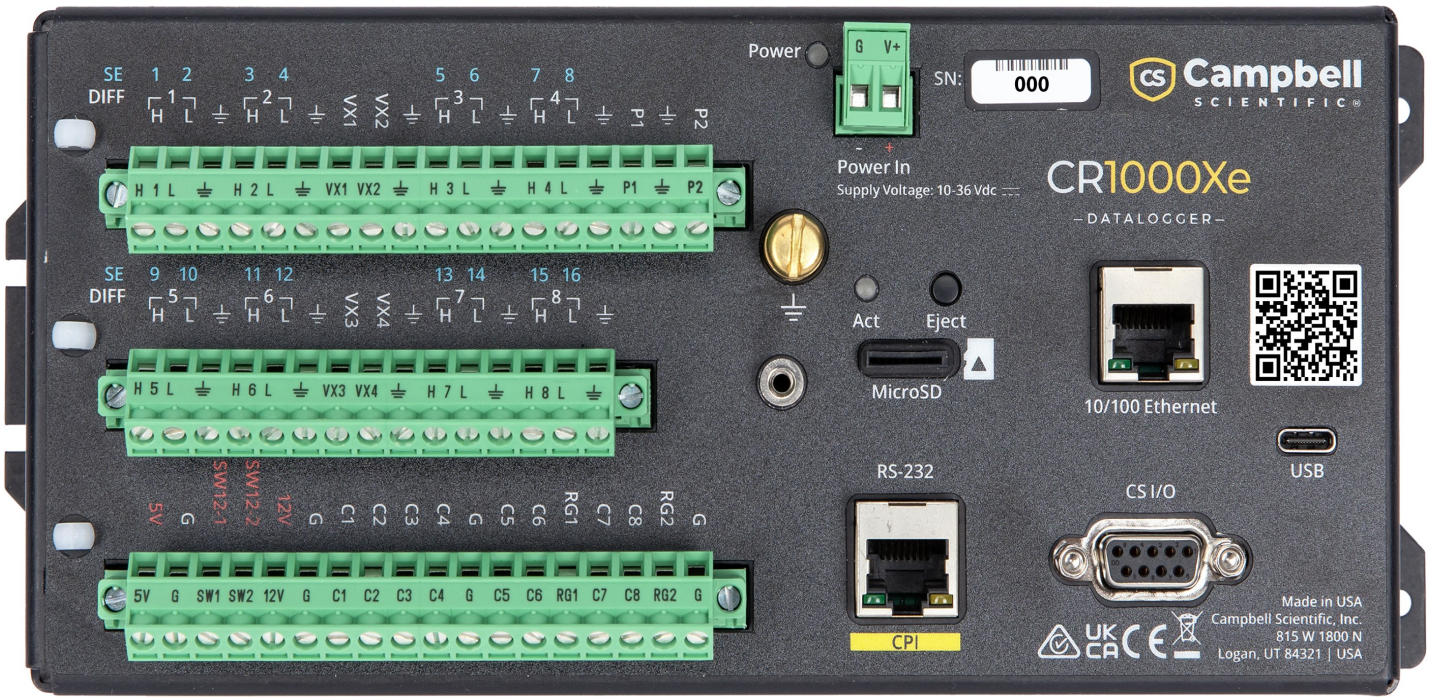
PRODUCT MANUAL

Data Logger

CR1000X/CR1000Xe

Measurement and Control Datalogger



Revision: 09/2026

Copyright © 2017 – 2026
Campbell Scientific, Inc.

RELIABLE MONITORING SINCE 1974

Campbell SCIENTIFIC®

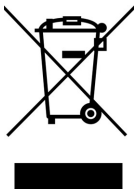
Please read first

About this manual

Please note that this manual was produced by Campbell Scientific Inc. primarily for the North American market. Some spellings, weights and measures may reflect this. 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 Europe, Middle East, and Africa (EMEA) or Asia Pacific (APAC) 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 EMEA or APAC use. Please note, however, *that when a power supply adapter is ordered from Campbell Scientific it will be suitable for use in your country*.

Reference to some radio transmitters, digital cell phones and aerials (antennas) 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 EMEA or APAC market; in some cases alternatives are offered.

Recycling information for countries subject to WEEE regulations 2012/19/EU



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 product's life should be removed from the product and sent to an appropriate recycling facility, per The Waste Electrical and Electronic Equipment (WEEE) Regulations 2012/19/EU. Campbell Scientific 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 support, please contact Campbell Scientific, or your local agent.

Table of Contents

1. Introduction **1**

2. Precautions **2**

3. Initial inspection **3**

4. CR1000X/CR1000Xe data acquisition system components **4**

4.1 The CR1000X/CR1000Xe Datalogger 5

4.1.1 Overview 5

4.1.2 Operations 6

4.1.3 Programs 6

4.2 Sensors 6

5. Wiring panel and terminal functions **8**

5.1 Power input 12

5.1.1 Powering a data logger with a vehicle 13

5.1.2 Power LED indicator 13

5.2 Power output 14

5.3 Grounds 15

5.4 Communications ports 17

5.4.1 USB device port 17

5.4.2 Ethernet port 17

5.4.3 C terminals for communications 17

5.4.3.1 SDI-12 ports 18

5.4.3.2 RS-232, RS-422, RS-485, TTL, and LVTTL ports 18

5.4.3.3 SDM ports 18

5.4.4 CS I/O port 19

5.4.5 RS-232/CPI port 20

5.5 Programmable logic control 21

6. Setting up the CR1000X/CR1000Xe **23**

6.1 Setting up communications with the data logger 23

6.1.1 USB or RS-232 communications 24

6.1.2 Virtual Ethernet over USB (RNDIS) 26

6.1.2.1 Web interface 27

Using an RNDIS connection 28
Using an IP connection 29
Through Device Configuration Utility (option 1) 30
Through the Web Interface (option 2) 32
Web interface recovery 34
6.1.3 Ethernet communications option 35
6.1.3.1 Configuring data logger Ethernet settings 35
6.1.3.2 Ethernet LEDs 36
6.1.3.3 Setting up Ethernet communications between the data logger and computer 37
6.2 Testing communications with EZSetup 38
6.3 Making the software connection 39
6.4 Creating a Short Cut data logger program 40
6.5 Sending a program to the data logger 43
7. Pre-installation 45
7.1 Data logger security 45
7.1.1 Secure by default 45
7.1.2 Other Secure by default features 46
7.1.3 Obtaining a Unique Identification Number (UID) 47
7.1.4 Device Configuration Utility Security Check 52
7.1.4.1 PakBus 53
7.1.4.2 Web services 53
HTTP 54
HTTPS 54
7.1.4.3 Network services 54
FTP 55
Telnet 56
Ping 56
7.1.4.4 Operating System Status 56
7.1.5 Other security measures reviewed by Device Configuration Utility 57
7.1.5.1 PakBus TCP Enabled 57
7.1.5.2 Accounts editor 57
7.1.5.3 IP Broadcast Filtered 57
7.1.5.4 Other communications protocols 58
7.1.6 TLS 58
7.1.6.1 Obtaining certificate and private key 59

From a Certificate Authority ... 59
From your IT department ... 61
7.1.7 Additional security measures ... 62
7.1.7.1 Security codes ... 62
7.1.7.2 CRBasic ... 63
7.1.7.3 USB ... 63
USB Disable ... 63
USB Not Trusted ... 63
7.1.7.4 Other ... 64
7.2 Default program ... 64
7.3 Power budgeting ... 66

8. Installation ... 67

8.1 Field work ... 68
8.2 Data logger enclosures ... 68
8.3 Electrostatic discharge and lightning protection ... 69

9. Onboarding a data logger to CampbellCloud ... 72

9.1 MQTT Auto Publish Data versus MQTTPublishTable() ... 74

10. Working with data ... 76

10.1 Default data tables ... 76
10.2 Collecting data ... 77
10.2.1 Collecting data using LoggerNet ... 77
10.2.2 Collecting data using PC400 ... 77
10.3 Viewing historic data ... 78
10.4 Data types and formats ... 79
10.4.1 Variables ... 79
10.4.2 Constants ... 81
10.4.3 Data storage ... 81
10.5 About data tables ... 82
10.5.1 Table definitions ... 83
10.5.1.1 Header rows ... 83
10.5.1.2 Data records ... 85
10.6 Creating data tables in a program ... 85

11. Data memory ... 88

11.1 Data tables ... 88

11.2 Memory allocation 89
11.3 SRAM 89
11.3.1 USR drive 90
11.4 Flash memory 91
11.4.1 CPU drive 91
11.5 MicroSD (CRD: drive) 91
11.5.1 Formatting microSD cards 93
11.5.2 MicroSD card precautions 93
11.5.3 Act LED indicator 93
11.5.4 Card data retrieval 94
11.5.4.1 Via a communications link 94
11.5.4.2 Card transport to computer 95

12. Measurements 99

12.1 Voltage measurements 99
12.1.1 Single-ended measurements 100
12.1.2 Differential measurements 101
12.1.2.1 Reverse differential 101
12.2 Current-loop measurements 101
12.2.1 Example current-loop measurement connections 102
12.3 Resistance measurements 103
12.3.1 Resistance measurements with voltage excitation 104
12.3.2 RTD and PRT 107
12.3.3 Strain measurements 107
12.3.4 AC excitation 109
12.3.5 Accuracy for resistance measurements 110
12.4 Thermocouple Measurements 110
12.5 Period-averaging measurements 111
12.6 Pulse measurements 112
12.6.1 Low-level AC measurements 114
12.6.2 High-frequency measurements 114
12.6.2.1 P terminals 115
12.6.2.2 C terminals 115
12.6.3 Switch-closure and open-collector measurements 115
12.6.3.1 P Terminals 116
12.6.3.2 C terminals 116
12.6.4 Edge timing and edge counting 116

12.6.4.1 Single edge timing 117
12.6.4.2 Multiple edge counting 117
12.6.4.3 Timer input NAN conditions 117
12.6.5 Quadrature measurements 117
12.6.6 Pulse measurement tips 119
12.6.6.1 Input filters and signal attenuation 119
12.6.6.2 Pulse count resolution 119
12.7 Vibrating wire measurements 120
12.7.1 VSPECT™ 120
12.8 Sequential and pipeline processing modes 120
12.8.1 Sequential mode 121
12.8.2 Pipeline mode 121
12.8.3 Slow Sequences 122

13. Communications protocols 123

13.1 General serial communications 124
13.1.1 RS-232 126
13.1.2 RS-485 127
13.1.3 RS-422 127
13.1.4 TTL 128
13.1.5 LVTTL 128
13.1.6 TTL-Inverted 129
13.1.7 LVTTL-Inverted 129
13.2 Modbus communications 130
13.2.1 About Modbus 131
13.2.2 Modbus protocols 132
13.2.3 Understanding Modbus Terminology 133
13.2.4 Connecting Modbus devices 133
13.2.5 Modbus client-server protocol 133
13.2.6 About Modbus programming 134
13.2.6.1 Endianness 135
13.2.6.2 Function codes 135
13.2.7 Modbus information storage 136
13.2.7.1 Registers 136
13.2.7.2 Coils 136
13.2.7.3 Data Types 137
Unsigned 16-bit integer 137

Signed 16-bit integer ... 137
Signed 32-bit integer ... 138
Unsigned 32-bit integer ... 138
32-Bit floating point ... 138
13.2.8 Modbus tips and troubleshooting ... 138
13.2.8.1 Error codes ... 138
Result code -01: illegal function ... 139
Result code -02: illegal data address ... 139
Result code -11: COM port error ... 139
13.3 Internet communications ... 140
13.3.1 IP address ... 141
13.3.2 HTTPS server ... 141
13.3.3 FTP server ... 141
13.4 DNP3 communications ... 143
13.5 Serial peripheral interface (SPI) and I2C ... 143
13.6 PakBus communications ... 143
13.7 SDI-12 communications ... 144
13.7.1 SDI-12 transparent mode ... 145
13.7.1.1 Watch command (sniffer mode) ... 146
13.7.1.2 SDI-12 transparent mode commands ... 147
13.7.1.3 aXLOADOS! command ... 147
13.7.2 SDI-12 programmed mode/recorder mode ... 149
13.7.3 Programming the data logger to act as an SDI-12 sensor ... 149
13.7.4 SDI-12 power considerations ... 150

14. CR1000X/CR1000Xe maintenance ... 152

14.1 Data logger calibration ... 152
14.1.1 About background calibration ... 153
14.2 Internal battery ... 154
14.2.1 Replacing the internal battery ... 155
14.3 Updating the operating system ... 157
14.3.1 Sending an operating system to a local data logger ... 158
14.3.2 Sending an operating system to a remote data logger ... 158
14.4 gzip ... 160
14.5 File management via powerup.ini ... 161
14.5.1 Syntax ... 161
14.5.2 Example powerup.ini files ... 163

15. Tips and troubleshooting 165
15.1 Checking station status 166
15.1.1 Viewing station status 166
15.1.2 Watchdog errors 167
15.1.3 Results for last program compiled 168
15.1.4 Skipped scans 168
15.1.5 Skipped records 168
15.1.6 Variable out of bounds 168
15.1.7 Battery voltage 169
15.2 Understanding NAN and INF occurrences 169
15.3 Timekeeping 170
15.3.1 Clock best practices 170
15.3.2 Time stamps 170
15.3.3 Avoiding time skew 171
15.4 CRBasic program errors 171
15.4.1 Program does not compile 172
15.4.2 Program compiles but does not run correctly 172
15.5 Resetting the data logger 173
15.5.1 Processor reset 173
15.5.2 Program send reset 173
15.5.3 Manual data table reset 174
15.5.4 Formatting drives 174
15.5.5 Full memory reset 174
15.6 Troubleshooting power supplies 174
15.7 Using terminal mode 175
15.7.1 Serial talk through and comms watch 178
15.7.2 SDI-12 transparent mode 178
15.7.2.1 Watch command (sniffer mode) 180
15.7.2.2 SDI-12 transparent mode commands 180
15.8 Ground loops 181
15.8.1 Common causes 181
15.8.2 Detrimental effects 181
15.8.3 Severing a ground loop 183
15.8.4 Soil moisture example 184
15.9 Improving voltage measurement quality 185
15.9.1 Deciding between single-ended or differential measurements 186
15.9.2 Minimizing ground potential differences 187

15.9.2.1 Ground potential differences ... 187
15.9.3 Detecting open inputs ... 188
15.9.4 Minimizing power-related artifacts ... 188
15.9.4.1 Minimizing electronic noise ... 189
15.9.5 Filtering to reduce measurement noise ... 190
15.9.5.1 CR1000X/CR1000Xe filtering details ... 190
15.9.6 Minimizing settling errors ... 191
15.9.6.1 Measuring settling time ... 191
15.9.7 Factors affecting accuracy ... 193
15.9.7.1 Measurement accuracy example ... 194
15.9.8 Minimizing offset voltages ... 195
15.9.8.1 Compensating for offset voltage ... 197
15.9.8.2 Measuring ground reference offset voltage ... 198
15.10 Field calibration ... 199
15.11 File system error codes ... 199
15.12 File name and resource errors ... 200
15.13 Background calibration errors ... 200

16. Information tables and settings (advanced) ... 202

16.1 DataTableInfo table system information ... 203
16.1.1 DataFillDays ... 203
16.1.2 DataRecordSize ... 203
16.1.3 DataTableName ... 203
16.1.4 RecNum ... 204
16.1.5 SecsPerRecord ... 204
16.1.6 SkippedRecord ... 204
16.1.7 TimeStamp ... 204
16.2 Status table system information ... 204
16.2.1 Battery ... 204
16.2.2 BuffDepth ... 205
16.2.3 CalCurrent ... 205
16.2.4 CalGain ... 205
16.2.5 CalOffset ... 205
16.2.6 CalRefOffset ... 205
16.2.7 CalRefSlope ... 205
16.2.8 CalVolts ... 205
16.2.9 CardStatus ... 205

16.2.10 CommsMemFree 206
16.2.11 CompileResults 206
16.2.12 ErrorCalib 206
16.2.13 FullMemReset 206
16.2.14 LastSystemScan 206
16.2.15 LithiumBattery 206
16.2.16 Low12VCount 206
16.2.17 MaxBuffDepth 207
16.2.18 MaxProcTime 207
16.2.19 MaxSystemProcTime 207
16.2.20 MeasureOps 207
16.2.21 MeasureTime 207
16.2.22 MemoryFree 207
16.2.23 MemorySize 207
16.2.24 Messages 208
16.2.25 OSDate 208
16.2.26 OSSignature 208
16.2.27 OSVersion 208
16.2.28 PakBusRoutes 208
16.2.29 PanelTemp 208
16.2.30 PortConfig 208
16.2.31 PortStatus 209
16.2.32 ProcessTime 209
16.2.33 ProgErrors 209
16.2.34 ProgName 209
16.2.35 ProgSignature 209
16.2.36 RecNum 209
16.2.37 RevBoard 210
16.2.38 RunSignature 210
16.2.39 SerialNumber 210
16.2.40 SkippedScan 210
16.2.41 SkippedSystemScan 210
16.2.42 StartTime 210
16.2.43 StartUpCode 210
16.2.44 StationName 211
16.2.45 SW12Volts 211
16.2.46 SystemProcTime 211

16.2.47 TimeStamp 211
16.2.48 VarOutOfBound 211
16.2.49 WatchdogErrors 211
16.2.50 WiFiUpdateReq 212
16.3 CPIStatus system information 212
16.3.1 BusLoad 212
16.3.2 ModuleReportCount 212
16.3.3 ActiveModules 213
16.3.4 BuffErr (buffer error) 213
16.3.5 RxErrMax 213
16.3.6 TxErrMax 213
16.3.7 FrameErr (frame errors) 213
16.3.8 ModuleInfo array 213
16.4 Settings 214
16.4.1 Baudrate 214
16.4.2 Beacon 214
16.4.3 CentralRouters 215
16.4.4 CommsMemAlloc 215
16.4.5 ConfigComx 215
16.4.6 CSIOxnetEnable 216
16.4.7 CSIOInfo 216
16.4.8 DeleteCardFilesOnMismatch 216
16.4.9 DisableLithium 216
16.4.10 DisableTCPDelayAck 217
16.4.11 DNS 217
16.4.12 EthernetInfo 217
16.4.13 EthernetPower 218
16.4.14 FilesManager 218
16.4.15 FTPEnabled 218
16.4.16 FTPPassword 218
16.4.17 FTPPort 218
16.4.18 FTPUserName 218
16.4.19 HTTPEnabled 219
16.4.20 HTTPHeader 219
16.4.21 HTTPPort 219
16.4.22 HTTPSEnabled 219
16.4.23 HTTPSPort 219

16.4.24 IncludeFile 219
16.4.25 IPAddressCSIO 220
16.4.26 IPBroadcastFiltered 220
16.4.27 IPAddressEth 220
16.4.28 IPGateway 220
16.4.29 IPGatewayCSIO 221
16.4.30 IPMaskCSIO 221
16.4.31 IPMaskEth 221
16.4.32 IPTrace 221
16.4.33 IPTraceCode 221
16.4.34 IPTraceComport 221
16.4.35 IsRouter 222
16.4.36 KeepAliveURL (Ping keep alive URL) 222
16.4.37 KeepAliveMin (Ping keep alive timeout value) 222
16.4.38 MaxPacketSize 222
16.4.39 Neighbors 222
16.4.40 NTPServer 223
16.4.41 PakBusAddress 223
16.4.42 PakBus Encryption Key 223
16.4.43 PakBusNodes 223
16.4.44 PakBusPort 224
16.4.45 PakBusTCPClients 224
16.4.46 PakBusTCPEnabled 224
16.4.47 PakBusTCPPassword 224
16.4.48 PingEnabled 224
16.4.49 PCAP 224
16.4.50 pppDial 225
16.4.51 pppDialResponse 225
16.4.52 pppInfo 225
16.4.53 pppInterface 225
16.4.54 pppIPAddr 225
16.4.55 pppPassword 226
16.4.56 pppUsername 226
16.4.57 RouteFilters 226
16.4.58 RS232Handshaking 226
16.4.59 RS232Power 227
16.4.60 RS232Timeout 227

16.4.61 Security(1), Security(2), Security(3) 227
16.4.62 ServicesEnabled 227
16.4.63 TCPClientConnections 227
16.4.64 TCP_MSS 227
16.4.65 TCPPort 227
16.4.66 TelnetEnabled 227
16.4.67 TLSConnections (Max TLS Server Connections) 228
16.4.68 TLSPassword 228
16.4.69 TLSStatus 228
16.4.70 Configure USB 228
16.4.71 USB Disable 228
16.4.72 USBEnumerate 229
16.4.73 USB Not Trusted 229
16.4.74 USB Virtual Ethernet Address (RNDIS) 229
16.4.75 USRDriveFree 229
16.4.76 USRDriveSize 230
16.4.77 UTCOffset 230
16.4.78 Verify 230
16.4.79 MQTT settings 230
16.4.79.1 MQTT Auto-Publish Data 231
16.4.79.2 MQTTBaseTopic (MQTT base topic) 231
16.4.79.3 MQTTCleanSession (MQTT connection) 231
16.4.79.4 MQTTClientID (MQTT client identifier) 231
16.4.79.5 MQTTEnable (Enable or disable MQTT) 231
16.4.79.6 MQTTEndpoint (MQTT broker URL) 232
16.4.79.7 MQTTKeepAlive (MQTT keep alive) 232
16.4.79.8 MQTTPassword (MQTT password) 232
16.4.79.9 MQTTPortNumber (MQTT port number) 232
16.4.79.10 MQTTStatusInterval (Status information publish interval) 232
16.4.79.11 MQTTState (MQTT state) 233
16.4.79.12 MQTTStateInterval (State publish interval) 233
16.4.79.13 MQTTUserName (MQTT user name) 234
16.4.79.14 MQTTWillMessage (MQTT last will message) 234
16.4.79.15 MQTTWillQoS (Quality of service) 234
16.4.79.16 MQTTWillRetain (MQTT last will message retained by broker) 234
16.4.79.17 MQTTWillTopic (MQTT last will topic) 234
16.4.80 GOES settings 234

16.4.80.1 GOESComPort 235
16.4.80.2 GOESEnabled 235
16.4.80.3 GOESGainSetting 235
16.4.80.4 GOESMsgWindow 236
16.4.80.5 GOESPlatformID 236
16.4.80.6 GOESRepeatCount 236
16.4.80.7 GOESRTBaudRate 236
16.4.80.8 GOESRTChannel 236
16.4.80.9 GOESRTInterval 236
16.4.80.10 GOESSTBaudRate 237
16.4.80.11 GOESSTChannel 237
16.4.80.12 GOESSTInterval 237
16.4.80.13 GOESSTOffset 237

17. CR1000Xe specifications 238

17.1 System specifications 238
17.2 Physical specifications 239
17.3 Power requirements 239
17.4 CR1000Xe Power output specifications 240
17.4.1 System power output current limits 240
17.4.2 Shared 12 V and SW12 power output 240
17.4.3 Individual maximum current for 12 V and SW12 output terminals 241
17.4.4 5 V fixed output 241
17.4.5 Control port as power output 241
17.4.6 CS I/O pin 1: 5 V fixed output 242
17.4.7 CS I/O pin 8: 12 V switched output 242
17.4.8 Voltage excitation 242
17.5 Analog measurement specifications 242
17.5.1 Voltage measurements 243
17.5.2 Resistance measurement specifications 245
17.5.3 Period-averaging measurement specifications 245
17.5.4 Current-loop measurement specifications 246
17.6 Pulse measurement specifications 247
17.6.1 Low-level AC input 247
17.6.2 Switch closure input 247
17.6.3 High-frequency input 248
17.7 Digital input/output specifications 248

17.7.1 Switch closure input 249
17.7.2 High-frequency input 249
17.7.3 Edge timing 249
17.7.4 Edge counting 249
17.7.5 Quadrature input 249
17.7.6 Pulse-width modulation 250
17.8 Communications specifications 250
17.9 Standards compliance specifications 251

18. CR1000X specifications 252

18.1 System specifications 252
18.2 Physical specifications 253
18.3 Power requirements 253
18.4 Power output specifications 254
18.4.1 System power out limits (when powered with 12 VDC) 254
18.4.2 12 V and SW12 V power output terminals 255
18.4.3 5 V fixed output 255
18.4.4 C as power output 255
18.4.5 CS I/O pin 1 256
18.4.6 Voltage excitation 256
18.5 Analog measurement specifications 256
18.5.1 Voltage measurements 257
18.5.2 Resistance measurement specifications 259
18.5.3 Period-averaging measurement specifications 259
18.5.4 Current-loop measurement specifications 260
18.6 Pulse measurement specifications 260
18.6.1 Low-level AC input 261
18.6.2 Switch closure input 261
18.6.3 High-frequency input 262
18.7 Digital input/output specifications 262
18.7.1 Switch closure input 262
18.7.2 High-frequency input 263
18.7.3 Edge timing 263
18.7.4 Edge counting 263
18.7.5 Quadrature input 263
18.7.6 Pulse-width modulation 263
18.8 Communications specifications 264

18.9 Standards compliance specifications 265

Appendix A. Glossary **266**

1. Introduction

> NOTE:
> An online version of this manual is available at help.campbellsci.com/CR1000X/Content/Home.htm. The online manual includes links to training videos and additional support resources.

The CR1000X and CR1000Xe are our flagship data loggers that provide measurement and control for a wide variety of applications. Their reliability and ruggedness make them excellent choices for remote environmental applications, including weather stations, mesonet systems, wind profiling, air quality monitoring, hydrological systems, water quality monitoring, and hydrometeorological stations.

The CR1000X and CR1000Xe are low-powered devices that measure sensors, drive direct communications and telecommunications, analyze data, control external devices, and store data and programs in onboard, nonvolatile storage. The electronics are RF-shielded by a unique sealed, stainless-steel canister. A battery-backed clock assures accurate timekeeping. The onboard, BASIC-like programming language supports data processing and analysis routines.

The Getting Started Guide provides an introduction to data acquisition and walks you through a procedure to set up a simple system. You may not find it necessary to progress beyond this. However, should you want to dig deeper into the complexity of the data logger functions or quickly look for details, extensive information is available in this and other Campbell Scientific manuals.

Additional Campbell Scientific publications are available online at www.campbellsci.com. Video tutorials are available at www.campbellsci.com/videos. Generally, if a particular feature of the data logger requires a peripheral hardware device, more information is available in the manual written for that device.

2. Precautions

READ AND UNDERSTAND the Safety section at the back of this manual.

An authorized technician shall verify that the installation and use of this product is in accordance to the manufacturer's instructions, recommendations and intended use.

Although the CR1000X/CR1000Xe is rugged, it should be handled as a precision scientific instrument.

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

3. Initial inspection

Upon receiving the CR1000X/CR1000Xe, inspect the packaging and contents for damage. File damage claims with the shipping company.

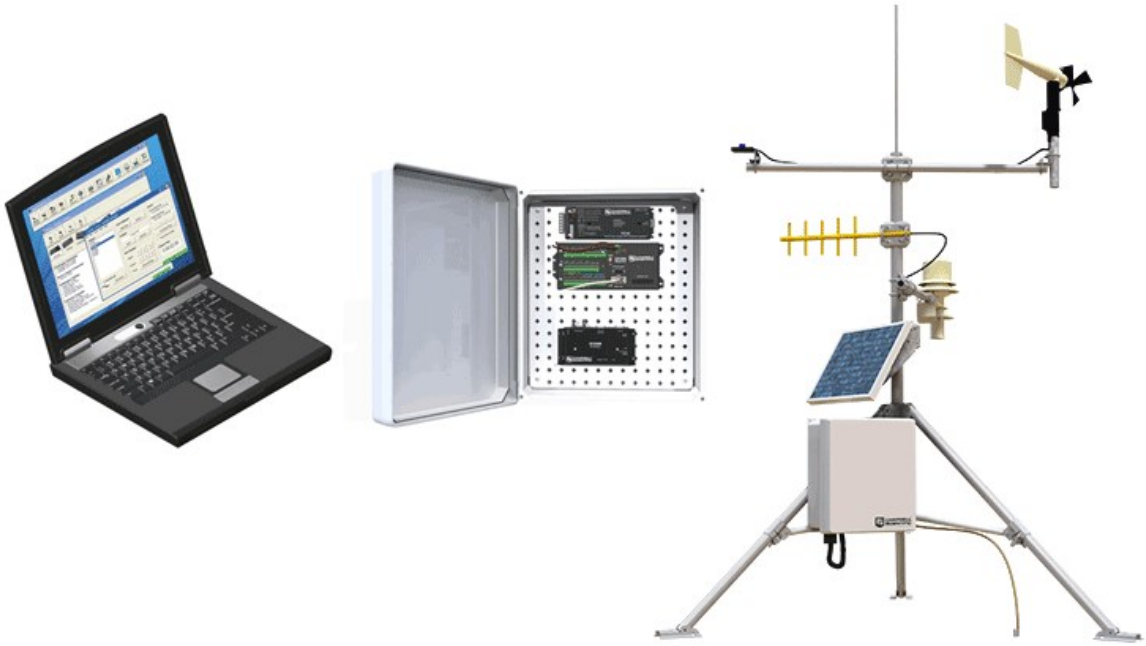
Immediately check package contents. Thoroughly check all packaging material for product that may be concealed. Check model numbers, part numbers, and product descriptions against the shipping documents. Model or part numbers are found on each product. Report any discrepancies to Campbell Scientific.

Check the CR1000X/CR1000Xe operating system version as outlined in Updating the operating system (p. 157), and update as needed. CR1000X data loggers with Serial Numbers 34000 and newer have hardware requiring the use of OS version 5.02 or newer.

4. CR1000X/CR1000Xe 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. 6) 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. 43).
- **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. 21) 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.

4.1 The CR1000X/CR1000Xe Datalogger

The CR1000X/CR1000Xe is used in a broad range of measurement and control projects. Rugged enough for extreme conditions and reliable enough for remote environments, it plays a critical role in numerous complex applications. Used in applications all over the world, it is a powerful core component for your data acquisition system.

4.1.1 Overview

The CR1000X/CR1000Xe data logger is the main part of a data acquisition system (see CR1000X/CR1000Xe data acquisition system components (p. 4) for more information). It has a central-processing unit (CPU), analog and digital measurement inputs, analog and digital outputs, and memory. An operating system (firmware) coordinates the functions of these parts in conjunction with the onboard clock and the CRBasic application program.

The CR1000X/CR1000Xe 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 CR1000X/CR1000Xe temporarily suspends operations when primary power drops below 9.6 V, reducing the possibility of inaccurate measurements.

The electronics are RF shielded and protected by the sealed, stainless-steel canister, making the CR1000X/CR1000Xe economical, small, and very rugged. A battery-backed clock assures accurate timekeeping.

4.1.2 Operations

The CR1000X/CR1000Xe 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.

4.1.3 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 CR1000X/CR1000Xe 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. See also:

- Creating a Short Cut data logger program (p. 40)
- https://www.campbellsci.com/videos/datalogger-programming

For more demanding programs, use the full-featured ***CRBasic Editor***. The ***CRBasic Editor*** help contains program structure details, instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

4.2 Sensors

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

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

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

- Analog
  - Voltage
  - Current
  - Strain
  - Thermocouple
  - Resistive bridge
- Pulse
  - High frequency
  - Switch-closure
  - Low-level AC
  - Quadrature
- Period average
- Vibrating wire (through interface modules)
- Smart sensors
  - SDI-12
  - RS-232
  - Modbus
  - DNP3
  - TCP/IP
  - RS-422
  - RS-485

5. Wiring panel and terminal functions

The CR1000X/CR1000Xe 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:

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

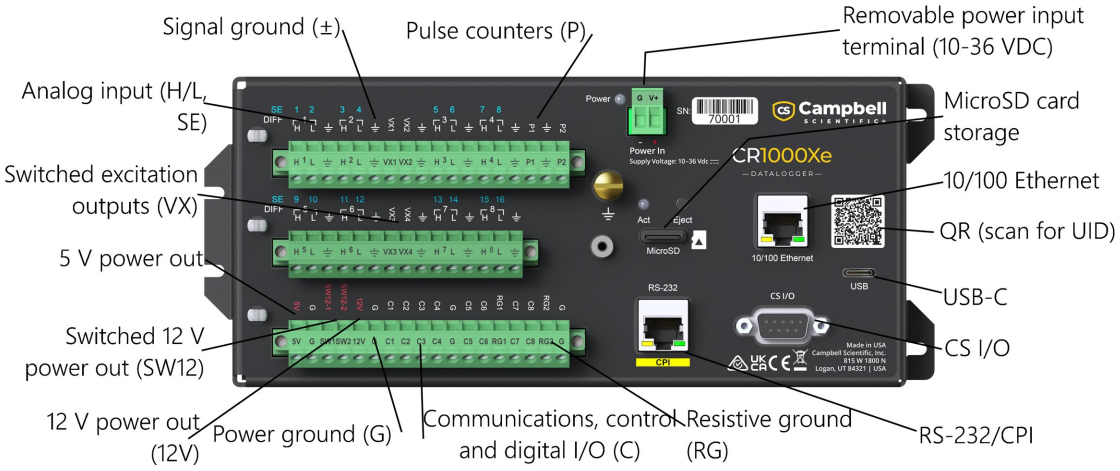


Figure 5-1. CR1000Xe wiring panel

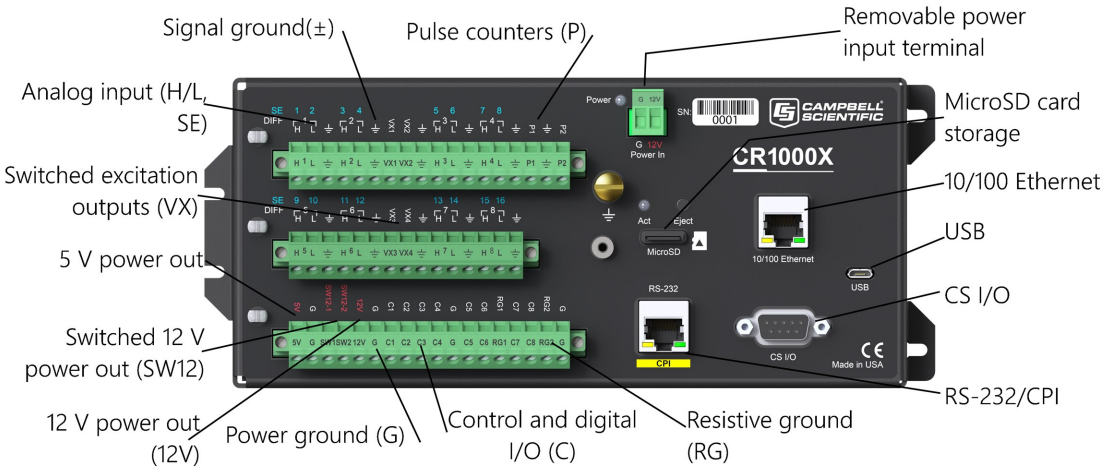


Figure 5-2. CR1000X Wiring panel

Table 5-1: Analog input terminal functions																		
SE DIFF	1 2 ┌1┐ H L		3 4 ┌2┐ H L		5 6 ┌3┐ H L		7 8 ┌4┐ H L		9 10 ┌5┐ H L		11 12 ┌6┐ H L		13 14 ┌7┐ H L		15 16 ┌8┐ H L		RG1	RG2
Single-Ended Voltage	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Differential Voltage	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L		
Ratiometric/Bridge	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Thermocouple	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Current Loop																	✓	✓
Period Average	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		

Table 5-2: Pulse counting terminal functions			
	P1	P2	C1-C8
Switch-Closure	✓	✓	✓
High Frequency	✓	✓	✓
Low-level AC	✓	✓	

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()`.

Table 5-3: Analog output terminal functions

	VX1-VX4
Switched Voltage Excitation	✓

Table 5-4: Voltage Output

	C1-C8[1]	VX1-VX4	5V	12V	SW12-1	SW12-2	SW12-CSIO
5 VDC	✓	✓	✓				
3.3 VDC	✓	✓					
12 VDC				✓	✓	✓	✓ (CR1000Xe only)

[1]C terminal voltage levels are configured in pairs. The default voltage output from C terminals is 5 V. Use the `PortPairConfig` instruction in CRBasic to configure a C terminal pair to output 3.3 V.

Table 5-5: Communications terminal functions

	C1	C2	C3	C4	C5	C6	C7	C8	RS-232/CPI
SDI-12	✓		✓		✓		✓		
GPS	PPS	Rx	Tx	Rx	Tx	Rx	Tx	Rx	
TTL 0-5 V[1]	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	
LVTTL 0-3.3 V[1]	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	

Table 5-5: Communications terminal functions

	C1	C2	C3	C4	C5	C6	C7	C8	RS-232/CPI
RS-232*	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx	✓
	(CR1000Xe only)								
RS-485 (Half Duplex)	A-	B+	A-	B+	A-	B+	A-	B+	
	(CR1000Xe only)								
RS-485[2] (Full Duplex)	Tx-	Tx+	Rx-	Rx+	Tx-	Tx+	Rx-	Rx+	
	(CR1000Xe only)								
I2C	SCL	SDA	SCL	SDA	SCL	SDA	SCL	SDA	
SPI	SCLK	COPI	CIPO		SCLK	COPI	CIPO		
SDM[3]	Data	Clk	Enabl		Data	Clk	Enabl		
CPI/CDM									✓

*ComC1 and ComC3 on the CR1000X are not designed to handle RS-232 signals, and long-term exposure—such as when connecting an RV50(X) modem—can damage the data logger.

[1] TTL and LVTTL are configured with the CommsMode option of the `SerialOpen` instruction in CRBasic.

[2] RS-422 compatible.

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

Communications functions also include Ethernet and USB.

WARNING:
While ComC1–ComC3 are compatible with RS-232 signals on the CR1000Xe, only ComC5 and ComC7 are compatible with RS-232 signals on the CR1000X. ComC1 and ComC3 on the CR1000X are not designed to handle RS-232 signals, and long-term exposure—such as when connecting an RV50(X) modem—can damage the data logger. An advantage of the CR1000Xe is that ComC1, ComC3, ComC5, and ComC7 all support RS-232 signals, whereas the CR1000X supports RS-232 only on ComC5 and ComC7.

Table 5-6: Digital I/O terminal functions	
	C1-C8
General I/O	✓
Pulse-Width Modulation Output	✓
Timer Input	✓
Interrupt	✓
Quadrature	✓

5.1 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 face of the wiring panel. See Wiring panel and terminal functions (p. 8). The positive power wire connects to the **12V** (CR1000X) or **+** (CR1000Xe) terminal. The negative wire connects to **G**. The power terminals are internally protected against polarity reversal and high voltage transients. For the CR1000X, if the voltage on the **POWER IN** terminals exceeds 19 V, power is shut off to certain parts of the data logger to prevent damaging connected sensors or peripherals. For the CR1000Xe, limit the voltage on the **POWER IN** terminals to 38Vdc or less.

The primary power source, which is often a transformer, power converter, or solar panel, connects to the charging regulator, as does a rechargeable battery. A third connection connects the charging regulator to the terminals of the **POWER IN** port. A 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. An external UPS consists of a primary-power source, a charging regulator external to the data logger, and an external battery.

WARNING:
Sustained supply (input) voltages in excess of those listed in Power requirements can damage the transient voltage suppression. For the CR1000Xe, see Power requirements (p. 239). For the CR1000X, see Power requirements (p. 253).

