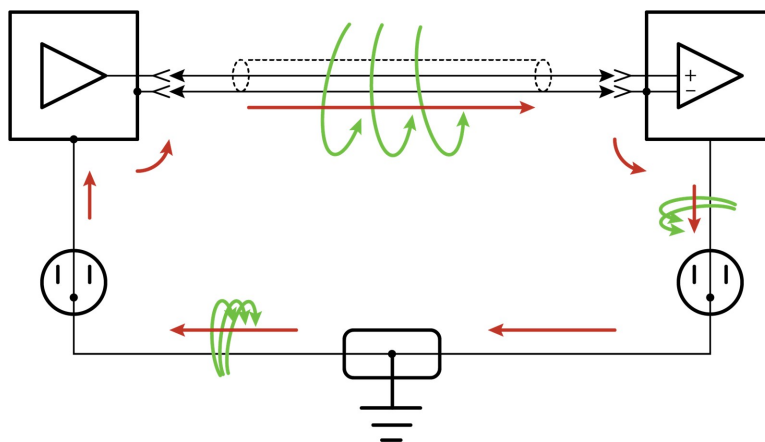


FIGURE 9-1. Stray AC magnetic fields picked up in loop antenna

- Relatively small electromagnetic energy: This could come from AC current on a nearby power cable, or RF energy transmitting through the air, and can cause electrical noise that disrupts digital communications.
- Larger electromagnetic energy: The antenna loop scenario can have a more damaging effect when a large current is discharged nearby. The creation of an electromagnetic pulse can induce a surge that damages attached electronic devices.

Another way ground loops affect a system is by allowing ground current to flow between devices. This can be either a DC or AC effect. For various reasons, the voltage potential between two different points on the surface of the earth is not always 0 V. Therefore, when two electrical devices are both connected to a local earth ground, there may exist a voltage difference between the two devices. When a cable is connected between the two devices at different voltages, physics dictates that an electrical current must flow between the two points through the cable. See [FIGURE 9-2](#) (p. 132).

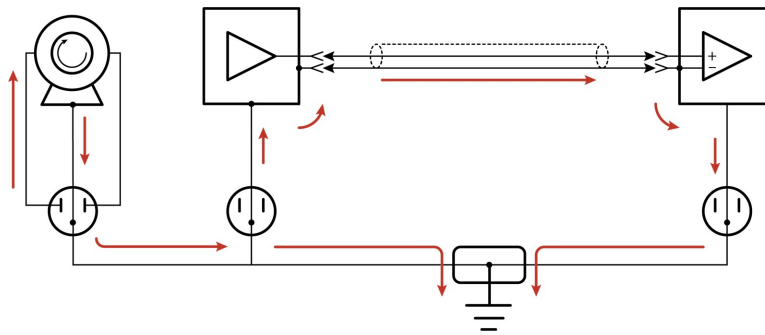


FIGURE 9-2. Leakage current (AC or DC) from nearby load

- For digital communications, an offset in the ground voltage reduces the dynamic range of the digital signals. This makes them more susceptible to noise corruption. If the ground voltage changes by one volt or more, the digital communications could stop working because the signals no longer reach the thresholds for determining the state of each bit.
- If the ground voltage differences reach several volts, damaging effects may occur at the terminals of the electronics devices. Damage occurs when the maximum allowable voltage on the internal components is exceeded.

9.7.3 Severing a ground loop

To avoid or eliminate ground loops, when they are detected, requires severing the loop. Suggestions for severing ground loops include:

- Connect the shield wire of a signal cable to ground only at one end of the cable. Leave the other end floating (not connected to ground).
- Never intentionally use the shield (or drain wire) of a cable as a signal ground or power ground.
- Use the mechanical support structure only as a connection for the safety ground (usually the ground lug). Do not intentionally return power ground through the structure.
- Do not use shielded Cat5e cables for Ethernet, CPI or EPI communications.
- For long distance communication protocols such as RS-485, RS-422, and CAN, use the RG terminal for the ground connection. The RG terminal has a 100-ohm resistor in series with ground to limit the amount of DC current that can flow between the two endpoints while keeping the common-mode voltage in range of the transceivers. The transceivers themselves have enhanced voltage range inputs allowing for ground voltage differences of up to 7 V between endpoints.
- For exceptional cases, use optical or galvanic isolation devices to provide a signal connection without any accompanying ground connection. These should be used only when ground loops are causing system problems and the other methods of breaking a ground loop don't apply. These devices add expense and tend to consume large amounts of power.

9.8 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. See the *CRBasic Editor* help for detailed instruction information and program examples: <https://help.campbellsci.eu/crbasic/granite10/>, <https://help.campbellsci.eu/crbasic/granite9/>.

9.9 File system error codes

Errors can occur when attempting to access files on any of the available drives. All occurrences are rare, but they are most likely to occur when using optional memory cards. Often, formatting the drive will resolve the error. The errors display in the **File Control** messages box or in the **CardStatus** field of the **Status** table. See [Information tables and settings \(advanced\)](#) (p. 136) for more information.

- 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

9.10 File name and resource errors


The maximum file name size that can be stored, run as a program, or FTP transferred in the data logger 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.

9.11 Background calibration errors

A system slow sequence runs at a one-minute interval to measure the CPU temperature, external battery voltage, and lithium battery voltage. When GRANITE Measurement Modules or CDMs are connected, this background task triggers the external modules to run their independent calibrations.

10. 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 GRANITE 9/10 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 data logger to your computer (see [Making the software connection](#) (p. 31) 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 data logger to your computer (see [Making the software connection](#) (p. 31) for more information).
 - **LoggerNet** users, select **DataTableInfo** from the **Table Monitor** list.
- **Settings:** Settings can be accessed from the **LoggerNet** Connect Screen **Datalogger** > **Settings Editor**, or using **Device Configuration Utility Settings Editor** tab. 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 data logger terminal mode (from **Device Configuration Utility**, click the **Terminal** tab) using command "F".
- Status, DataTableInfo and Settings values may be accessed programmatically using **Tablename.FieldName** syntax. For example: `Variable = Settings.FieldName`. For more information see: <https://www.campbellsci.eu/blog/programmatically-access-stored-data-values>.