Ensure that power supply components match the specifications of the device to which they are connected.

NOTE: For the CR1000X, to prevent voltage input issues with sensors and peripherals, do not use more than 16 V when powering them through the 12V, SW12-1, SW12-2, or CS I/O port on the data logger.

When connecting power, switch off the power supply, insert the connector, then turn the power supply on. See Troubleshooting power supplies (p. 174) for more information.

The CR1000X/CR1000Xe can receive power via the **POWER IN** port as well as 5 VDC via a **USB** connection. If both **POWER IN** and **USB** are connected, power will be supplied by whichever has the highest voltage. If **USB** is the only power source, then the **CS I/O** port and the **12V**, **SW12**, and **5V** terminals will not be operational. When powered by USB (no other power supplies connected) **Status** field **Battery** = **0**. Functions that will be active with a 5 VDC source (**USB**) include sending programs, adjusting data logger settings, and making some measurements.

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. 154).

5.1.1 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.

5.1.2 Power LED indicator

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

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

5.2 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 12V power outputs differ between the CR1000X and CR1000Xe models. Unlike the CR1000X, the CR1000Xe provides a regulated 12V power output. See 12V Power output comparison between CR1000X and CR1000Xe (p. 14) for details.

Table 5-7: 12V Power output comparison between CR1000X and CR1000Xe

Port(s)	CR1000X	CR1000Xe	Benefits of CR1000Xe
12V, SW12-1, SW12-2	Unregulated 12V power directly from the battery. 12V outputs allow up to ~1A per channel, with a maximum of 3A per system	Regulated 12 Vdc power. 12V outputs support up to 2A per channel, with a maximum of 3.5A per system	Guarantees 12V output to power sensors and peripherals, even when a 24V power supply is used
12V over CS I/O	Unregulated 12V power over CS I/O port via battery for simpler wiring of CS measurement peripherals and radios	Regulated 12V power and SW12 control are accessible via pin 8 of the CS I/O port	Allows for a hard power reset on external CS modems without the requirement for an inline device

Additional power outputs include:

- **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 designed to power sensors or devices requiring a 5 VDC power supply. It is not intended as an excitation source for bridge measurements. The current output is shared with the CS I/O port; meaning the total current must remain within the specified limit. See 5 V fixed output (p. 241) specifications for the CR1000Xe and 5 V fixed output (p. 255) specifications for the CR1000X.

CAUTION:
For the CR1000X (not CR1000Xe), the voltage levels at the **12V** and switched **SW12** terminals, as well as pin **8** on the **CS I/O** port, are tied closely to the voltage levels of the main power

supply. Therefore, if 16 VDC is supplied to the **POWER IN 12V** and **G** terminals, the **12V**, **SW12** terminals, and pin **8** on the **CS I/O** port will also supply 16 VDC to connected peripherals. This could damage any connected peripheral or sensor not designed to handle that voltage.

- **VX** terminals: supply precise output voltage used by analog sensors to generate high resolution and accurate signals. In this case, these terminals are regularly used with resistive-bridge measurements (see Resistance measurements (p. 103) for more information). Using the **SWVX()** instruction, **VX** terminals can also supply a selectable, switched, regulated 3.3 or 5 VDC power source to power digital sensors and toggle control lines.
- **C** terminals: can be set low or high as output terminals . With limited drive capacity, digital output terminals are normally used to operate external relay-driver circuits. See also Digital input/output specifications (p. 262).

See also Power output specifications (p. 254).

5.3 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:

- **Signal Ground (⏚)** - reference for single-ended analog inputs, excitation returns, and a ground for sensor shield wires.
  - 11 common terminals
  - sometimes called analog ground (AG)
- **Power Ground (G)** - return for 3.3 V, 5 V, 12 V, and digital sensors. Use of **G** grounds for these outputs minimizes potentially large current flow through the analog-voltage-measurement section of the wiring panel, which can cause single-ended voltage measurement errors.
  - 4 common terminals

- **Resistive Ground (RG)** - used for non-isolated 0-20 mA and 4-20 mA current loop measurements (see Current-loop measurements (p. 101) for more information). Also used for decoupling ground on RS-485 signals. Includes 100 Ω resistance to ground. Maximum voltage for RG terminals is ±16 V.
    - 2 common terminals
- **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. 181)
- Minimizing ground potential differences (p. 187)
- Understanding the Importance of Grounding
- Connecting a Data Acquistion System to Earth Ground
- Avoiding Ground Loops

5.4 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 outstation networks
- Ethernet
- Modems
- Campbell Scientific PakBus™ networks
- Other Campbell Scientific data loggers

Campbell Scientific data logger communications ports include:

- CS I/O
- RS-232/CPI
- USB Device
- Ethernet
- C terminals

5.4.1 USB device port

The 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.

5.4.2 Ethernet port

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

5.4.3 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)

- LVTTL (0 to 3.3 V)
- SDM

Some communications types require more than one terminal, and some are only available on specific terminals. See Communications specifications (p. 264) for more information.

5.4.3.1 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. 144).

5.4.3.2 RS-232, RS-422, RS-485, TTL, and LVTTL ports

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

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

> **NOTE:**
> The maximum cable length for RS-232 communication is typically limited to 50 feet (15 meters) at 19200 baud. Higher baud rates may result in shorter transmission distances due to signal degradation.
>
> RS-485 supports a theoretical maximum point-to-point communication distance of 1200 meters (4000 feet). To achieve this distance, it's essential to use well-shielded and insulated cable, ensure careful installation, apply bus termination, and maintain low data rates (baud) of less than 115200 bps.

Configure **C** terminals as serial ports using ***Device Configuration Utility*** or by using the `SerialOpen()` CRBasic instruction. Terminals are configured in pairs for TTL, LVTTL, RS-232, and half-duplex RS-422 and RS-485 communications. For full-duplex RS-422 and RS-485, four terminals are required. See also Communications protocols (p. 123).

5.4.3.3 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. 264).

5.4.4 CS I/O port

One nine-pin port, labeled **CS I/O**, is available for communicating with a computer through Campbell Scientific communications interfaces, modems, and peripherals. Campbell Scientific recommends keeping CS I/O cables short (maximum of a few feet). See also Communications specifications (p. 264).

Table 5-8: CS I/O pinout

Pin number	Function	Input (I) Output (O)	Description
1	5 VDC	O	5 VDC: sources 5 VDC, used to power peripherals.
2	SG		Signal ground: provides a power return for pin 1 (5V), and is used as a reference for voltage levels.
3	RING	I	Ring: raised by a peripheral to put the CR1000X/CR1000Xe in the telecom mode.
4	RXD	I	Receive data: serial data transmitted by a peripheral are received on pin 4.
5	ME	O	Modem enable: raised when the CR1000X/CR1000Xe 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.

Table 5-8: CS I/O pinout			
Pin number	Function	Input (I) Output (O)	Description
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.

5.4.5 RS-232/CPI 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 Scientific CDM 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. CDM devices require more power to operate in general than do SDM devices. CPI ports also enable networking between compatible Campbell Scientific data loggers. Consult the manuals for CDM modules for more information.

CPI port power levels are controlled automatically by the CR1000X/CR1000Xe:

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

Table 5-9: RS-232/CPI pinout	
Pin number	Description
1	RS-232: Transmit (Tx)
2	RS-232: Receive (Rx)

Table 5-9: RS-232/CPI pinout	
Pin number	Description
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

5.5 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.

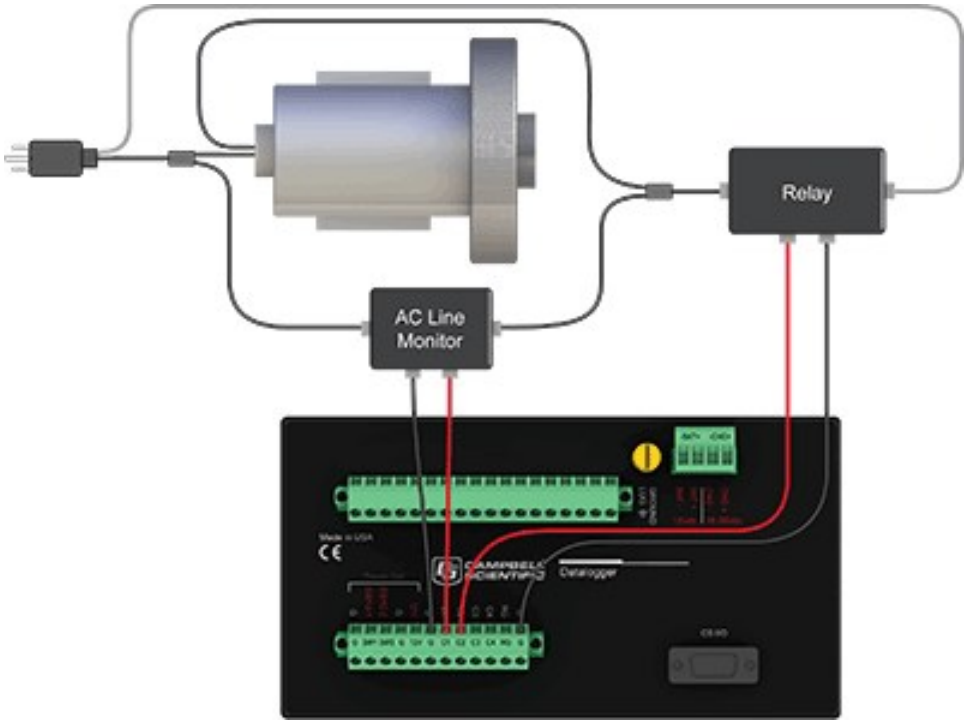
Control decisions can be based on time, an event, or a measured condition. Controlled devices can be physically connected to **C**, **VX**, or **SW12** terminals. ***Short Cut*** has provisions for simple on/off control. Control modules and relay drivers are available to expand and augment data logger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

- **C** terminals are selectable as binary inputs, control outputs, or communications ports. These terminals can be set low (0 VDC) or high (3.3 or 5 VDC) using the `PortSet()` or `WriteIO()` instructions. See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/. 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. Terminals C4, C5, and C7 can also be configured for pulse width modulation with a maximum period of 36.4 s. A C terminal configured for digital I/O is normally used to operate an external relay-driver circuit because the terminal itself has limited drive capacity.

- VX terminals can be set low or high using the `PortSet()` or `SWVX()` instruction. For more information on these instructions, see the CRBasic help.
- 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
```

6. Setting up the CR1000X/CR1000Xe

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

6.1 Setting up communications with the data logger 23
6.2 Testing communications with EZSetup 38
6.3 Making the software connection 39
6.4 Creating a Short Cut data logger program 40
6.5 Sending a program to the data logger 43

6.1 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.

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

6.1.1 USB or RS-232 communications 24
6.1.2 Virtual Ethernet over USB (RNDIS) 26
6.1.3 Ethernet communications option 35

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.com/videos/loggernet-software-network-planner.

Additional information is found in your specific peripheral manual, and the data logger support software manual and help. See also:

- www.campbellsci.com/cellular-communications for links to CELL200-series, RV50X, cellular data services, Konect PakBus router, and other cellular products
- www.campbellsci.com/wi-fi-communications for links to the NL241, and other Wi-Fi products
- www.campbellsci.com/spread-spectrum-radios for links to the RF452 and RF407-series spread spectrum radios
- www.campbellsci.com/satellite-communications for links to the TX325, TX326, HUGHES9502, and other satellite products
- www.campbellsci.com/loggernet
- www.campbellsci.com/pc400

Manuals for retired products are found at: www.campbellsci.com/manuals. These include, but are not limited to: RF401, RV50, TX321, TX320, and TX312.

6.1.1 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.com/videos.

> **TIP:**
> You will physically connect your data logger to your computer in step 6.

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.

   > **NOTE:**
   > New software installations automatically open the EZSetup Wizard the first time they run.

   - *LoggerNet* users, click **Setup**, select the **View** menu and ensure you are in the **EZ (Simplified)** view, then click **Add Datalogger**.
   - *PC400* users, click **Add Datalogger**.
2. Click **Next**.
3. Select your data logger from the list. In the **Datalogger Name** field, type a meaningful name for your data logger (for example, a site identifier or project name), and click **Next**.
4. Select the **Direct Connect** connection type and click **Next**.

5. If this is the first time connecting this computer to a CR1000X/CR1000Xe via USB, click **Install USB Driver**, select your data logger, click **Install**, and follow the prompts to install the USB driver.
6. Plug the data logger into your computer using a USB or RS-232 cable. The USB connection supplies 5 V power as well as a communications link, which is adequate for setup. A 12 V battery will be needed for field deployment. If using RS-232, external power must be provided to the data logger, and a CPI/RS-232 RJ45 to DB9 cable is required to connect to the computer.

   > **NOTE:**
   > The Power LED on the data logger indicates the program and power states. Because the data logger ships with a program set to run on power-up, the Power LED flashes three times every 10 seconds when powered over USB. When powered with a 12 V battery, it flashes once every 10 seconds. When no program is running, the LED is always on.

7. From the **COM Port** list, select the COM port used for your data logger. It will appear as CR1000X/CR1000Xe (COM number).
8. USB and RS-232 connections do not typically require a **COM Port Communication Delay**; this type of delay allows time for hardware devices to "wake up" and negotiate a communications link. Accept the default value of **00 seconds** and click **Next**.
9. You **must match** the baud rate and PakBus address hardware settings of your data logger. A USB connection does not require a baud rate selection, keep the default. RS-232 connections default to 115200 baud. The default PakBus address is **1**.
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 communications failure error. Accept the default value of **00 seconds**.
11. Set a **Max Time On-Line** to limit the amount of time the data logger remains connected. When the data logger is connected, communications with it are 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. Leave the **Neighbor PakBus Address** as the default of **0**.
13. Click **Next**.
14. By default, the data logger does not use a security code. Therefore, the **Security Code** can be left at **0**. If the code has been changed in the data logger, enter the new code.

    The **PakBus Encryption Key** can be left blank for direct USB connections.

15. Click **Next**.
16. Review the **Setup Summary**. If you need to make changes, click **Previous** to return to a previous window and change the settings.
17. Setup is now complete. The EZSetup Wizard allows you to **Finish**, or you may click **Next** to test communications, set the data logger clock, and send a program to the data logger. See Testing communications with EZSetup (p. 38) for more information.

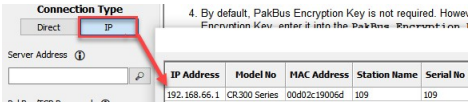
6.1.2 Virtual Ethernet over USB (RNDIS)

The data logger supports RNDIS (virtual Ethernet over USB). This allows the data logger to communicate via TCP/IP over USB. Watch a video at https://www.campbellsci.com/videos/ethernet-over-usb or use the following instructions.

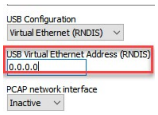
1. Supply power to the data logger. If connecting via USB for the first time, you must first install USB drivers by using ***Device Configuration Utility*** (select your data logger, then on the main page, click **Install USB Driver**). Alternatively, you can install the USB drivers using EZ Setup. A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

   > **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. By default, the RNDIS address is **192.168.66.** You can see the address by selecting **IP** as the Connection Type.



   If you wish to change the RNDIS address, click the **Settings Editor** tab >**Advanced** sub-tab > **USB Virtual Ethernet Address (RNDIS)** and enter the desired address. Note that 0 for the address defaults to 192.168.66.1.



4. The **Virtual Ethernet (RNDIS)** 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 the IP address (for example, **192.168.66.1**) into the address bar. For more information, see Web interface (p. 27).

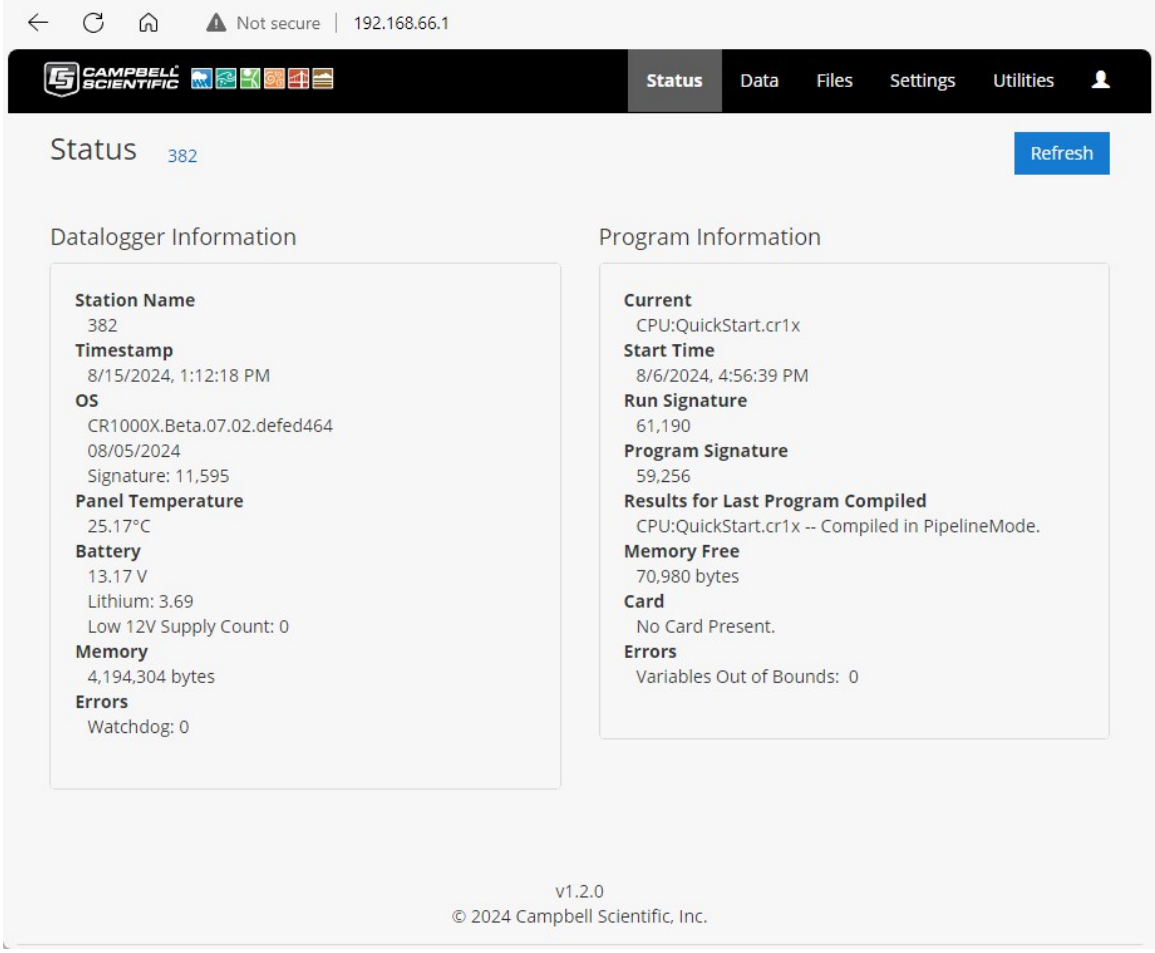
To secure your data logger from others who have access to your network, we recommend that you set security. For more information, see Data logger security (p. 45).

> **NOTE:**
> Ethernet over USB (RNDIS) is considered a direct communications connection. Therefore, it is a trusted connection and **Administrator** privileges are automatically granted for all functionality.

6.1.2.1 Web interface

For data loggers with an IP or RNDIS (virtual Ethernet over USB) connection, the built-in web interface provides access to real-time and stored data logger data, status and setting information, files, and more.

For more information on the web interface, watch an instructional video at: https://www.campbellsci.com/videos/web-interface.



Using an RNDIS connection

The CR1000X/CR1000Xe supports RNDIS. This allows the data logger to communicate via TCP/IP over USB.

Use the following steps.

1. Supply power to the data logger. A USB connection supplies 5 V power (as well as a communications link), which is adequate for setup, but a 12 V battery will be needed for field deployment.

   > **NOTE:**
   > Ensure the data logger is connected directly to the computer USB port (not to a USB hub). We recommend always using the same USB port on your computer each time you connect.

2. Ensure that the USB drivers have been installed on your computer. If you have connected to a CR1000X/CR1000Xe with this computer, the drivers have been installed. If this is the first time using this computer, see USB or RS-232 communications (p. 24).
3. The default RNDIS address is **192.168.66.1**. Type this address into a web browser to access the web interface.

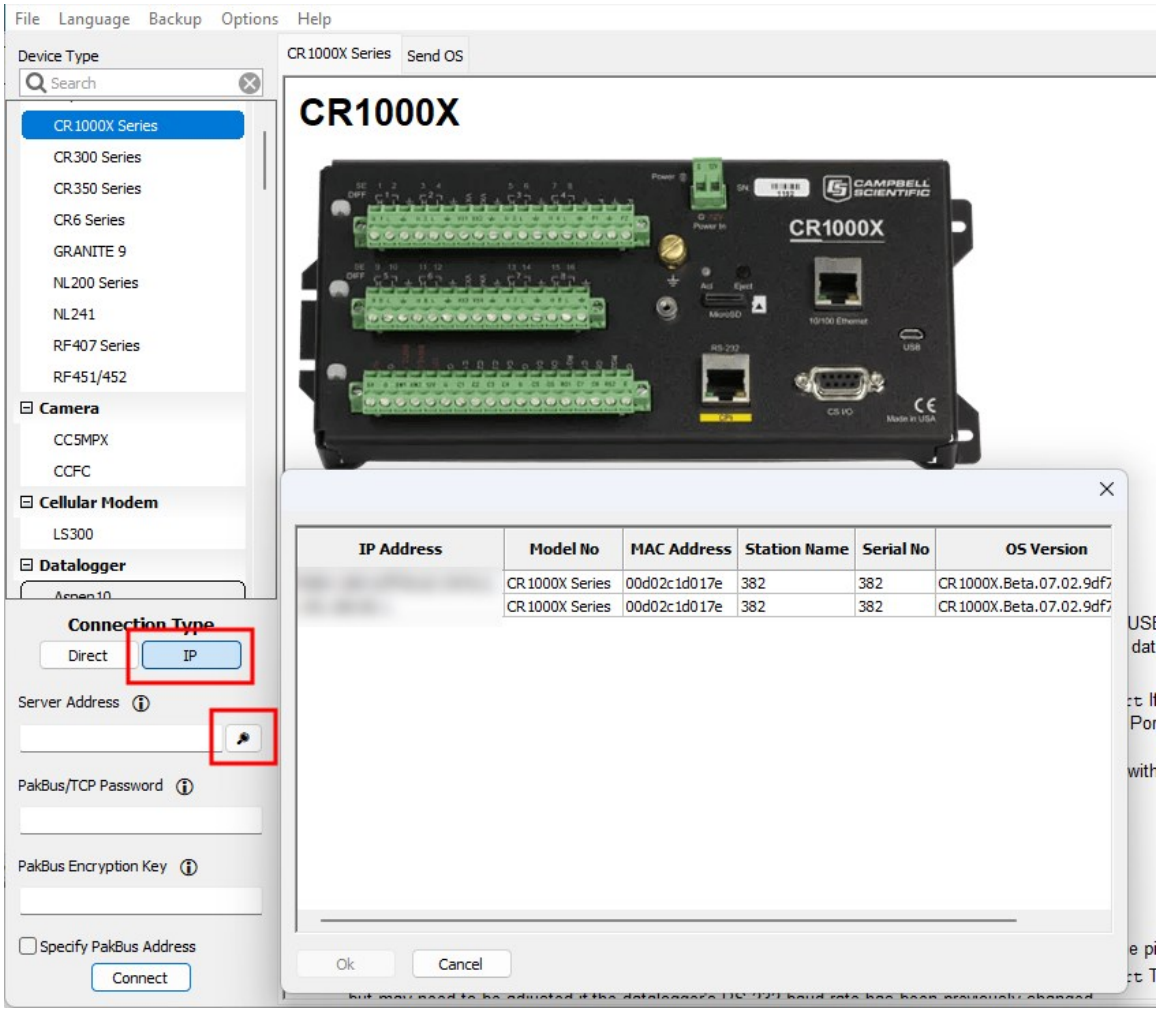
Using an IP connection

To use a standard IP connection (not RNDIS) to access the web interface, HTTP or HTTPS access must be enabled. By default, HTTP is enabled and HTTPS is disabled. Additionally, **anonymous** HTTP access is disabled by default. The default HTTP login credentials are **admin** for the username and the data logger *UID* as the admin password (if the data logger has a UID).

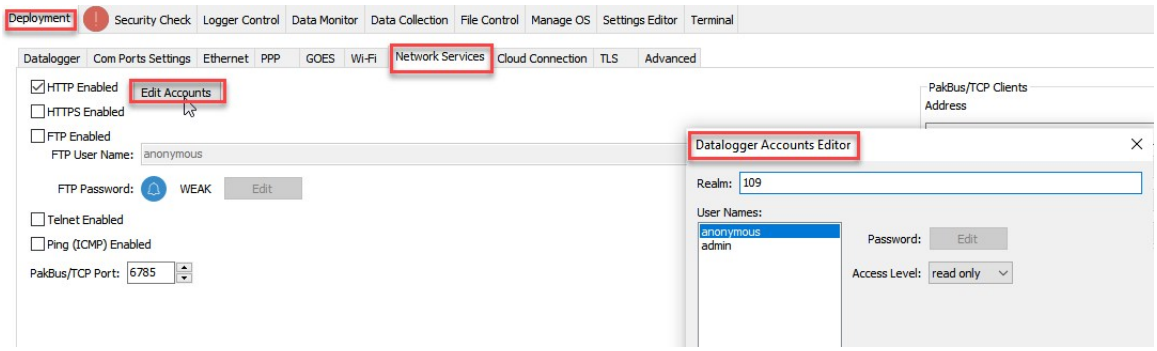
There are two ways to edit or set permissions for access.

Through *Device Configuration Utility* (option 1)

1. If you do not know the data logger IP address, retrieve it through ***Device Configuration Utility***. 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. 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).



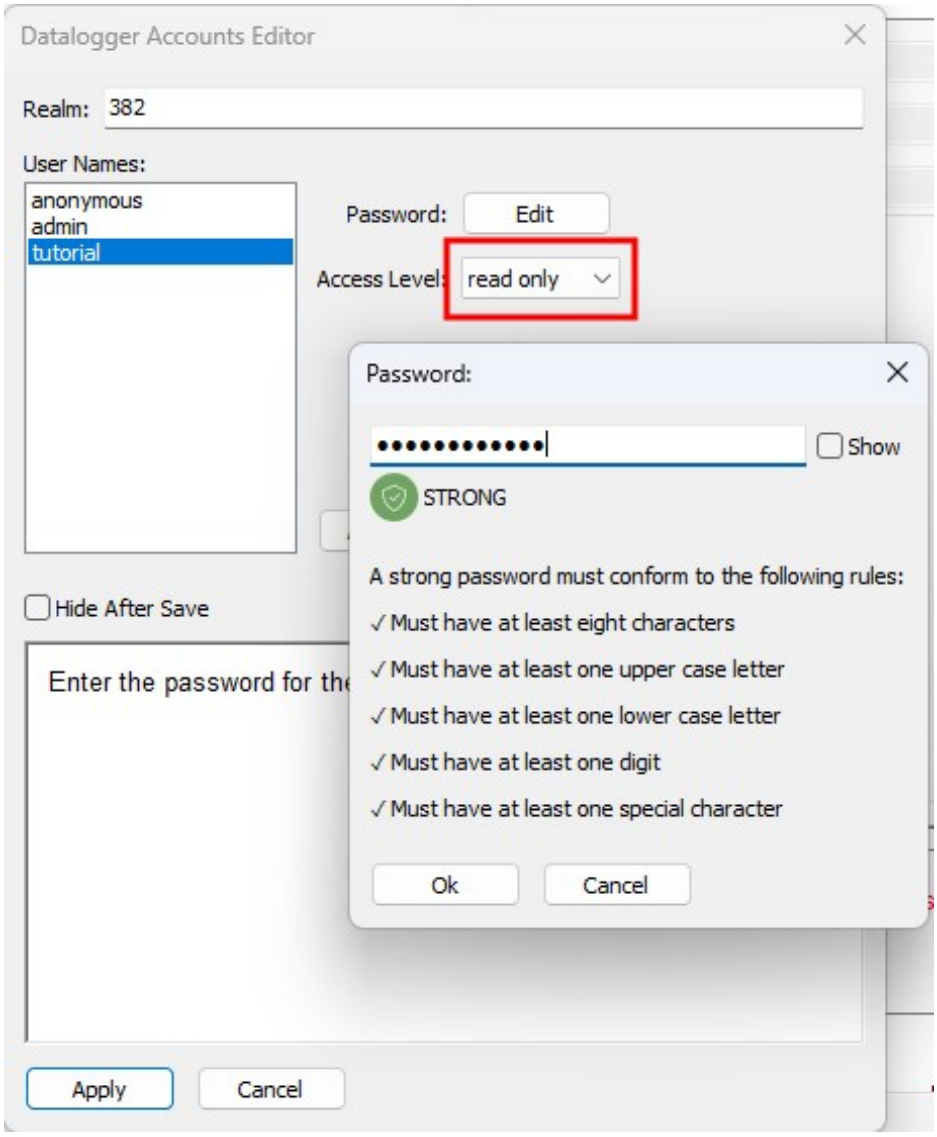
2. Enter the PakBus Encryption Key. For data loggers that have a UID, **PakBus Encryption** is enabled by default. The default **PakBus Encryption Key** is the UID.
3. To edit or set additional permissions to access the web interface, use ***Device Configuration Utility*** > **Networks Services** > **Edit Accounts** (in older versions this button is labeled **Edit .csipasswd File**). To add permissions, click **Add User**. Multiple user accounts with differing levels of access can be defined for one data logger.



Four levels of access are available:

- **Anonymous**: Read-only access, disabled by default. This account cannot be removed, but privileges can be disabled.
- **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 (Administrator)**: Data collection is unrestricted. Clock, writable variables, and settings can be changed. Programs can be viewed, stopped, deleted, and retrieved. Hidden tables can be viewed. Files, including programs can be sent to the data logger.

4. Assign an **Access Level** and **Password**.

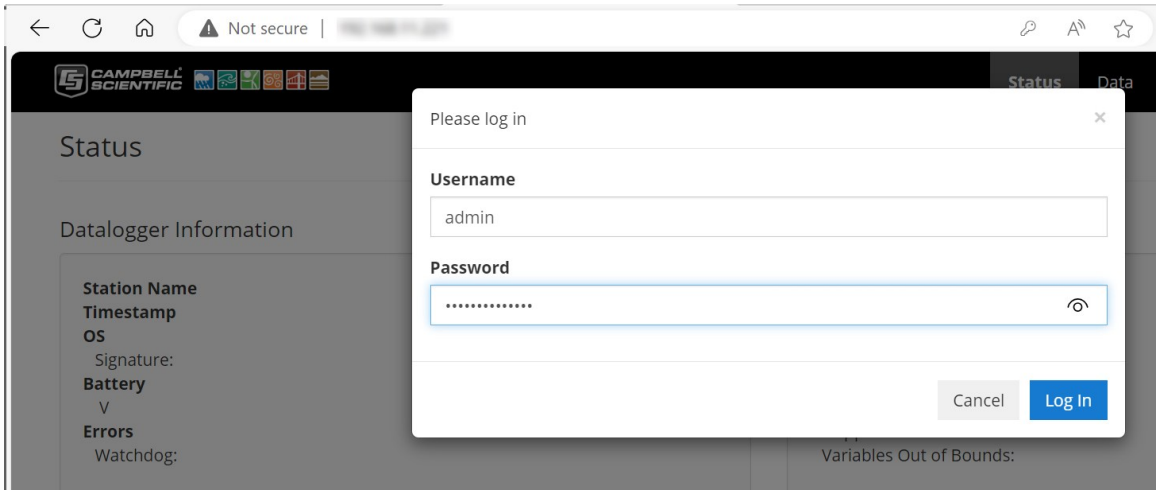


5. Type the data logger IP address into a web browser. If prompted, enter the user name and password that you just configured through **Edit Accounts**.

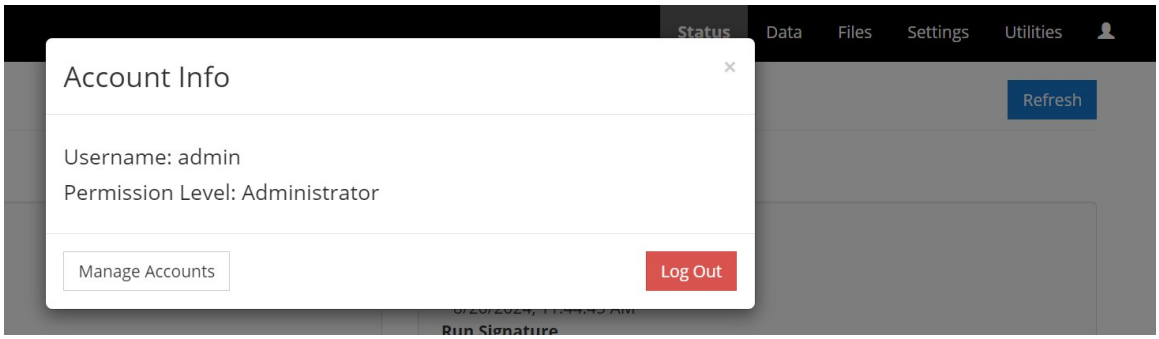
Through the Web Interface (option 2)

1. If you do not know the data logger IP address, retrieve it through ***Device Configuration Utility***. See the previous step 1.
2. Once you have the data logger IP address, type it into a web browser.

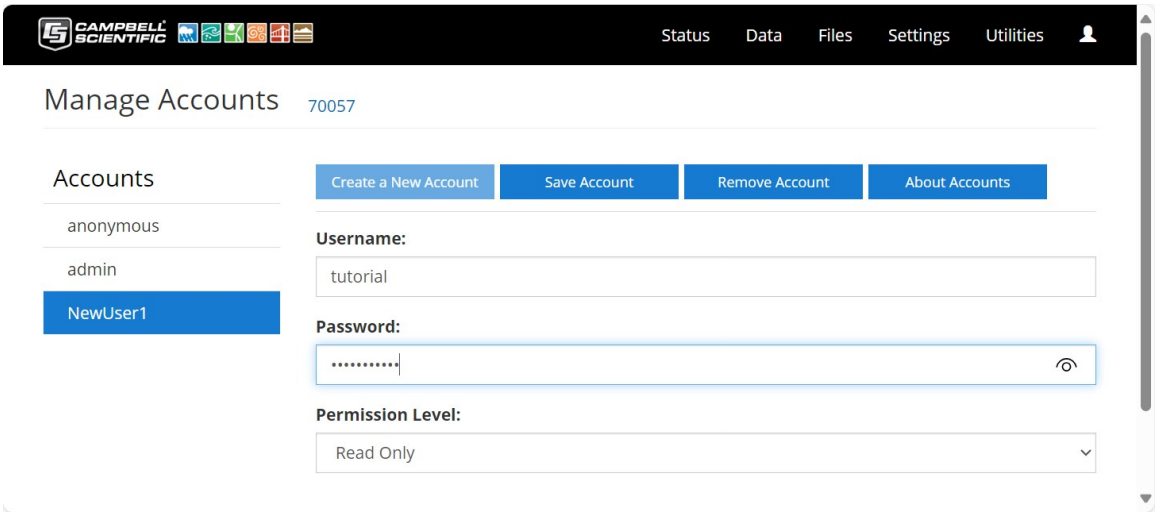
3. When prompted, enter the user name **admin** and the *UID* as the password. You may have to enter credentials twice, once for the browser and once for the CR1000X/CR1000Xe. Click **Log In**.



4. To edit or set additional permissions to access the web interface, select **Manage Accounts**. This is also accessible through the icon.



5. To add permissions, click **Create a New Account**. Multiple user accounts with differing permission levels can be defined for one data logger. Enter a **Username**, **Password**, and **Permission Level**. Click **Save Account**.



Four levels of access are available:

- **Anonymous**: Read-only access, disabled by default. This account cannot be removed, but privileges can be disabled.
- **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 (Administrator)**: Data collection is unrestricted. Clock, writable variables, and settings can be changed. Programs can be viewed, stopped, deleted, and retrieved. Hidden tables can be viewed. Files, including programs can be sent to the data logger.

Web interface recovery

In the unlikely event that web interface files are lost or corrupted, a data logger-hosted recovery page can be used to restore the web interface. This page can also be used to update the web interface when new versions are available. The recovery page is accessed by navigating to: **datalogger IP Address/recovery**. For example: **111.222.333.444/recovery**. This will open a **Recovery** page in the browser.



The **Send File** button on the recovery web page allows users to upload .obj or web.obj.gz files to restore web interface files on the data logger. The upload process may take several minutes.

6.1.3 Ethernet communications option

The CR1000X/CR1000Xe 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. 37). Watch a video at https://www.campbellsci.com/videos/datalogger-ethernet-configuration or use the following instructions.

6.1.3.1 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**). Alternatively, you can install the USB drivers using EZ Setup. A USB connection supplies 5 V power (as well as a communications link), which is adequate for setup, but a 12 V battery will be needed for field deployment.
2. Connect an Ethernet cable to the **10/100 Ethernet** port on the 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. 36).
3. Using data logger support software (***LoggerNet***, or ***PC400***), open ***Device Configuration Utility***.
4. Select the **CR1000X Series** 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. Beginning with operating system 8.00, the data logger is configured to be secure by default. Therefore, for data loggers that have a UID, **PakBus Encryption** is enabled by default. The default **PakBus Encryption Key** is the UID. Enter the data logger UID or, if the setting has been changed, enter the new key. See Data logger security (p. 45) 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.

6.1.3.2 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.

6.1.3.3 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 at https://www.campbellsci.com/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**.
   - *PC400* users, click **Add Datalogger**.
2. Click **Next**.
3. Select the **CR1000X Series** 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. 35) 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* and *PC400* 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. Therefore, the **Security Code** can be left at **0**. If the code has been changed in the data logger, enter the new code.

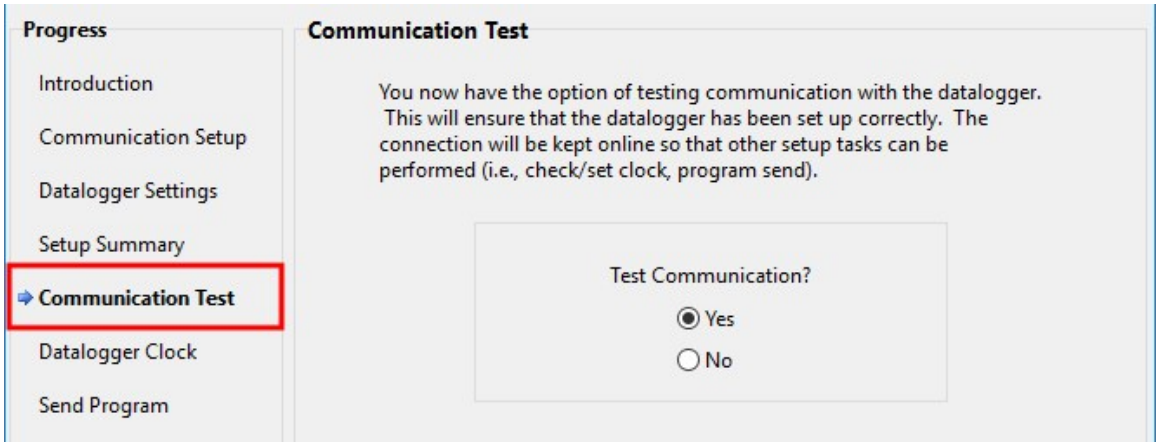
   Beginning with operating system 8.00, the data logger is configured to be secure by default. Therefore, for data loggers that have a UID, **PakBus Encryption** is enabled by default. The default **PakBus Encryption Key** is the UID. Enter the data logger UID or, if the setting has been changed, enter the new key. See Data logger security (p. 45) for more information.

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. 38) for more information.

6.2 Testing communications with EZSetup

1. Advance to, or select, the **Communication Test** step in EZ Setup. See USB or RS-232 communications (p. 24) for more information.



2. Ensure the data logger is physically 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. 165).
3. With a successful connection, the **Connection Time** with the data logger is displayed in the lower-left corner of the wizard. Click **Next**.
4. The **Datalogger Clock** window displays the time for both the data logger and the computer (server).

- The **Adjusted Server Date/Time** displays the current reading of the clock for the computer running your data logger support software. If the **Datalogger Date/Time** and **Adjusted Server Date/Time** do not match, click **Set Datalogger Clock** to set the data logger clock to the computer clock.
- Optionally, specify a positive or negative **Time Zone Offset** to apply when setting the data logger clock. This offset allows you to set the clock for a data logger that is in a different time zone than the computer (or to accommodate for changes in daylight saving time).

5. Click **Next**.
6. 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**.
7. ***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. Do not set up a scheduled collection during this tutorial. Click **Next**.
8. Click **Finish**, or you may click **Next** to test communications, set the data logger clock, and send a program to the data logger.

6.3 Making the software connection

Once you have configured your hardware connection (see Setting up communications with the data logger [p. 23]), your data logger and computer can communicate. 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**.
- ***PC400*** users, select the data logger from the list and click **Connect**.

To disconnect, click **Disconnect**.

For more information, see the Connect Window Tutorial.

6.4 Creating a *Short Cut* data logger program

You must provide a program for the data logger in order for it to make measurements, store data, or control external devices. There are several ways to write a program. The simplest is to use the program generator called **Short Cut**. For more complex programming, ***CRBasic Editor*** is used. The program file may use the extension **.CR1X**, **.CRB**, or **.DLD**.

Data logger programs are executed on a precise schedule termed the scan interval, based on the data logger internal clock.

Measurements are first stored in temporary memory called variables in the **Public** table. Data stored in variables is usually overwritten each scan. Periodically, generally on a time interval, the data logger stores data in tables. The data tables are later copied to a computer using your data logger support software.

Use ***Short Cut*** software to generate a program for your data logger. ***Short Cut*** is included with your data logger support software.

This section guides you through programming a CR1000X/CR1000Xe data logger to measure the voltage of the data logger power supply, the internal temperature of the data logger, and a thermocouple. With minor changes, these steps can apply to other measurements. Use the following instructions or watch the Quickstart part 3 video:

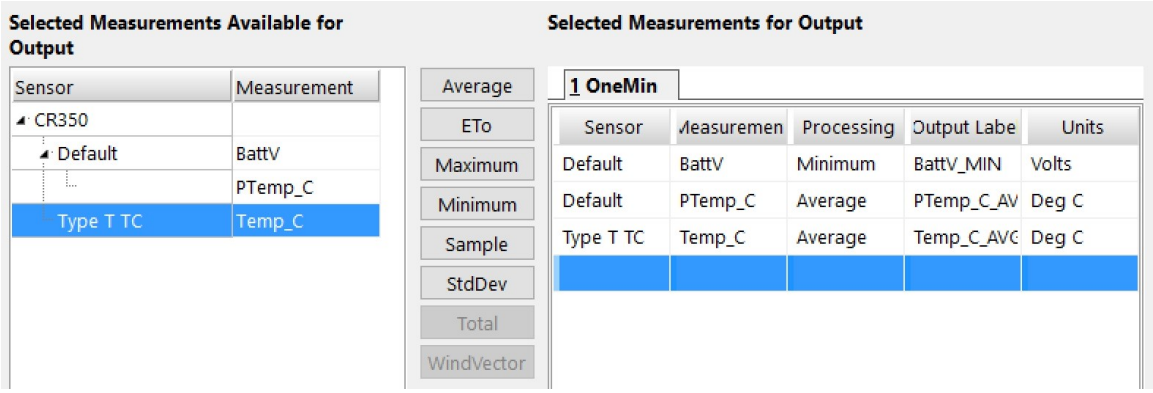
1. Using data logger support software, launch ***Short Cut***.
    - ***LoggerNet*** users, click **Program** then ***Short Cut***.
    - ***PC400*** users, click ***Short Cut***.
2. Click **Create New Program**.
3. Select **CR1000X Series** and click **Next**.

    > NOTE:
    > The first time ***Short Cut*** is run, a prompt asks for a noise rejection choice. Select **60 Hz Noise Rejection** for North America and areas using 60 Hz ac voltage. Select **50 Hz Noise Rejection** for most of the Eastern Hemisphere and areas that operate at 50 Hz.
    >
    > A second prompt lists sensor support options. **Campbell Scientific, Inc. (US)** is usually the best fit outside of Europe.
    >
    > To change the noise rejection or sensor support option for future programs, use the **Program** menu.

4. Lists of **Available Sensors and Devices** and **Selected Measurements Available for Output** are displayed. Battery voltage **BattV** and internal temperature **PTemp_C** are selected by

default. During operation, battery and temperature should be recorded at least daily to assist in monitoring system status.

5. Use the Search feature or expand folders to locate your sensor or device. Double-click on a sensor or measurement in the **Available Sensors and Devices** list to configure the device (if needed) and add it to the **Selected** list. For the example program, expand the **Sensors** and **Temperature** folders and double-click **Type T Thermocouple**.
6. If the sensor or device requires configuration, a window displays with configuration options. Click **Help** at the bottom of the window to learn more about any field or option. For the example program, accept the default options:
    - **1** **Type T TC** sensor
    - **Temp_C** as the **Temperature** label, and set the units to **Deg C**
    - **PTemp_C** as the **Reference Temperature Measurement**
7. Click the **Wiring** tab at the top of the page to see how to wire the sensor to the data logger. With the power disconnected from the data logger, insert the wires as directed in the diagram. Ensure you clamp the terminal on the wire, not the colored insulation. Use the included flat-blade screwdriver to open and close the terminals.
8. Click **OK**.
9. Click **Next**.
10. Use the **Output Setup** options to specify how often to make measurements and how often outputs are to be stored. Type **1** in the **How often should the data logger measure its sensor(s)?** box. Leave the units as **Seconds**.
11. Multiple output intervals can be specified, one for each output table (**Table1** and **Table2** tabs). For the example program, only one table is needed. Click the **Table2** tab and click **Delete Table**.
12. In the **Table Name** box, type a name for the table. For example: **OneMin**.
13. Select a **Data Output Storage Interval**. For example: **1** minute.
14. Click **Next**.
15. Select a measurement from the **Selected Measurements Available for Output** list, then click an output processing option to add the measurement to the **Selected Measurements for Output** list. For the example program, select **BattV** and click the **Minimum** button to add it to the **Selected Measurements for Output** list. Do not store the exact time that the minimum occurred. Repeat this procedure for an **Average PTemp_C** and **Average Temp_C**.



16. Click **Finish** and give the program a meaningful name such as a site identifier. Click **Save**.
17. If ***LoggerNet*** or other data logger support software is running on your computer, and the data logger is connected to the computer (see Making the software connection (p. 39) for more information), you can choose to send the program. Generally it is best to collect data first; so, we recommend sending the program using the instructions in Sending a program to the data logger (p. 43). Click **No**, do not send the program to the data logger.

    TIP:
    It is good practice to always retrieve data from the data logger before sending a program; otherwise, data may be lost. See Collecting data (p. 77) for detailed instruction.

18. Make note of the newly generated program location and filename. By default, programs created with ***Short Cut*** are stored in `C:\Campbellsci\SCWin\`.
19. Close ***Short Cut***.

If your data acquisition requirements are simple, you can probably create and maintain a data logger program exclusively with ***Short Cut***. If your data acquisition needs are more complex, the files that ***Short Cut*** creates are a great source for programming code to start a new program or add to an existing custom program using CRBasic. See the ***CRBasic Editor*** help for detailed information on program structure, syntax, and each instruction available to the data logger https://help.campbellsci.com/crbasic/cr1000x/.

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

6.5 Sending a program to the data logger

TIP:
It is good practice to always retrieve data from the data logger before sending a program; otherwise, data may be lost. See Collecting data (p. 77) 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 4 video.

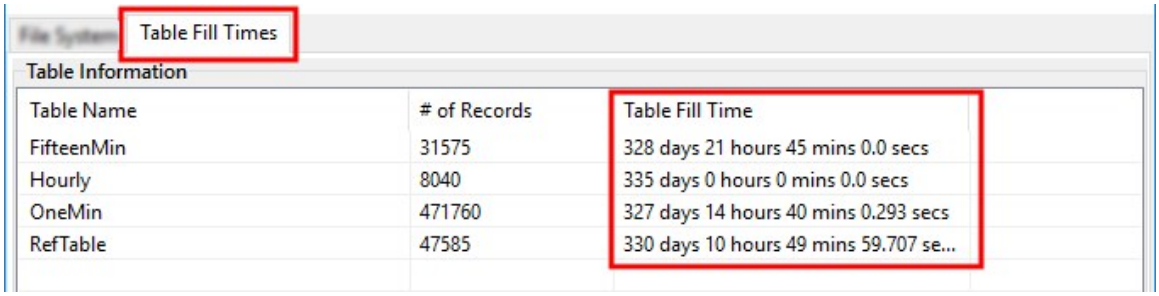
1. Connect the data logger to your computer (see Making the software connection (p. 39) for more information).
   - *LoggerNet* users, select **Main** and **Connect** on the *LoggerNet* toolbar, select the data logger from the **Stations** list, then **Connect**.
   - *PC400* users, select the data logger from the list and click **Connect**.
2. *LoggerNet* users, click **Send New...** (located in the **Current Program** section on the right side of the window).

   PC400 users, click **Send Program...** (located in the **Datalogger Program** section on the right side of the window).
3. *PC400* users, confirm that you would like to proceed and erase all data tables saved on the data logger. Click **Yes**.
4. Navigate to the program, select it, and click **Open**. For example: navigate to **C:\Campbellsci\SCWin** and select **MyTemperature.CR1X**. Click **Open**.
5. *LoggerNet* users, confirm that you would like to proceed and erase all data tables saved on the data logger. Click **Yes**.
6. The program is sent and compiled.
7. Review the **Compile Results** window for errors, messages and warnings.

8. *LoggerNet* users, click **Details**, select the **Table Fill Times** tab.

   PC400 user click **OK** then click *Station Status* , select the **Table Fill Times** tab.

   Ensure that the times shown are expected for your application. Click **OK**.



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. 76) for more information.

7. Pre-installation

This section describes some topics that should be considered before installing your station in the field.

7.1 Data logger security 45
7.1.1 Secure by default 45
7.1.2 Other Secure by default features 46
7.1.3 Obtaining a Unique Identification Number (UID) 47
7.1.4 Device Configuration Utility Security Check 52
7.1.5 Other security measures reviewed by Device Configuration Utility 57
7.1.6 TLS 58
7.1.7 Additional security measures 62
7.2 Default program 64
7.3 Power budgeting 66

7.1 Data logger security

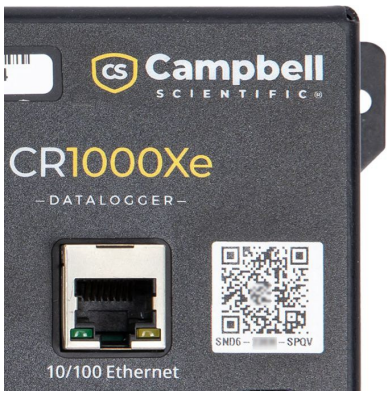
Data logger security concerns include:

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

7.1.1 Secure by default

Data loggers with UIDs are secure by default. This means a PakBus Encryption Key is enabled by default. The default **PakBus Encryption Key** is the UID.

The UID can be found on a QR-code on the front of the data logger.



CR1000X/CR1000Xe data loggers shipped before the release of UIDs and running operating system 8.00 or later may request a UID through ***Device Configuration Utility***. See Obtaining a Unique Identification Number (UID) (p. 47).

> **NOTE:**
> Direct USB connections do not require that you enter the PakBus Encryption Key, unless **USB Not Trusted** is enabled in the data logger.

7.1.2 Other Secure by default features

Anonymous HTTP access is disabled by default. HTTP will be accessible with **admin** as the username and *UID* as the admin password. See Web services (p. 53) for more information.

FTP is disabled by default. However, when enabled, the UID will be the default FTP password. See Network services (p. 54) for more information.

Wi-Fi, when configured as an Access Point ("Create a Network"), will use the UID as the default password.

When a data logger has a UID and is reset to factory defaults, the UID will be used as the default password.

> **CAUTION:**
> Passwords may be changed from the default UID. However, when the data logger is reset to factory defaults or a new OS is sent, the default password will revert to the UID.

Check the CR1000X/CR1000Xe operating system version as outlined in Updating the operating system (p. 157), and update as needed.

For more information on security, see Communications Protocols and Security Options for Campbell Scientific Data Loggers.

7.1.3 Obtaining a Unique Identification Number (UID)

As discussed in Data logger security (p. 45), new data loggers are given a Unique Identification Number (UID) at the factory. The UID is also used as a password specific to each data logger. Beginning with operating system 8.00 data loggers which were released prior to the UID being implemented can also be assigned a UID. This is done using ***Device Configuration Utility*** found in ***LoggerNet***, ***PC400***, or downloadable directly from the Campbell Scientific website. ***Device Configuration Utility*** version 2.31 or later is required.

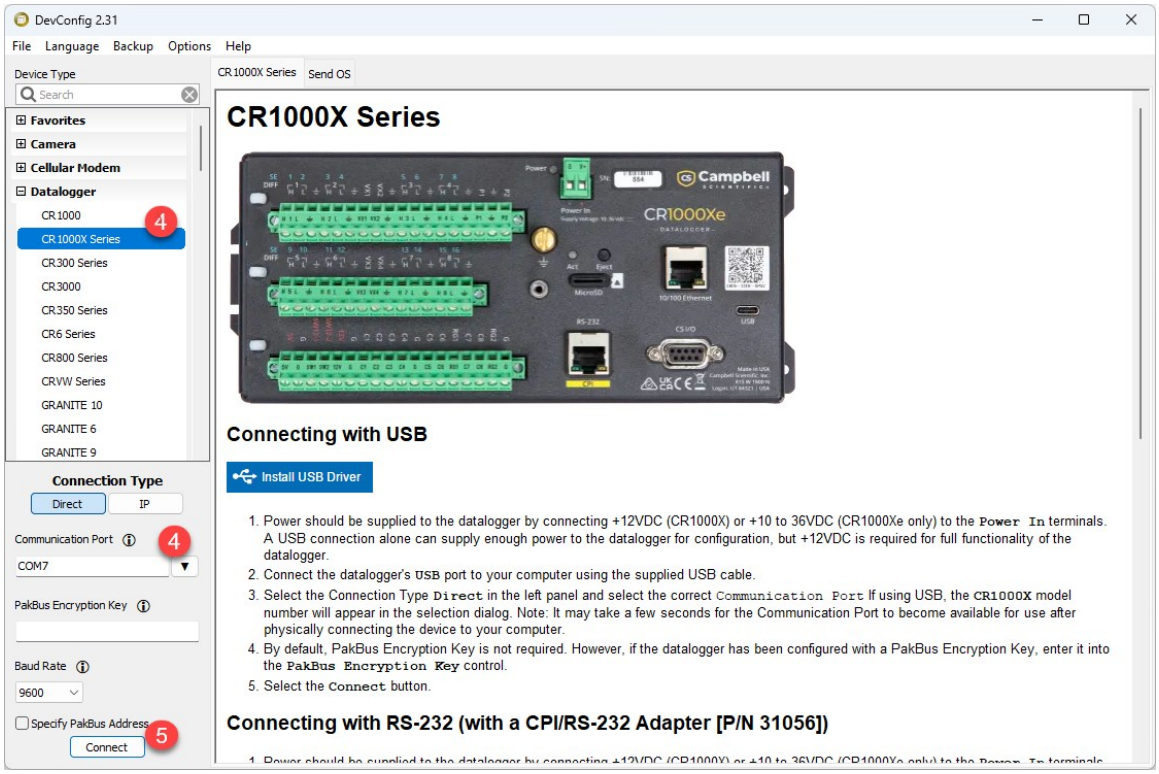
> NOTE:
> A UID cannot be obtained without a CampbellCloud account. To create an account, see Creating a CampbellCloud organization account in the CampbellCloud online manual.

> NOTE:
> When a UID is assigned to a data logger, it is written to one-time programmable memory. Therefore, any future OS updates or factory default resets of the device will not delete the UID.

Follow these steps to obtain a UID for your data logger:

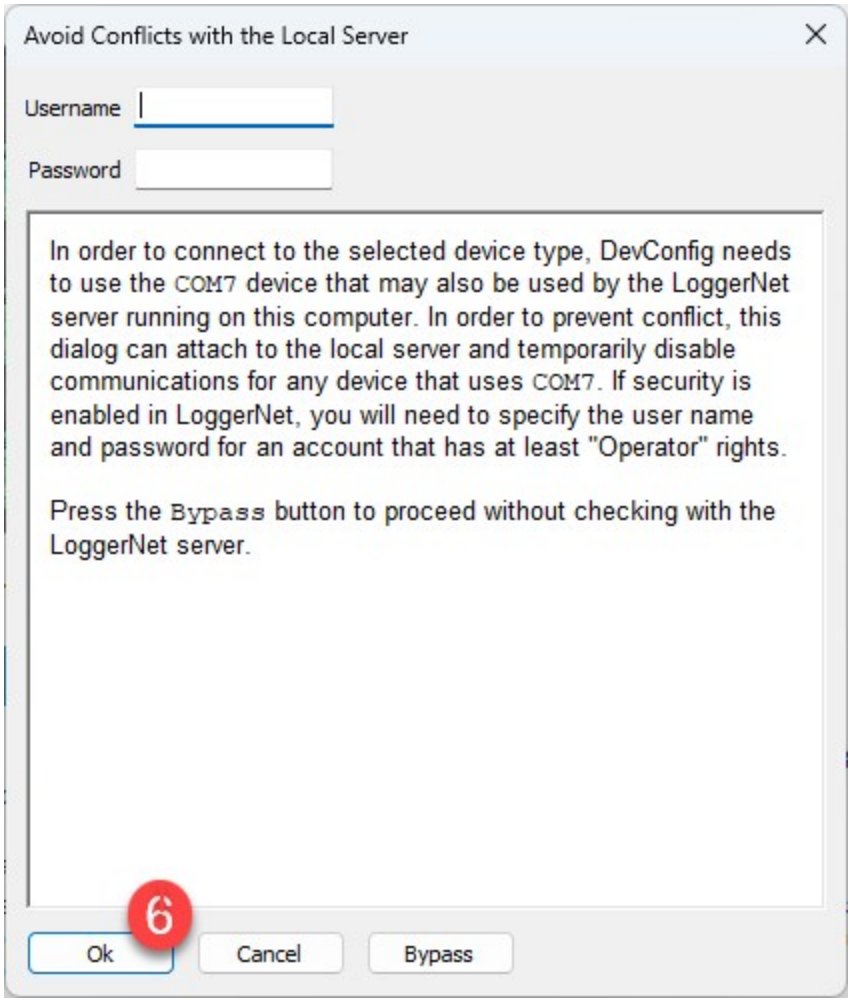
1. Ensure the data logger has the latest operating system installed. Operating systems are available from the Campbell Scientific website: https://www.campbellsci.com/downloads/operating-systems-datalogger.
2. Connect the data logger to a computer using a USB cable or TCP/IP connection.
3. Open ***Device Configuration Utility*** on the computer.

4. Select the appropriate data logger from the left side, then select the correct communications port or IP address to connect.



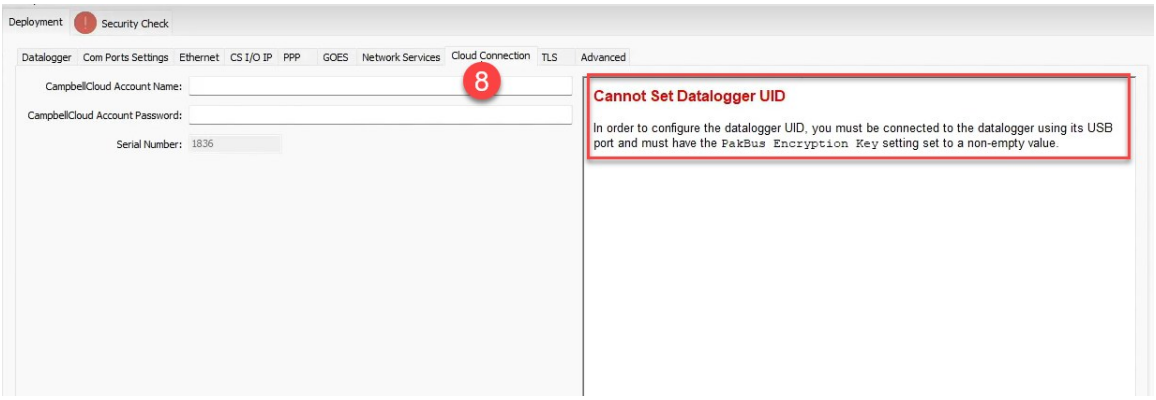
5. Click **Connect**.

6. If *Device Configuration Utility* was opened from *LoggerNet* or *PC400*, a window opens titled **Avoid Conflicts with the Local Server**. Click **OK**.



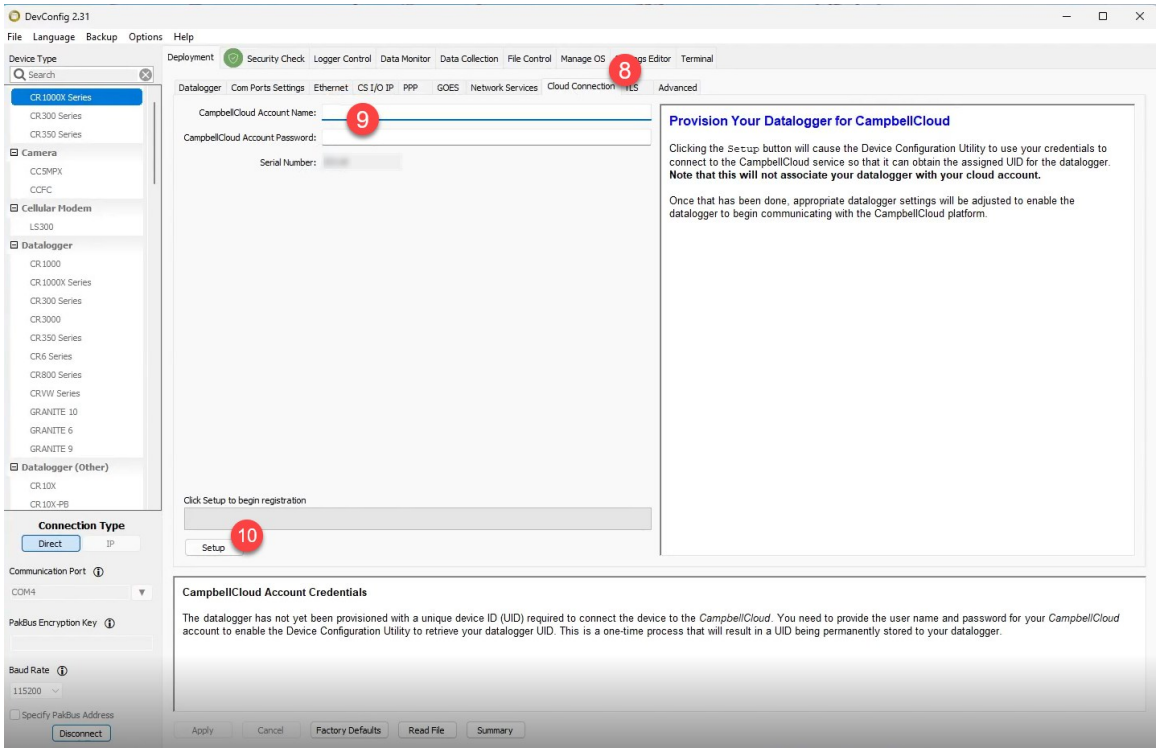
7. *Device Configuration Utility* will identify the PakBus address of the data logger and connect to it. Once connected to the data logger, *Device Configuration Utility* will open the **Datalogger** tab under **Deployment**.

8. Select the **Cloud Connection** tab.



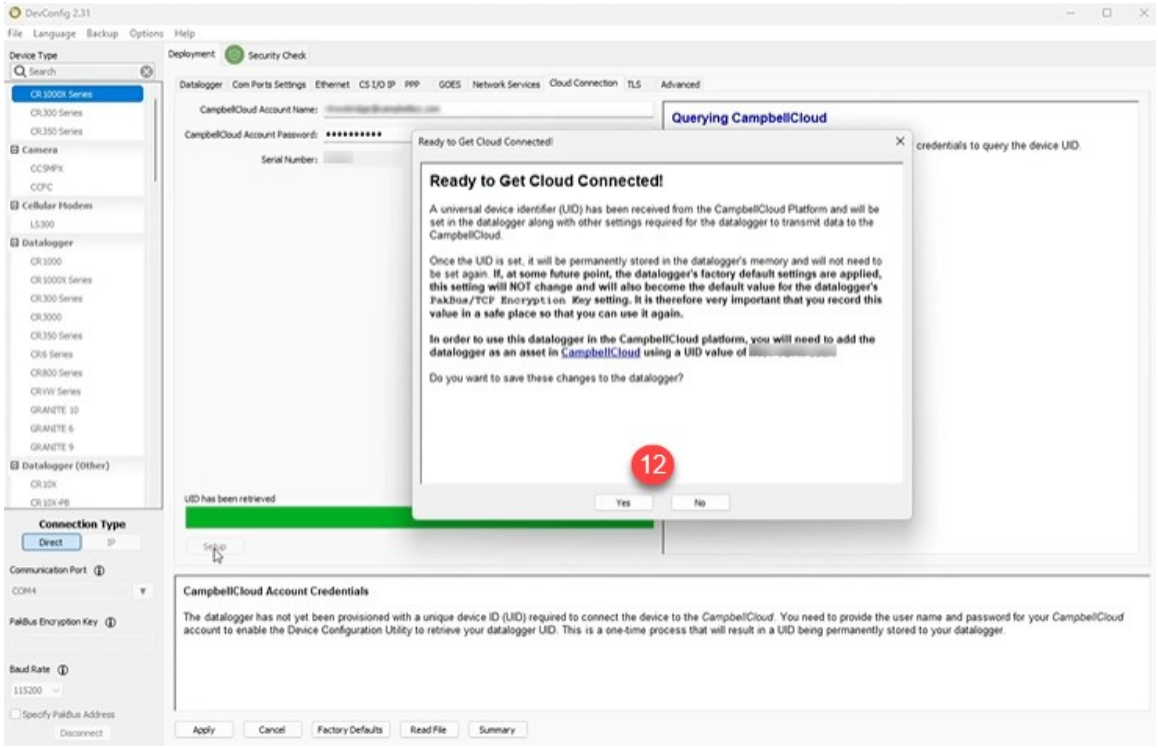
> **NOTE:**
> If a **PakBus Encryption Key** is not set, ***Device Configuration Utility*** will prompt you to set one before obtaining a UID. To do this, navigate back to the **Deployment** tab, click **Edit** next to **PakBus Encryption Key** and enter a key. Once a **PakBus Encryption Key** is set, it will be required for users and other devices to communicate with the data logger. Be sure to record it for future reference. Click **OK**, then **Apply** and **Confirm** your setting changes. Finally, click **OK** on the resulting dialog box and return to Step 5.

9. To receive a UID, enter a valid CampbellCloud Account Name and Password.

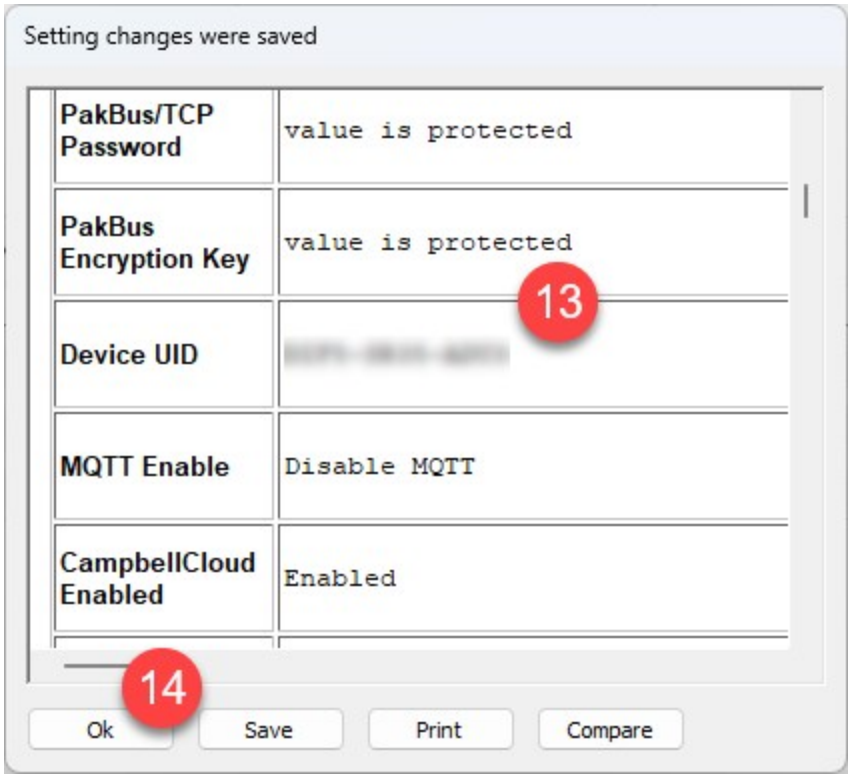


10. Click **Setup**

11. *Device Configuration Utility* uses the entered credentials to connect to CampbellCloud and retrieve the UID. A progress bar displays the progress of the UID retrieval process.
12. When complete, a new window opens with information about the UID, including the new UID number. The window includes a prompt to save the changes to the data logger. Click **Yes** to close the window.



13. Another window opens with the new data logger configuration. Scroll down through the settings to find the UID number. Record this number for future reference.



14. Click **OK**

The data logger now has a UID assigned, enabling its use with CampbellCloud.

Refer to Onboarding a data logger to CampbellCloud (p. 72) for the next steps to configure a data logger for publishing to CampbellCloud.

For more information on obtaining a UID, watch an instructional video at: Unique Identification Number (UID) and new security features.

7.1.4 *Device Configuration Utility* Security Check

A Security Check is provided through ***Device Configuration Utility***, starting with version 2.29. This check helps you identify areas where security can be improved.

All suggestions shown in ***Device Configuration Utility*** are optional and no changes will be made unless you make them. For example, ***Device Configuration Utility*** uses a simple set of criteria to suggest a strong password. If you have your own criteria, you can use it. Because every deployment can be different, ***Device Configuration Utility*** will provide you with the information you need to ensure your data logger security is optimized for your application.

In general, green, blue, and red icons indicate password strength.

- Green: strong password
- Blue: weak password
- Red: no password set

A strong password has the following:

- Eight or more characters
- One upper case letter
- One lower case letter
- One digit
- One special character

The green, blue, and red icons may also show the potential severity of a security vulnerability.

- Green: good, no action needed
- Blue: advisory information
- Red: action recommended

7.1.4.1 PakBus

PakBus encryption is the best data logger option to secure PakBus communications. For data loggers that have a UID, **PakBus Encryption** is enabled by default. The default **PakBus Encryption Key** is the UID.The PakBus Encryption (AES-128) Key can be changed or cleared (deleted) in *Device Configuration Utility*.

> CAUTION:
> Passwords may be changed from the default UID. However, when the data logger is reset to factory defaults or a new OS is sent, the default password will revert to the UID.

A PakBus Encryption Key forces PakBus data to be encrypted during transmission. The Security Check will check if the data logger has a PakBus Encryption Key set and a PakBus/TCP Password.

When neither of these is set, the Security Check will indicate that action is recommended.

The Security Check will not suggest setting a PakBus/TCP password if PakBus Encryption is already enabled. However, a PakBus/TCP password can be used with PakBus encryption.

7.1.4.2 Web services

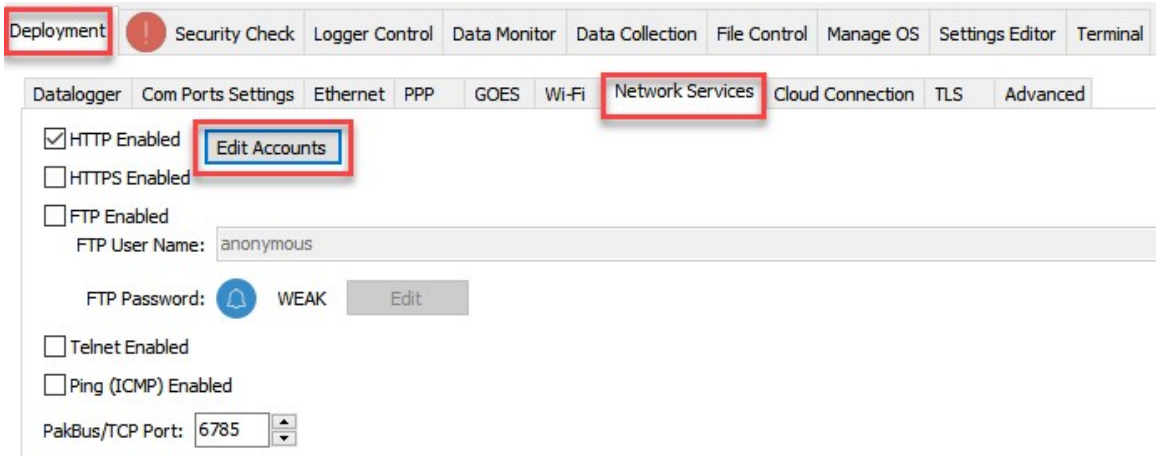
The **Security Check** will check to see if HTTP or HTTPS is enabled. If these protocols are not being used but are enabled, we recommend disabling them.

HTTP

Anonymous HTTP access is disabled by default. HTTP will be accessible with **admin** as the username and *UID* as the admin password.

> **CAUTION:**
> Passwords may be changed from the default UID. However, when the data logger is reset to factory defaults or a new OS is sent, the default password will revert to the UID.

HTTP is an insecure protocol and can allow access to data logger data, settings, and programming. HTTP should only be enabled if it is required, and the data logger is on a secure network. If the data logger is not using web API or a hosted web page, then HTTP should be disabled. If HTTP protocol is used, user accounts should be configured with user names and passwords. See ***Device Configuration Utility*** > **Deployment** > **Network Services** > **Edit Accounts** to make changes.



See Web interface (p. 27) for more information.

HTTPS

HTTPS is a secure protocol, but it is disabled by default. It should only be enabled if necessary, as it grants access to data logger data, settings, and programming. If the data logger is not using web API or a hosted web page, then HTTPS should be disabled. If you do enable HTTPS, ensure that user accounts are also set up for security. Expect delays when using HTTPS, especially when first connecting.

7.1.4.3 Network services

The Security Check will check to see if any of the following network services are enabled. These services can be used to discover your data logger on an IP network. See ***Device Configuration Utility*** > **Deployment** > **Network Services** tab, to make changes.

> **NOTE:**
> FTP, Telnet, and Ping services are disabled by default.

FTP

FTP is an insecure protocol that allows access to data logger data and files. FTP is disabled by default. However, when enabled, the UID will be the default FTP password.

> **CAUTION:**
> Passwords may be changed from the default UID. However, when the data logger is reset to factory defaults or a new OS is sent, the default password will revert to the UID.

FTP should only be enabled if it is required. If the FTP file transfer is not needed, then FTP should be disabled. Some data loggers support using SFTP (only as a client); this can improve file transfer security. Consider using a public/private key pair for SFTP authentication. Load a **.PEM** format file through the *Device Configuration Utility* **Settings Editor** > **Advanced** tab.

- **Maximum key file size**: 4 KB public, 4 KB private
- **Key exchange methods**:
  - ecdh-sha2-nistp256
  - ecdh-sha2-nistp384
  - ecdh-sha2-nistp521
  - diffie-hellman-group-exchange-sha256
  - diffie-hellman-group16-sha512
  - diffie-hellman-group18-sha512
  - diffie-hellman-group14-sha256
  - diffie-hellman-group14-sha1
  - diffie-hellman-group1-sha1
  - diffie-hellman-group-exchange-sha1
- **Host key types**:
  - ecdsa-sha2-nistp256
  - ecdsa-sha2-nistp384
  - ecdsa-sha2-nistp521
  - ecdsa-sha2-nistp256-cert-v01@openssh.com

        - ecdsa-sha2-nistp384-cert-v01@openssh.com
        - ecdsa-sha2-nistp521-cert-v01@openssh.com
        - rsa-sha2-512
        - rsa-sha2-256
        - ssh-rsa
        - ssh-rsa-cert-v01@openssh.com
    - Supported ciphers:
        - aes256-ctr
        - aes192-ctr
        - aes128-ctr
        - aes256-cbc
        - aes192-cbc
        - aes128-cbc

For more information, see How to Generate SFTP Keys Easily.

Telnet

Telnet, disabled by default, is an insecure protocol. It should only be enabled if it is required, and the data logger is on a secure network. PakBus is the preferred way of accessing the data logger terminal interface.

Ping

Ping, disabled by default, makes the data logger visible to network scans. It should only be enabled if it is required.

7.1.4.4 Operating System Status

The Security Check will check to see if your data logger is running the latest operating system. Newer operating systems may have enhanced security measures. See Updating the operating system (p. 157) for more information.

7.1.5 Other security measures reviewed by *Device Configuration Utility*

Many of these settings can be accessed from the ***Device Configuration Utility*** **Settings Editor** tab. **Settings** are organized in tabs and can be searched for.

7.1.5.1 PakBus TCP Enabled

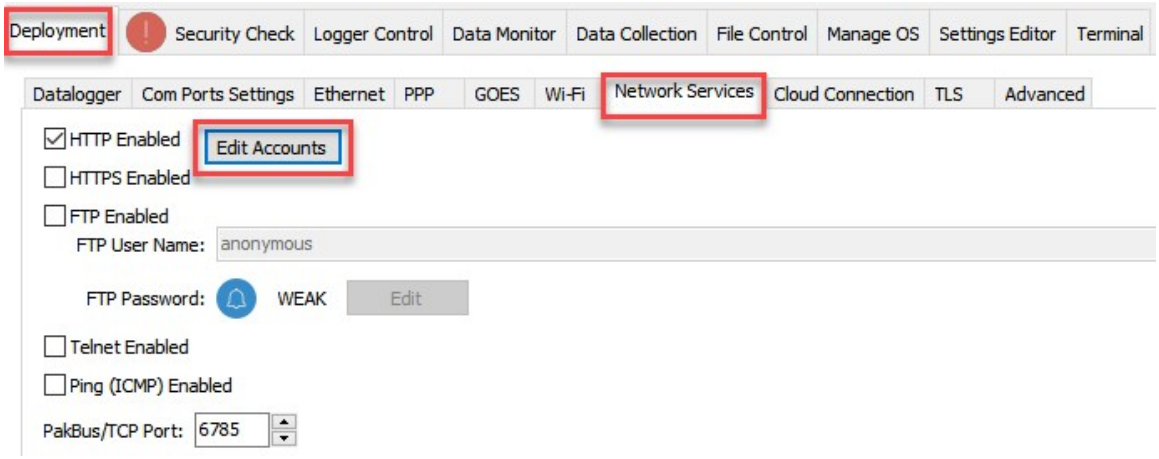
By default, PakBus TCP communications are enabled. Normally this would not be disabled as it would prevent data logger support software from connecting to a data logger using TCP.

See ***Device Configuration Utility*** > **Settings Editor** > **Network Services** tab.

7.1.5.2 Accounts editor

The Security Check only checks password strength when entering passwords through **Edit Accounts** in ***Device Configuration Utility***. It does no other checking.

See ***Device Configuration Utility*** > **Deployment** > **Network Services** > **Edit Accounts** to make changes.



7.1.5.3 IP Broadcast Filtered

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 to one, the data logger will fail to respond to the broadcast requests.

See ***Device Configuration Utility*** > **Settings Editor** > **Advanced** tab.

7.1.5.4 Other communications protocols

Specific protocols such as PPP or MQTT have settings that involve security. The Security Check does not consider these settings other than password strength.

7.1.6 TLS

Transport Layer Security (TLS) is an internet communications security protocol. TLS settings are necessary for **server** applications, not for client applications. The primary reason for using TLS is to encrypt communications between a server and its clients. Using TLS is recommended when connecting to a data logger over an IP connection using the web interface. TLS does not affect PakBus communications.

Example server application instructions include:

- HTTPS server
- `DNP()` using the optional **DNPTLS** parameter

Example client application instructions include:

- `HTTPGet()`, `HTTPPut()` and `HTTPPost()`
- `EmailRelay()`
- `EmailSend()` and `EmailRecv()`
- `FTPClient()`
- `MQTTConnect()`
- `MQTTPublishTable()`
- `MQTTPublishConstTable()`

CSI Web Server can also use TLS.

> **NOTE:**
> For enhanced security, TLS settings are only shown in *Device Configuration Utility* when using a direct USB connection, or an IP connection using PakBus Encryption.

Use the following steps to configure TLS:

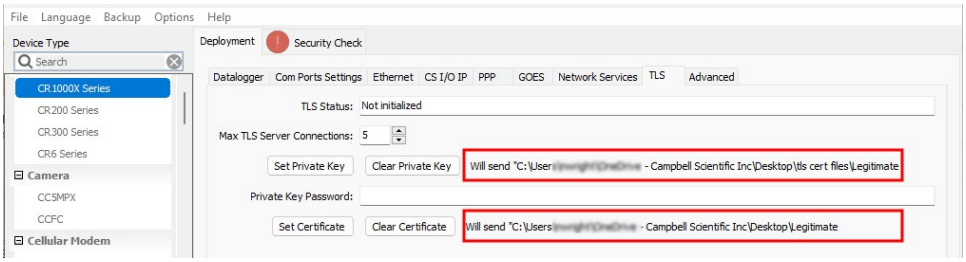
1. Use the *Device Configuration Utility* to enable HTTPS and disable HTTP. See **Deployment** > **Network Services** tab.
2. Use the *Device Configuration Utility* to enable and set up TLS. See **Deployment** > **Datalogger** > **TLS** tab.

a. Increase the number of **Max TLS Server Connections** to greater than zero. Each additional connection uses about 20 KB of memory. For general use, such as publishing web pages, use a minimum of five connections. Add more if multiple users may access the hosted web pages at the same time. See Web interface (p. 27).

b. Use **Set Private Key** and **Set Certificate** to upload files in **.PEM** format. These can either be self-signed or issued from a trusted third party organization. See Obtaining certificate and private key (p. 59) for more information.

   - **Maximum key file size**: 4 KB public, 4 KB private

   Review the **Will send *file path*** message to ensure you have the correct files.



3. **Apply** to save your changes.
4. Confirm your TLS security settings by connecting to the data logger using a web browser. See Web interface (p. 27). This connection can initially take up to 30 seconds as the data logger negotiates the TLS with the web browser. If the default data logger web page loads then TLS has been set up correctly.

> **NOTE:**
> If the certificates uploaded to the data logger are from an unknown source, such as most self-signed certificates, the web browser will likely display a warning. If the issuer can be trusted, this warning can be bypassed.

7.1.6.1 Obtaining certificate and private key

This section is provided as general guidance only. Have your IT department provide you with the required certificate and key files, or work with them to obtain them.

From a Certificate Authority

Some things you will need to know before starting the process with a Certificate Authority:

- Your website domain name, or common name
- Proof that you control the domain. This could include the email associated with the domain name.

The general steps when using an outside source for the certificate and keys are as follows:

1. Select a Certificate Authority (CA) such as DigiCert, Symantec, or GoDaddy, to generate your certificate and key files.

   > NOTE:
   > Generally there is a cost associated with the this process, and it may take several days. It is common to refile the application several times to get the correct files in the correct format.

2. Create an account with your selected CA. Sign in.
3. Generate a private key and Certificate Signing Request (CSR). Save these files to a secure location on your computer. Some Certificate Authorities may offer to generate these for you. If not, then they will require the CSR and private key you generated.

   > NOTE:
   > This is the private key file you will need later. If the file saved has a `.txt` extension, make a copy and change the extension to `.PEM`.

   > TIP:
   > If your CA generates your private key and CSR, save a copy of both. For your security, the CA will not keep a record of either the CSR or private key. If you fail to save them, you will have to generate new ones and this will take additional time.

4. Provide proof that you control the domain. Often additional instructions are received in an email from the CA.
5. Receive the certificate from the CA. This may take two or more business days. Save this file to a secure location on your computer.
6. Verify that the key file is in `.PEM` format. The contents of a valid `.PEM` formatted key will look similar to the following when viewed as a text file. The `----BEGIN RSA PRIVATE KEY-----` header and `-----END RSA PRIVATE KEY-----` footer indicate that the key was generated in the correct format.



7. Verify that the certificate file is in `.PEM` format. The contents of a valid `.PEM` formatted certificate will look similar to the following when viewed as a text file. The `----BEGIN CERTIFICATE-----` header and `-----END CERTIFICATE-----` footer indicate that the key was generated in the correct format.



From your IT department

If your IT department provides the key and certificate files, you need to determine if the key requires a private key password. To determine if your `.PEM` formatted key requires a private key password:

1. Open the key file in a text editor.
2. The following header is an example of the key without a private key password.

   ```
   -----BEGIN RSA PRIVATE KEY-----
   MIIEpQIBAAKCAQEAo8GRTJKW+grlRfuuUNrlqCc4aodqaRnNd+L+/Wjpz
   ```

3. The following header is an example of the key with a private key password.

   ```
   -----BEGIN RSA PRIVATE KEY-----
   Proc-Type: 4,ENCRYPTED
   DEK-Info: DES-EDE3-CBC,556C1115CDA822F5
   AHi/3++BAAKCAQEAo8GRTJKW+grlRfuuUNrlqCc4aodqaRnNd+L+/Wjpz
   ```

4. If the key header is similar to that shown in step 3, you need to specify a private key password. Your IT department should provide this.

> NOTE:
> Never specify a key password if your key does not have one.

7.1.7 Additional security measures

Following are some additional security measures that may be taken to secure your data logger.

7.1.7.1 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 communications 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**.

Table 7-1: Functions affected by security codes			
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. 227) for the related fields in the Settings Editor.

7.1.7.2 CRBasic

Encrypt program files if they contain sensitive information. See CRBasic help `FileEncrypt()` or use *CRBasic Editor* > **File** > **Save and Encrypt**.

Hide program files for extra protection. See CRBasic help `FileManage()` instruction.

7.1.7.3 USB

Beginning with operating system 8.00, the data logger has the following settings to control security of the USB port. See Settings (p. 214) for more information.

USB Disable

USBDisable controls whether the USB port is enabled or disabled. If set to **0** (the default), the port is enabled. **1** disables it. If the setting is changed, a data logger reboot is required for the change to take effect.

> **NOTE:**
> When USB is disabled, you will need to use another communications method (Ethernet, serial, CR1000KD) to re-enable it.

USB Not Trusted

USBNotTrusted controls how the USB port behaves, as a COM port, with respect to security. If set to **0** (default), the port will allow PakBus communications to occur when the PakBus Encryption Key (p. 223) is set. Also, if RNDIS is enabled (USB Disable (p. 228) = **0**), no security challenges will

be given to gain access via HTTP(S). If set to **1**, PakBus communications will follow the **PakBusEncryption Key** setting and RNDIS will require login credentials.

> NOTE:
> In ***Device Configuration Utility***, RNDIS is treated an IP connection rather than a direct connection. As a result, the **USBNotTrusted** setting does not apply when using RNDIS through ***Device Configuration Utility***.

7.1.7.4 Other

Monitor your data logger for changes by tracking program and operating system signatures, as well as CPU, USR, and CRD file contents.

Secure the physical data logger and power supply under lock and key.

> WARNING:
> Some 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.

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

- Setting a PakBus/TCP password. The PakBus TCP password controls access to PakBus communications 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***.
- Disabling HTTP/HTTPS or creating a user account to secure HTTP/HTTPS (see ***Device Configuration Utility*** > **Deployment** > **Network Services** > **Edit Accounts** to make changes.
- Enabling HTTPS and disabling HTTP. To prevent data collection via the web interface, both HTTP and HTTPS must be disabled.

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 to Generate SFTP Keys Easily

7.2 Default program

Many data logger settings can be changed remotely over a communications link. This convenience comes with the risk of inadvertently changing settings and disabling

communications. For example, external cellular modems are often controlled by a switched 12 VDC (SW12) terminal. SW12 is normally off; so, if the program controlling SW12 is disabled, such as by changing a setting or sending a new operating system, the cellular modem is switched off and the remote data logger will not turn it on. This could require an on-site visit to correct the problem unless a special program called **default** has been installed.

Having a **default.CR1X** program stored on the data logger will also ensure that a non-compiling CRBasic program does not lock out a remote user.

> NOTE:
> The default program may use the extension **.CR1X**, **.CRB** or **.DLD**.

When a file named **default.CR1X** is stored on the data logger CPU: drive, it is loaded if no other program takes priority. Program execution priorities are as follows:

1. When the CR1000X/CR1000Xe powers up, it executes commands in the **powerup.ini** file (on an attached USB drive or memory card) including commands to set the CRBasic program file attributes to **Run Now** or **Run On Power-up**.
2. When the CR1000X/CR1000Xe powers up, a program file marked as **Run On Power-up** will run. If that program includes a file specified by the **Include File** setting, it will be incorporated into the program that runs.
3. If there is no program file marked **Run Now** or **Run On Power-up** (or if the program selected to run cannot be compiled), the data logger will run the program specified by the **IncludeFile** setting. For more information see IncludeFile (p. 219).
4. If the **IncludeFile** program cannot be compiled or if no program is specified, the data logger will attempt to run the program named **default.CR1X** on its CPU: drive.
5. If there is no **default.CR1X** file or it cannot be compiled, the CR1000X/CR1000Xe will not automatically run any program.

See File management via powerup.ini (p. 161) for more information.

The **default.CR1X** program generally contains instructions to preserve critical datalogger settings such as communications settings, but should not be more than a few lines of code.

CRBasic Example 1: **default.CR1X** example