Communications and processor bandwidth are consumed when generating the **Status** and other information tables. If data logger 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 data logger program to recompile, which may cause loss of data. Before changing settings, it is a good practice to collect your data (see [Collecting data](#) (p. 37) for more information). Examples of settings that force the data logger 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
- PakBus encryption key
- PakBus/TCP server port
- HTTP service port
- FTP service port
- PakBus/TCP service port
- PakBus/TCP client connections
- Communications allocation

10.1 DataTableInfo table system information

The **DataTableInfo** table is automatically created when a program produces other data tables. View the **DataTableInfo** table by connecting the data logger to your computer (see [Making the software connection](#) (p. 31) for more information).

Most fields in the **DataTableInfo** table are **read only** and of a **numeric data type** unless noted. Error counters (for example **SkippedRecord**) may be reset to **0** for troubleshooting purposes.

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

10.1.1 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.

10.1.2 DataRecordSize

Reports the number of records allocated to a data table.

10.1.3 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

10.1.4 RecNum

Record number is incremented when any one of the **DataTableInfo** fields change, for example **SkippedRecord**.

10.1.5 SecsPerRecord

Reports the data output interval for a data table.

10.1.6 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.

10.1.7 TimeStamp

Scan time that a record was generated.

- NSEC data type

10.2 Status table system information

The **Status** table is an automatically created data table. View the **Status** table by connecting the data logger to your computer (see [Making the software connection](#) (p. 31) for more information).

Most fields in the **Status** table are **read only** and of a **numeric data type** unless noted. Error counters (for example, **WatchdogErrors** or **SkippedScan**) may be reset to **0** for troubleshooting purposes.

Status table values may be accessed programatically using `SetStatus()` or `TableName.FieldName` syntax. For example: `Variable = Status.FieldName`. For more information see: <https://www.campbellsci.eu/blog/programmatically-access-stored-data-values>.

10.2.1 Battery

Voltage (VDC) of the battery powering the system. Updates once per minute, when viewing the **Status** table, or programatically.

10.2.2 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.

10.2.3 CardStatus

Contains a string with the most recent status information for the removable memory card.

10.2.4 CommsMemFree

Memory allocations for communications. Numbers outside of parentheses reflect current memory allocation. Numbers inside parentheses reflect the lowest memory size reached.

10.2.5 CompileResults

Contains messages generated at compilation or during runtime. Updated after compile and for runtime errors such as variable out of bounds.

- String data type

10.2.6 ErrorCalib

Number of erroneous calibration values measured. Erroneous values are discarded. Updated at startup.

10.2.7 FullMemReset

Enter **98765** to start a full-memory reset, all data and programs will be erased.

10.2.8 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](#).

10.2.9 LithiumBattery

Voltage of the internal lithium battery. Updated at GRANITE 9/10 power up. For battery information, see [Internal battery](#) (p. 107).

10.2.10 Low12VCount

Counts the number of times the primary GRANITE 9/10 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.

10.2.11 MaxBuffDepth

Maximum number of buffers the GRANITE 9/10 will use to process lagged measurements. Enter **0** to reset.

10.2.12 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.

10.2.13 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

10.2.14 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.

10.2.15 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.

10.2.16 MemoryFree

Unallocated MemoryFree(1) is the battery backed SRAM. MemoryFree(2) is the DDR SDRAM. 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-data memory, may be created. Updated after compile completes.

10.2.17 MemorySize

Total final-data memory size (bytes) in the GRANITE 9/10. Updated at startup. MemorySize(1) is the battery backed SRAM. MemorySize(2) is the DDR SDRAM.

10.2.18 Messages

Contains a string of manually entered messages.

10.2.19 OSDate

Release date of the operating system in the format mm/dd/yyyy. Updated at startup.

- String data type

10.2.20 OSSignature

Signature of the operating system.

10.2.21 OSVersion

Version of the operating system in the GRANITE 9/10. Updated at OS startup.

- String data type

10.2.22 PakBusRoutes

Lists routes or router neighbours known to the data logger at the time the setting was read. Each route is represented by four components separated by commas and enclosed in parentheses:(port, via neighbour address, pakbus address, response time in ms). Updates when routes are added or deleted.

- String data type

10.2.23 CPUTemp

Current processor board temperature (°C). Updates once per minute, when viewing the **Status** table, or programatically.

10.2.24 PortConfig

Provides information on the configuration settings (input, output, SDM, RS-485, SDI-12, COM port) for **C** terminals in numeric order of terminals. Default = **Input**. Updates when the port configuration changes.

- String data type

10.2.25 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. Enter **-1** to set to **true**. Enter **0** to set to **false**.

- Boolean data type

10.2.26 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.

10.2.27 ProgErrors

Number of compile or runtime errors for the running program. Updated after compile.

10.2.28 ProgName

Name of current (running) program; updates at startup.

- String data type

10.2.29 ProgSignature

Signature of the running CRBasic program including comments. Does not change with operating-system changes. Updates after compiling the program.

10.2.30 RecNum

Record number increments when the Status Table is requested by support software. Range = 0 to 2^{32} .

- Long data type

10.2.31 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

10.2.32 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.

10.2.33 SerialNumber

GRANITE 9/10 serial number assigned by the factory when the data logger was calibrated. Stored in flash memory. Updated at startup.

10.2.34 SkippedScan

Number of skipped program scans (see [Checking station status](#) (p. 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. Enter **0** to reset.

10.2.35 SkippedSystemScan

Number of scans skipped in the background calibration. Enter **0** to reset. See [LastSystemScan](#), [MaxSystemProcTime](#), and [SystemProcTime](#).

10.2.36 StartTime

Time (date and time) the CRBasic program started. Updates at beginning of program compile.

- NSEC data type

10.2.37 StartUpCode

Indicates how the running program was compiled. Updated at startup. 65 = Run on powerup is running and normal powerup occurred.

10.2.38 StationName

Station name stored in flash memory. This is not the same name as that is entered into your data logger 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 data logger is currently running a program with a [CardOut\(\)](#) instruction.

- String data type

10.2.39 SW12Volts

Status of switched, 12 VDC terminal. On/high (**true**) or off/low (**false**) Enter **-1** to set to **true**. Enter **0** to set to **false**. Updates when the state changes.

- Boolean data type

10.2.40 SystemProcTime

Time (μs) required to process auto (background) calibration. Default is 0 until background calibration runs.

10.2.41 TimeStamp

Scan-time that a record was generated.