```
'This program turns ON the SW12 switched
'power terminal, for 30 seconds every 60 seconds.
BeginProg
  Scan(1,Sec,0,0)
    If TimeIsBetween (15,45,60,Sec) Then SW12(SW12_1,1)
  NextScan
EndProg
```

Downloading operating systems over communications requires much of the available CR1000X/CR1000Xe memory. If the intent is to load operating systems via a communications link and have a **default.CR1X** file in the CR1000X/CR1000Xe, the **default.CR1X** program should not allocate significant memory, as might happen by allocating a large USR: drive. Also, do not auto-allocate tables in `DataTable()` instructions; if it is necessary to use `DataTable()` instructions, set small fixed table sizes. Refer to Sending an operating system to a remote data logger (p. 158) for information about sending the operating system.

Execution of **default.CR1X** at power-up can be aborted by holding down the DEL key on a CR1000KD Keyboard/Display.

7.3 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, 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 Supplies Application Note
- Battery Care Blog
- Troubleshooting Your Solar Panel blog
- Power Budget Spreadsheet
- Power Budgeting Video

See also:

- Power input (p. 12)
- Power output (p. 14)
- Power requirements (p. 253)
- Power output specifications (p. 254)

8. Installation

Campbell Scientific data loggers support research and operations all over the world in a variety of applications. The limits of the CR1000X/CR1000Xe are defined by your application needs. Therefore, every installation will be unique. See www.campbellsci.com/solutions.

> **TIP:**
> Time spent in the office, setting up and testing hardware and software, will make time in the field more efficient.

Recommended tools:

- Voltmeter
- Screwdrivers
  - Flat-blade
  - Phillips-head
  - Small flat-blade
- Wire cutter/stripper
- Crescent wrench
- Pliers
- Pad and pen
- Laptop computer, fully charged, with software and drivers installed
- USB cable

Tools required to install a Campbell Scientific tripod or tower:

- Shovel
- Rake
- Open-end wrench set
- Socket wrench set
- Magnetic compass
- Tape measure
- Nut driver
- Level
- Sledgehammer
- Pliers
- Flat-bladed screwdrivers
- Phillips screwdrivers

For more information, watch a video at: http://www.campbellsci.com/videos/toolbox-for-installation-and-maintenance.

8.1 Field work

Field installation is site- and application- specific. This section lists resources to aid with system installation.

- Data logger enclosures (p. 68)
- Grounds (p. 15)
- Electrostatic discharge and lightning protection (p. 69)
- Assembling a CM110 Series Tripod
- Weather Station Siting and Installation Technical Paper
- Protect Station from Birds Blog
- Tripod Manual
- Tripod Installation Videos
- Tower Manual (UT20 and UT30)
- UTBASE Installation Video
- Surge Protector Kits: Installation and Troubleshooting White Paper
- Preparing for a Site Visit

8.2 Data logger enclosures

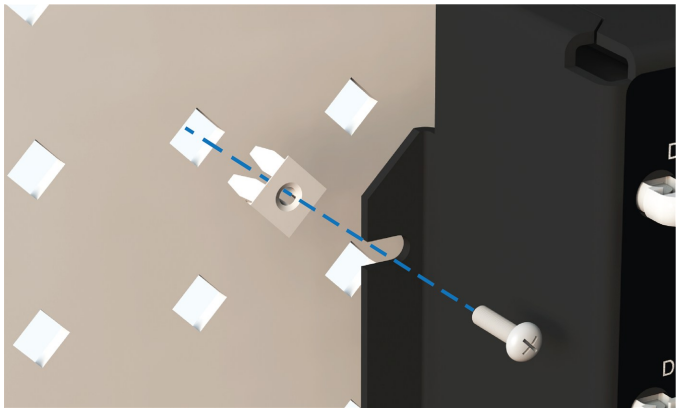
The data logger and most of its peripherals must be protected from moisture and humidity. Moisture in the electronics will seriously damage the data logger. In most cases, protection from moisture is easily accomplished by placing the data logger in a weather-tight enclosure with desiccant and elevating 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.

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

1. Insert the included nylon anchors into the backplate. Position them to align with the mounting holes on the base of the data logger.

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

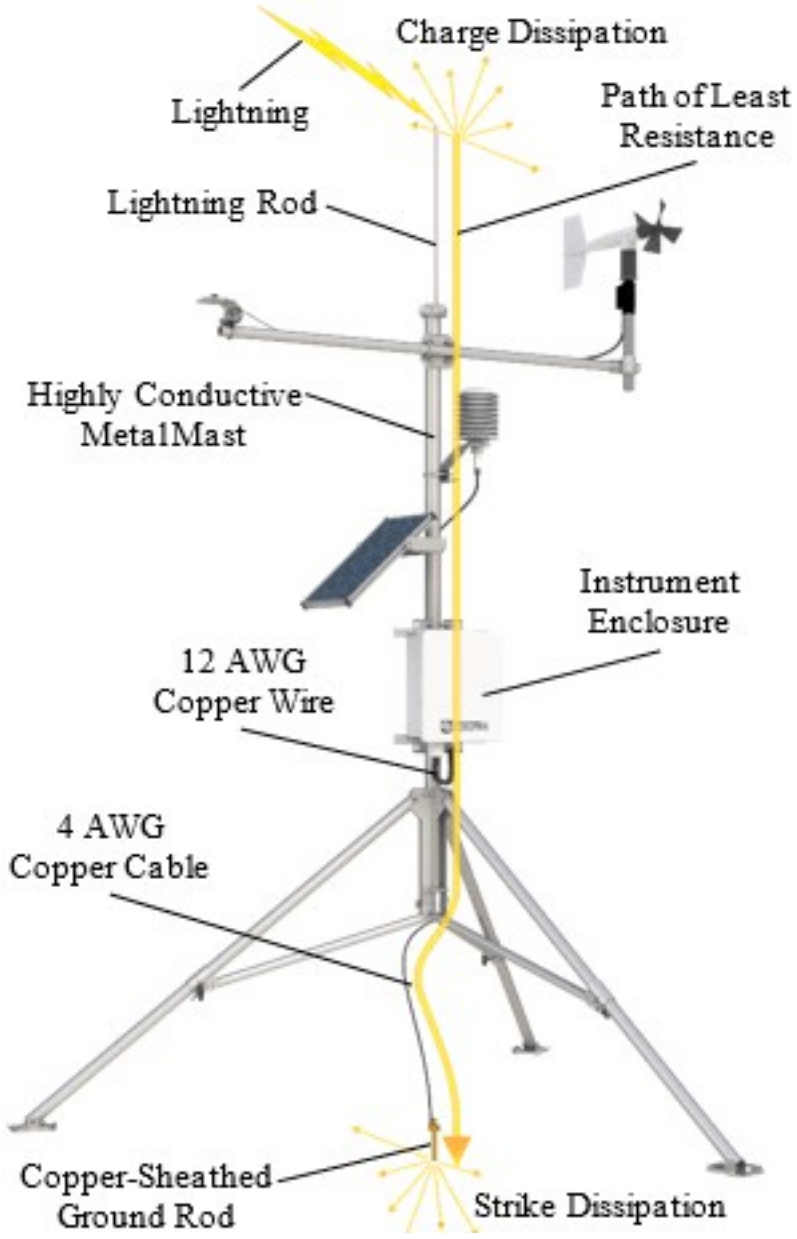


See also Physical specifications (p. 253).

8.3 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. 15).

See also

- Understanding the Importance of Grounding
- Connecting a Data Acquistion System to Earth Ground
- Avoiding Ground Loops

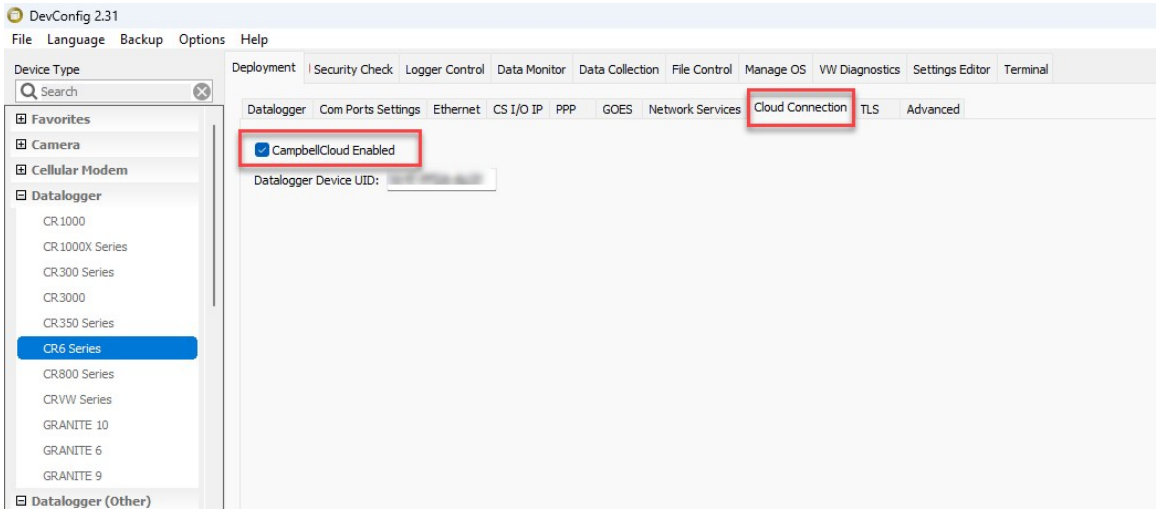
9. Onboarding a data logger to CampbellCloud

A Campbell Scientific data logger with a UID can be configured to publish to CampbellCloud.

> **NOTE:**
> If your data logger was not configured with a UID from the factory, see: Obtaining a Unique Identification Number (UID) (p. 47)

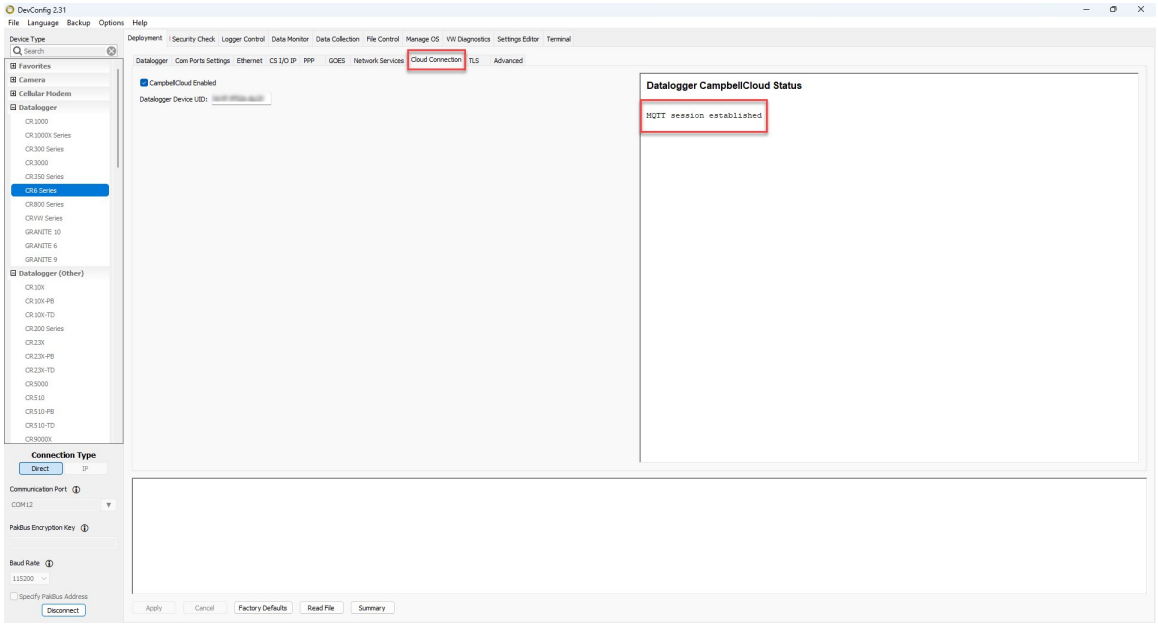
Follow these steps to prepare your data logger for CampbellCloud:

1. Ensure the data logger has the latest operating system installed. Operating systems are available from the Campbell Scientific website: https://www.campbellsci.com/downloads/operating-systems-datalogger.
2. Ensure that your data logger has internet access.
3. Connect to your data logger with ***Device Configuration Utility***.
4. Navigate to the **Cloud Connection** tab.
5. Select the **CampbellCloud Enabled** checkbox.



6. **Apply** the settings. The data logger will then restart and begin connecting to CampbellCloud, including authentication with the platform. This process may take a few minutes.

7. Reconnect to your data logger with ***Device Confguration Utility***. Navigate to the **Cloud Connection** tab. The **Datalogger CampbellCloud Status** box on this tab displays the status of MQTT communications. Once the device is successfully connected, you should see **MQTT session established** in this box.



The above steps will configure the device for communications with CampbellCloud. To start publishing data to CampbellCloud, you will need to follow the steps below and refer to the associated topics in the CampbellCloud Help:

- **Onboard the data logger to CampbellCloud:** Refer to Adding an asset for detailed instructions on onboarding your data logger.
- **Link the data logger to a station:** Use the process described in Adding a station to a network to connect the data logger to a station.
- **Activate a subscription:** Ensure the data logger has an active subscription by following the steps in Ordering and activating subscriptions.
- **Select the data to be published:** See MQTT Auto Publish Data versus MQTTPublishTable() (p. 74) for more information.

9.1 MQTT Auto Publish Data versus MQTTPublishTable()

There are two ways to publish data to CampbellCloud: 1) Auto-publish data by enabling the **MQTT Auto-Publish Data** setting in the data logger settings, or 2) use the `MQTTPublishTable()` instruction in the data table.

To enable auto publishing:

In ***Device Configuration Utility***, on the **MQTT** tab, set **MQTT Auto-Publish Data** to **Enabled**.

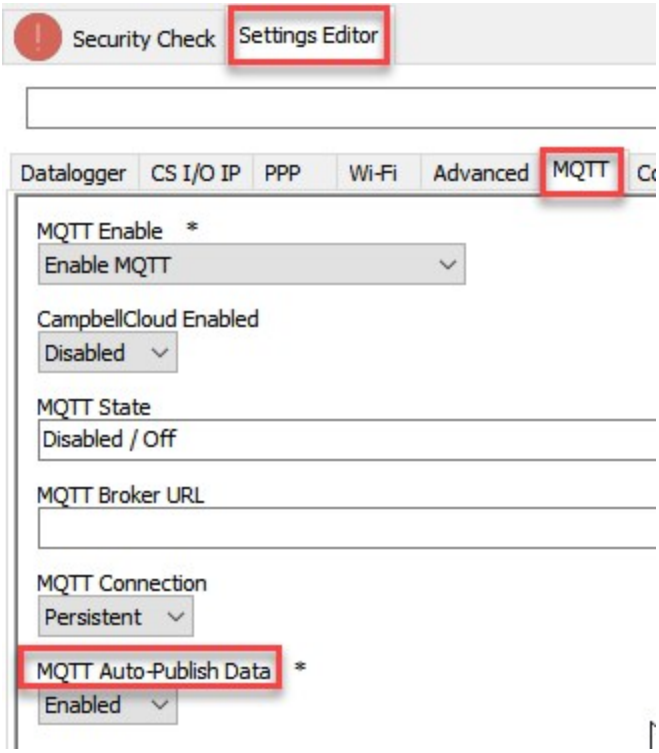


*Figure 9-1. MQTT Auto-Publish Data setting in **Device Configuration Utility***

Data tables intended for publishing must include a `DataInterval()` instruction. For example:

```
DataTable (Test,1,-1)
  DataInterval (0,10,Min,10)
  Minimum (1,Batt_volt,FP2,False,False)
  Sample (1,PTemp,FP2)
EndTable
```

By default, the fastest auto-publish interval is 10 minutes. Data is published as follows:

- If the `DataInterval()` is less than 1 minute, the table is excluded from publishing.
- If the `DataInterval()` is between 1 and 10 minutes, data stored in the data table is published every 10 minutes.
- If `DataInterval()` is greater than 10 minutes, data is published at the specified interval.
- To publish data more frequently than the 10-minute auto-publish interval, disable **MQTT Auto-Publish Data** and use the `MQTTPublishTable()` instruction in the data table; specifying the desired publish rate. Set the `OutputFormat` parameter in the `MQTTPublishTable()` to 2, **GeoJSON**.

> **NOTE:**
> Once `MQTTPublishTable()` is included in a data table in a CRBasic program, **Auto-Publish** is no longer applicable.

10. Working with data

10.1 Default data tables

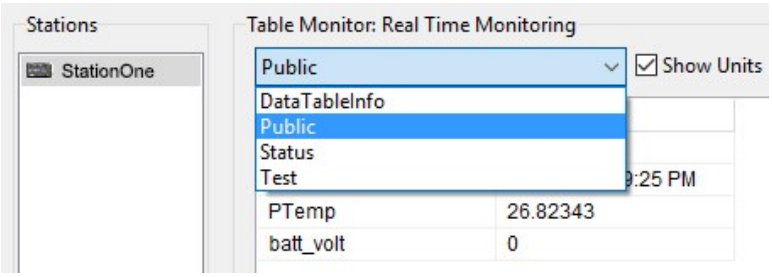
By default, the data logger includes three tables: **Public**, **Status**, and **DataTableInfo**. Each of these tables only contains the most recent measurements and information.

- 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.

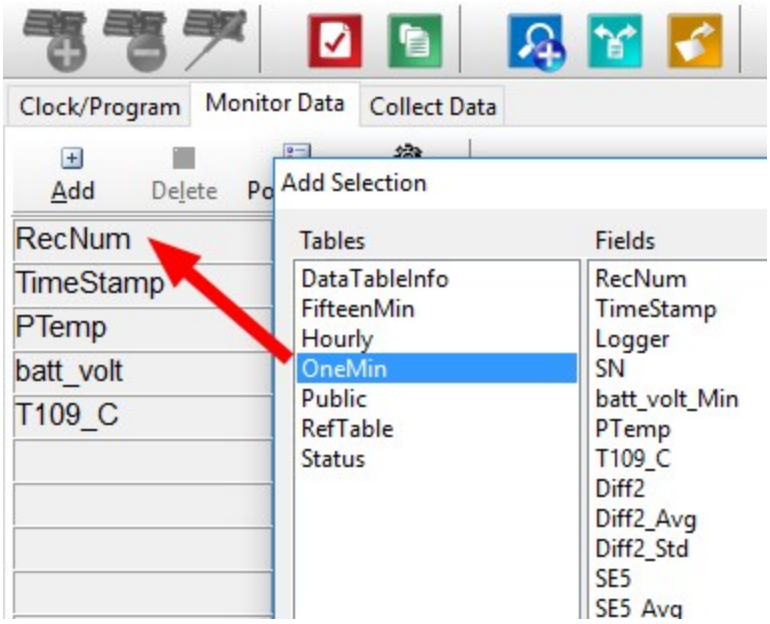
For information on collecting your data, see Collecting data (p. 77).

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, then select the data logger from the **Stations** list, then click **Connect**. Once connected, select a table to view in the **Table Monitor**.



PC400 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.



10.2 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. 85) 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 instruction or follow the Collect Data Tutorial.

10.2.1 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. 78).

10.2.2 Collecting data using *PC400*

1. Click **Connect** on the main ***PC400*** window.
2. Go to the **Collect Data** tab.

3. By default, all output tables set up in the data logger program are selected for collection. Typically, the default tables (DataTableInfo, Public, and Status) are not collected.
4. Select an option for **What to Collect**. Either option creates a new file if one does not already exist.
    - **New data from data logger (Append to data files)**: This is the default, and most often used option. Collect only the data, in the selected tables, stored since the last data collection from this instance of *PC400* and append this data to the end of the existing files on the computer.
    - **All data from data logger (Overwrite data files)**: Collects all of the data in the selected tables and overwrites (or replaces) the existing data files on the computer.
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. 78).

10.3 Viewing historic data

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 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. 77) for more information).
2. Open View Pro:
    - *LoggerNet* users click **Data** then **View Pro** on the *LoggerNet* toolbar.
    - *PC400* users click **View Data Files via View Pro**.
3. Click **Open**, navigate to the directory where you saved your tables (the default directory is **C:\Campbellsci***[your data logger software application]*). For example: navigate to the **C:\Campbellsci\LoggerNet** folder and select **OneMin.dat**.
4. Click **Open**.

10.4 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.

See the ***CRBasic Editor*** help for additional data types and formats, and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

Table 10-1: Data types, ranges and resolutions

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
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

10.4.1 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 CR1000X/CR1000Xe Operating System can do math on integers faster than with `Float`s, 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 CR1000X/CR1000Xe 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 `AVW200()`, `GPS()`, 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.com/crbasic/cr1000x/.

For more information on variables, see Fundamentals of CRBasic Programming Part 1: Variables.

10.4.2 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 Floatdata type.

10.4.3 Data storage

Data can be stored in IEEE4 or FP2 formats. The format is selected in the program instruction that outputs the data, such as Minimum() and Maximum().

Additionally, data can be stored in IEEE8 format when high precision is needed. For more information on double-precision math, watch an instructional video at: http://www.campbellsci.com/videos/double-precision.

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).

Table 10-2: Resolution and range limits of FP2 data

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).

Table 10-3: FP2 decimal location

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**.

10.5 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. 88). 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, which includes the **Public**, **Status**, **DataTableInfo**, and **ConstTable**. You can retrieve data based on a schedule or by manually choosing to collect data using data logger support software. See Collecting data (p. 77).

Table 10-4: Example data				
TOA5, MyStation, CR1000X, 1142, CR1000X.Std.01, CPU:MyTemperature.CR1X, 1958, OneMin				
TIMESTAMP TS	RECORD RN	BattV_Avg Volts	PTemp_C_Avg Deg C	Temp_C_Avg 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

10.5.1 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.

10.5.1.1 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. Other formats include: TOB1 and TOACI1.
- **MyStation** - Station name. Changed via ***LoggerNet*** **Setup**, ***Device Configuration Utility***, or CRBasic program.
- **CR1000X** - Data logger model.
- **1142** - Data logger serial number.
- **CR1000X.Std.01** - Data logger OS version.
- **CPU:MyTemperature.CR1X** - Data logger program name. Changed by sending a new program (see Sending a program to the data logger (p. 43) 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. 43) for more information).

- OneMin - Table name as declared in the running program (see Creating data tables in a program (p. 85) 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. 84).

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. 83).

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.

Table 10-5: Data processing abbreviations

Data processing name	Abbreviation
Totalize	`Tot`
Average	`Avg`
Maximum	`Max`
Minimum	`Min`
Sample at Max or Min	`SMM`
Standard Deviation	`Std`
Moment	`MMT`
Sample	No abbreviation
Histogram	`Hst`

Table 10-5: Data processing abbreviations	
Data processing name	Abbreviation
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

10.5.1.2 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. 83), **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.

> **NOTE:**
> TableName.Timestamp syntax can be used to return the timestamp of a data table record, expressed either as a time into an interval (for example seconds since 1970 or seconds since 1990) or as a date/time string. For more information,
> see: https://www.campbellsci.com/blog/programmatically-access-stored-data-values.

10.6 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 Creating a Short Cut data logger program (p. 40) 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.

```
Public 'Declare Public Variables

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

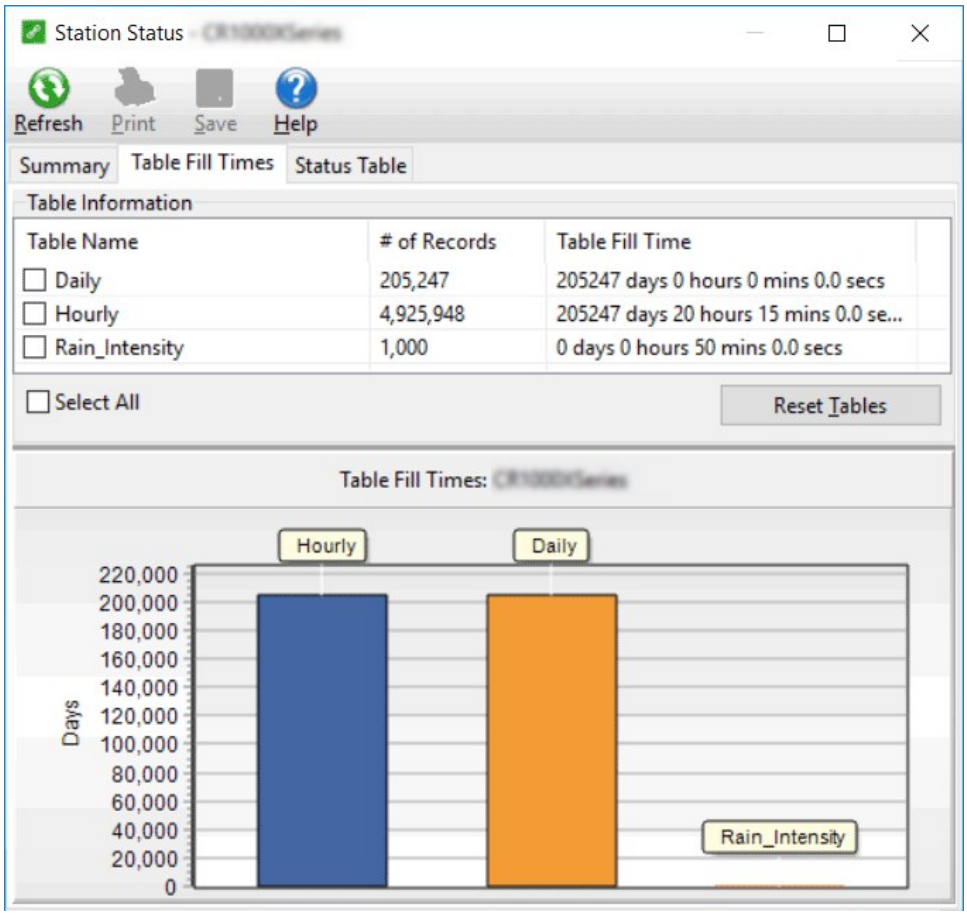
'Main Program
BeginProg
```

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 program scan.

See the *CRBasic Editor* help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

For additional information on data tables, watch Fundamentals of CRBasic Programming Part 2: Tables and Fundamentals of CRBasic Programming Part 5: Using Conditional Tables.

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. 166) for more information). An example of the Table Fill Times tab follows. For information on data table storage see Data memory (p. 88).



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

11. Data memory

The data logger includes three types of memory: SRAM, Flash, and Serial Flash. A memory card slot is also available for an optional microSD card. Note that the data logger USB port does not support USB flash or thumb drives (see Communications ports (p. 17) for more information).

- Total onboard: 128 MB of flash + 4 MB battery-backed SRAM
  - Data storage: 4 MB SRAM + 72 MB flash (extended data storage automatically used for auto-allocated Data Tables not being written to a card)
  - CPU drive: 30 MB flash
  - OS load: 8 MB flash
  - Settings: 1 MB flash
  - Reserved (not accessible): 10 MB flash
- Data storage expansion: Removable microSD flash memory, up to 16 GB

11.1 Data tables

Measurement data is primarily stored in data tables within SRAM. 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.com/crbasic/cr1000x/.

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. 202).

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

11.2 Memory allocation

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

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

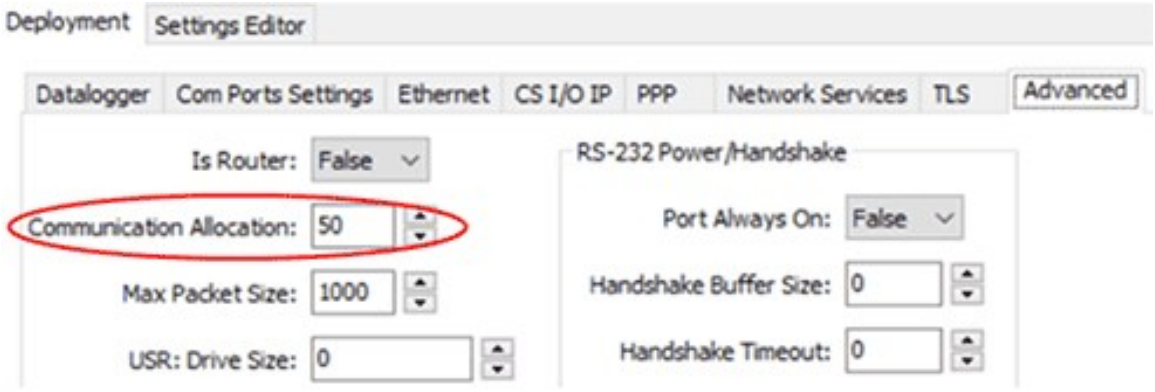
11.3 SRAM

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

The structure of the 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 neighbor lists. This memory also stores dynamic properties such as known routes and communications timeouts.
- CRBasic Program Operating Memory: This memory stores the currently compiled and running user program. This sector is rebuilt on power-up, recompile, and watchdog events.
- Variables & Constants: This memory stores constants and public variables used by the CRBasic program. Variables may persist through power-up, recompile, and watchdog events if the `PreserveVariables` instruction is in the running program.
- Final-Data Memory: This memory stores data. Auto-allocated tables fill whatever memory remains after all other demands are satisfied. A compile error occurs if insufficient memory is available for user-allocated data tables. This memory is given lowest priority in SRAM memory allocation.
- Communication Memory 1: Memory used for construction and temporary storage of PakBus packets.
- Communication Memory 2: Memory used to store the list of known nodes and routes to nodes. Routers use more memory than leaf nodes because routes store information about

other routers in the network. You can increase the **Communication Allocation** field in *Device Configuration Utility* to increase this memory allocation.



- **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**.

11.3.1 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.

11.4 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. 157).

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.

11.4.1 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. 252) for information on data logger memory.

11.5 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.

For additional information on when to use a MicroSD card, watch Fundamentals of CRBasic Programming Part 7: Determine if External Memory is Required/Intro to MicroSD.

When storing high-frequency data, or when storing data to cards greater than 2 GB, `TableFile()` with **Option 64** is recommended to write final storage data to a card. In other applications `CardOut()` can be used to store data to a card.

> **NOTE:**
> Sub-folders are not supported.

See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

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 advantages Campbell Scientific cards have over less expensive commercial-grade cards:

- Verified compatibility with Campbell Scientific data loggers
- 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. Sometimes this error may be resolved by reformatting the card. If the error repeats itself, try an industrial-grade card. For more information on errors, see File system error codes (p. 199).

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. Note that with `TableFile()`, the number of files stored on the card is controlled by the **MaxFiles** parameter.

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.

NOTE:
A card must be exchanged before it fills, or the oldest data will be overwritten, by incoming new records, and lost. During the card exchange, once the old card is removed, the new card must be inserted before the data table in data logger CPU memory rings, or data will be overwritten and lost.

A microSD card can also facilitate the use of **powerup.ini** (see File management via powerup.ini (p. 161) for more information).

11.5.1 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. There are several ways to format cards such as using: File Control, CR1000KD, and Windows. Formatting on the data logger is recommended because this ensures correct FAT32 format.

11.5.2 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. 252) for information on maximum card size.

11.5.3 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**: This LED 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.

11.5.4 Card data retrieval

Data stored on cards can be retrieved through a communications link to the data logger or by removing the card and carrying it to a computer with a card adapter. With large files, transferring the card to a computer may be faster than collecting the data over a communications link.

For additional information on card data retrieval, watch Fundamentals of CRBasic Programming Part 9: Retrieving Data from a MicroSD Card.

CAUTION:
Removing a card while it is active can cause corrupted data and can damage the card. **Always** press the **Eject** button and wait for a green light before removing card. Do not switch off the data logger power if a card is present and active.

CAUTION:
File Control (in *LoggerNet* or *PC400*) should not be used to retrieve an open file (for example, a file created by using `CardOut()` or the latest file created by `TableFile()`, **Option 64**) from a card. Using **File Control** to retrieve the data can result in a corrupted data file. However, **File Control** can be used to retrieve closed files such as JPEG images or files (other than the latest) created by `TableFile()`, **Option 64**.

11.5.4.1 Via a communications link

Data can be copied to a computer via a communications link by using one of Campbell Scientific data logger support software packages (for example, *LoggerNet* or *PC400*). There is no need to distinguish whether the data is to be collected from the CPU memory or a card. The software package will look for data in both the CPU memory and the card.

The data logger manages data on a card as final-storage data, accessing the card as needed to fill data-collection requests initiated with the **Collect** button in data logger support software. If desired, binary data can be collected by using the **File Control** utility in data logger support software. Before collecting data this way, stop the data logger program to ensure data is not written to the card while data is retrieved; this will avoid data corruption.

Fast storage/data-collection constraints

Factors affecting how fast the data logger stores data include the data storage rate, number of table values, and number of tables. For more information, see Creating data tables in a program (p. 85).

When data logger support software collects data from ring tables that have filled, there is the possibility of missing records due to the collection process. When a ring table has filled, the oldest data is overwritten by the newest data. *LoggerNet* and *PC400* use a collection algorithm that collects data from multiple tables in small blocks as they collect from all the tables. Collection starts with the oldest data for each table.

With filled ring tables, as collection begins, the data collection software queries the data logger for the oldest data starting with the first table. When this data block is returned, the software goes to the next table and so on until all of the tables are initially collected. By the time *LoggerNet* or *PC400* make the second pass requesting more data from the tables, the possibility exists that some of that data may have been overwritten.

Normally, data is collected without gaps; however, if the data logger is storing data fast enough, it is possible to get into an always-behind scenario where the data collection never catches up and the data logger repeatedly overwrites uncollected data.

> CAUTION:
> The possibility of missing records is greater when collecting data over high-latency communications links, such as RF or busy IP networks. This is due to the high demand of communications on processor time.

11.5.4.2 Card transport to computer

With large files, transferring the card to a computer may be faster than collecting the data over a communications link.

> CAUTION:
> Removing a card while it is active can cause corrupted data and can damage the card. **Always** press the **Eject** button and wait for a green light before removing card. Do not switch off the data logger power if a card is present and active.

To remove a card, first press the **Eject** button. The data logger will copy any buffered data to the card and then power the card off. The Act LED will turn green when it is OK to physically remove the card. The card will be reactivated after 15 seconds if it is not removed.

When the card is inserted into a computer, the data files can be copied to another drive or used directly from the card just as one would from any other disk. In most cases, however, it will be necessary to convert the file format before using the data.

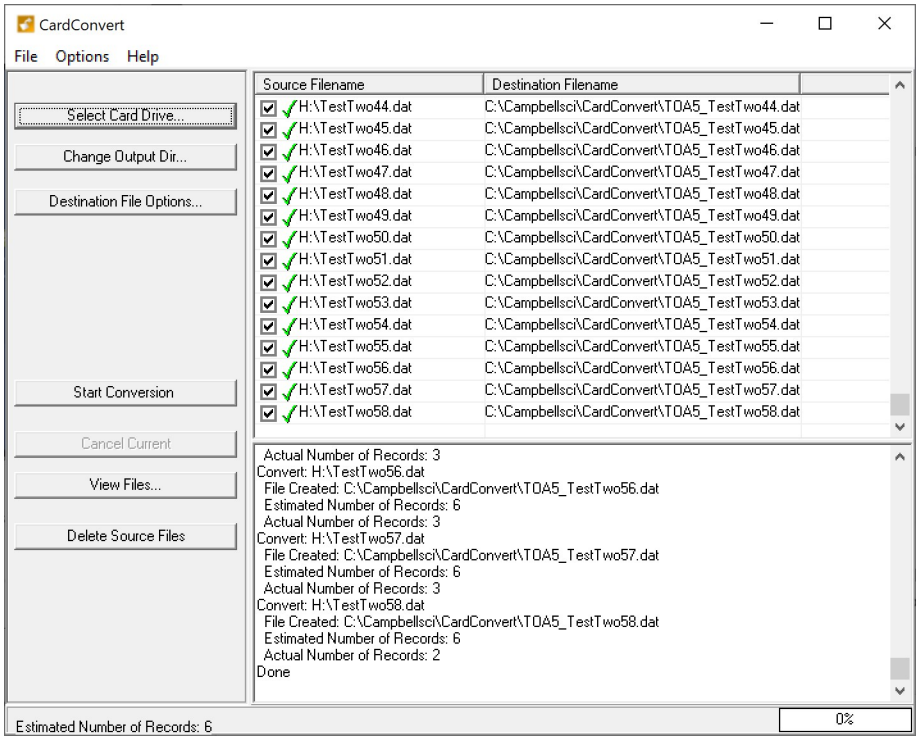
Note that for both `CardOut()` and `TableFile()` **Option 64**, data is stored on the card in binary (TOB3) format. TOB3 is a binary format that incorporates features to improve reliability of cards. TOB3 format is different from the data file formats created when data is collected via a communications link, which is ASCII (TOA5) format. Hence, data files that are read directly from

the card need to be converted into another format to be human readable. You can convert files from binary or other formats using ***CardConvert*** software that is included in your data logger support software.

Converting file formats

Use ***CardConvert*** to convert data to a different format.

1. Open ***CardConvert***.
   - On the ***LoggerNet*** toolbar select the **Data** category.
   - In ***PC400*** select the **Tools** menu.
2. Click **Select Card Drive**.
3. Select where the files to be converted are stored and press **OK**.
4. Click **Change Output Dir** and select where to store the converted files.
5. Place check marks next to the files to be converted. A default destination *filename* is given. It can be changed by right-clicking with the *filename* highlighted.
6. Press **Destination File Options** to select what file format to convert to and other options.
7. Press **Start Conversion** to begin converting files. Green check marks will appear next to each *filename* as conversion is complete. Refer to the data logger support software manual or built-in ***CardConvert*** help for more information.



Reinserting the card

If the same card is inserted again into the data logger, the data logger will store all data to the card that has been generated since the card was removed that is still in the CPU memory. If the data tables have been left on the card, new data will be appended to the end of the old files. If the data tables have been deleted, new ones will be created.

> CAUTION:
> Check the status of the card before leaving the data logger. If a card was not properly accepted, the LED will flash orange. In that case, reformat and erase all data contained on the card. Formatting or erasing a card might be done on a computer or data logger. See MicroSD (CRD: drive) (p. 91) for information on formatting a card.

Card swapping

When transporting a card to a computer to retrieve data, most users will want to use a second card to ensure that no data is lost. For this method of collection, use the following steps.

1. Insert formatted card ("card-A") into the data logger card slot. See Formatting microSD cards (p. 93).
2. Send program containing `TableFile()` or `CardOut()` instruction(s).
3. When ready to retrieve data (hours, days, or months later), press the **Eject** button. The LED will be red while the most-current data is stored to the card and then turn green. Remove the card while the LED is green.
4. Insert the clean card ("card-B").
5. Use ***CardConvert*** to copy data from card-A to computer and convert. The default ***CardConvert*** filename will be TOA5_*stationname_tablename*.dat. Once the data is copied, use Windows Explorer to **delete all data files from the card**.
6. At the next card swap, eject card-B, press the **Eject** button. The LED will be red while the most-current data is stored to the card and then turn green. Remove the card while the LED is green.
7. Insert the clean card-A.
8. Running ***CardConvert*** on card-B will result in separate data files containing records since card-A was ejected. ***CardConvert*** can increment the *filename* to TOA5_*stationname_tablename*_0.dat.

9. The data files can be joined by using text editing software such as ***WordPad*** or a spreadsheet such as ***Excel***.

CardConvert file	Card-A record numbers	Card-B record numbers
TOA5_*tablename*.dat	0-100	
TOA5_*tablename*.dat		101-1234
TOA5_*tablename*.dat	1235-….	

12. Measurements

12.1 Voltage measurements 99
12.2 Current-loop measurements 101
12.3 Resistance measurements 103
12.4 Thermocouple Measurements 110
12.5 Period-averaging measurements 111
12.6 Pulse measurements 112
12.7 Vibrating wire measurements 120
12.8 Sequential and pipeline processing modes 120

12.1 Voltage measurements

Voltage measurements are made using an Analog-to-Digital Converter (ADC). A high-impedance Programmable-Gain Amplifier (PGA) amplifies the signal. Internal multiplexers route individual terminals within the amplifier. The CRBasic measurement instruction controls the ADC gain and configuration – either single-ended or differential input. Information on the differences between single-ended and differential measurements can be found here: Deciding between single-ended or differential measurements (p. 186).

A voltage measurement proceeds as follows:

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

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

In general, use the smallest input range that accommodates the full-scale output of the sensor. This results in the best measurement accuracy and resolution (see Analog measurement specifications (p. 256) for more information).

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

> **WARNING:**
> Sustained voltages in excess of ±20 V applied to terminals configured for analog input will damage CR1000X/CR1000Xe circuitry.

12.1.1 Single-ended measurements

A single-ended measurement measures the difference in voltage between the terminal configured for single-ended input and the reference ground. For example, single-ended channel 1 is comprised of terminals **SE 1** and ⏚. Single-ended terminals are labeled in blue. For more information, see Wiring panel and terminal functions (p. 8). The single-ended configuration is used with the following CRBasic instructions:

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

See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

12.1.2 Differential measurements

A differential measurement measures the difference in voltage between two input terminals. For example, **DIFF** channel **1** is comprised of terminals **1H** and **1L**, with **1H** as high and **1L** as low. For more information, see Wiring panel and terminal functions (p. 8). The differential configuration is used with the following CRBasic instructions:

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

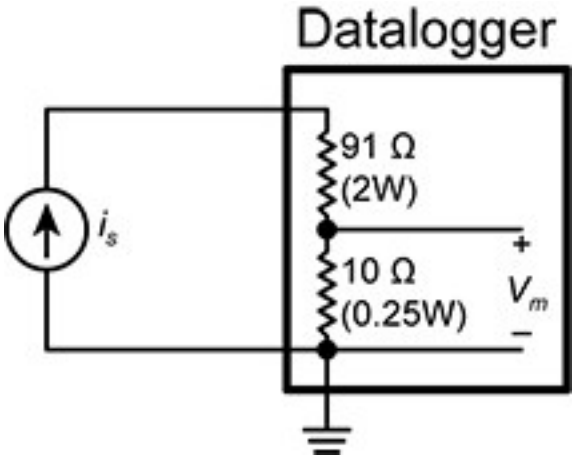
12.1.2.1 Reverse differential

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

For more information on voltage measurements, see Improving voltage measurement quality (p. 185) and Analog measurement specifications (p. 256).

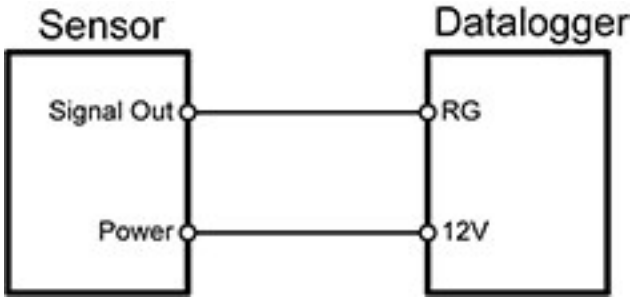
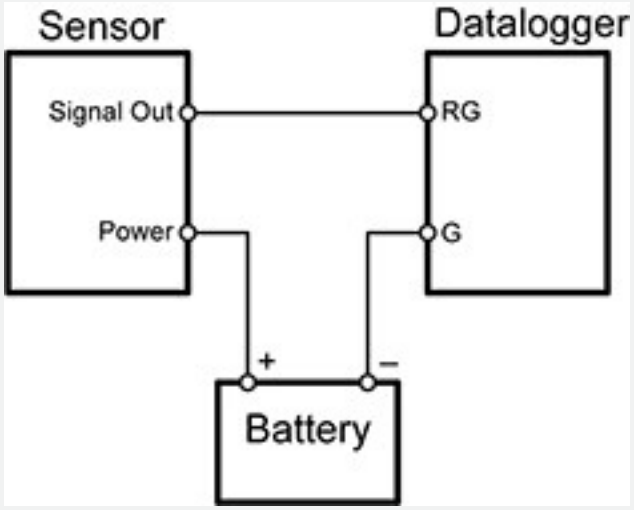
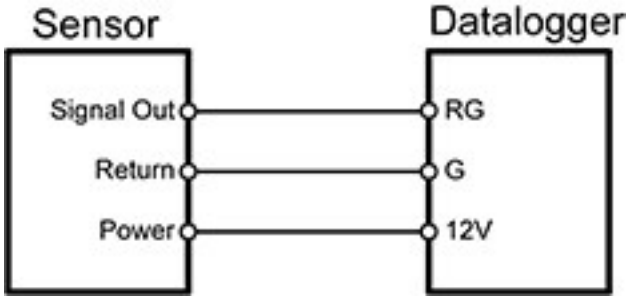
12.2 Current-loop measurements

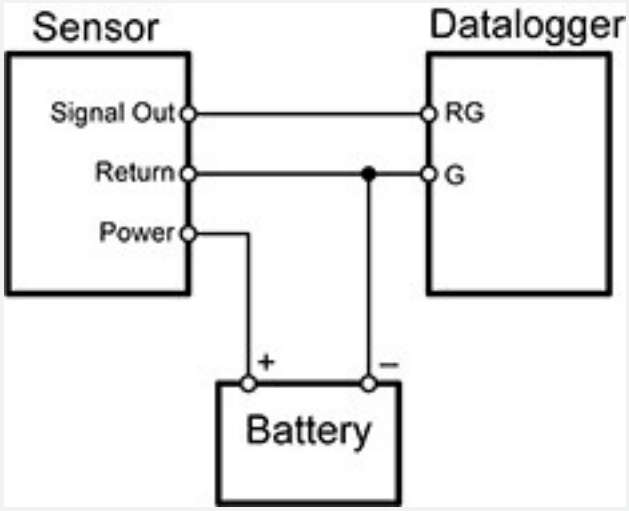
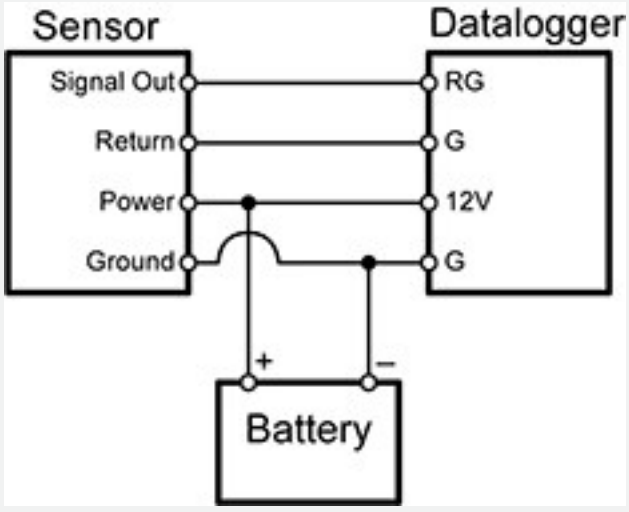
RG terminals can be configured to make analog current measurements using the `CurrentSE()` instruction. When configured to measure current, terminals each have an internal resistance of 101 Ω in the current measurement loop. The return path of the sensor must be connected directly to the **RG** terminal.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



12.2.1 Example current-loop measurement connections

The following table shows example schematics for connecting typical current sensors and devices. See also Current-loop measurement specifications (p. 260).

Sensor type	Connection example
2-wire transmitter using data logger power	
2-wire transmitter using external power	
3-wire transmitter using data logger power	

Sensor type	Connection example
3-wire transmitter using external power	
4-wire transmitter using data logger power	
4-wire transmitter using external power	

12.3 Resistance measurements

Bridge resistance is determined by measuring the difference between a known voltage applied to the excitation (input) of a resistor bridge and the voltage measured on the output arm. The data logger supplies a precise voltage excitation via VX terminals. Return voltage is measured on

analog input terminals configured for single-ended (**SE**) or differential (**DIFF**) input. The result of the measurement is a ratio of measured voltages.

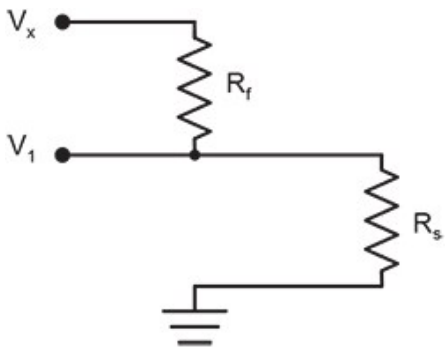
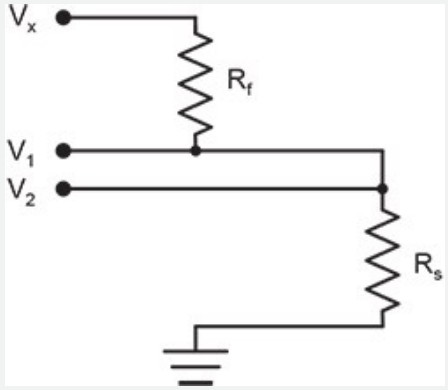
See also Resistance measurement specifications (p. 259).

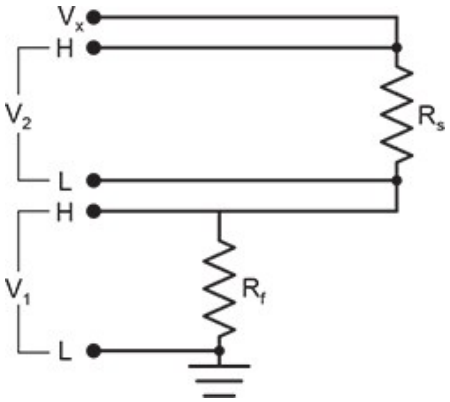
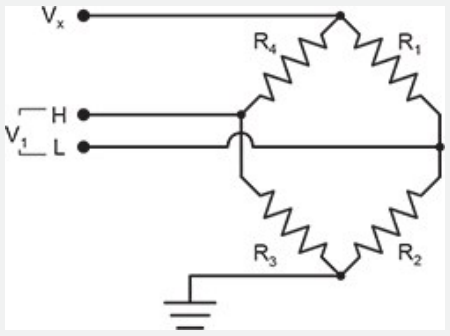
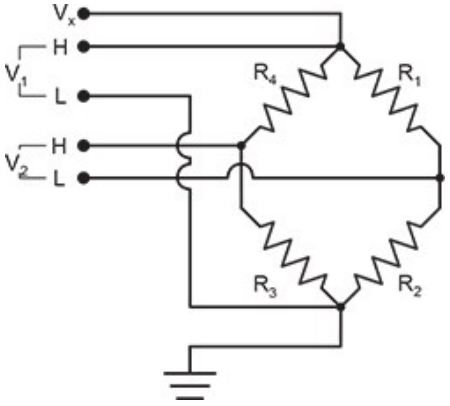
12.3.1 Resistance measurements with voltage excitation

CRBasic instructions for measuring resistance with voltage excitation include:

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

See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

Resistive-bridge type and circuit diagram	CRBasic instruction and fundamental relationship	Relational formulas
Half Bridge[1] 	CRBasic Instruction: `BrHalf()` Fundamental Relationship: X = result w/mult = 1, offset = 0 $X = \frac{V_1}{V_x} = \frac{R_s}{R_s+R_f}$	$R_s = R_f \frac{X}{1-X}$ $R_f = \frac{R_s(1-X)}{X}$
Three Wire Half Bridge[1,2] 	CRBasic Instruction: `BrHalf3W()` Fundamental Relationship: X = result w/mult = 1, offset = 0 $X = \frac{2V_2-V_1}{V_x-V_1} = \frac{R_s}{R_f}$	$R_s = R_f X$ $R_f = \frac{R_s}{X}$

<table>
<tr><th>Resistive-bridge type and circuit diagram</th><th>CRBasic instruction and fundamental relationship</th><th>Relational formulas</th></tr>
<tr><td>Four Wire Half Bridge[1,2]

</td><td>CRBasic Instruction:
BrHalf4W()
Fundamental Relationship:
X = result w/mult = 1, offset = 0
$X = \frac{V_2}{V_1} = \frac{R_s}{R_f}$</td><td>$R_f = \frac{R_s}{X}$
$R_s = R_f X$</td></tr>
<tr><td>Full Bridge[1,2]

</td><td>CRBasic Instruction:
BrFull()
Fundamental Relationship:
X = result w/mult = 1, offset = 0
$X = 1000 \frac{V_1}{V_x} = 1000 \left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$</td><td rowspan="2">These relationships apply to
BrFull()
and BrFull6W()
$R_1 = \frac{R_2(1-X_1)}{X_1}$
$R_2 = \frac{R_1 X_1}{1-X_1}$
where $X_1 = \frac{-X}{1000} + \frac{R_3}{R_3+R_4}$
$R_3 = \frac{R_4 X_2}{1-X_2}$
$R_4 = \frac{R_3(1-X_2)}{X_2}$
where $X_2 = \frac{X}{1000} + \frac{R_2}{R_1+R_2}$</td></tr>
<tr><td>Six Wire Full Bridge[1]

</td><td>CRBasic Instruction:
BrFull6W()
Fundamental Relationship:
X = result w/mult = 1, offset = 0
$X = 1000 \frac{V_2}{V_1} = 1000 \left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$</td></tr>
<tr><td colspan="3">[1] Key: V_x = excitation voltage; V_1, V_2 = sensor return voltages; R_f = fixed, bridge or completion resistor; R_s = variable or sensing resistor.
[2] Campbell Scientific offers terminal input modules to facilitate this measurement.</td></tr>
</table>

Offset voltage compensation applies to bridge measurements. In addition to `RevDiff` and `MeasOff` parameters discussed in Minimizing offset voltages (p. 195), CRBasic bridge

measurement instructions include the RevEx parameter that provides the option to program a second set of measurements with the excitation polarity reversed. Much of the offset error inherent in bridge measurements is canceled out by setting RevDiff, RevEx, and MeasOff to True.

Measurement speed may be reduced when using RevDiff, MeasOff, and RevEx. When more than one measurement per sensor is necessary, such as occurs with the BrHalf3W(), BrHalf4W(), and BrFull6W() instructions, input and excitation reversal are applied separately to each measurement. For example, in the four-wire half-bridge (BrHalf4W()), when excitation is reversed, the differential measurement of the voltage drop across the sensor is made with excitation at both polarities and then excitation is again applied and reversed for the measurement of the voltage drop across the fixed resistor. The results of the measurements (X) must then be processed further to obtain the resistance value, which requires additional program execution time.

CRBasic Example 2: Four-wire full-bridge measurement and processing