- NSEC data type

10.2.42 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 runtime when the error occurs. Enter **0** to reset.

10.2.43 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.

10.2.44 WiFiUpdateReq

Shows if WiFi operating system update is available. Update available (**true**) or not (**false**). Updates when state changes.

- Boolean data type

10.3 CPIStatus system information

The **CPIStatusA** and **CPIStatusB** tables are automatically created when a program uses one of the CPI busses. View the **CPIStatus** table by connecting the data logger to your computer (see [Making the software connection](#) (p. 31) for more information).

Most fields in the **CPIStatus** table are **read/write** and of a **numeric data type** unless noted. Error counters (for example **BuffErr**) may be reset to **0** for troubleshooting purposes.

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

For more information on the CPI bus and how to design a CDM network, see the technical paper at: <https://s.campbellsci.eu/documents/us/technical-papers/cpi-bus.pdf>

10.3.1 BusLoad

Percentage of the possible CPI network bandwidth use over the scan interval. $\text{BusLoad} = \text{Used capacity} / \text{Maximum capacity}$.

- Read only
- Percentage (0.000 to 100)

TIP:

Use **CPISpeed(C)** to change the CPI bit rate. The default bit rate is 250 kbps. Use a higher bit rate if the **BusLoad** exceeds 75 percent.

10.3.2 ModuleReportCount

Reports the number of times measurement modules report in to the CPI bus. Modules report in on program send or when settings in the CPIStatus table are edited remotely. Activity that could cause the number of modules to be reported differently will cause ModuleReportCount to increment. Also, if there are devices on the network that are connected but not active, (such as those not in the running program) they will report in once minute, advertising their presence, and incrementing ModuleReportCount.

10.3.3 ActiveModules

Reports the number of measurement modules that are active on the CPI bus.

- Read only

10.3.4 BuffErr (buffer error)

Reports how many times there is an error in the buffer. Enter **0** to reset.

10.3.5 RxErrMax

Reports the maximum number of receive errors. Enter **0** to reset.

10.3.6 TxErrMax

Reports the maximum number of transmit errors. Enter **0** to reset.

10.3.7 FrameErr (frame errors)

Reports how many times a frame has an error. Enter **0** to reset.

10.3.8 ModuleInfo array

Reports: CDM Type, Serial Number, Device Name, CPI Address, Activity, OS Version.

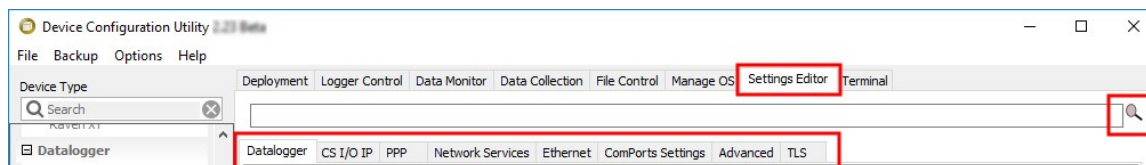
- String data type
- Read only

Possible responses and meanings in the **Activity** field are below:

- **Active**: The module is connected to the CPI bus and is making measurements according to the data logger program.
- **Offline**: The module was present after startup but is no longer responding.
- **Unused**: The module is or was connected and powered but is not included in the data logger program.
- **Wait Config**: The module has not yet responded to a data logger attempts to configure it.
- **Config Fail**: The module could not be configured. A configuration error message is appended to this response.
- **CAN Errors, resetting CPI**: The CDM module is not used in the data logger program.

10.4 Settings

Settings can be accessed from the *LoggerNet* Connect Screen **Datalogger** > **Setting Editor**, or using *Device Configuration Utility* **Settings Editor** tab. **Settings** are organized in tabs and can be searched for.



Most **Settings** are **read/write** and of a **numeric data type** unless noted.

Settings may be accessed programmatically using `SetSetting()` or `TabName.FieldName` syntax. For example: `Variable = Settings.FieldName`. For more information see: <https://www.campbellsci.eu/blog/programmatically-access-stored-data-values>.

NOTE:

A list of **Settings** fieldnames is also available from the data logger terminal mode using command **F**.

10.4.1 Baudrate

This setting governs the baud rate that the data logger 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) support auto-baud synchronization while the other ports support only fixed baud. With auto-baud synchronization, the data logger 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.

10.4.2 Beacon

This setting, in units of seconds, governs the rate at which the data logger will broadcast PakBus messages on the associated port in order to discover any new PakBus neighbouring nodes. If this setting value is set to a value of 0 or 65,535, the data logger 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).

10.4.3 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 data logger will be configured as a Branch Router meaning that it will not be required to keep track of neighbours of any routers except those in its own branch. Configured in this fashion, the data logger will ignore any neighbour 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

10.4.4 CommsMemAlloc

Replaces **PakBusNodes**.

10.4.5 DNS

This setting specifies the addresses of up to two domain name servers that the data logger 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

10.4.6 EthernetInfo

Reports the IP address, network mask, and default gateway for each of the data logger'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

10.4.7 EthernetPower

This setting specifies how the data logger controls power to its Ethernet interface. This setting provides a means of reducing the data logger power consumption while Ethernet is not connected. Always on, 1 Minute, or Disable.

10.4.8 FilesManager

This setting controls how the data logger 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

10.4.9 FTPEnabled

Set to **1** if to enable FTP service. Default is **0**.

10.4.10 FTPPassword

Specifies the password that is used to log in to the FTP server.

- String data type

10.4.11 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**.

10.4.12 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

10.4.13 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.

10.4.14 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.

10.4.15 HTTPSEnabled

Set to 1 to enable the HTTPS (secure web server) service.

10.4.16 HTTPSPort

Configures the TCP port on which the HTTPS (secure 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.

10.4.17 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:) on which the file is located. The extension of the file must also be valid for a data logger program (.CRB, .DLD).

See also [File management via powerup.ini](#) (p. 114).

- String data type

10.4.18 IPAddressEth

- String data type

10.4.19 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 data logger will

configure the effective value of this setting using DHCP. This setting is the equivalent to the `IPGateway` status table variable.

- String data type

10.4.20 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 data logger will configure the effective value of this setting using DHCP.

- String data type

10.4.21 IPTrace

Discontinued; aliased to `IPTraceComport`

10.4.22 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.

10.4.23 IPTraceComport

Specifies the port (if any) on which TCP/IP trace information is sent. Information type is controlled by `IPTraceCode`.