```
'This program example demonstrates the measurement and
'processing of a four-wire resistive full bridge.
'In this example, the default measurement stored
'in variable X is deconstructed to determine the
'resistance of the R1 resistor, which is the variable
'resistor in most sensors that have a four-wire
'full-bridge as the active element.
'Declare Variables
Public X
Public X_1
Public R_1
Public R_2 = 1000 'Resistance of fixed resistor R2
Public R_3 = 1000 'Resistance of fixed resistor R3
Public R_4 = 1000 'Resistance of fixed resistor R4
'Main Program
BeginProg
  Scan(500,mSec,1,0)
    'Full Bridge Measurement:
    BrFull(X,1,mV250,1,Vx1,1,4000,True,True,0,60,1.0,0.0)
    X_1 = ((-1 * X) / 1000) + (R_3 / (R_3 + R_4))
    R_1 = (R_2 * (1 - X_1)) / X_1
  NextScan
EndProg
```

12.3.2 RTD and PRT

RTDs (resistance temperature detectors) are resistive devices made of platinum, nickel, copper, or other material. Platinum RTDs, known as PRTs (platinum resistance thermometers) are very accurate temperature measurement sensors.

A PRT element is a specialized resistor with two connection points. Most PRTs are either 100 Ω or 1000 Ω. This number is the resistance the PRT has at 0 °C. The resistance of a PRT increases as it is warmed. Industry standards define how PRTs respond to temperature.

`BrHalf4W()` or `CDM_BrHalf4W()` in combination with `PRTCalc()` are the recommended CRBasic instructions for measuring RTDs.

See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

12.3.3 Strain measurements

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

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

Table 12-1: StrainCalc() configuration codes	
BrConfig code	Configuration
1	Quarter-bridge strain gage: $\mu\varepsilon = \frac{-4 * 10^6 V_r}{GF(1 + 2V_r)}$
2	Half-bridge strain gage. One gage parallel to strain, the other at 90° to strain: $\mu\epsilon = \frac{-4 * 10^6 V_r}{GF[(1 + v) - 2V_r(v - 1)]}$
3	Half-bridge strain gage. One gage parallel to +ε, the other parallel to -ε: $\mu\varepsilon = \frac{-2 * 10^6 V_r}{GF}$
4	Full-bridge strain gage. Two gages parallel to +ε, the other two parallel to -ε: $\mu\varepsilon = \frac{-10^6 V_r}{GF}$

Table 12-1: StrainCalc() configuration codes

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

Where:

ν : Poisson's Ratio (0 if not applicable).

GF: Gage Factor.

V_r: 0.001 (Source-Zero) if **BRConfig** code is positive (+).

V_r: –0.001 (Source-Zero) if **BRConfig** code is negative (–).

and where:

"source": the result of the full-bridge measurement (X = 1000 • V1 / Vx) when multiplier = 1 and offset = 0.

"zero": gage offset to establish an arbitrary zero.

12.3.4 AC excitation

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

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

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

NOTE:
Take precautions against ground loops when measuring sensors that require AC excitation. See also Ground loops (p. 181).

For more information, see Accuracy for resistance measurements (p. 110).

12.3.5 Accuracy for resistance measurements

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

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

NOTE:
Error discussed in this section and error-related specifications of the CR1000X/CR1000Xe do not include error introduced by the sensor, or by the transmission of the sensor signal to the data logger.

For accuracy specifications of ratiometric resistance measurements, see Resistance measurement specifications (p. 259). Voltage measurement is variable V_1 or V_2 in resistance measurements. Offset is the same as that for simple analog voltage measurements.

Assumptions that support the ratiometric-accuracy specification include:

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

12.4 Thermocouple Measurements

Thermocouple measurements are special case voltage measurements.

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

A thermocouple consists of two wires, each of a different metal or alloy, joined at one end to form the measurement junction. At the opposite end, each wire connects to terminals of a voltage measurement device, such as the data logger. These connections form the reference junction. If the two junctions (measurement and reference) are at different temperatures, a voltage proportional to the difference is induced in the wires. This phenomenon is known as the Seebeck effect.

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

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

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

See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

12.5 Period-averaging measurements

Use `PeriodAvg()` to measure the period (in microseconds) or the frequency (in Hz) of a signal on a single-ended channel. For these measurements, the data logger uses a high-frequency digital clock to measure time differences between signal transitions, whereas pulse-count measurements simply accumulate the number of counts. As a result, period-average measurements offer much better frequency resolution per measurement interval than pulse-count measurements. See also Pulse measurements (p. 112).

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

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

A threshold other than zero results in offset voltage drift, limited accuracy (approximately ±10 mV) and limited resolution (approximately 1.2 mV).

See also Period-averaging measurement specifications (p. 259).

TIP:
Both pulse count and period-average measurements are used to measure frequency output sensors. However, their measurement methods are different. Pulse count measurements use dedicated hardware - pulse count accumulators, which are always monitoring the input signal, even when the data logger is between program scans. In contrast, period-average measurements use program instructions that only monitor the input signal during a program scan. Consequently, pulse count scans can occur less frequently than period-average scans. Pulse counters may be more susceptible to low-frequency noise because they are always "listening", whereas period-averaging measurements may filter the noise by reason of being "asleep" most of the time.

Pulse count measurements are not appropriate for sensors that are powered off between scans, whereas period-average measurements work well since they can be placed in the scan to execute only when the sensor is powered and transmitting the signal.

12.6 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. Both pulse count and period-average measurements are used to measure frequency-output sensors. For more information, see Period-averaging measurements (p. 111).

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

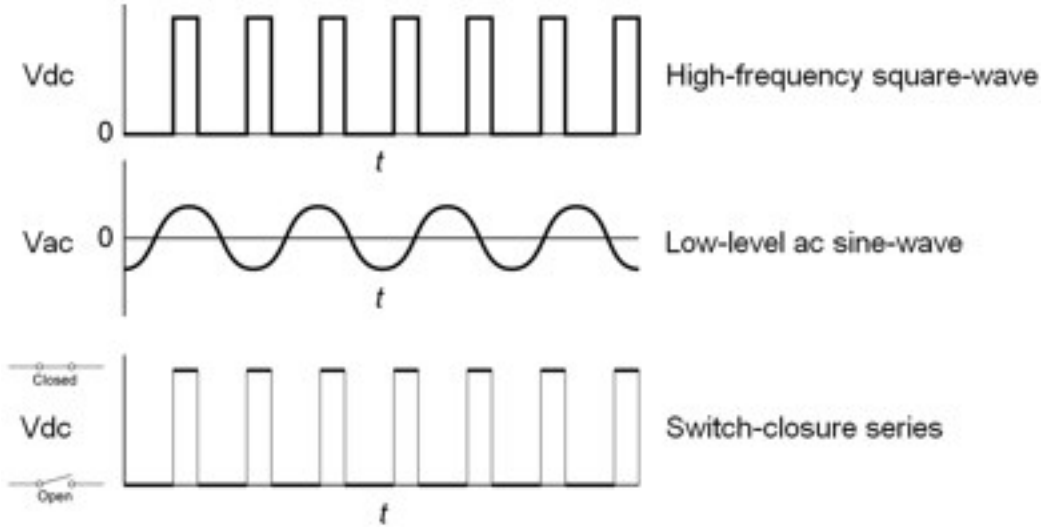


Table 12-2: Pulse input terminals and the input types they can measure	
Input type	Pulse input terminal
High-frequency	**P1** **P2** **C** (all)
Low-level AC	**P1** **P2**
Switch-closure	**P1** **P2** **C** (all)

Using the `PulseCount()` instruction, **P C** terminals are configurable for pulse input to measure counts or frequency. Maximum input frequency is dependent on signal voltage. If pulse input voltages exceed the maximum voltage, third-party external-signal conditioners should be employed. Do not measure voltages greater than 2019 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. 119).

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

For more information, see Pulse measurement specifications (p. 260).

See the *CRBasic Editor* help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

12.6.1 Low-level AC measurements

Low-level AC (alternating current or sine-wave) signals can be measured on P terminals. AC generator anemometers typically output low-level AC.

Measurement output options include the following:

- Counts
- Frequency (Hz)
- Running average

Rotating magnetic-pickup sensors commonly generate AC voltage ranging from millivolts at low-rotational speeds to several volts at high-rotational speeds.

CRBasic instruction: `PulseCount()`. See the *CRBasic Editor* help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

Low-level AC signals cannot be measured directly by C terminals. Peripheral terminal expansion modules, such as the Campbell Scientific LLAC4, are available for converting low-level AC signals to square-wave signals measurable by C terminals.

For more information, see Pulse measurement specifications (p. 260).

12.6.2 High-frequency measurements

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

- P or C

Common sensors that output high-frequency pulses include:

- Photo-chopper anemometers
- Flow meters

Measurement output optionss include counts, frequency in hertz, and running average. Note that the resolution of a frequency measurement can be different depending on the terminal used in the `PulseCount()` instruction. See the CRBasic help for more information.

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.

12.6.2.1 P terminals

- CRBasic instruction: `PulseCount()`

High-frequency pulse inputs are routed to an inverting CMOS input buffer with input hysteresis. See Pulse measurement specifications (p. 260) for more information.

12.6.2.2 C terminals

- CRBasic instructions: `PulseCount()`

See Pulse measurement specifications (p. 260) for more information.

12.6.3 Switch-closure and open-collector measurements

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

- P or 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 gages
- 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.

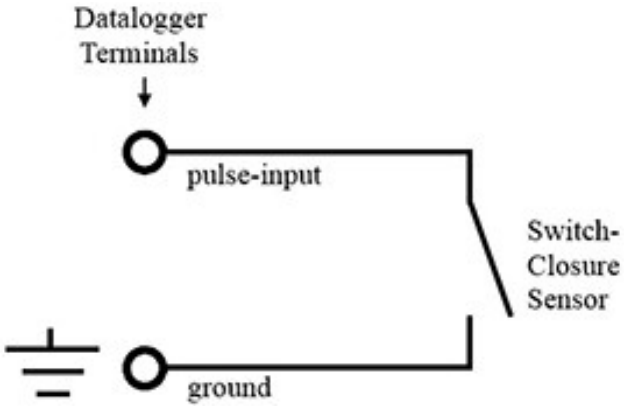
Data output options include counts, frequency (Hz), and running average.

12.6.3.1 P Terminals

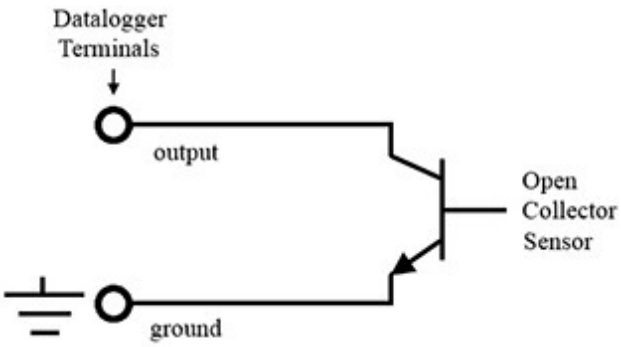
An internal 100 kΩ pull-up resistor pulls an input to 5 VDC with the switch open, whereas a switch-closure to ground pulls the input to 0 V.

- CRBasic instruction: `PulseCount()`. See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

Switch Closure on P Terminal



Open Collector on P Terminal



12.6.3.2 C terminals

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

- CRBasic instruction: `PulseCount()`.

See also Pulse measurement specifications (p. 260).

12.6.4 Edge timing and edge counting

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

12.6.4.1 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()`

12.6.4.2 Multiple edge counting

Time between edges, time from an edge on the previous terminal, and edges that span the scan interval can be measured on C terminals. Measurements can be expressed as a time (µs), frequency (Hz) or period (µs).

- CRBasic instruction: `TimerInput()`

12.6.4.3 Timer input NAN conditions

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

- Measurement timer expires
- The signal frequency is too fast

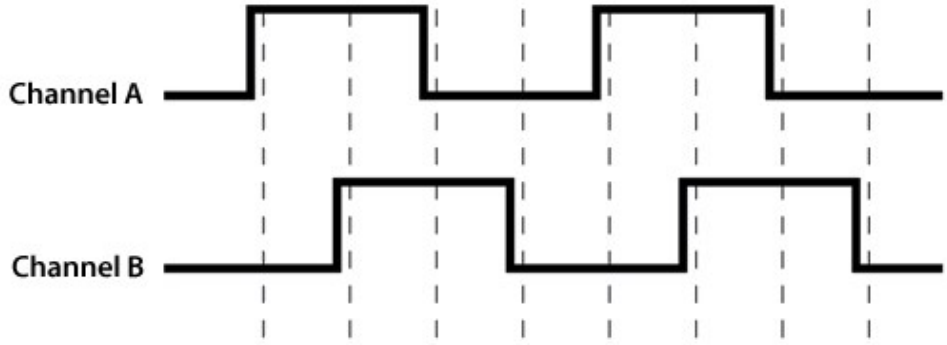
For more information, see:

- Pulse measurement specifications (p. 260)
- Digital input/output specifications (p. 262)
- Period-averaging measurement specifications (p. 259)

12.6.5 Quadrature measurements

The `Quadrature()` instruction is used to measure shaft or rotary encoders. A shaft encoder outputs a signal to represent the angular position or motion of the shaft. Each encoder will have two output signals, an A line and a B line. As the shaft rotates the A and B lines will generate digital pulses that can be read, or counted, by the 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 A leads channel B.
- Counting the increase at each rising and falling edge of both channels when channel A leads channel B. Counting the decrease at each rising and falling edge of both channels when channel B leads channel A.

For more information, see Digital input/output specifications (p. 262).

12.6.6 Pulse measurement tips

The `PulseCount()` instruction uses dedicated 24-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 16,777,216 (2^{24}), resulting in erroneous measurements. See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

Counts are the preferred `PulseCount()` output option when measuring the number of tips from a tipping-bucket rain gage or the number of times a door opens. Many pulse-output sensors, such as anemometers and flow meters, are calibrated in terms of frequency (Hz) so are usually measured using the `PulseCount()` frequency-output option.

Use the **LLAC4** module to convert non-TTL-level signals, including low-level AC signals, to TTL levels for input to **C** terminals

Conflicts can occur when 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. 113).

12.6.6.1 Input filters and signal attenuation

Terminals configured for pulse input have internal filters that reduce electronic noise. The electronic noise can result in false counts. However, input filters attenuate (reduce) the amplitude (voltage) of the signal. Attenuation is a function of the frequency of the signal. Higher-frequency signals are attenuated more. If a signal is attenuated too much, it may not pass the detection thresholds required by the pulse count circuitry. See Pulse measurement specifications (p. 260) for more information. The listed pulse measurement specifications account for attenuation due to input filtering.

12.6.6.2 Pulse count resolution

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

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

12.7 Vibrating wire measurements

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

12.7.1 VSPECT™

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

12.8 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." Or watch an instructional video at: http://www.campbellsci.com/videos/pipeline-sequential.

12.8.1 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).

12.8.2 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()` and `ExciteV()`, may not be completed until the next scan.

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

12.8.3 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 1 msec time slice, after which, the next task in the queue runs for 1 msec. The data logger cycles through the queue until all instructions for all sequences are complete.

13. 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 outstation, CPI, SPI, and TCP/IP. Several industry-specific protocols are also supported. CAN-bus is supported when using the Campbell Scientific SDM-CAN communications module. See also Communications specifications (p. 264) and Communications Protocols and Security Options for Campbell Scientific Data Loggers.

13.1 General serial communications 124
13.2 Modbus communications 130
13.3 Internet communications 140
13.4 DNP3 communications 143
13.5 Serial peripheral interface (SPI) and I2C 143
13.6 PakBus communications 143
13.7 SDI-12 communications 144

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.com/crbasic/cr1000x/.

- 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** IsRouter (p. 222).
- 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: https://www.campbellsci.com/videos/loggernet-software-network-planner.

13.1 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()`
- `SerialInChk()`
- `SerialOut()`
- `SerialOutBlock()`
- `SerialBrk()`
- `SerialFlush()`

See this application note for more information on interfacing serial sensors with Campbell Scientific data loggers: Serial Sensors | Interfacing with CS Data Loggers.

See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

To communicate over a serial port, it is important to be familiar with the 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. Terminals are configured in pairs for TTL, LVTTL, RS-232, and half-duplex RS-422 and RS-485 communications. For full-duplex RS-422 and RS-485, four terminals are required.

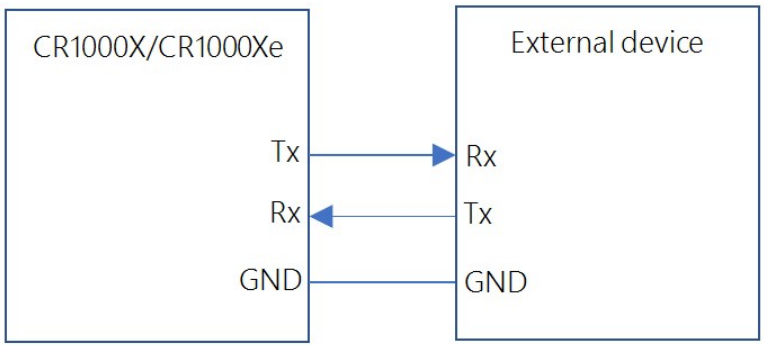


Figure 13-1. RS-232 single-ended full-duplex communications

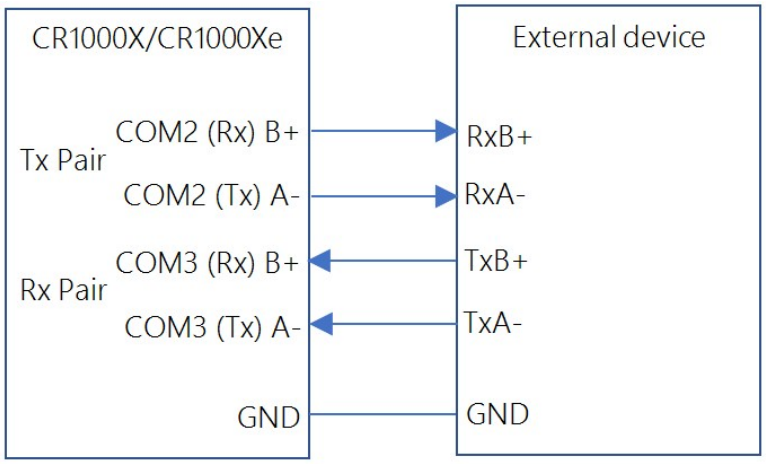


Figure 13-2. RS-485/RS-422 differential-pair full-duplex communications

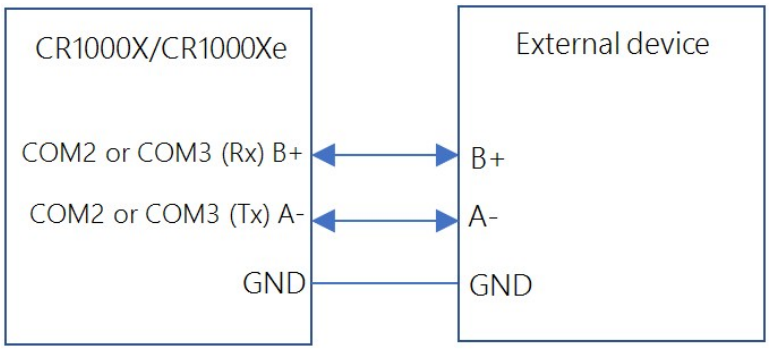


Figure 13-3. RS-485 differential-pair half-duplex communications

13.1.1 RS-232

RS-232 supports point-to-point communications between one base (usually the data logger) and one external device. See Figure 13-1 (p. 125). Data bits are sent from the base to external devices 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 with the data logger using TTL-inverted communications.

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

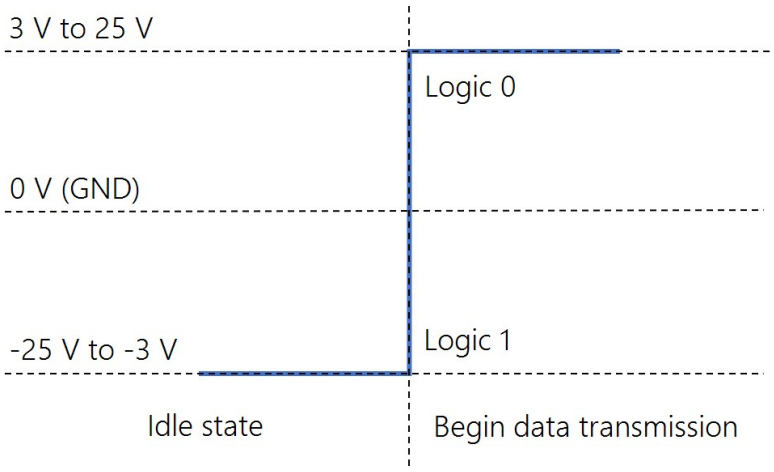


Figure 13-4. RS-232 Tx voltage with respect to GND

13.1.2 RS-485

RS-485 supports communications between 32 base and 32 external devices. See Figure 13-3 (p. 126) and Figure 13-2 (p. 125). Differential voltage between two lines (A & B) transmit data. When the voltage of B with respect to A is between -0.2 V and -5 V that is interpreted as logic 0. When the differential voltage in the range of positive 0.2 V to 5 V that is interpreted as logic 1.

> **NOTE:**
> The CR1000X/CR1000Xe uses about -1 V to represent logic 0, and about 1 V to represent logic 1.

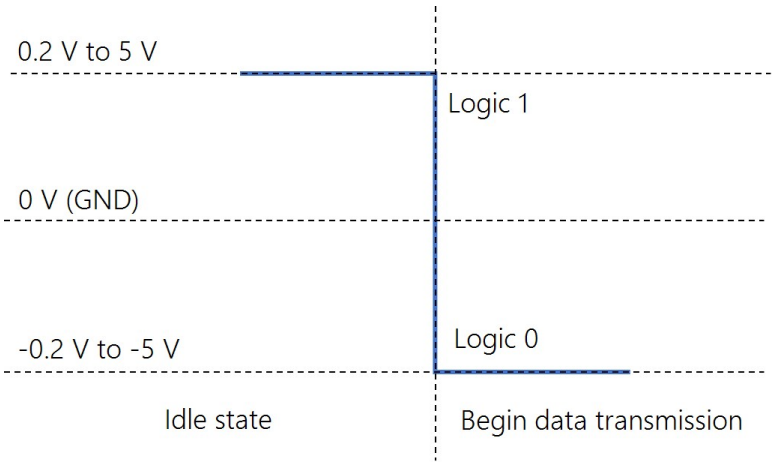


Figure 13-5. RS-485 Voltage B with respect to A

13.1.3 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 base to 10 external devices, but not return communications from all 10 external devices. In full-duplex point-to-point (1 base, 1 external) 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 13-3 (p. 126) and Figure 13-2 (p. 125).

> **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.

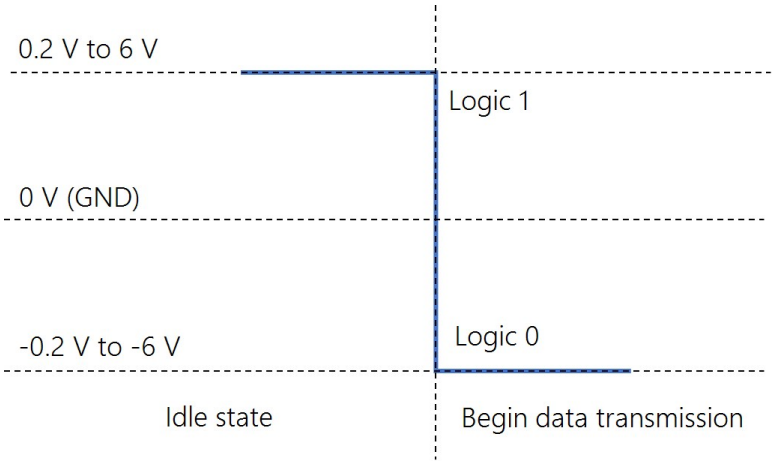


Figure 13-6. RS-422 Voltage B with respect to A

13.1.4 TTL

TTL supports point-to-point communications between one base and one external device. See Figure 13-1 (p. 125). Data bits are sent from base to external device 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).

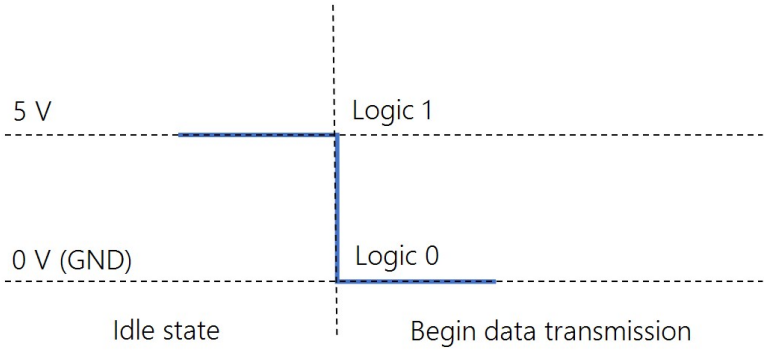


Figure 13-7. TTL Tx voltage with respect to GND

13.1.5 LVTTL

The only difference between low-voltage TTL (LVTTL) and TTL is that the voltage range is 0 V to 3.3 V. See Figure 13-1 (p. 125).

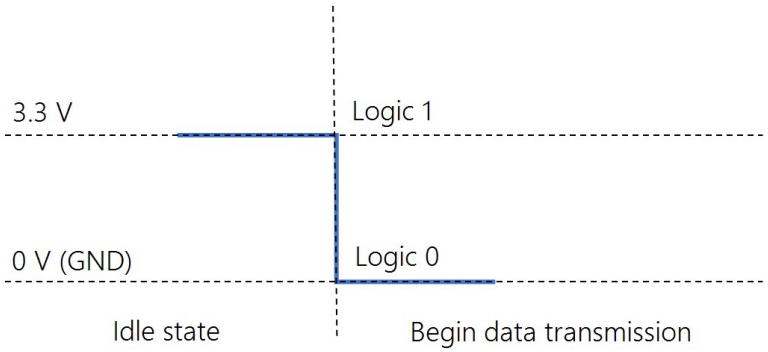


Figure 13-8. LVTTL Tx voltage with respect to GND

13.1.6 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 13-1 (p. 125). Data is sent after the voltage is pulled high (to 5 V).

> **NOTE:**
> Many RS-232 devices are compatible with this communications protocol.

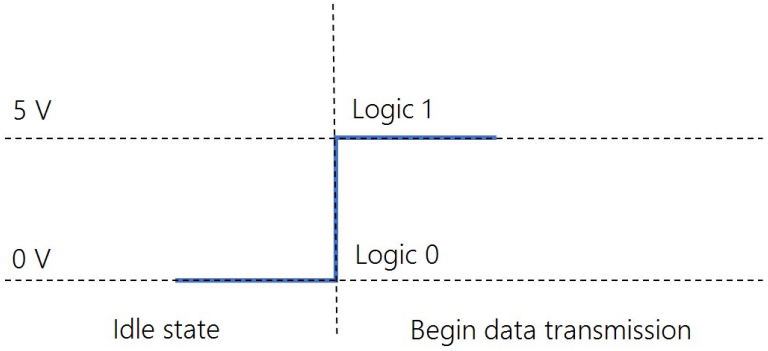


Figure 13-9. TTL-inverted Tx voltage with respect to GND

13.1.7 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 13-1 (p. 125).

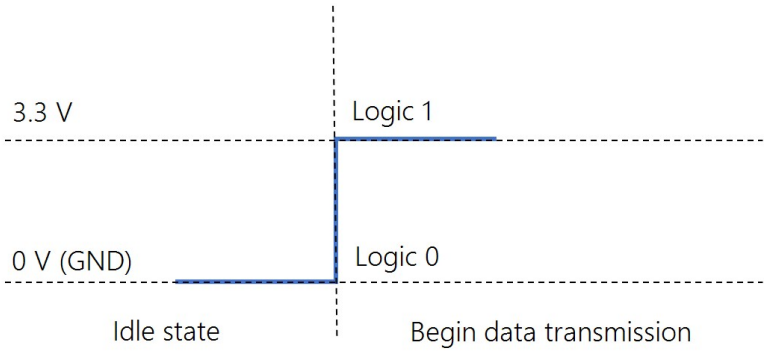


Figure 13-10. LVTTL-inverted Tx voltage with respect to GND

13.2 Modbus communications

The data logger supports Modbus RTU, Modbus ASCII, and Modbus TCP protocols and can be programmed as a Modbus client (master) or Modbus server (slave). These protocols are often used in SCADA networks. Data loggers can communicate using Modbus on all available communications ports. The data logger conducts Modbus over TCP using an Ethernet or Wireless connection. The data logger supports RTU and ASCII communications modes on RS-232 and RS-485 connections.

CRBasic Modbus instructions include:

- `ModbusClient()`
- `ModbusServer()`
- `MoveBytes()`

See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

For additional information on Modbus, see:

- About Modbus (p. 131)
- Why Modbus Matters: An Introduction
- How to Access Live Measurement Data Using Modbus
- Using Campbell Scientific Data Loggers 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 communications 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 client 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

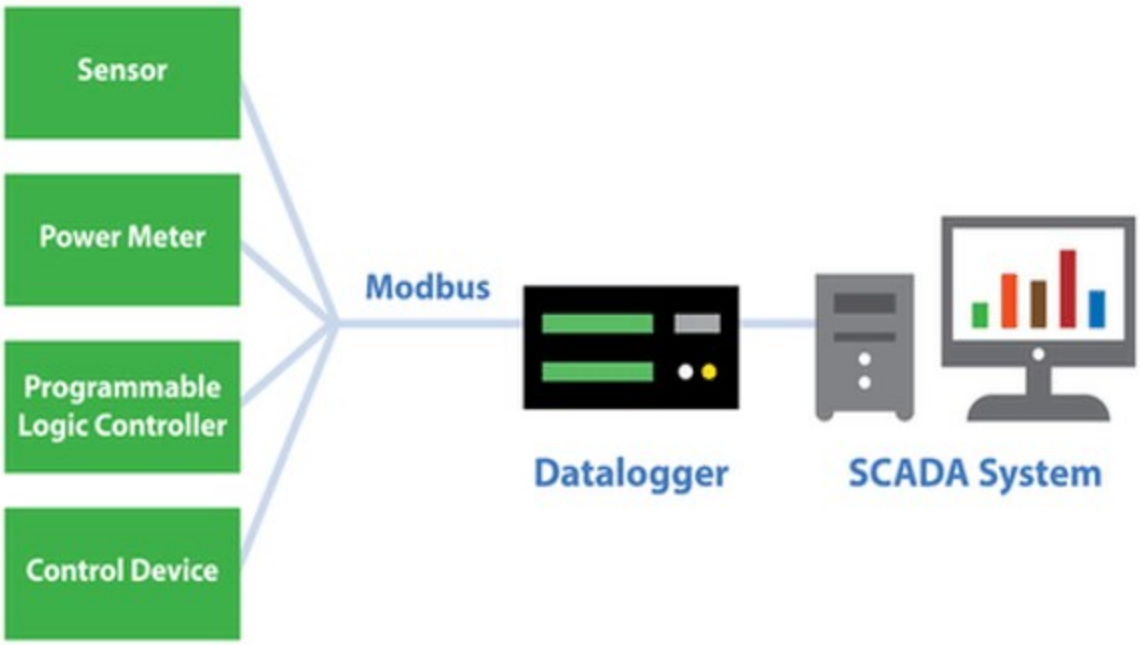
13.2.1 About Modbus

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

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

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



13.2.2 Modbus protocols

There are three standard variants of Modbus protocols:

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

  > NOTE:
  > The Modbus RTU protocol standard does not allow a delay between characters of 1.5 times or more the length of time normally required to receive a character. This is analogous to “pizza” being understood, and “piz za” being gibberish. It's important to note that communications hardware used for Modbus RTU, such as radios, must transfer data as entire packets without injecting delays in the middle of Modbus messages.

- Modbus ASCII — Used in serial communications, data is transmitted as an ASCII representation of the hexadecimal values. Timing requirements are loosened, and a simpler longitudinal redundancy check checksum is used.
- Modbus TCP/IP or Modbus TCP — Used for communications over TCP/IP networks. The TCP/IP format does not require a checksum calculation, as lower layers already provide

checksum protection. The packet structure is similar to RTU, but 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 server functionality over other comports use RTU. When acting as a client, the data logger can be switched between ASCII and RTU protocols using an option in the `ModbusClient()` instruction. See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

13.2.3 Understanding Modbus Terminology

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

Information is stored in the server 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.

13.2.4 Connecting Modbus devices

Data loggers can communicate with Modbus on all available communications 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. 15).

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.

13.2.5 Modbus client-server protocol

Modbus is a client-server protocol. The device requesting the information is called the Modbus client, and the devices supplying information are Modbus servers. In a standard Modbus

network, there is one client and up to 247 servers. A client does not have a Modbus address. However, each Modbus server on a shared network has a unique address from 1 to 247.

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

Every message from the client will begin with the server address, followed by the function code, function parameters, and a checksum. The server 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 server. Server devices only process one command at a time. So, the client needs to wait for a response, or timeout before sending the next command.

A broadcast address is specified to allow simultaneous communications with all servers. Because response time of server 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 client will need to be adjusted to account for the observed response time of the servers.

Campbell Scientific data loggers can be programmed to be a Modbus client or Modbus server - 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 client) from localized sensors or other data sources, and then in the other, it serves that data up (as a server) to another Modbus client.

13.2.6 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:

- `ModbusClient()`
- `ModbusServer()`
- `MoveBytes()`

See the *CRBasic Editor* help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

13.2.6.1 Endianness

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

In big-endian format, the byte containing the most significant bit is stored first, then the following bytes are stored in decreasing significance order, with the byte containing the least significant bit stored last. Little-endian format reverses this order: the sequence stores the least significant byte first and the most significant byte last. Endianness is used in some Modbus programming so it is important to note that the CR1000X/CR1000Xe is a big-endian instrument.

13.2.6.2 Function codes

A function code tells the server 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 `ModbusClient()` instruction refers to a source, not a destination.

13.2.7 Modbus information storage

With the Modbus protocol, most of the data values you want to transmit or receive are stored in registers. Information is stored in the server 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.com/crbasic/cr1000x/.

13.2.7.1 Registers

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

Campbell Scientific uses 1-based numbering (a common convention for numbering registers in equipment) in the `ModbusClient()` 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 `ModbusClient()`.

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.

13.2.7.2 Coils

Discrete digital I/O channels in Modbus are referred to as coils. The term coil has its roots in digital outputs operating solenoid coils in an industrial environment. Coils may be read only or

read/write. A read only coil would be a digital input. A read/write coil is used as an output. Coils are read and manipulated with their own function codes, apart from the registers. Many modern devices do not 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.

13.2.7.3 Data Types

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

13.2.8 Modbus tips and troubleshooting

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

For additional information on Modbus troubleshooting, see Modbus Troubleshooting Guide.

13.2.8.1 Error codes

Modbus defines several error codes, which are reported back to a client from a server. `ModbusClient()` 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 server 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 server 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 `ModbusClient()` 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 server device. Once you resolve the connection between the computer and the server device, the connection from the data logger should work.

13.3 Internet communications

See the Communications specifications (p. 264) 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:

- DHCPRenew()
- DNSQuery()
- EmailRelay()
- EmailSend()
- EmailRecv()
- FTPClient()
- HTTPGet()
- HTTPOut()
- HTTPPost()
- HTTPPut
- IPInfo()
- MQTTConnect()
- MQTTPublish()
- MQTTPublishConstTable()
- MQTTPublishMeta()
- MQTTPubishTable()
- NetworkTimeProtocol()
- PingIP()
- PPPOpen()
- PPPClose()
- TCPOpen()
- TCPClose()
- UDPDatagram()
- UDPOpen()
- UDPSocketClose()
- UDPSocketOpen()
- UDPSocketRecv()
- UDPSocketSend()

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.

See the FTP streaming technical paper for information on using FTPClient() or HTTPPut() to stream data.

See Using MQTT with Campbell Scientific Data Loggers for information on using MQTT.

See the HTTP troubleshooting technical paper for HTTP troubleshooting information.

13.3.1 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 CR1000X/CR1000Xe 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.

> **NOTE:**
> When setting a static IP address, first manually set a **DNS Server Address** in **Settings Editor** > **Advanced**.

13.3.2 HTTPS server

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

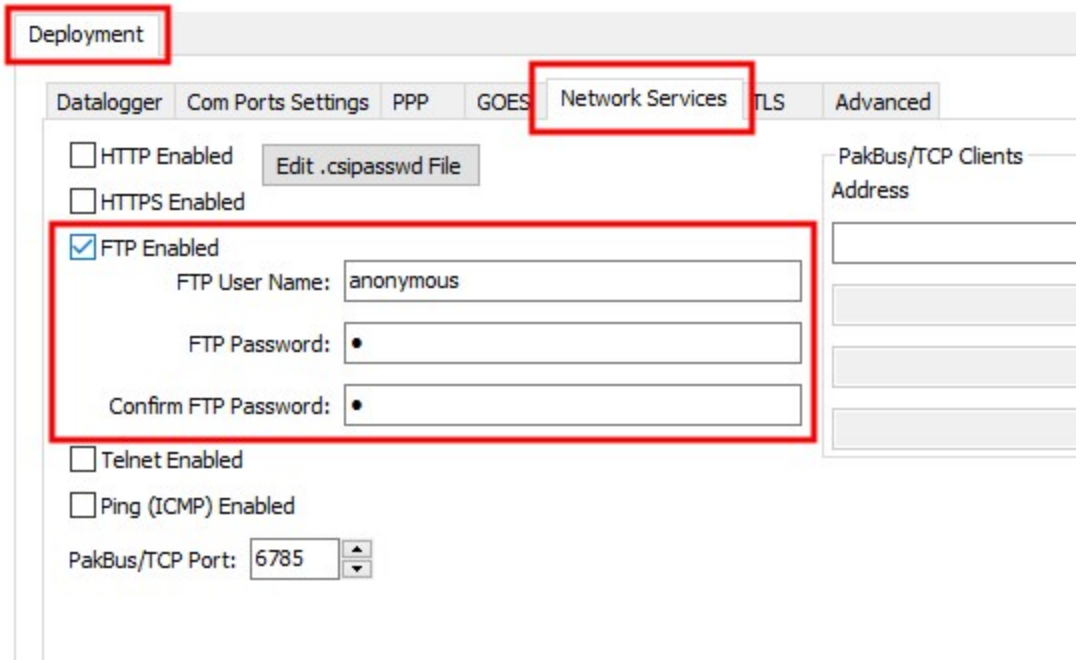
13.3.3 FTP server

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.

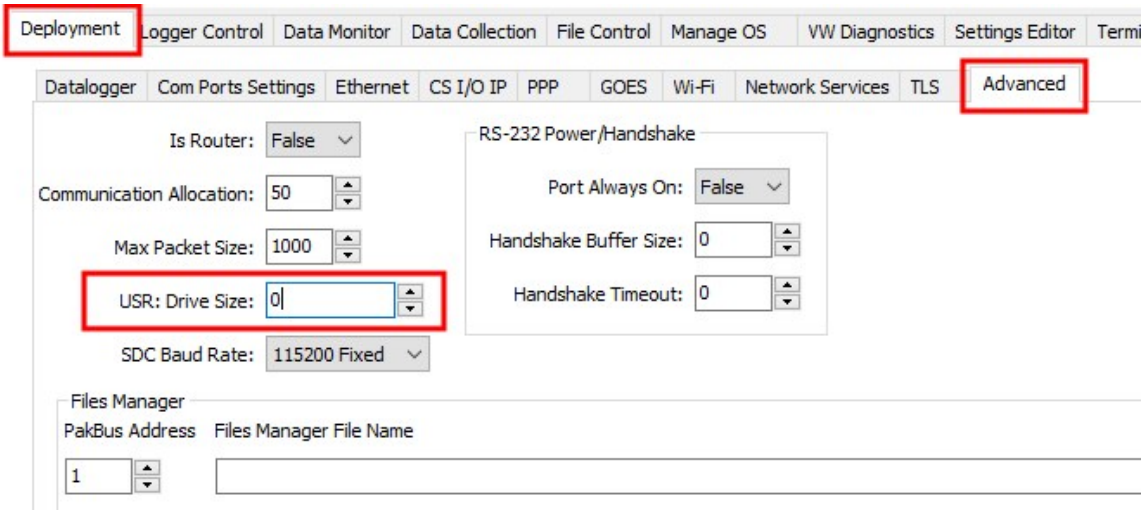
Select FTPEnabled (p. 218) and assign a **User Name** and **Password**.

> **NOTE:**
> FTP is disabled by default. However, when enabled, the UID will be the default FTP password.

> **CAUTION:**
> Passwords may be changed from the default UID. However, when the data logger is reset to factory defaults or a new OS is sent, the default password will revert to the UID.



Allocate memory where the received files will be stored. Often this is on the USR drive. Data memory (p. 88)



WARNING:
Partitioning or changing the size of the USR 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 USR drive, use a command similar to **put image.jpg /USR/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 and FTP Troubleshooting technical paper for more information.

13.4 DNP3 communications

DNP3 is designed to optimize transmission of data and control commands from a master computer to one or more remote devices or outstations. The 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.com/crbasic/cr1000x/.

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

13.5 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-controller (master), multi-peripheral (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.

13.6 PakBus communications

PakBus is a Campbell Scientific communications protocol. By using signed data packets, PakBus increases the number of communications 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.

13.7 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. 18) 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. 178) 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. 149) 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.com/crbasic/cr1000x/.

The data logger uses SDI-12 version 1.4.

See also

- www.sdi-12.org
- SDI-12 Sensors Troubleshooting tips

13.7.1 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.

Table 13-1: SDI-12 probe connections

Wire function	Data logger connection
SDI-12 signal	C
Shield	G
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: https://www.campbellsci.com/videos/sdi12-sensors-transparent-mode.

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



1. Press **Enter** until the data logger responds with the prompt **CR1000X>**.
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.

13.7.1.1 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: https://www.campbellsci.com/videos/sdi12-sensors-watch-or-sniffer-mode or use the following instructions.

1. Enter the transparent mode as described previously.
2. Press **Enter** until a **CR1000X>** prompt appears.
3. Type **W** and then press **Enter**.
4. In response, the query **Select 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.

13.7.1.2 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).

13.7.1.3 **aXL0AD0S!** command

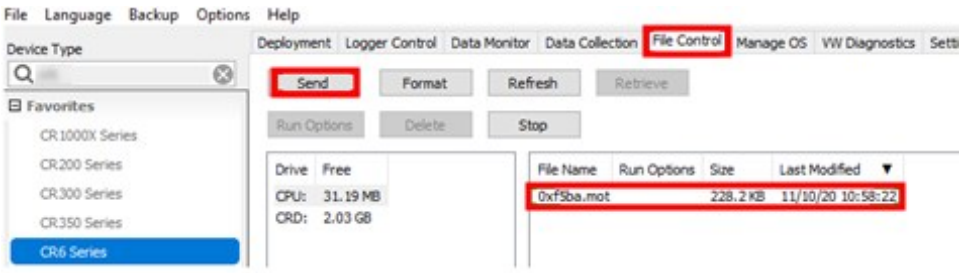
aXL0AD0S! is an example of an SDI-12 transparent mode command. It is used to send an operating system (OS) update from a data logger to an SDI-12 sensor.

> **NOTE:**
> Verify with the sensor manufacturer that the sensor supports this command.

Use **aXL0AD0S!** in the following procedure to send an OS update to an SDI-12 sensor.

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**). Alternatively, you can install the USB drivers using EZ Setup. A USB connection supplies 5 V power (as well as a communications link), which is adequate for setup, but a 12 V battery will be needed for field deployment.
2. Physically connect your data logger to your computer using a USB cable, then in ***Device Configuration Utility*** select your data logger.
3. Copy the sensor OS file from the computer to the data logger.
    a. Select **File Control** > **CPU:** drive > **Send** and navigate to the file on the computer.
    b. Click **Open**.

c. Click **OK**.



4. Enter the transparent mode as described in SDI-12 transparent mode (p. 178).
5. An `Entering SDI12 Terminal` response indicates that SDI-12 transparent mode is active and ready to transmit SDI-12 commands and display responses.

   The load OS command has the following format:

   `aXLOADOS Baudrate drive:filename!`.

   For example: `0XLOADOS 9600 CPU:0XF5BA.MOT!`.

   Type the command, including the ending exclamation point (!) then press **Enter**.

```
CR6>SDI12
1: C1
2: C3
3: [illegible]
4: [illegible]
5: [illegible]
6: [illegible]
7: [illegible]
8: [illegible]
Select SDI12 Port: 1
Entering SDI12 Terminal
0XLOADOS  9600  CPU:0XF5BA.MOT!
```

6. The screen will show OS send updates and the **bytes sent** will continue to increase. The process is slow, it can take several minutes, but not hours.

```
0XLOADOS  9600  CPU:0XF5BA.MOT!
  file size 228234, bytes sent 9098
```

7. A **SUCCESS** message indicates the process is complete.

```
0XLOADOS 9600 CPU:0X49ED.MOT!
  file size 228582, bytes sent 225465 SUCCESS
```

13.7.2 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 `aD0!` to initiate a measurement and get data, with the operator providing the proper time delay between the request for measurement and the request for data. In programmed mode, the 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 `aD0!` 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.com/crbasic/cr1000x/.

13.7.3 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.com/crbasic/cr1000x/.

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.

13.7.4 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 13-2: 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

Table 13-2: 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)
30			Yes	66	66	66	66	264
Sleep				0.25	0.25	0.25	0.25	1

14. CR1000X/CR1000Xe maintenance

Protect the data logger from humidity and moisture. The data logger is designed to operate over a wide range of temperature and humidity. However, when dew point is reached, condensation can occur, potentially damaging data logger electronics. To prevent this, place adequate desiccant inside the instrumentation enclosure to control humidity and keep the dew point below the temperature of the data logger electronics. Replace the desiccant 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. 165)
- Does Your Data Logger Need to be Repaired Blog
- Troubleshooting Data Acquisition System Blog

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

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

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

14.1 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.com/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

See also About background calibration (p. 153).

Campbell calibration certificates confirm that an instrument meets or exceeds published specifications and has been calibrated using standards and instruments with accuracies traceable to the National Institute of Standards and Technology, an accepted value of a natural physical constant, or a ratio calibration technique. The collective measurement uncertainty of the calibration process meets or exceeds a 4:1 accuracy ratio. Our facility's policies and procedures comply with ISO 9001 standards.

You can download and print Campbell calibration certificates for many of your purchased products by logging in to the Campbell Scientific website and going to: https://www.campbellsci.com/calcerts. You will need the product's serial number to access the certificate.

Watch an instructional video at: http://www.campbellsci.com/videos/calibration-certs.

In addition to Campbell calibration certificates, ISO1725 Calibration certificates are available. For ISO 17025 calibration, the data logger is verified by an independent certifying body accredited by A2LA to ISO 17025 standards. A paper certificate of the A2LA ISO/IEC 17025 accredited calibration is provided.

> **NOTE:**
> ISO 17205 calibration certificates are not available for download from our website. To request a ISO17025 calibration certificate, email support@campbellsci.com or, in the United States, call 435-227-9100.

14.1.1 About background calibration

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

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

Disable background calibration when it interferes with execution of very fast programs and less accuracy can be tolerated. With background calibration disabled, measurement accuracy over the operational temperature range is specified as less accurate by a factor of 10. That is, over the extended temperature range of –55 °C to 85 °C, the accuracy specification of ±0.08 % of reading can degrade to ±0.8 % of reading with background calibration disabled. If the temperature of the data logger remains the same, there is little calibration drift when background calibration is disabled.

14.2 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 LithiumBattery (p. 206) 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 to 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. 152).

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 communications logs (relearned without user intervention).
- Time. Clock will need resetting when the battery is replaced.
- Final-memory data tables.

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

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

See Power requirements (p. 253) 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.

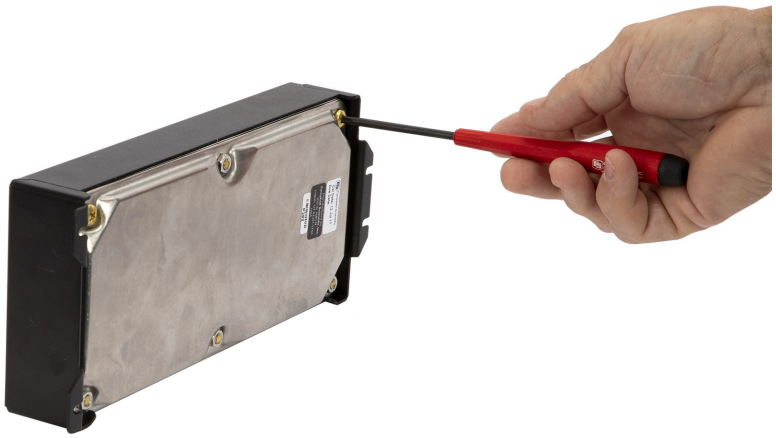
For additional information on the internal battery, visit the Campbell Scientific blog article, Get to Know Your Data Logger's Spare Tire: The Lithium Battery.

14.2.1 Replacing the internal battery

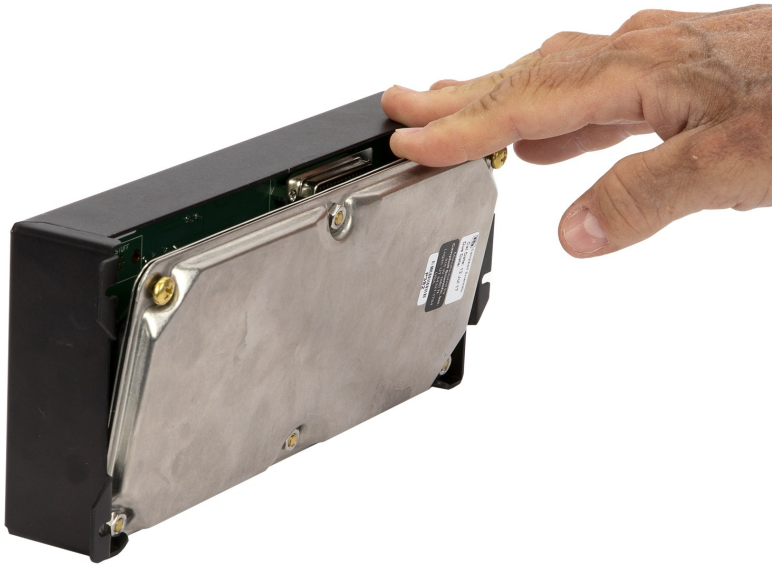
It is recommended that you send the data logger in for scheduled calibration, which includes internal battery replacement. See Data logger calibration (p. 152).

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

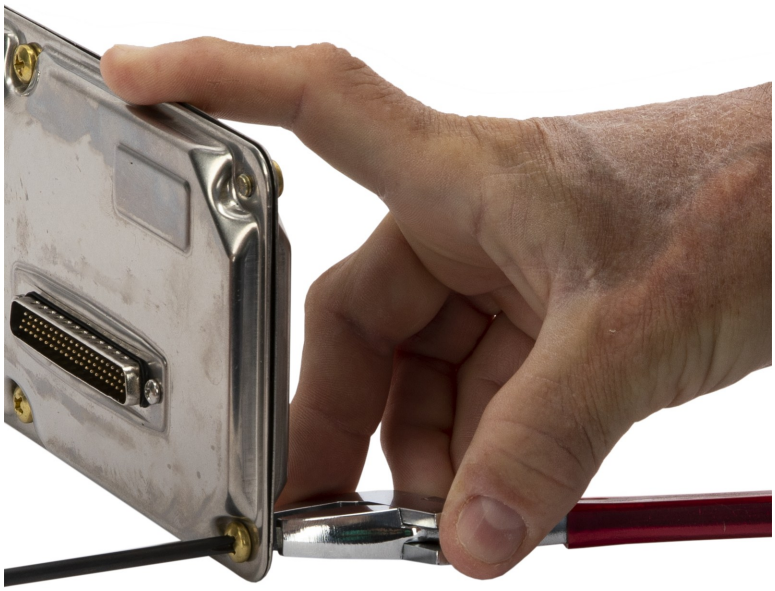
1. Remove the two screws from the back of the panel.



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



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



5. Remove the lithium battery by gently prying it out with a small flat-bladed screwdriver. Replace it with a new battery.



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

14.3 Updating the operating system

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

CAUTION:
CR1000X data loggers with Serial Numbers 34000 and newer have hardware requiring the use of OS version 5.02 or newer.

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**.

14.3.1 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.com/downloads.
2. Locate the .exe download and double-click to run the file. This will extract the .obj OS file to the **C:\Campbellsci\Lib\OperatingSystems** folder.
3. Supply power to the 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**). Alternatively, you can install the USB drivers using EZ Setup. A USB connection supplies 5 V power (as well as a communications link), which is adequate for setup, but a 12 V battery will be needed for field deployment.
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.
10. The Power and Act LEDs on the CR6 will flash red as the operating system is updated. After the flashing stops, return to the **Deployment | Datalogger** tab and check that the operating system shows the updated version. You may have to press the F5 key to refresh the displayed information.

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

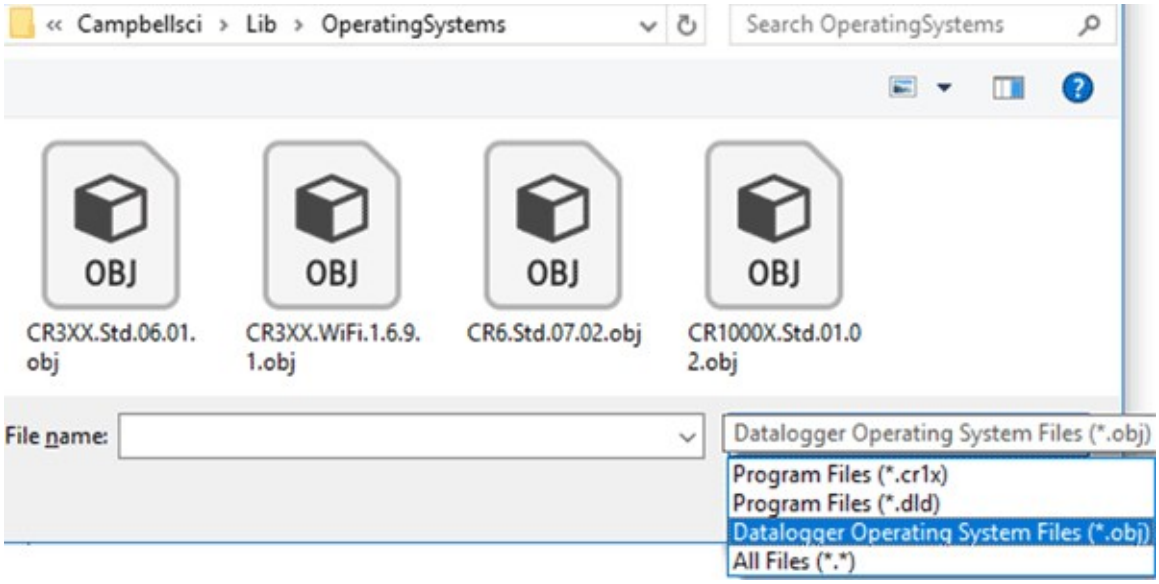
14.3.2 Sending an operating system to a remote data logger

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.

The `default.CR1X` program can be edited to preserve critical data logger settings such as communications settings. See Default program (p. 64) for more information.

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

14.4 gzip

The CR1000X/CR1000Xe supports the ability to extract the contents of program, operating system, and other files that have been created using **gzip**. The file name must be in the format: *filename.fileextension.gz* (for example: **TestPgm.CR1X.gz**, **CR1000X.Std.01.obj.gz**, or **CR1000X.Std.01.web.obj.gz**).

For more information see: www.gzip.org.

Zipping a file can significantly reduce its size, resulting in fewer bytes to transfer when sending a zipped file to a data logger. This is especially beneficial over slow, high-latency, or costly telecommunications links. Therefore, those using low-baud-rate radios, satellite, or restricted cellular data plans should consider gzipping operating systems and large programs before sending.

Compatible files can be created using any utility that supports the **gzip** file format. Use a file **tarball** (***filename*.tar.gz**) to compress multiple files. Several free utilities provide zipping to these formats.

Send the zipped file to the **CPU:**, **CRD:** or **USB:** drive using data logger support software. Files sent using **Connect** > **Send Program** will be unzipped automatically. However, the data logger will not automatically unzip files that are sent using **File Control** > **Send File**. To unzip files sent with **File Control**, mark them as **Run Now**.

Unzipping and installing file contents takes a long time; expect several minutes for operating systems and additional time for **.web** files. The details of unzipping and installing files from a **gzip** file are as follows:

1. The data logger receives the **gzip** file and restarts.
2. The data logger unzips the .gz file to the same drive to which it was sent.
3. The .tar portion of the file, if available, is processed.
4. Operating system (.obj or .iobj files) are programmed to the respective destination.
5. The data logger restarts.
6. When an .obj file is involved the OS will be loaded by the boot code resulting in another restart.
7. Web user interface (**.web**) files, if available, are installed. This may take over ten minutes.

NOTE:
Compression has little effect on an encrypted program (`FileEncrypt()`) and on files that already employ compression such as JPEG or MP4.

TIP:
The data logger also has the ability to compress files using `GZip()`. See the *CRBasic Editor* help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

14.5 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.

1. When the CR1000X/CR1000Xe powers up, it executes commands in the **powerup.ini** file (on an attached USB drive or memory card) including commands to set the CRBasic program file attributes to **Run Now** or **Run On Power-up**.
2. When the CR1000X/CR1000Xe powers up, a program file marked as **Run On Power-up** will run. If that program includes a file specified by the **Include File** setting, it will be incorporated into the program that runs.
3. If there is no program file marked **Run Now** or **Run On Power-up** (or if the program selected to run cannot be compiled), the data logger will run the program specified by the **IncludeFile** setting. For more information see IncludeFile (p. 219).
4. If the **IncludeFile** program cannot be compiled or if no program is specified, the data logger will attempt to run the program named **default.CR1X** on its CPU: drive.
5. If there is no **default.CR1X** file or it cannot be compiled, the CR1000X/CR1000Xe will not automatically run any program.

See Default program (p. 64) for more information.

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

14.5.1 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 14-1: Powerup.ini commands

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

Table 14-1: Powerup.ini commands		
Command	Action	Details
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.

14.5.2 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.CR1X,cpu:
```