10.4.24 IsRouter

This setting controls whether the data logger is configured as a router or as a leaf node. If the value of this setting is **true**, the data logger 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 data logger configured as a router will maintain its own list of neighbours and send this list to other routers in the PakBus network. It will also obtain and receive neighbour lists from other routers.

If the value of this setting is **false**, the data logger will be configured to act as a leaf node. In this configuration, the data logger will not be able to forward packets from one port to another and it will not maintain a list of neighbours. Under this configuration, the data logger can still communicate with other data loggers and wireless sensors. It cannot, however, be used as a means of reaching those other data loggers. The default value is false.

- Boolean data type

10.4.25 MaxPacketSize

Specifies the maximum number of bytes per data collection packet.

10.4.26 Neighbours

This setting specifies, for a given port, the explicit list of PakBus node addresses that the data logger will accept as neighbours. If the list is empty (the default value) any node will be accepted as a neighbour. This setting will not affect the acceptance of a neighbour if that neighbour's address is greater than 3999.

- String data type

10.4.27 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 data logger network. Duplication of PakBus addresses can lead to failures and unpredictable behaviour in the PakBus network.

When a device has an allowed neighbour 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 neighbour list.

10.4.28 PakBusEncryptionKey

This setting specifies text that will be used to generate the key for encrypting PakBus messages sent or received by this data logger. If this value is specified as an empty string, the data logger will not use PakBus encryption. If this value is specified as a non-empty string, however, the data logger will not respond to any PakBus message unless that message has been encrypted.

- String data type

10.4.29 PakBusNodes

Discontinued; aliased to **CommsMemAlloc**

10.4.30 PakBusPort

This setting specifies the TCP service port for PakBus communications with the data logger. Unless firewall issues exist, this setting probably does not need to be changed from its default value. Default 6785.

10.4.31 PakBusTCPClients

This setting specifies outgoing PakBus/TCP connections that the data logger should maintain. Up to four addresses can be specified.

- String data type

10.4.32 PakBusTCPEnabled

By default, PakBus TCP communications are enabled. To disable PakBus TCP communications, set the PakBusPort setting to **65535**.

10.4.33 PakBusTCPPassword

This setting specifies a password that, if not empty, will make the data logger authenticate any incoming or outgoing PakBus/TCP connection. This type of authentication is similar to that used by CRAM-MD5.

- String data type

10.4.34 PingEnabled

Set to one to enable the ICMP ping service.

10.4.35 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 data logger 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 data logger will add a delay of one second for each additional semicolon.

If a value of PPP is specified for this setting, will configure the data logger to act as a PPP client without any modem dialing. Finally, an empty string (the default) will configure the data logger to listen for incoming PPP connections also without any modem dialing.

- String data type

10.4.36 pppDialResponse

Specifies the response expected after dialing a modem before a PPP connection can be established.

- String data type

10.4.37 pppInfo

Reports the IP address, network mask, and default gateway for each of the data logger'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

10.4.38 pppInterface

This setting controls which data logger port PPP service will be configured to use.

10.4.39 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

10.4.40 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

10.4.41 pppUsername

Specifies the user name that is used to log in to the PPP server.

- String data type

10.4.42 RouteFilters

This setting configures the data logger to restrict routing or processing of some PakBus message types so that a "state changing" message can only be processed or forwarded by this data logger 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 data logger 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

10.4.43 RS232Power

Controls whether the RS-232 port will remain active even when communications are not taking place.

- Boolean data type

10.4.44 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 [Data logger security](#) (p. 101).

10.4.45 ServicesEnabled

Discontinued; replaced by/aliased to **HTTPEnabled**, **PingEnabled**, **TelnetEnabled**.

10.4.46 TCPClientConnections

Discontinued; replaced by / aliased to **PakBusTCPClients**.

10.4.47 TCPPort

Discontinued; replaced by / aliased to **PakBusPort**.

10.4.48 TelnetEnabled

Enables (1) or disables (0) the Telnet service.

10.4.49 TLSConnections (Max TLS Server Connections)

This setting controls the number of concurrent TLS (secure or encrypted) client socket connections that the data logger 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 data logger program.

This setting will control the amount of RAM that the data logger 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 data logger to recompile its program so that it can reallocate memory

10.4.50 TLSPassword

This setting specifies the password that will be used to decrypt the TLS Private Key setting.

- String data type

10.4.51 TLSStatus

Reports the current status of the data logger TLS network stack.

- String data type
- Read only

10.4.52 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 data logger will fail to respond to the broadcast requests.

Default = 0.

10.4.53 UTCOffset

Specifies the offset, in seconds, of the data logger'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.

10.4.54 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 data logger will attempt to start a hello transaction with a neighbour if no other communications have taken place within the interval.

10.4.55 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.

Where to find:

- All settings: **Settings Editor** tab in *Device Configuration Utility*: **Wi-Fi** tab.
- Key settings: in *Device Configuration Utility*: **Deployment** > **Wi-Fi** tab.

See also [Wi-Fi communications](#) (p. 26)

NOTE:

A list of **Settings** fieldnames is also available from the data logger terminal mode using command **F**.

10.4.55.1 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.

- String data type

10.4.55.2 IPGatewayWiFi

Specifies the address of the IP router to which the data logger will forward all non-local IP packets for which it has no route.

- String data type

10.4.55.3 IPMaskWiFi

Specifies the subnet mask for the WiFi interface.

- String data type

10.4.55.4 WiFiChannel

This setting is only applicable when the device is configured to create a network (**Configuration**). It then specifies in which channel the network should be created. If **Auto** is selected, the device will use only channels 1, 6, and 11 to minimize 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

10.4.55.5 WiFiConfig

In addition to joining, creating or disabling a network, this setting controls how the device acts when the WIFI **Mode** button is pressed.

When the **Mode** button is used to temporarily create the Wi-Fi network, it will stay powered for at least 5 minutes. There is a 5 minute timeout that is refreshed every time communications are detected. Once the timeout has expired the device will power off the Wi-Fi network. If the Mode button is pressed again while the temporary network is active, the device will power off the network.

- Long data type, where:
 - 0 = Join network
 - 1 = Create network
 - 4 = Disable network
 - 5 = Normally off; join network on button press
 - 6 = Normally off; create network on button press
 - 7 = Join network; create network on button press

10.4.55.6 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

10.4.55.7 WiFiEAPPassword

If joining an Enterprise Security-enabled network enter password here.

- String data type

10.4.55.8 WiFiEAPUser

If joining an Enterprise Security-enabled network enter user name here.

- String data type

10.4.55.9 Networks

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

10.4.55.10 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

10.4.55.11 WiFiFwdCode (Forward Code)

This is an advanced setting to allow some customized filtering by the Wi-Fi module. It specifies which incoming packets are forwarded to the data logger. This decreases the amount of processing required by the data logger.

- Long data type

10.4.55.12 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

10.4.55.13 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 **Configuration** option.

- Long data type

10.4.55.14 WiFiSSID (Network Name)

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, including hidden networks, to join here. If the device is configured to **Create a Network** using the **Configuration** option, then the SSID entered here will be the SSID of the network created.

NOTE:

When creating a network, the device will not create the network if the Network Name (SSID) specified is the same as one that already exists in the area.

- String data type

10.4.55.15 WiFiStatus

Specifies the current status of the Wi-Fi module.

- String data type
- Read only

10.4.55.16 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

10.4.55.17 WLANDomainName

The **WLAN Domain Name** is only relevant when the device is configured to create a network. When attempting to communicate with the device, attached Wi-Fi client devices can simply use the domain name specified here which will be resolved to the device's IP address. For example, the data logger web page can be accessed simply by entering the domain name specified here into a web browser.

- String data type

11. GRANITE 9/10 Specifications

Electrical specifications are valid over a -40 to +70 °C, non-condensing environment, unless otherwise specified. Extended electrical specifications (noted as XD 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.

11.1 System specifications	160
11.2 Physical specifications	162
11.3 Power requirements	162
11.4 Power output specifications	163
11.5 Pulse measurement specifications	164
11.6 Digital input/output specifications	165
11.7 Communications specifications	167
11.8 Standards compliance specifications	169

11.1 System specifications

Processor: NXP iMX6 Quad core running at 1 GHz

Memory (see [Data memory](#) (p. 48) for more information):

- 2 GB DDR SDRAM
- 8 GB eMMC NAND OS storage
- 128 MB NOR FLASH
- 4 MB SRAM battery backed
- Data storage expansion: Removable microSD flash memory, up to 16 GB
- USB host provides for portable data storage on a mass storage device (MSD) formatted as FAT32. Not intended for long term unattended data storage other than what is available with [TableFile\(\)](#).

GRANITE 9 Solid State Drive (SSD):

- **SSD:** Enhanced MLC
- **SSD (XD):** SLC

- **Total onboard:** 64 GB
- **Humidity:** 8% to 95%, non-condensing
- **JESD219A client work load:** 86 terabytes written (TBW) (standard)
- **Random write:** 914 TBW (XD)
- **Sequential write:** 5333 TBW (XD)
- **Block PE cycle:** 100000 (XD)
- **Data Retention at 40 °C:** 10 years with 10% PE cycle (XD)
- **MTBF (hours) at 25 °C:** 1,500,000 (standard); 2,000,000 (XD)
- **Typical power consumption at 12 VDC:** 120.8 mA (standard); 191.7 mA (XD)
- **Maximum sustained write power consumption at 12 VDC:** 295.8 mA (XD only)

GRANITE 10 Solid State Drive (SSD):

- **SSD:** Enhanced MLC
- **SSD (XD):** SLC
- **Total onboard:** 128 GB
- **Humidity:** 8% to 95%, non-condensing
- **JESD219A client work load:** 172 86 terabytes written (TBW) (standard)
- **Random write:** 1828 TBW (XD)
- **Sequential write:** 10666 TBW (XD)
- **Block PE cycle:** 100000 (XD)
- **Data Retention at 40 °C:** 10 years with 10% PE cycle (XD)
- **MTBF (hours) at 25 °C:** 1,500,000 (standard); 2,000,000 (XD)
- **Typical power consumption at 12 VDC:** 175 mA (standard version); 212.5 mA (XD)
- **Maximum sustained write power consumption at 12 VDC:** 316.7 mA (XD only)

Real-Time Clock:

- Battery backed while external power is disconnected
- **Resolution:** 1 ms
- **Accuracy:** ±3 min. per year
- **GPS Phase Lock** to within 200 nS if used

GPS:

- SMA Female 50 Ω input impedance
- Active antenna design, 3.3 Vdc
- 25 dBm maximum input
- Integrated SAW filtering and jam resistance
- 1 S time-to-fix during normal operation
- 35 S time-to-fix on power up or reboot
- 13 min. for leap second, once per day auto

- PPS $\pm 1 \mu\text{s}$ to full UTC second
- Receive sensitivity -161 dBm

Wiring Panel Temperature: Measured using a thermistor, located on the main processor board.

11.2 Physical specifications

Case Material: Stainless Steel 304 and Aluminium 6061

GRANITE 10

Dimensions: 21.4 x 12.0 x 7.5 cm (8.4 x 4.7 x 3.0 in); additional clearance required for cables, wires, and antennas.

Weight/Mass: 1.2 kg (2.7 lb)

GRANITE 9

Dimensions: 21.4 x 12.0 x 5.0 cm (8.4 x 4.7 x 2.0 in); additional clearance required for cables, wires, and antennas.

Weight/Mass: 1.0 kg (2.2 lb)

11.3 Power requirements

Protection: Power inputs are protected against surge, over-voltage, over-current, and reverse power. IEC 61000-4 Class 4 level.

Power In Terminal:

- **Voltage Input:** 9.6 to 32 VDC
- **Input Current Limit at 12 VDC:**
 - Total system current is fused at 5 A with replaceable automotive mini-blade fuse
- 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 **Power** 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.

Internal Lithium Battery: 1/2AA, 1.2 Ah, 3.6 VDC (Tadiran L5902S) for battery-backed memory and clock. 5-year life with no external power source. See also [Internal battery](#) (p. 107).

Average Current Drain:

- **Active:** ~6 Watts
 - 24 V input: 255 mA input
 - 12 V input: 495 mA input

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 start-up.