Example: Run Program on Power Up

```
'Copy program file pwrup.CR1X from the external drive to CPU:
'File will run only when the data logger is powered-up later.
2,pwrup.CR1X,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,CR1000X.Std.01.obj
```

Example: Run Program from SC115 Flash Memory Drive

```
'A program file is carried on an SC115 Flash Memory drive.
'Do not copy program file from SC115
'Run program always, erase data.

13,toobigforcpu.CR1X,usb:
```

Example: Always Run Program, Erase Data

```
13,pwrup_1.CR1X,cpu:
```

Example: Run Program Now and Erase Data Now

```
14,run.CR1X,cpu:
```

15. Tips and troubleshooting

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

1. Ensure your system is well grounded. See Grounds (p. 15). The symptoms of a poorly grounded system range from bad measurements, to intermittent communications, to damaged hardware.
2. Using a voltmeter, check the voltage of the primary power source at the **POWER IN** terminals on the face of the data logger, it should be 10 to 18 VDC.
3. 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.
4. 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.
5. Reset the data logger. Sometimes the easiest way to resolve a problem is by resetting the data logger (see Resetting the data logger (p. 173) for more information).

For additional troubleshooting options, see:

15.1 Checking station status 166
15.2 Understanding NAN and INF occurrences 169

15.3 Timekeeping 170
15.4 CRBasic program errors 171
15.5 Resetting the data logger 173
15.6 Troubleshooting power supplies 174
15.7 Using terminal mode 175
15.8 Ground loops 181
15.9 Improving voltage measurement quality 185
15.10 Field calibration 199
15.11 File system error codes 199
15.12 File name and resource errors 200
15.13 Background calibration errors 200

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

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

15.1 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 at: https://www.campbellsci.com/videos/connect-station-status or use the following instructions.

15.1.1 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.
- From *PC400*: Select the **Datalogger** menu and **Station Status** to view the **Summary** tab.

15.1.2 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.

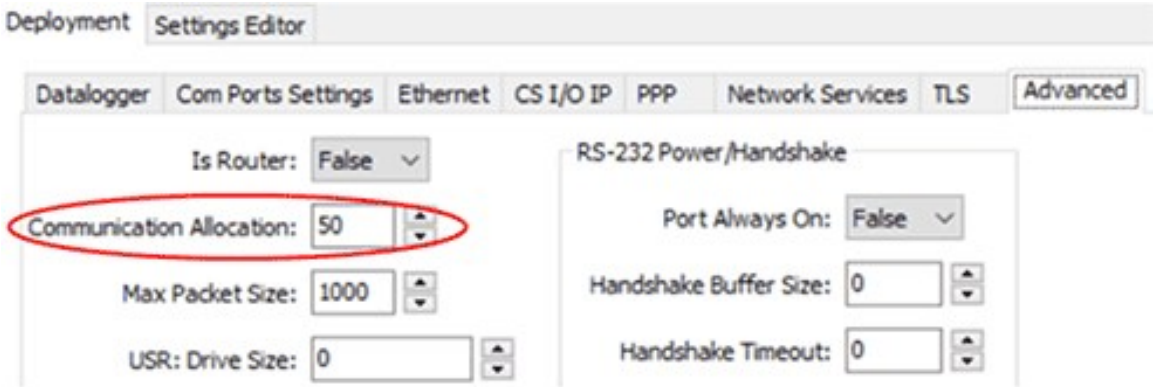
Watchdog errors could be due to:

- 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.

An IP panic watchdog error may be caused by insufficient communications memory. Try increasing the **Communication Allocation** field in *Device Configuration Utility*.



> **TIP:**
> It is good practice to always retrieve data from the data logger before changing settings; otherwise, data may be lost. See Collecting data (p. 77) for detailed instruction.

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

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

Additionally, a `watchdogInfo.txt` file may be created on the CPU drive when the CR1000X/CR1000Xe experiences a software reset (rather than a hardware reset that increments the **WatchdogErrors** field in the **Status Table**). Postings of `watchdoginfo.txt` files are rare. If ths file appears, contact Campbell Scientific for assistance (see https://www.campbellsci.com/support).

15.1.3 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. 171).

15.1.4 Skipped scans

Skipped scans are caused when a program takes longer to process than the scan interval 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.

15.1.5 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.

15.1.6 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.

15.1.7 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. 12)
- Power requirements (p. 253)

15.2 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 ÷ 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.

15.3 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. 252). An internal lithium battery backs the clock when the data logger is not externally powered. See Internal battery (p. 154).

15.3.1 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.

15.3.2 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.

15.3.3 Avoiding time skew

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

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

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

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

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

- Making measurements in the scan before time-consuming code.
- Programming the 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.com/crbasic/cr1000x/.

15.4 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 .

15.4.1 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.com. When run, the executable file updates the computer compiler. To update the data logger operating system, see Updating the operating system (p. 157).
- 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 USR drive size. If it is too large it may be using memory needed for the program.
  - Ensure a memory card is available when a program is attempting to access the CRD drive.

15.4.2 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 Status table system information (p. 204).

See the *CRBasic Editor* help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

15.5 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

15.5.1 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>).

15.5.2 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 to always retrieve data from the data logger before sending a program; otherwise, data may be lost. See Collecting data (p. 77) 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()`.

15.5.3 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**.

15.5.4 Formatting drives

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

15.5.5 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 FullMemReset (p. 206). 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. 157) for more information.

15.6 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. 66).

See also

- Measuring Data Logger Output Voltage With a Multimeter
- Checking Continuity
- Measuring Current Drain

15.7 Using terminal mode

Table 15-1 (p. 176) lists terminal mode options. With exception of perhaps the **C** command, terminal options are not necessary to routine CR1000X/CR1000Xe operations.

To enter terminal mode, connect a computer to the CR1000X/CR1000Xe. See Setting up communications with the data logger (p. 23). Open a terminal emulator program from Campbell Scientific data logger support software:

- **Connect** window > **Datalogger** menu item> **Terminal Emulator...**
- *Device Configuration Utility* **Terminal** tab

After entering a terminal emulator, press **Enter** a few times until the prompt **CR1000X>** is returned. Terminal commands consist of specific characters followed by **Enter**. Sending an **H** and **Enter** will return the terminal emulator menu.

ESC or a 40 second timeout will terminate on-going commands. Concurrent terminal sessions are not allowed and will result in dropped communications.

Terminal commands are subject to change. Please consult Campbell Scientific for assistance if you are not familiar with the effects of a command.

Table 15-1: CR1000X/CR1000Xe terminal commands		
Command	Description	Use
0	Scan processing time; real time in seconds	Lists technical data concerning program scans.
1	Serial FLASH data dump	Campbell Scientific engineering tool
2	Read clock chip	Lists binary data concerning the CR1000X/CR1000Xe clock chip.
3	Status	Lists the CR1000X/CR1000Xe **Status** table.
4	Card status and compile errors	Lists technical data concerning an installed memory card.
5	Scan information	Technical data regarding the CR1000X/CR1000Xe scan.
6	Raw A/D values	Technical data regarding analog-to-digital conversions.
7	VARS	Lists **Public** table variables.
8	Suspend / start data output	Outputs all table data. This is not recommended as a means to collect data, especially over comms. Data are dumped as non-error checked ASCII.
9	Read inloc binary	Lists binary form of **Public** table.
A	Operating system copyright	Lists copyright notice and version of operating system.
B	Task sequencer op codes	Technical data regarding the task sequencer.
C	Modify constant table	Edit constants defined with **ConstTable** / **EndConstTable**. Only active when **ConstTable** / **EndConstTable** in the active program.
D	MTdbg() task monitor	Campbell Scientific engineering tool
E	Compile errors	Lists compile errors for the current program download attempt.
F	Settings and predefined constants names	Lists predefined constants and settings
G	CPU serial flash dump	Campbell Scientific engineering tool

Table 15-1: CR1000X/CR1000Xe terminal commands

Command	Description	Use
H	Terminal emulator menu	Lists main menu.
I	Calibration data	Lists gains and offsets resulting from internal calibration of analog measurement circuitry.
J	Download file dump	Sends text of current program including comments.
L	Peripheral bus read	Campbell Scientific engineering tool
M	Memory check	Lists memory-test results.
N	File system information	Lists files in CR1000X/CR1000Xe memory.
O	Data table sizes	Lists technical data concerning data-table sizes.
P	Serial talk through	Issue commands from keyboard that are passed through the logger serial port to the connected device. Similar in concept to SDI12 Talk Through. No timeout when connected via PakBus.
PING	PING <HOST>	The HOST can be a DNS name or an IP address. For example, PING CAMPBELLSCI.COM.
REBOOT	Program recompile	Typing "REBOOT" rapidly will recompile the CR1000X/CR1000Xe program immediately after the last letter, "T", is entered. Table memory is retained. NOTE: When typing **REBOOT**, characters are not echoed (printed on terminal screen).
SDI12	SDI12 talk through	Issue commands from keyboard that are passed through the CR1000X/CR1000Xe SDI-12 port to the connected device. Similar in concept to Serial Talk Through. See also SDI-12 transparent mode (p. 178)
T	Unused	
U	Data recovery	Provides the means by which data lost when a new program is loaded may be recovered. Contact Campbell Scientific support.
V	Low level memory dump	Campbell Scientific engineering tool

Table 15-1: CR1000X/CR1000Xe terminal commands		
Command	Description	Use
W	Comms Watch (Sniff)	Enables monitoring of CR1000X/CR1000Xe communications traffic. No timeout when connected via PakBus.
X	Peripheral bus module identify	Campbell Scientific engineering tool

15.7.1 Serial talk through and comms watch

The **P: Serial Talk Through** and **W: Comms Watch** ("sniff") modes do not have a timeout when connected in terminal mode via PakBus. Otherwise, the timeout can be changed from the default of 40 seconds to any value ranging from 1 to 86400 seconds (86400 seconds = 1 day).

When using options **P** or **W** in a terminal session, consider the following:

- Concurrent terminal sessions are not allowed by the CR1000X/CR1000Xe.
- Opening a new terminal session will close the current terminal session.
- The data logger will attempt to enter a terminal session when it receives non-PakBus characters on the **RS-232** port or **CS I/O** port, unless the port is first opened with the `SerialOpen()` instruction.

If the data logger attempts to enter a terminal session on the **RS-232** port or **CS I/O** port because of an incoming non-PakBus character, and that port was not opened using `SerialOpen()`, any currently running terminal function, including the comms watch, will immediately stop. So, in programs that frequently open and close a serial port, the probability is higher that a non-PakBus character will arrive at the closed serial port, thus closing an existing talk-through or comms watch session. If this occurs, use the **FilesManager** setting to send comms watch or sniffer to a file.

For more information on Comms Watch, see a video at: https://www.campbellsci.com/videos/sdi12-sensors-watch-or-sniffer-mode.

15.7.2 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.

Table 15-2: SDI-12 probe connections	
Wire function	Data logger connection
SDI-12 signal	C
Shield	G
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: https://www.campbellsci.com/videos/sdi12-sensors-transparent-mode.

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



1. Press **Enter** until the data logger responds with the prompt **CR1000X>**.
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.

15.7.2.1 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: https://www.campbellsci.com/videos/sdi12-sensors-watch-or-sniffer-mode or use the following instructions.

1. Enter the transparent mode as described previously.
2. Press **Enter** until a **CR1000X>** prompt appears.
3. Type **W** and then press **Enter**.
4. In response, the query `Select 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.

15.7.2.2 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).

15.8 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.

For more information on ground loops, watch an instructional video: Avoiding Ground Loops.

See also Grounds (p. 15)

15.8.1 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, communications cables, and power cables. Any time one of these cables connects to the same two endpoints as another cable, a ground loop is formed.

15.8.2 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 AC noise or an AC pulse. As seen in

Figure 15-1 (p. 182) the parallel conductive paths form an electrical loop that acts as an antenna to pick up electromagnetic energy.

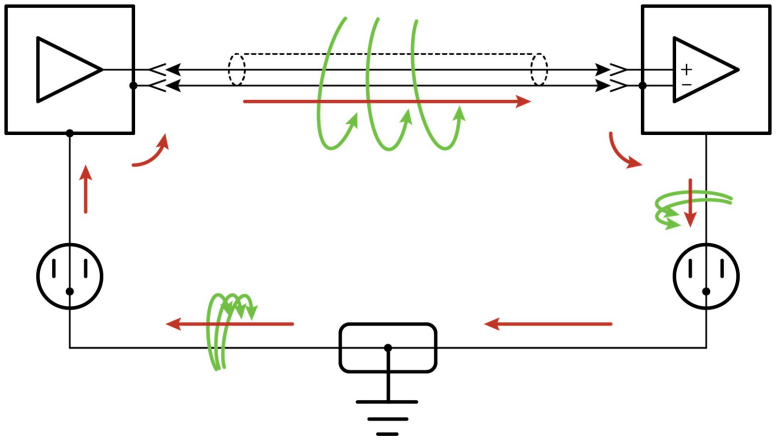


Figure 15-1. Stray AC magnetic fields picked up in loop antenna

- 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 either corrupts an analog signal or 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 than an electrical current must flow between the two points through the cable. See Figure 15-2 (p. 183).

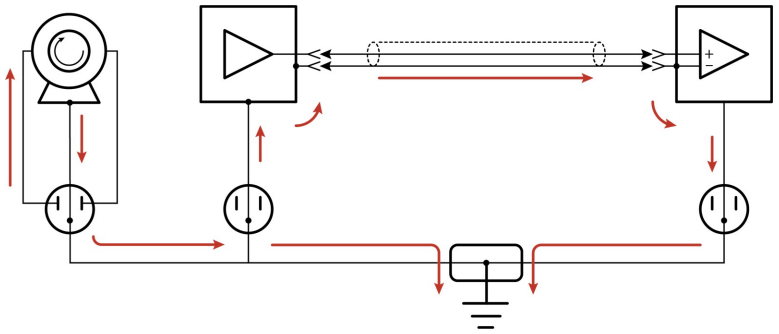


Figure 15-2. Leakage current (AC or DC) from nearby load

- One effect of this DC ground current-flow is a voltage offset error in analog measurements. Errors of this sort are usually not obvious but can have meaningful effects on measurements.
- 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.

15.8.3 Severing a ground loop

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

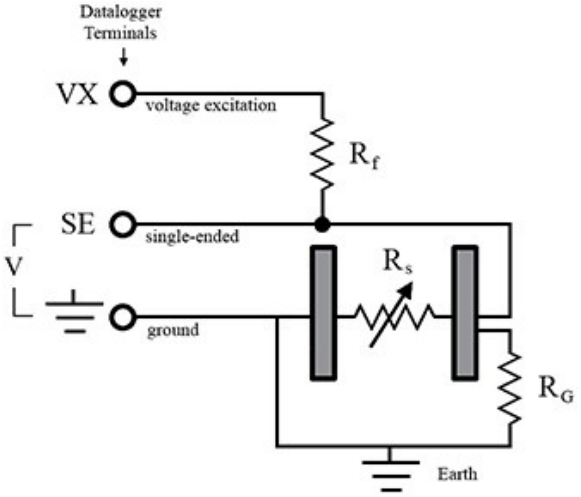
- Connect the shield wire of a signal cable to ground only at one end of the cable. Leave the other end floating (not connected to ground).
- Never intentionally use the shield (or drain wire) of a cable as a signal ground or power ground.
- Use the mechanical support structure only as a connection for 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 communications protocols such as RS-485, RS-422, and CAN, use a Resistive Ground (**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.

15.8.4 Soil moisture example

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



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

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

where

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

$R_s R_f / R_g$ is the source of error due to the ground loop. When R_g is large, the error is negligible. Note that the geometry of the electrodes has a great effect on the magnitude of this error. The Delmhorst gypsum block used in the Campbell Scientific 227 probe has two concentric cylindrical electrodes. The center electrode is used for excitation; because it is encircled by the ground electrode, the path for a ground loop through the soil is greatly reduced. Moisture blocks that consist of two parallel plate electrodes are particularly susceptible to ground loop problems. Similar considerations apply to the geometry of the electrodes in water conductivity sensors.

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

15.9 Improving voltage measurement quality

The following topics discuss methods of generally improving voltage measurements:

15.9.1 Deciding between single-ended or differential measurements 186
15.9.2 Minimizing ground potential differences 187
15.9.3 Detecting open inputs 188
15.9.4 Minimizing power-related artifacts 188
15.9.5 Filtering to reduce measurement noise 190
15.9.6 Minimizing settling errors 191
15.9.7 Factors affecting accuracy 193
15.9.8 Minimizing offset voltages 195

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

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

15.9.1 Deciding between single-ended or differential measurements

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

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

- Passive Noise Rejection
  - No voltage reference offset
  - Common-mode noise rejection, which filters capacitively coupled noise
- Active Noise Rejection
  - Input reversal
  - For more information, see Compensating for offset voltage (p. 197).

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

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

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

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

Because a single-ended measurement is referenced to data logger ground, any difference in ground potential between the sensor and the data logger will result in an error in the measurement. For more information on grounds, see Grounds (p. 15) and Minimizing ground potential differences (p. 187).

15.9.2 Minimizing ground potential differences

Low-level, single-ended voltage measurements (<200 mV) are sensitive to ground potential fluctuation due to changing return currents from **5V**,**12V**, **SW12**, and **C** terminals. The data logger grounding scheme is designed to minimize these fluctuations by separating signal grounds (⏚) from power grounds (**G**). For more information on data logger grounds, see Grounds (p. 15). To take advantage of this design, observe the following rules:

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

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

15.9.2.1 Ground potential differences

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

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

- **Soil Temperature Thermocouple**: If the measuring junction of a thermocouple is not insulated when in soil or water, and the potential of earth ground is, for example, 1 mV greater at the sensor than at the point where the data logger is grounded, the measured voltage will be 1 mV greater than the thermocouple output. With a Type T (copper-constantan) thermocouple, 1 mV equates to approximately 25 °C measurement error.

- **External Signal Conditioner**: External instruments with integrated signal conditioners, such as an infrared gas analyzer (IRGA), are frequently used to make measurements and send analog information to the data logger. These instruments are often powered by the same VAC-line source as the data logger. Despite being tied to the same ground, differences in current drain and wire resistance result in different ground potentials at the two instruments. For this reason, a differential measurement should be made on the analog output from the external signal conditioner.

For additional information, see Minimizing offset voltages (p. 195).

15.9.3 Detecting open inputs

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

The **C** option may not detect an open circuit in the following situations:

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

15.9.4 Minimizing power-related artifacts

Some VAC-to-VDC power converters produce switching noise or AC ripple as an artifact of the ac-to-dc rectification process. Excessive switching noise on the output side of a power supply can increase measurement noise, and so increase measurement error. Noise from grid or mains

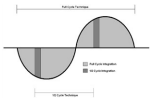
power also may be transmitted through the transformer, or induced electromagnetically from nearby motors, heaters, or power lines.

High-quality power regulators typically reduce noise due to power regulation. Using the 50 Hz or 60 Hz first notch frequency (`f_N1`) option for CRBasic analog input measurement instructions often improves rejection of noise sourced from power mains. The CRBasic standard deviation output instruction, `StdDev()`, can be used to evaluate measurement noise.

The data logger includes adjustable digital filtering, which serves two purposes:

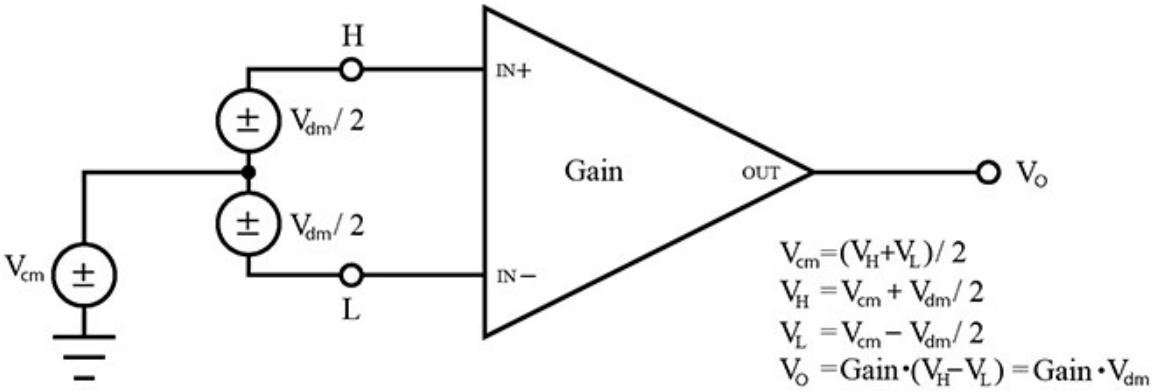
- Arrive as close as possible to the true input signal
- Filter out measurement noise at specific frequencies, the most common being noise at 50 Hz or 60 Hz, which originate from mains-power lines.

Filtering time is inversely proportional to the frequency being filtered.



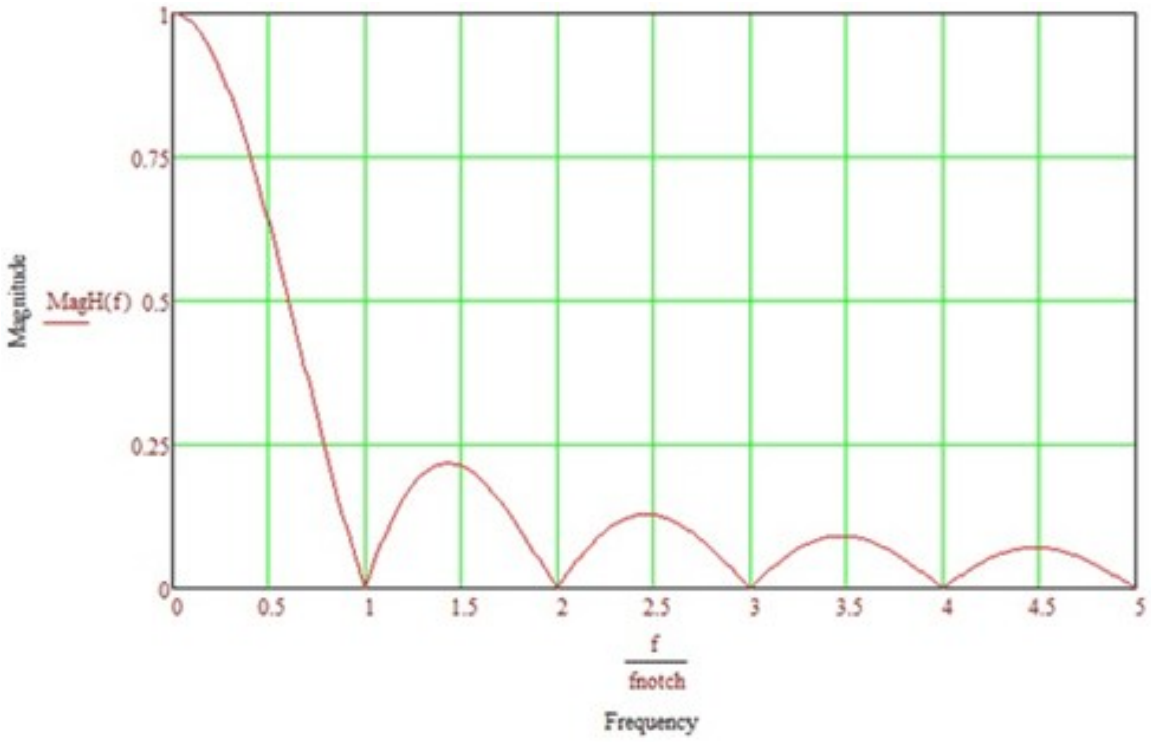
15.9.4.1 Minimizing electronic noise

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



15.9.5 Filtering to reduce measurement noise

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



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

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

15.9.5.1 CR1000X/CR1000Xe filtering details

The data logger utilizes a sigma-delta ADC that outputs digitized data at a rate of 31250 samples per second. User-specified filtering is achieved by averaging several samples from the ADC. Recall that averaging the signal over a period of $1/f_{N1}$ seconds will filter signal components at f_{N1}

Hz. The final result, then, is the average calculated from 31250/f_{N1} samples. For example, if **fN1** is set to 50 Hz, 625 samples (31250 / 50) are averaged to generate the final filtered result.

The actual f_{N1} may deviate from the user-specified setting since a whole integer number of samples must be averaged. For example, if **fN1** is set to 60 Hz, 521 samples (31250 / 60 = 520.83) will be averaged to produce the filtered result. The rounding of 520.83 to 521 moves the actual f_{N1} to 31250 / 521 = 59.98 Hz.

15.9.6 Minimizing settling errors

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

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

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

15.9.6.1 Measuring settling time

Settling time for a particular sensor and cable can be measured with the CR1000X/CR1000Xe. Programming a series of measurements with increasing settling times will yield data that indicate at what settling time a further increase results in negligible change in the measured voltage. The programmed settling time at this point indicates the settling time needed for the sensor / cable combination.

The following **CRBasic Example: Measuring Settling Time** presents CRBasic code to help determine settling time for a pressure transducer using a high-capacitance semiconductor. The code consists of a series of full-bridge measurements () with increasing settling times. The

pressure transducer is placed in steady-state conditions so changes in measured voltage are attributable to settling time rather than changes in pressure.

CRBasic Example 3: Measuring settling time

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

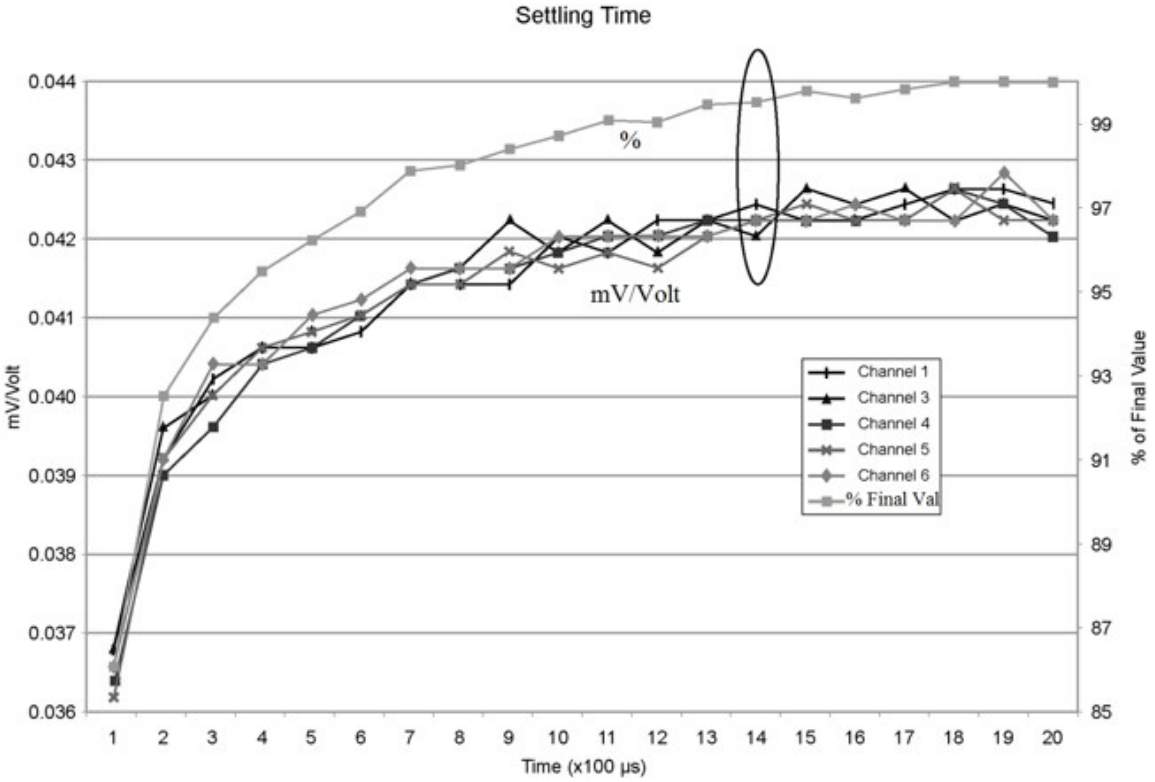
The first six measurements are shown in the following table:

Table 15-3: Example data from Measuring settling time program

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

Table 15-3: Example data from Measuring settling time program							
Timestamp	Record number	PT(1) Smp	PT(2) Smp	PT(3) Smp	PT(4) Smp	PT(5) Smp	PT(6) Smp
8/3/2017 23:34	3	0.03658813	0.03941815	0.03982244	0.04042887	0.04103531	0.04103531
8/3/2017 23:34	4	0.03679027	0.03921601	0.04022673	0.04063102	0.04063102	0.04083316

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



15.9.7 Factors affecting accuracy

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

accuracy = percent-of-reading + offset

where percents-of-reading and offsets are displayed in the Analog measurement specifications (p. 256).

> **NOTE:**
> Error discussed in this section and error-related specifications of the data logger do not include error introduced by the sensor, or by the transmission of the sensor signal to the data logger.

15.9.7.1 Measurement accuracy example

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

Example:

- Sensor-signal voltage: approximately 1050 mV
- CRBasic measurement instruction: `VoltDiff()`
- Programmed input-voltage range (`Range`) : **mV 5000** (±5000 mV)
- Input measurement reversal (`RevDiff`): **True**
- Data logger circuitry temperature: 10° C

Accuracy of the measurement is calculated as follows:

accuracy = percent-of-reading + offset

where

percent-of-reading = 1050 mV • ±0.04%

=±0.42 mV

and

offset = 0.5 µV

Therefore,

accuracy = ±(0.42 mV + 0.5 µV) = ±0.4205 mV

15.9.8 Minimizing offset voltages

Voltage offset can be the source of significant error. For example, an offset of 3 µV on a 2500 mV signal causes an error of only 0.00012%, but the same offset on a 0.25 mV signal causes an error of 1.2%. Measurement offset voltages are unavoidable, but can be minimized. Offset voltages originate with:

- Ground currents. See Minimizing ground potential differences (p. 187).
- Seebeck effect
- Residual voltage from a previous measurement

Remedies include:

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

Single-ended measurements are susceptible to voltage drop at the ground terminal caused by return currents from another device that is powered from the data logger wiring panel, such as another manufacturer's communications modem, or a sensor that requires a lot of power. Currents greater than 5 mA are usually undesirable. The error can be avoided by routing power grounds from these other devices to a power ground **G** terminal, rather than using a signal ground (⏚) terminal. Ground currents can be caused by the excitation of resistive-bridge sensors, but these do not usually cause offset error. These currents typically only flow when a voltage excitation is applied. Return currents associated with voltage excitation cannot influence other single-ended measurements because the excitation is usually turned off before the data logger moves to the next measurement. However, if the CRBasic program is written in such a way that an excitation terminal is enabled during an unrelated measurement of a small voltage, an offset error may occur.

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

measuring critical low-level voltages, especially those below 200 mV, such as are output from pyranometers and thermocouples.

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

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

The following table lists some of the tools available to minimize the effects of offset voltages:

Table 15-4: Offset voltage compensation options

CRBasic measurement instruction	Input reversal (RevDiff=True)	Excitation reversal (RevEx=True)	Measure offset during measurement (MeasOff=True)	Measure offset during background calibration (RevDiff=False) (RevEx=False) (MeasOff=False)
`BrHalf()`		✓		✓
`BrHalf3W()`		✓		✓
`BrHalf4W()`	✓	✓		✓
`BrFull()`	✓	✓		✓
`BrFull6W()`	✓	✓		✓
`TCDiff()`	✓			✓
`TCSe()`			✓	✓
`VoltDiff()`	✓			✓
`VoltSe()`			✓	✓

15.9.8.1 Compensating for offset voltage

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

Ratiometric measurements use an excitation voltage to excite the sensor during the measurement process. Reversing excitation polarity also reduces offset voltage error. Setting the `RevEx` parameter to `True` programs the measurement for excitation reversal. Excitation reversal results in a polarity change of the measured voltage so that two measurements with opposite polarity can be subtracted and divided by 2 for offset reduction similar to input reversal for differential measurements.

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

5.003 mV – (–4.997 mV) = 10.000 mV

10.000 mV / 2 = 5.000 mV

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

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

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

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

1. Switch to the measurement terminals.
2. Set the excitation, settle, and then measure.

3. Reverse the excitation, settle, and then measure.
4. Reverse the excitation, reverse the input terminals, settle, measure.
5. Reverse the excitation, settle, measure.

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

Read More: The Benefits of Input Reversal and Excitation Reversal for Voltage Measurements.

15.9.8.2 Measuring ground reference offset voltage

Single-ended and differential measurements without input reversal use an offset voltage measurement with the PGIA inputs grounded. This offset voltage is subtracted from the subsequent measurement. For differential measurements without input reversal, this offset voltage measurement is performed as part of the routine background calibration of the data logger. See About background calibration (p. 153). Single-ended measurement instructions `VoltSE()` and `TCSe()` include the `MeasOff` parameter determines whether the offset voltage measured is done at the beginning of the measurement instruction, or as part of self-calibration. This option provides you with the opportunity to weigh measurement speed against measurement accuracy. When `MeasOff = True`, a measurement of the single-ended offset voltage is made at the beginning of the `VoltSE()` or `TCSe()` instruction. When `MeasOff = False`, measurements will be corrected for the offset voltage determined during self-calibration. For installations experiencing fluctuating offset voltages, choosing `MeasOff = True` for the `VoltSE()` or `TCSe()` instruction results in better offset voltage performance.