Wi-Fi Additional Current Contribution at 12 VDC:

Mode	Wi-Fi Option
Client Mode	7 mA idle, 70 mA communicating
Access Point Mode	62 mA idle, 70 mA communicating
Sleep (use <code>IPNetPower()</code> or DevConfig setting to disable)	<1 mA

11.4 Power output specifications

11.4.1 System power out limits (when powered with 12 VDC)

Total system current is fused at 5 A with replaceable automotive mini-blade fuse

11.4.2 12 V and SW12 power output terminals

12V, SW12-1, and SW12-2: Provide 12 VDC power $\pm 10\%$ when the power input supply voltage is ≥ 13.7 VDC. When the supply voltage is < 13.7 V the output voltage will be at least the supply voltage minus 1.7 volts.

SW12-1 and SW12-2 can be independently set to a regulated 12 V under program control.

SW12 current limit: 1100 mA

12 VDC outputs limited to 3300 mA, which is shared by all 12 V outputs including 12V, SW12-1, SW12-2 and CS I/O pin 8.

11.4.3 5 V fixed output

5V: One regulated 5 V output. Supply is shared between the 5V terminal and CS I/O pin 1.

- **Voltage Output:** Regulated 5 V output ($\pm 5\%$)
- **Current Limit:** 250 mA

11.4.4 C as power output

Operating at the current limit is OK if voltage fluctuation can be tolerated. Drive capacity is determined by the logic level of the VDC supply and the output resistance (R_o) of the C terminal. It is expressed as: $V_o = 5\text{ V} - (R_o \cdot I_o)$, where V_o is the drive limit, and I_o is the current required by the external device. For example: at the maximum current limit of 20 mA on C1 the voltage level would reduce from 5 V to 3.5 V.

- C Terminals:
 - Output Resistance (R_o): 150 Ω
 - 5 V Logic Level Drive Capacity: 10 mA @ 3.5 VDC; $V_o = 5\text{ V} - (150\ \Omega \cdot I_o)$
 - 3.3 V Logic Level Drive Capacity: 10 mA @ 1.8 VDC; $V_o = 3.3\text{ V} - (150\ \Omega \cdot I_o)$

11.4.5 CS I/O pin 1

5 V Current Limit: 250 mA

11.4.6 CS I/O pin 8

12 V Current Limit: 1100 mA

11.5 Pulse measurement specifications

NOTE:

Conflicts can occur when a control port pair is used for different instructions (`TimerInput()`, `PulseCount()`, `SDI12Recorder()`, `WaitDigTrig()`). For example, if C1 is used for `SDI12Recorder()`, C2 cannot be used for `TimerInput()`, `PulseCount()`, or `WaitDigTrig()`.

Maximum Input Voltage: ± 20 VDC

Maximum Counts Per Channel: 2^{32}

Maximum Counts Per Scan: 2^{32}

Input Resistance: 5 k Ω

Accuracy: $\pm(6\text{ ppm of reading} + 0.00001)$

11.5.1 Switch closure input

Terminals: C1-C8

Pull-Down Resistance: Configurable in terminal pairs with 100 k Ω

Pull-Up Resistance: Configurable in terminal pairs with 100 k Ω (weak) or 2.2 k Ω (strong)

Maximum Input Frequency: 250 Hz

Minimum Switch Closed Time: 1 ms

Minimum Switch Open Time: 1 ms

Maximum Bounce Time: 1 ms open without being counted

Software Debounce Time: 1 ms

11.5.2 High-frequency input

Terminals: C1-C8

Pull-Down Resistance: Configurable in terminal pairs with 100 k Ω

Pull-Up Resistance: Configurable in terminal pairs with 100 k Ω (weak) or 2.2 k Ω (strong)

Maximum Input Frequency: 1 MHz

11.5.3 Low-level AC input

DC-offset rejection: Internal AC coupling eliminates DC-offset voltages up to ± 0.05 VDC

Input Hysteresis: 12 mV at 1 Hz

Low-Level AC Pulse Input Ranges:

Sine wave (mV RMS)	Range (Hz)
20	1.0 to 20
200	0.5 to 200
2000	0.3 to 10,000
5000	0.3 to 20,000

11.6 Digital input/output specifications

Terminals configurable for digital input and output (I/O) including status high/low, pulse width modulation, external interrupt, edge timing, switch closure pulse counting, high-frequency pulse counting, UART, RS-232, RS-422, 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.

NOTE:

Conflicts can occur when a control port pair is used for different instructions (`TimerInput()`, `PulseCount()`, `SDI12Recorder()`, `WaitDigTrig()`). For example, if C1 is used for `SDI12Recorder()`, C2 cannot be used for `TimerInput()`, `PulseCount()`, or `WaitDigTrig()`.

Terminals: C1-C8

Maximum Input Voltage: ± 20 V

Logic Levels and Drive Current:

Terminal Pair Configuration	5 V Source	3.3 V Source
Logic low	≤ 1.5 V	≤ 0.8 V
Logic high	≥ 3.5 V	≥ 2.5 V

11.6.1 Switch closure input

Terminals: C1-C8

Pull-Down Resistance: Configurable in terminal pairs with 100 k Ω

Pull-Up Resistance: Configurable in terminal pairs with 100 k Ω (weak) or 2.2 k Ω (strong)

Maximum Input Frequency: 250 Hz

Minimum Switch Closed Time: 1 ms

Minimum Switch Open Time: 1 ms

Maximum Bounce Time: 1 ms open without being counted

Software Debounce Time: 1 ms

11.6.2 High-frequency input

Terminals: C1-C8

Pull-Down Resistance: Configurable in terminal pairs with 100 k Ω

Pull-Up Resistance: Configurable in terminal pairs with 100 k Ω (weak) or 2.2 k Ω (strong)

Maximum Input Frequency: 1 MHz

11.6.3 Edge timing

Terminals: C1-C8

Maximum Input Frequency: 1 MHz

Resolution: 20 ns

11.6.4 Edge counting

Terminals: C1-C8

Maximum Input Frequency: 1 MHz

11.6.5 Quadrature input

Terminals: C1-C8 can be configured as digital pairs to monitor the two sensing channels of an encoder.

Maximum Frequency: 500 kHz

Resolution: 20 ns or 50 MHz

11.6.6 Pulse-width modulation

Modulation Voltage: Logic high

Maximum Period: 43 seconds

Resolution: 10 ns

11.6.7 Maximum time between counter or timer instructions

- 86 seconds

See also [Pulse measurements](#) (p. 56) and [Pulse measurement specifications](#) (p. 164).

11.7 Communications specifications

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

Ethernet Port: RJ45 jack, 10/100/1000 Base Mbps, full and half duplex, Auto-MDIX, magnetic isolation, and TVS surge protection, IEEE 802.3 compliant. See also [Ethernet communications option](#) (p. 23).

Internet Protocols: Ethernet, PPP, RNDIS, ICMP/Ping, Auto-IP(APIPA), IPv4, IPv6, UDP, TCP, TLS (v1.2), DNS, DHCP, SLAAC, Telnet, HTTP(S), SFTP, FTP(S), POP3/TLS, NTP, SMTP/TLS, SNMPv3, CS I/O IP

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

USB Device: Micro-B device for computer connectivity

USB Host: USB 2.0 full speed host 12 Mbps, Type-A for mass storage devices

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

0 – 5 V Serial(C1 to C8): Eight independent TX/RX pairs

SDI-12 (C1, C3, C5, C7): Four independent SDI-12 compliant terminals are individually configured and meet SDI-12 Standard v 1.4.

RS-485 (C1 to C8): Two full duplex or four half duplex. Optional 120 Ohm termination resistor between pairs.

RS-422 (C1 to C8): Two full duplex or four half duplex. Use RS-485 configuration.

RS-232 (C1 to C8): Four independent Tx/Rx pairs.

CPI A/B and RS-232 A/B: Two RJ45 module ports that can operate in one of two modes: CPI or RS-232. CPI interfaces with Campbell Scientific CDM measurement peripherals and sensors. RS-232 connects, with an adapter cable, to computer, sensor, or communications devices serially.

CAN (GRANITE 10 only): Four general purpose ports, CAN 2.0 up to 1 Mbps, or CAN FD up to 5 Mbps. Screw terminal or DSUB 15-pin connections. Supports DBC files.

EPI: One EPI bus. 100 Mbps data rate. IEEE 1588 synchronization to 50 nS. 100 m (330 ft) maximum cable length per network connection. Up to 15 devices. EPI is a proprietary interface for communications between Campbell Scientific data loggers and Campbell Scientific CDM peripheral devices. It is based on Ethernet and IEEE 1588 Precision Time Protocol. It consists of a physical layer definition and a data protocol.

CPI: Two independent CPI buses. Up to 1 Mbps data rate each. Synchronization of devices to 5 μ S. Total cable length up to 610 m (2000 ft). Up to 20 devices per bus. CPI is a proprietary interface for communications between Campbell Scientific data loggers and Campbell Scientific CDM peripheral devices. It consists of a physical layer definition and a data protocol.

Wireless: Wi-Fi

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

Satellite: GOES, Argos, Inmarsat Hughes, Iridium

11.7.1 Wi-Fi specifications

WLAN (Wi-Fi)

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 data logger): 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 b/g/n, WPA/WPA2-Personal, WPA/WPA2-Enterprise Security, WEP

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

Access Point Mode: WPA2-Personal

Receive Sensitivity: -97 dBm

For WiFi additional current contribution specifications, see .

11.8 Standards compliance specifications

View EU Declarations of Conformity at www.campbellsci.eu/granite10.

View EU Declarations of Conformity at www.campbellsci.eu/granite9.

EMI and ESD protection:

- **Immunity:** Meets or exceeds following standards:
 - **ESD:** per IEC 61000-4-2; ±15 kV air, ±8 kV contact discharge
 - **Radiated RF:** per IEC 61000-4-3; 10 V/m, 80-1000 MHz
 - **EFT:** per IEC 61000-4-4; 4 kV power, 4 kV I/O
 - **Surge:** per IEC 61000-4-5; 4 kV power, 4kV I/O
 - **Conducted RF:** per IEC 61000-4-6; 10 V power, 10 V I/O
- Emissions and immunity performance criteria available on request.
- United States FCC ID: XF6-RS9113SB
- Industry Canada (IC): 8407A-RS9113SB

NOTE:

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

Appendix A. 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

Analogue to digital conversion. The process that translates analogue voltage levels to digital values.

alias

A second name assigned to variable in CRBasic.

allowed neighbour list

In PakBus networking, an allowed neighbour list is a list of neighbours with which a device will communicate. If a device address is entered in an allowed neighbour 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 neighbour 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.

analogue

Data presented as continuously variable electrical signals.

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 "neighbour" devices. A beacon in a PakBus network ensures that all devices in the network are aware of other devices that are viable.

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 data logger initiates communications with a computer running appropriate Campbell Scientific data logger 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 data loggers.

CDM/CPI

CPI is a proprietary interface for communications between Campbell Scientific data loggers and Campbell Scientific CDM peripheral devices. It consists of a physical layer definition and a data protocol.

CF

CompactFlash®

code

A CRBasic program, or a portion of a program.

Collect button

Button or command in data logger 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 data logger 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. Data logger user programs are compiled internally by the data logger 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 data logger support software.

CPU

Central processing unit. The brains of the data logger.

cr

Carriage return.

CRBasic

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

CRBasic Editor

The CRBasic programming editor; stand-alone software and also included with LoggerNet, PC400, and RTDAQ software.

CRC

Cyclic Redundancy Check

CRD

An optional memory drive that resides on a memory card.

CVI

Communication verification interval. The interval at which a PakBus® device verifies the accessibility of neighbours in its neighbour list. If a neighbour 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 neighbour is removed from the neighbour list.

D

DAC

Digital to analogue conversion. The process that translates digital voltage levels to analogue 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 data logger support software. A binary file is created for each table in each data logger. These files mimic the storage areas in data logger memory, and by default are two times the size of the data logger storage area. When the software collects data from a data logger, the data is stored in the binary file for that data logger. Various software functions retrieve data from the data cache instead of the data logger directly. This allows the simultaneous sharing of data among software functions.

data logger support software

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

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 data logger drives, and in binary or ASCII files that result from collecting final-data memory with data logger support software.

DC

Direct current.

DCE

Data Communication Equipment. While the term has much wider meaning, in the limited context of practical use with the data logger, it denotes the pin configuration, gender,

and function of an RS-232 port. The RS-232 port on the data logger 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 data logger enclosure, it prevents condensation.

Device Configuration Utility

Software tool used to set up data loggers 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 analogue 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 data logger 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 data logger, it denotes the pin configuration, gender, and function of an RS-232 port. The RS-232 port on the data logger 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 data logger 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 analogue voltage measurements. Note that most objects have a "an electrical potential" and the potential at different places on the earth - even a few metres 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 data logger base analogue-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 data logger support software. It provides a view of the data logger 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-data memory

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

final-storage data

Data that resides in final-data memory.

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 data logger data type for Public or Dim variables. Same format as IEEE4.

FP2

Two-byte floating-point data type. Default data logger data type for stored data. While IEEE4 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 gauge 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 data logger 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 data logger circuitry that are supposed to be at ground or 0 Volts. This difference in potentials can cause errors when measuring single-ended analogue 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 data logger 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 neighbour.

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.

HTTPS

Hypertext Transfer Protocol Secure. A secure version of HTTP.

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 data logger 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 in data logger memory.

isolation

Hardwire communication devices and cables can serve as alternate paths to earth ground and entry points into the data logger 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 data logger 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, neighbour lists, routing table, and communication timeouts.

keyboard/display

The data logger has an optional external keyboard/display.

L

leaf node

A PakBus node at the end of a branch. When in this mode, the data logger is not able to forward packets from one of its communication ports to another. It will not maintain a list

of neighbours, but it still communicates with other PakBus data loggers and wireless sensors. It cannot be used as a means of reaching (routing to) other data loggers.

If

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 data loggers.

LoggerNet

Campbell Scientific's data logger support software for programming, communications, and data retrieval between data loggers 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

Removable memory-card technology.

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 data logger in the communication command state, or an asynchronous serial communication port that can be configured to communicate with the data logger.

modulo divide

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

MSB

Most significant bit (the leading bit).

multimeter

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

multiplier

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

mV

The SI abbreviation for millivolts.

N

NAN

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

neighbour 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 datal oggers in PakBus networks so that they can communicate with each other and the LoggerNet server. For more information, see <https://www.campbellsci.eu/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 data logger 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 $^{\circ}\text{F} = ^{\circ}\text{C} \times 1.8 + 32$. The factor 32 is the offset.

ohm

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

Ohm's Law

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

on-line data transfer

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

operating system

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

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 data logger. A peripheral requires the data logger to operate. Peripherals include measurement, control, and data retrieval and communication modules.

PGA

Programmable Gain 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 data logger 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 data logger 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 data logger to commence communications. Ring line is pin 3 of the CS I/O 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 data logger configured as a router maintains its own list of neighbours and sends this list to other routers in the PakBus network. It also obtains and receives neighbour 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 neighbours 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 data loggers to computer communications is quite rigid, but transparent to most users. Features in the data logger that implement RS-232 communication with smart sensors are flexible.

RS-422

Communications protocol similar to RS-485. Most RS-422 sensors will work with RS-485 protocol.

RS-485

Recommended Standard 485. A standard defining how two computing devices can communicate with each other.

RTDAQ

Real Time Data Acquisition software for high-speed data acquisition applications. RTDAQ supports a variety of telecommunication options, manual data collection, and extensive data display. It includes Short Cut for creating data logger programs, as well as full-featured program editors.

RTU

Remote Telemetry Units

Rx

Receive

S

sample rate

The rate at which measurements are made by the data logger. 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 gauge 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 data logger 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 data logger 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 data logger 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 data logger 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 data logger support software. Sends a CRBasic program or operating system to a data logger.

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 data logger 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 data logger may be simplex.

single-ended

Denotes a sensor or measurement terminal wherein the analogue 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 data logger 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: data logger 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 coordin-

ation 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 data logger 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 data logger or some other compatible device. A terminal emulator is available in most data logger 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 data logger 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 data logger 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.

V

VAC

Volts alternating current.

variable

A packet of SRAM given an alphanumeric name. Variables reside in variable memory.

VDC

Volts direct current.

VisualWeather

Data logger support software specialized for weather and agricultural applications. The software allows you to initialize the setup, interrogate the station, display data, and generate reports from one or more weather stations.

volt meter

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

voltage divider

A circuit of resistors that ratiometrically divides voltage. For example, a simple two-resistor voltage divider can be used to divide a voltage in half. So, when fed through the voltage divider, 1 mV becomes 500 μ V, 10 mV becomes 5 mV, and so forth. Resistive-bridge circuits are voltage dividers.

volts

SI unit for electrical potential.

VSPECT®

® 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



Global Sales & Support Network

A worldwide network to help meet your needs



Campbell Scientific regional offices

Australia

Location: Garbutt, QLD Australia
Phone: 61.7.4401.7700
Email: info@campbellsci.com.au
Website: www.campbellsci.com.au

Brazil

Location: São Paulo, SP Brazil
Phone: 11.3732.3399
Email: vendas@campbellsci.com.br
Website: www.campbellsci.com.br

Canada

Location: Edmonton, AB Canada
Phone: 780.454.2505
Email: dataloggers@campbellsci.ca
Website: www.campbellsci.ca

China

Location: Beijing, P. R. China
Phone: 86.10.6561.0080
Email: info@campbellsci.com.cn
Website: www.campbellsci.com.cn

Costa Rica

Location: San Pedro, Costa Rica
Phone: 506.2280.1564
Email: info@campbellsci.cc
Website: www.campbellsci.cc

France

Location: Vincennes, France
Phone: 0033.0.1.56.45.15.20
Email: info@campbellsci.fr
Website: www.campbellsci.fr

Germany

Location: Bremen, Germany
Phone: 49.0.421.460974.0
Email: info@campbellsci.de
Website: www.campbellsci.de

India

Location: New Delhi, DL India
Phone: 91.11.46500481.482
Email: info@campbellsci.in
Website: www.campbellsci.in

South Africa

Location: Stellenbosch, South Africa
Phone: 27.21.8809960
Email: sales@campbellsci.co.za
Website: www.campbellsci.co.za

Spain

Location: Barcelona, Spain
Phone: 34.93.2323938
Email: info@campbellsci.es
Website: www.campbellsci.es

Thailand

Location: Bangkok, Thailand
Phone: 66.2.719.3399
Email: info@campbellsci.asia
Website: www.campbellsci.asia

UK

Location: Shepshed, Loughborough,
 UK
Phone: 44.0.1509.601141
Email: sales@campbellsci.co.uk
Website: www.campbellsci.co.uk

USA

Location: Logan, UT USA
Phone: 435.227.9120
Email: info@campbellsci.com
Website: www.campbellsci.com