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

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

> NOTE:
> When measurement duration must be minimal to maximize measurement frequency, consider disabling `RevDiff`, `RevEx`, and `MeasOff` when data logger temperatures and return currents are slow to change.

15.10 Field calibration

Calibration increases accuracy of a measurement device by adjusting its output, or the measurement of its output, to match independently verified quantities. Adjusting sensor output directly is preferred, but not always possible or practical. By adding the `FieldCal()` or `FieldCalStrain()` instruction to a CRBasic program, measurements of a linear sensor can be adjusted by modifying the programmed multiplier and offset applied to the measurement, without modifying or recompiling the CRBasic program. See the ***CRBasic Editor*** help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/cr1000x/.

15.11 File system error codes

Errors can occur when attempting to access files on any of the available drives. All occurrences are rare, but they are most likely to occur when using optional memory cards. 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. 202) 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

15.12 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.

15.13 Background calibration errors

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

- Check all analog inputs to make sure they are not greater than ±5 VDC by measuring the voltage between the input and a **G** terminal. Do this with a multimeter.
- Check for condensation, which can sometimes cause leakage from a 12 VDC source terminal.
- Check for a lose ground wire on a sensor powered from a **12V** or **SW12** terminal.

- If a multimeter is not available, disconnect sensors, one at a time, that require power from 9 to 16 VDC. If measurements return to normal, you have found the cause.

16. 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 CR1000X/CR1000Xe 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. 39) 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. 39) for more information).
  - *PC400* users, click the **Monitor Data** tab and add the **DataTableInfo** to display it.
  - *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.com/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. 77) 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
- USR drive size
- PakBus encryption key
- PakBus/TCP server port
- HTTP service port
- FTP service port
- PakBus/TCP service port
- PakBus/TCP client connections
- Communications allocation

16.1 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. 39) 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.
- *PC400* users, click the **Monitor Data** tab and add the **DataTableInfo** to display it.

16.1.1 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.

16.1.2 DataRecordSize

Reports the number of records allocated to a data table.

16.1.3 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

16.1.4 RecNum

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

16.1.5 SecsPerRecord

Reports the data output interval for a data table.

16.1.6 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.

16.1.7 TimeStamp

Scan time that a record was generated.

- NSEC data type

16.2 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. 39) 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.com/blog/programmatically-access-stored-data-values.

16.2.1 Battery

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

16.2.2 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.

16.2.3 CalCurrent

Shows the offset calibration factor for the resistor used in 0-20 and 4-20 mA measurements on RG terminals. Measured once during production calibration.

16.2.4 CalGain

Array of floating-point values reporting calibration gain (mV) for each integration / range combination.

16.2.5 CalOffset

Displays the offset calibration factor for the different voltage ranges.

16.2.6 CalRefOffset

Displays voltage reference temperature compensation offset.

16.2.7 CalRefSlope

Displays voltage reference temperature compensation slope.

16.2.8 CalVolts

Array of floating-point values reporting a factory calibrated correction factor for the different voltage ranges.

16.2.9 CardStatus

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

- String data type

16.2.10 CommsMemFree

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

16.2.11 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

16.2.12 ErrorCalib

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

16.2.13 FullMemReset

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

16.2.14 LastSystemScan

Reports the time of the of the last auto (background) calibration, which runs in a hidden slow-sequence type scan. See MaxSystemProcTime (p. 207), SkippedSystemScan (p. 210), and SystemProcTime (p. 211).

16.2.15 LithiumBattery

Voltage of the internal lithium battery. Updated at CR1000X/CR1000Xe power up. For battery information, see Internal battery (p. 154).

16.2.16 Low12VCount

Counts the number of times the primary CR1000X/CR1000Xe supply voltage drops below ≈9.0 VDC. Updates with each **Status** table update. Reset by entering 0. Incremented prior to scan (slow or fast) with measurements if the internal hardware signal is asserted.

16.2.17 MaxBuffDepth

Maximum number of buffers the CR1000X/CR1000Xe will use to process lagged measurements. Enter **0** to reset.

16.2.18 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.

16.2.19 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

16.2.20 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.

16.2.21 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.

16.2.22 MemoryFree

Unallocated final-data memory on the CPU (bytes). All free memory may not be available for data tables. As memory is allocated and freed, holes of unallocated memory, which are unusable for final-data memory, may be created. Updated after compile completes.

16.2.23 MemorySize

Total final-data memory size (bytes) in the CR1000X/CR1000Xe. Updated at startup.

16.2.24 Messages

Contains a string of manually entered messages.

- String data type

16.2.25 OSDate

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

- String data type

16.2.26 OSSignature

Signature of the operating system.

16.2.27 OSVersion

Version of the operating system in the CR1000X/CR1000Xe. Updated at OS startup.

- String data type

16.2.28 PakBusRoutes

Lists routes or router neighbors 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 neighbor address, pakbus address, response time in ms). Updates when routes are added or deleted.

- String data type

16.2.29 PanelTemp

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

16.2.30 PortConfig

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

- String data type

16.2.31 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

16.2.32 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.

16.2.33 ProgErrors

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

16.2.34 ProgName

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

- String data type

16.2.35 ProgSignature

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

16.2.36 RecNum

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

- Long data type

16.2.37 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

16.2.38 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.

16.2.39 SerialNumber

CR1000X/CR1000Xe serial number assigned by the factory when the data logger was calibrated. Stored in flash memory. Updated at startup.

16.2.40 SkippedScan

Number of skipped program scans (see Checking station status (p. 166) 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.

16.2.41 SkippedSystemScan

Number of scans skipped in the background calibration. Enter **0** to reset. See LastSystemScan (p. 206), MaxProcTime (p. 207), and SystemProcTime (p. 211).

16.2.42 StartTime

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

- NSEC data type

16.2.43 StartUpCode

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

16.2.44 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

16.2.45 SW12Volts

Status of switched, 12 VDC terminal(s). 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

16.2.46 SystemProcTime

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

16.2.47 TimeStamp

Scan-time that a record was generated.

- NSEC data type

16.2.48 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.

16.2.49 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.

16.2.50 WiFiUpdateReq

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

- Boolean data type

16.3 CPIStatus system information

The **CPIStatus** table is automatically created when a program uses the CPI bus. View the **CPIStatus** table by connecting the data logger to your computer (see Making the software connection (p. 39) 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.
- *PC400* users, click the **Monitor Data** tab and add **DataTableInfo**.

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

16.3.1 BusLoad

Percentage of the possible CPI network bandwidth use over the scan interval. BusLoad = Used capacity / Maximum capacity.

- Read only
- Percentage (0.000 to 100)

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

16.3.2 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.

16.3.3 ActiveModules

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

- Read only

16.3.4 BuffErr (buffer error)

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

16.3.5 RxErrMax

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

16.3.6 TxErrMax

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

16.3.7 FrameErr (frame errors)

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

16.3.8 ModuleInfo array

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

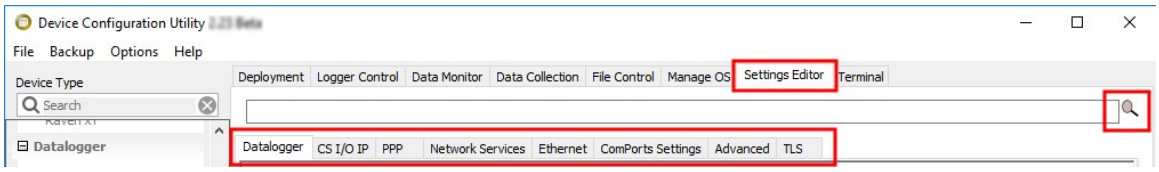
- String data type
- Read only

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.

16.4 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 programatically using `SetSetting()` or `Tablename.Fieldname` syntax. For example: *`Variable = Settings.Fieldname`*. For more information see: https://www.campbellsci.com/blog/programmatically-access-stored-data-values.

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

General data logger settings are listed below. For MQTT, GOES, and WIFI settings see:

- MQTT settings (p. 230).
- GOES settings (p. 234)
- Wi-Fi settings (p. 1)

16.4.1 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 and CS I/O ME) 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.

16.4.2 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 neighboring 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).

16.4.3 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 neighbors of any routers except those in its own branch. Configured in this fashion, the data logger will ignore any neighbor lists received from addresses in the central routers setting and will forward any messages that it receives to the nearest default router if it does not have the destination address for those messages in its routing table.

- String data type

16.4.4 CommsMemAlloc

Replaces **PakBusNodes**. Controls the amount of memory allocated for PakBus routing and communications in general. Increase the value of this setting if you require more memory dedicated to communications. Increase this value if the data logger will be used for routing a large number of PakBus nodes (>50). Increase this value if your data logger is dropping connections during short periods of high TCP/IP traffic. This setting will effect the values reported in CommsMemFree (p. 206).

16.4.5 ConfigCom*x*

Specifies the configuration for a data logger control port as it relates to serial communications. It is significant only when the associated port baud rate setting is set to something other than **Disabled**. This setting denotes the physical layer properties used for communications. It does not indicate the port's current configuration as it relates to standard or inverted logic. Options include:

- **RS-232**: Configures the port as RS-232 with standard voltage levels.

- TTL: The port is configured to use TTL, 0 to 5V voltage levels. By default, the port will use inverted logic levels. Use `SerialOpen()` to configure this port for standard TTL logic levels.
- LVTTL: The port is configured to use Low Voltage TTL (LVTTL), 0 to 3.3V voltage levels. By default, the port will use inverted logic levels. Use `SerialOpen()` to configure this port for standard TTL logic levels.
- RS-485 Half-Duplex PakBus: The port is configured as RS-485 half-duplex (two wire) and uses the PakBus/MDROP protocol. This allows reliable PakBus peer-to-peer networking of multiple devices including the MD485 and NL100 using the RS-485 interface.
- RS-485 Half-Duplex Transparent: The port is configured as RS-485 half-duplex (two wire). This setting is most commonly used when communicating with other non-PakBus RS-485 devices. Use this setting when communicating with devices such as Modbus RTUs or third-party serial sensors with RS-485 interfaces.
- RS-485/RS-422 Full Duplex Transparent: The port is configured as full-duplex (four wire). In this configuration, four adjacent control ports are required.

16.4.6 CSIO*x*netEnable

Controls whether the CS I/O IP #1 or #2 TCP/IP interface should be enabled.

16.4.7 CSIOInfo

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

16.4.8 DeleteCardFilesOnMismatch

Controls the behavior of the data logger when it restarts with a different program and it detects that data files created by the `CardOut()` are present but do not match the new program. If this value is set to one, the data logger will delete these files so that new files can be stored. If set to a value of zero, the data logger will retain the existing files and prevent any data from being appended to these files.

16.4.9 DisableLithium

Controls whether the data logger will maintain its real time clock and battery backed memory when it loses power. Setting this value to one will cause the data logger clock to lose time on

power loss. If this value is set to one, the data logger will not maintain its program or data after it powers down.

This value is useful when the data logger needs to be stored as it will prolong the shelf life of the lithium battery almost indefinitely.

If this value is set to one, the data logger will set it to zero when it powers up.

16.4.10 DisableTCPDelayAck

Controls whether the data logger uses TCP Delayed Acknowledgment for all TCP connections. The default value (0) enables TCP Delayed Acknowledgment, which is standard behavior on most computers and helps improve network efficiency. When TCP Delayed Acknowledgment is enabled, the data logger may wait up to 250 ms before acknowledging received data. In some cases, this delay can cause slow or uneven data delivery when reading data from a sensor over TCP—especially if the sensor uses Nagle's Algorithm. If you see delays of up to 250 ms between data packets, setting DisableTCPDelayAck to 1 may help. A value of 1 disables TCP Delayed Acknowledgment, causing the data logger to acknowledge data immediately as it is received. This can improve performance with devices that wait for acknowledgments before sending more data. See the CRBasic Help for TCPOpen for an example program.

16.4.11 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.

> NOTE:
> When setting a static IP address, first manually set a **DNS Server Address** in **Settings Editor** > **Advanced**.

- String data type

16.4.12 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

16.4.13 EthernetPower

This setting specifies how the data logger controls power to its Ethernet interface and provides a way to reduce power consumption when Ethernet is not connected. Options are Always On, 1 Minute, or Disable. The default is 1 Minute.

16.4.14 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

16.4.15 FTPEnabled

Set to **1** if to enable FTP service. Default is **0**. FTP is disabled by default. However, when enabled, the UID will be the default FTP password.

16.4.16 FTPPassword

Specifies the password that is used to log in to the FTP server. FTP is disabled by default. However, when enabled, the UID will be the default FTP password.

- String data type

16.4.17 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**.

16.4.18 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

16.4.19 HTTPEnabled

Set to **1** to enable HTTP (web server) service or **0** to disable it. Anonymous HTTP access is disabled by default. HTTP will be accessible with **admin** as the username and *UID* as the admin password.

16.4.20 HTTPHeader

Specifies additions to the HTTP header in the web service response. It can include multiple lines. Example: `Access-Control-Allow-Origin: *`

- String data type

16.4.21 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.

16.4.22 HTTPSEnabled

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

16.4.23 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.

16.4.24 IncludeFile

This setting specifies the name of a file to be implicitly included at the end of the current CRBasic program or can be run as the default program. In order to work as an include file, the file referenced by this setting cannot contain a `BeginProg()` statement or define any variable names or tables that are defined in the main program file.

This setting must specify both the name of the file to run as well as on the device (CPU:, USR:, or CRD:) on which the file is located. The extension of the file must also be valid for a data logger program (.CRB, .DLD, .CR1X).

See also File management via powerup.ini (p. 161).

- String data type

16.4.25 IPAddressCSIO

An array that specifies the CS I/O IP addresses for internet interfaces like the NL200 and NL240, which use the CS I/O bridge protocol. By default, the NL200 uses CS I/O IP address #1, and the NL240 uses CS I/O IP address #2. The "Interface Identifier" setting in the NL2xx can be used to change the default CS I/O IP address array number. If zero (the default) is specified for the IP address, the data logger will use DHCP to configure the IP address, network mask, and default gateway for that interface.

- String data type

16.4.26 IPBroadcastFiltered

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 to one, the data logger will fail to respond to the broadcast requests.

Default = **0**.

16.4.27 IPAddressEth

Specifies the IP address for the internet interface connected via the peripheral port to devices such as the NL115 and NL120. If this value is specified as "0.0.0.0" (the default), the data logger will use DHCP to configure the effective value for this setting as well as the `Ethernet Default Gateway` and `Ethernet Subnet Mask` settings. This setting is the equivalent to the `IPAddressEth` status table variable.

- String data type

16.4.28 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.

- String data type

16.4.29 IPGatewayCSIO

These settings specify the IP addresses of the router on the subnet to which the first or second CS I/O bridge internet interface is connected. The data logger will forward all non-local IP packets to this address when it has no other route. If the CS I/O IP Address setting is set to a value of "0.0.0.0", the data logger will configure the effective value of this setting using DHCP.

- String data type

16.4.30 IPMaskCSIO

These settings specify the subnet masks for the CS I/O bridge mode internet interface. If the corresponding CS I/O Address setting is set to a value of "0.0.0.0", the data logger will configure the effective value of this setting using DHCP.

- String data type

16.4.31 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

16.4.32 IPTrace

Discontinued; aliased to **IPTraceComport**

16.4.33 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.

16.4.34 IPTraceComport

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

16.4.35 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 neighbors and send this list to other routers in the PakBus network. It will also obtain and receive neighbor lists from other routers.

If the value of this setting is **false**, the 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 neighbors. 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

16.4.36 KeepAliveURL (Ping keep alive URL)

The URL to send a ping to when there has been no network activity for the **KeepAliveMin** interval. If there is no ping response then the network connection is reestablished.

- String data type

16.4.37 KeepAliveMin (Ping keep alive timeout value)

When there has been no network activity for this amount of time, in seconds, a ping will be sent to the **KeepAliveURL**. Default = **0** which disables keep alive pings.

- Long data type (allowed values: 0,5,10,15,30,60,120,180,240,300,360,480,720)

16.4.38 MaxPacketSize

Specifies the maximum number of bytes per data collection packet.

16.4.39 Neighbors

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

- String data type

16.4.40 NTPServer

This setting specifies an NTP Server to be queried (once per day) to adjust the data logger clock. This setting uses the **UTC Offset** setting. If UTC Offset setting is not set, it is assumed to be 0.

- String data type

16.4.41 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 behavior in the PakBus network.

When a device has an allowed neighbor list for a port, any device that has an address greater than or equal to 4000 will be allowed to connect to that device regardless of the allowed neighbor list.

16.4.42 PakBus Encryption Key

The **PakBusEncryptionKey** 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.

Beginning with operating system 8.00, the data logger is configured to be secure by default. Therefore, For data loggers that have a UID, **PakBus Encryption** is enabled by default. The default **PakBus Encryption Key** is the UID.

- String data type

16.4.43 PakBusNodes

Discontinued; aliased to **CommsMemAlloc**

16.4.44 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.

16.4.45 PakBusTCPClients

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

- String data type

16.4.46 PakBusTCPEnabled

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

16.4.47 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

16.4.48 PingEnabled

Set to one to enable the ICMP ping service.

16.4.49 PCAP

PCAP is a packet capture (PCAP) file of network packet data (network traffic) that can be opened by Wireshark. This setting specifies the network interface, file name, and maximum size of the PCAP file. For example:

- "usr:debug.pcap" saves the file to the USR drive with the file type .pcap.
- ".ring." found in name will create new files once the file size has been reached. "crd:debug.ring.pcap" creates crd:debug001.pcap, crd:debug002.pcap...
- If a number follows .ring. then only that number of files will be saved, with the oldest deleted. For example: "usr:debug.ring.3.pcap" will save three files.

If All Networks is selected as the Network Interface and PPP/Cell is active, then separate files will be opened for the PPP/Cell network with "ppp." prefixed on the file name.

16.4.50 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

16.4.51 pppDialResponse

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

- String data type

16.4.52 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

16.4.53 pppInterface

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

16.4.54 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

16.4.55 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

16.4.56 pppUsername

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

- String data type

16.4.57 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

16.4.58 RS232Handshaking

If non-zero, hardware handshaking is active on the RS-232 port. This setting specifies the maximum packet size sent between checking for CTS.

16.4.59 RS232Power

Controls whether the RS-232 port will remain active even when communications are not taking place. Note that if **RS232Handshaking** is enabled (handshaking buffer size is non-zero), that this setting must be set to **Yes**.

- Boolean data type

16.4.60 RS232Timeout

RS-232 hardware handshaking timeout. Specifies the time (tens of ms) that the CR1000X/CR1000Xe will wait between packets if CTS is not asserted.

16.4.61 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: Additional security measures (p. 62).

16.4.62 ServicesEnabled

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

16.4.63 TCPClientConnections

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

16.4.64 TCP_MSS

The maximum TCP segment size. This value represents the maximum TCP payload size. It is used to limit TCP packet size. A maximum TCP transmission unit (MTU) can be calculated by adding the IP Header size (20 bytes), the TCP Header size (20 bytes), and the payload size.

16.4.65 TCPPort

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

16.4.66 TelnetEnabled

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

16.4.67 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

16.4.68 TLSPassword

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

- String data type

16.4.69 TLSStatus

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

- String data type
- Read only

16.4.70 Configure USB

USBConfig controls the configuration of the data logger USB port. When set to a value of 1 it configures the data logger to enumerate USB as a virtual com port only. A value of 0 (the default) causes the data logger to enumerate as a composite device with both a virtual com port and a virtual Ethernet port (RNDIS) available.

Default = 0.

16.4.71 USB Disable

USBDisable controls whether the USB port is enabled or disabled. If set to 0 (the default), the port is enabled. 1 disables it. If the setting is changed, a data logger reboot is required for the change to take effect.

> NOTE:
> When USB is disabled, you will need to use another communications method (Ethernet, serial, CR1000KD) to re-enable it.

Default = **0**.

16.4.72 USBEnumerate

Controls the behavior of the data logger when its USB connector is plugged into the computer. If set to a value of **1**, the data logger will use its own serial number for identification in the USB enumeration. If set to a value of **0** (the default), the data logger will use a fixed serial number in the USB enumeration. This behavior controls whether the computer will allocate a new virtual serial port for the data logger USB connection or will use a previously allocated (but not currently used) virtual serial port.

Default = **0**.

16.4.73 USB Not Trusted

USBNotTrusted controls how the USB port behaves, as a COM port, with respect to security. If set to **0** (default), the port will allow PakBus communications to occur when the PakBus Encryption Key (p. 223) is set. Also, if RNDIS is enabled (USB Disable (p. 228) = **0**), no security challenges will be given to gain access via HTTP(S). If set to **1**, PakBus communications will follow the **PakBusEncryption Key** setting and RNDIS will require login credentials.

> NOTE:
> In ***Device Configuration Utility***, RNDIS is treated an IP connection rather than a direct connection. As a result, the **USBNotTrusted** setting does not apply when using RNDIS through ***Device Configuration Utility***.

Default = **0**.

16.4.74 USB Virtual Ethernet Address (RNDIS)

IPAddressUSB specifies the IP address for the USB Network Interface. When set to **0** it defaults to 192.168.66.1.

16.4.75 USRDriveFree

Provides information on the available bytes for the USR drive.

- Read only

16.4.76 USRDriveSize

Specifies the size in bytes allocated for the USR: ram disk drive. This memory is allocated from the memory that the data logger would normally use to store its compiled program or RAM based data tables. If this setting is too large, some programs may not be able to compile on the data logger.

Setting the USR: Drive Size setting will force the data logger to recompile its program and may result in the loss of data.

This setting controls the amount of memory set aside for the USR: size and is only indirectly related to the amount of storage within that file system. The amount of space available for storing files is always going to be less than this value because of the overhead of file system structures.

16.4.77 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.

16.4.78 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 neighbor if no other communications have taken place within the interval.

16.4.79 MQTT settings

Access MQTT 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***: **MQTT** tab, unless noted.

See also MQTT.

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

16.4.79.1 MQTT Auto-Publish Data

When enabled, this setting allows data tables to be published to CAMPBELL CLOUD without adding `MQTTPublishTable()` to a data table. See: MQTT Auto Publish Data versus MQTTPublishTable() (p. 74).

16.4.79.2 MQTTBaseTopic (MQTT base topic)

This is the base topic which will automatically be used. Use this setting to override the default format: CS/{CAMPBELL CLOUD Account ID}/{MQTT Client Id}/.

- String data type

16.4.79.3 MQTTCleanSession (MQTT connection)

Assigns the MQTT broker connection type. **Persistent** sessions save all relevant client information on the broker. The client gets messages that it misses offline.

If the connection between the client and broker is interrupted during a **Clean** session, topics may be lost and the client needs to subscribe again. The client does not get messages that it misses offline.

- Long data type, allowed values:
  - **0** = Clean
  - **1** = Persistent (default)

16.4.79.4 MQTTClientID (MQTT client identifier)

Unique identifier the data logger uses to connect to MQTT broker. The default is the hardware type_serial number. Example: CR1000X/CR1000Xe_123.

- String data type, maximum number of characters is 64

16.4.79.5 MQTTEnable (Enable or disable MQTT)

By default, MQTT is disabled.

- Long data type, allowed values:
  - 0 = Disable (default)
  - 1 = Enable with TLS-Mutual Authentication
  - 2 = Enable with TLS
  - 3 = Enable MQTT

> NOTE:
> Mutual authentication has become a widespread practice. Specifically, it is a requirement when using AWS IoT MQTT. It is also the practice used by CampbellCloud.

16.4.79.6 MQTTEndpoint (MQTT broker URL)

Server URL for MQTT broker.

- String data type

16.4.79.7 MQTTKeepAlive (MQTT keep alive)

When there has been no network activity for this amount of time, in seconds, a ping will be sent to the MQTTBrokerURL. Default = 0 which disables keep alive pings. Valid values are in the range 0 to 65535.

- Long data type

16.4.79.8 MQTTPassword (MQTT password)

Password, in association with MQTTUserName, required to connect to the MQTT broker.

- String data type

16.4.79.9 MQTTPortNumber (MQTT port number)

Port number to connect to the MQTT broker.

- Long data type, maximum number of characters is 256

16.4.79.10 MQTTStatusInterval (Status information publish interval)

Time (in minutes) between publishing MQTT status information. This interval determines how often the data logger publishes to the topic: {System Base Topic/}statusInfo. Valid values are in the range 0 to 1440.

- Long data type

16.4.79.11 MQTTState (MQTT state)

This is a read-only field indicating the current state of the data logger connection to the MQTT broker.

- Long data type, possible results:
  - 0 = Disabled / Off
  - 10 = Waiting for an IP network interface
  - 11 = Connection retry wait
  - 20 = Opening TCP connection
  - 21 = TCP Open failed
  - 22 = TCP connection opened
  - 24 = Closing TCP connection
  - 26 = TCP connection closed
  - 30 = TLS handshake started
  - 31 = TLS handshake failed
  - 32 = TLS handshake success
  - 50 = MQTT session established
  - 51 = Waiting for session start response
  - 52 = Publishing
  - 100 = Onboard started
  - 101 = Onboard retry
  - 102 = Onboard processing

16.4.79.12 MQTTStateInterval (State publish interval)

Time (in minutes) between publishing MQTT state information. This interval determines how often the data logger publishes to the topic: {System Base Topic/}State. Valid values are in the range 0 to 1440. Setting the value to 0 will not disable normal state publishing activity, only interval publishing.

- Long data type

16.4.79.13 MQTTUserName (MQTT user name)

User name, in association with **MQTTPassword**, used to connect to MQTT broker.

- String data type, maximum number of characters is 256

16.4.79.14 MQTTWillMessage (MQTT last will message)

Message published on last will topic by broker if disconnected without a disconnect command.

- String data type, maximum number of characters is 256

16.4.79.15 MQTTWillQoS (Quality of service)

This is an agreement that defines the guarantee of delivery for a specific message. Higher QoS levels are more reliable, but take more time and bandwidth.

- Long data type, allowed values:
  - **0** = At most once (default), no confirmation
  - **1** = At least once, confirmation required
  - **2** = Exactly once using a multi-step handshake

16.4.79.16 MQTTWillRetain (MQTT last will message retained by broker)

Enables or disables the broker to retain **MQTTWillMessage**.

- Long data type, allowed values:
  - **0** = Do not retain (default)
  - **1** = Retain

16.4.79.17 MQTTWillTopic (MQTT last will topic)

Broker will publish the **MQTTWillMessage** to this topic if disconnected without a disconnect command.

- String data type, maximum number of characters is 64

16.4.80 GOES settings

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

data loggers that have a TX325 or TX326 attached.

Where to find:

- All settings: **Settings Editor** tab in *Device Configuration Utility*: **GOES** tab, unless noted.

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

16.4.80.1 GOESComPort

Port used to communicate with the GOES transmitter.

- Long data type; allowed values:
  - **1** = RS-232
  - **4** = CS I/O SDC7
  - **5** = CS I/O SDC8
  - **7** = CS I/O SDC10
  - **8** = CS I/O SDC11
  - **9** = COMC1
  - **10** = COMC3

16.4.80.2 GOESEnabled

Controls whether the data logger polls the **GOESComPort** to see if a GOES radio is attached.

- Long data type, allowed values:
  - **0** = Disable (default). The data logger ignores all other GOES settings.
  - **1** = Enable

16.4.80.3 GOESGainSetting

Specifies the effective antenna gain (in units of 0.1 dbi). This is the maximum specified gain for the antenna minus the loss in the cable connecting the radio to the antenna.

- Long data type, allowed values:
  - **0** = Disable (default). The radio will operate as if the antenna used for its original certification is being used.
  - **1** to **140** = Enable for 300 Bps transmissions (14 dbi maximum)
  - **1** to **200** = Enable for 1200 Bps transmissions (20 dbi maximum)

16.4.80.4 GOESMsgWindow

Length, in seconds, of the assigned self-timed transmission window assigned by NESDIS (TX325) or EUMETSAT (TX326). Valid values are in the range **1** to **110** seconds.

- Long data type

16.4.80.5 GOESPlatformID

8-digit hexadecimal identification number assigned by NESDIS (TX325) or EUMETSAT (TX326).

- String data type

16.4.80.6 GOESRepeatCount

Number of times within the random transmit interval that the GOES transmitter will transmit the message data. Valid entries are **1** to **3**.

- Long data type

16.4.80.7 GOESRTBaudRate

Baud rate for the random transmissions. Valid settings are **100**, **300**, or **1200**. The baud rate must match the NESDIS (TX325) or EUMETSAT (TX326) channel assignment.

- Long data type

16.4.80.8 GOESRTChannel

Channel used for the random transmission assigned by NESDIS (TX325) or EUMETSAT (TX326).

- Long data type, allowed values:
  - **0** = Disable (default).
  - **0** to **566** = Channel

16.4.80.9 GOESRTInterval

Average time between random transmissions. Maximum interval is 24 hours; minimum interval is 1 minute.

- String data type entered in the format of “Hours:Minutes:Seconds”.

16.4.80.10 GOESSTBaudRate

Baud rate for self-timed transmissions. Valid settings are **300** or **1200**. The baud rate must match the NESDIS (TX325) or EUMETSAT (TX326) channel assignment.

- Long data type

16.4.80.11 GOESSTChannel

Channel used for the self-timed transmission assigned by NESDIS (TX325) or EUMETSAT (TX326).

- Long data type, allowed values:
  - **0** = Disable (default).
  - **0** to **566** = Channel

16.4.80.12 GOESSTInterval

Time between self-timed transmissions. Maximum interval is 14 days; minimum interval is 1 minute.

- String data type entered in the format of "Hours:Minutes:Seconds".

16.4.80.13 GOESSTOffset

Time after midnight for the first self-timed transmission as assigned by NESDIS (TX325) or EUMETSAT (TX326). Maximum offset is 23:59:59. A value of 0 results in no offset.

- String data type entered in the format of "Hours:Minutes:Seconds".

17. CR1000Xe specifications

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

See CR1000X specifications (p. 252) for CR1000X specifications.

17.1 System specifications 238
17.2 Physical specifications 239
17.3 Power requirements 239
17.4 CR1000Xe Power output specifications 240
17.5 Analog measurement specifications 242
17.6 Pulse measurement specifications 247
17.7 Digital input/output specifications 248
17.8 Communications specifications 250
17.9 Standards compliance specifications 251

17.1 System specifications

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

Memory (see Data memory (p. 88) for more information):

- Total onboard: 128 MB of flash + 4 MB battery-backed SRAM
  - Data storage: 4 MB SRAM + 72 MB flash (extended data storage automatically used for auto-allocated Data Tables not being written to a card)
  - CPU drive: 30 MB flash
  - OS load: 8 MB flash
  - Settings: 1 MB flash
  - Reserved (not accessible): 10 MB flash
- Data storage expansion: Removable microSD flash memory, up to 16 GB

Program Execution Period: 1 ms to 1 day

Real-Time Clock:

- Battery backed while external power is disconnected
- **Resolution**: 1 ms
- **Accuracy**: ±3 min. per year, optional GPS correction to ±10 µs

Wiring Panel Temperature: Measured using a 10K3A1A BetaTHERM thermistor, located between the two rows of analog input terminals.

17.2 Physical specifications

Dimensions: 23.8 x 10.1 x 6.2 cm (9.4 x 4.0 x 2.4 in); additional clearance required for cables and wires. For CAD files, see CR1000X Images and CAD 2D Drawings.

Weight/Mass: 0.86 kg (1.9 lb)

Case Material: Powder-coated aluminum

Operating Temperature Range:

- -40 °C to +70 °C (standard)
- -55°C to +85 °C (extended)

Storage Temperature Range: -60 to 85 °C

Operating Humidity: 0 to 100 %RH, non-condensing environment

17.3 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:

- **Supply Voltage**: 10 to 36 VDC
- **Sustained Supply Voltage without Damage**: 38 VDC

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.

USB Power: Functions that will be active with USB 5 VDC applied include sending programs, adjusting data logger settings, and making some measurements. If USB is the only power source, then the **CS I/O** port and the **5V**, **12V**, and **SW12** terminals will not be operational. When powered by only USB (no other power supplies connected) **Status table** field **Battery** = **0**.

Internal Lithium Battery: AA, 2.4 Ah, 3.6 VDC (Tadiran TL 5903/S) for battery-backed SRAM and clock. 3-year life with no external power source. See also Internal battery (p. 154).

Average Current Consumption (typ. at 20 °C):

Operating state	12 V Supply voltage	24 V Supply voltage
Idle	2.0 mA	<1.0 mA
Active 1 Hz Scan	2.0 mA	1.1 mA
Active 20 Hz Scan	57 mA	36 mA
Serial (RS-232/RS-485)	Active + 25 mA	Active + 16 mA
Ethernet Power Requirements:		
Ethernet 1 Minute	Active + 1 mA	Active + 0.7 mA
Ethernet Idle	Active + 4 mA	Active + 2.6 mA
Ethernet Link	Active + 47 mA	Active + 31 mA

17.4 CR1000Xe Power output specifications

17.4.1 System power output current limits

Temperature (°C)	12 V Supply voltage Current limit[1] (A)	24 V Supply voltage Current limit[1] (A)
–55°	3.4	4.4
–40°	3.4	4.4
20°	3.4	4.4
70°	2.5	4.2
85°	2.1	4.0

[1] Limited by self-resetting thermal fuse and maximum regulator output current.

17.4.2 Shared 12 V and SW12 power output

12V, **SW12-1**, and **SW12-2** provide regulated 12 VDC power. These outputs are disabled when operating on only USB power.

Temperature (°C)	12 V Supply voltage Current limit[1] (A)	24 V Supply voltage Current limit[1] (A)
–55°	3.3	3.3
–40°	3.3	3.3
20°	3.3	3.3
70°	2.5	3.3
85°	2.1	3.3
[1] Limited by self-resetting electronic and thermal fuses.		

17.4.3 Individual maximum current for 12 V and SW12 output terminals

Regulated 12 V output. System power output current limits may override one or more of these individual limits. These outputs are disabled when operating on only USB power.

- **Voltage Output**: Regulated 12 V output (±5%)
- **Current Limit**: 2000 mA

17.4.4 5 V fixed output

Regulated 5 V output. Supply is shared between the **5V** terminal and **CS I/O** DB9 5 V output.

- **Voltage Output**: Regulated 5 V output (±5%)
- **Current Limit**: 230 mA

17.4.5 Control port 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: $V_O = 5\ V – (R_O \bullet I_O)$, 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 10 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; $V_O = 5\ V - (150\ \Omega \bullet I_O)$
  - **3.3 V Logic Level Drive Capacity**: 10 mA @ 1.8 VDC; $V_O = 3.3\ V - (150\ \Omega \bullet I_O)$

17.4.6 CS I/O pin 1: 5 V fixed output

Regulated 5 V output. Supply is shared between the **5V** terminal and **CS I/O** DB9 5 V output.

- **Voltage Output**: Regulated 5 V output (±5%)
- **Current Limit**: 230 mA

17.4.7 CS I/O pin 8: 12 V switched output

Regulated 12 V output. Power output shared with system power output. This output is disabled when operating on only USB power.

- **Voltage Output**: Regulated 12 V output (±5%)
- **Current Limit**: 800 mA

17.4.8 Voltage excitation

VX: Four independently configurable voltage terminals (**VX1**-**VX4**). When providing voltage excitation, a single 16-bit DAC shared by all **VX** outputs produces a user-specified voltage during measurement only. In this case, these terminals are regularly used with resistive-bridge measurements (see Resistance measurements [p. 103] for more information). **VX** terminals can also be used to supply a selectable, switched, regulated 3.3 or 5 VDC power source to power digital sensors and toggle control lines.

	Range	Resolution	Accuracy	Maximum source/sink current[1]
Voltage Excitation	±4 V	0.12 mV	±(0.1% of setting + 2 mV)	±40 mA
Switched, Regulated	+3.3 or 5 V	3.3 or 5 V	±5%	50 mA

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

17.5 Analog measurement specifications

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

17.5.1 Voltage measurements

Terminals:

- **Differential Configuration**: DIFF 1H/1L – 8H/8L
- **Single-Ended Configuration**: SE1 – SE16

Input Resistance: 20 GΩ typical

Input Voltage Limits: ±5 V

Sustained Input Voltage without Damage: ±20 VDC

DC Common Mode Rejection:

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

Normal Mode Rejection: > 70 dB @ 60 Hz

Input Current @ 25 °C: ±1 nA typical

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

Analog Range and Resolution:

		Differential with input reversal		Single-ended and differential without input reversal	
Notch frequency (f_{N1}) (Hz)	Range[1] (mV)	RMS (µV)	Bits[2]	RMS (µV)	Bits[2]
15000	±5000	8.2	20	11.8	19
	±1000	1.9	20	2.6	19
	±200	0.75	19	1.0	18
50/60[3]	±5000	0.6	24	0.88	23
	±1000	0.14	23	0.2	23
	±200	0.05	22	0.08	22

		Differential with input reversal		Single-ended and differential without input reversal	
Notch frequency (f_{N1}) (Hz)	Range[1] (mV)	RMS (µV)	Bits[2]	RMS (µV)	Bits[2]
5	±5000	0.18	25	0.28	25
	±1000	0.04	25	0.07	24
	±200	0.02	24	0.03	23

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

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

[3] 50/60 corresponds to rejection of 50 and 60 Hz ac power mains noise.

Accuracy (does not include sensor or measurement noise):

- 0 to 40 °C: ±(0.04% of measurement + offset)
- –40 to 70 °C: ±(0.06% of measurement + offset)

Voltage Measurement Accuracy Offsets:

	Typical offset (µV RMS)	
Range (mV)	Differential with input reversal	Single-ended or differential without input reversal
±5000	±0.5	±2
±1000	±0.25	±1
±200	±0.15	±0.5

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

Multiplexed Measurement Time:

These are not maximum speeds. Multiplexed denotes circuitry inside the data logger that switches signals into the ADC.

$$Measurement\ Time = Setup\ Time + ((Settling\ Time + 1/fN1) \times M \times Repetitions$$

Where:

M = 1 (default)
M = 2 if reverse differential or measurement offset is used
Setup Time = 150 µs

	Differential with input reversal	Single-ended or differential without input reversal
Example fN1[1] (Hz)	Time[2] (ms)	Time[2] (ms)
15000	1.28	0.717
60	34.48	17.31
50	41.15	20.65
5	401.15	200.65

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

[2] Default settling time of 500 µs used.

See also Voltage measurements (p. 99).

17.5.2 Resistance measurement specifications

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

Accuracy:

Assumes input reversal for differential measurements `RevDiff` and excitation reversal `RevEx` for excitation voltage <1000 mV. Does not include bridge resistor errors or sensor and measurement noise.

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

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

17.5.3 Period-averaging measurement specifications

Use `PeriodAvg()` to measure the period (in microseconds) or the frequency (in Hz) of a signal on a single-ended channel.

Terminals: SE1-SE16

Accuracy: ±(0.01% of measurement + resolution), where resolution is 0.13 µs divided by the number of cycles to be measured

Ranges:

- Minimum signal centered around specified period average threshold.
- Maximum signal centered around data logger ground.
- Maximum frequency = 1/(2 * [minimum pulse width]) for 50% duty cycle signals

Gain code option	Voltage gain	Minimum peak to peak signal (mV)	Maximum peak to peak signal (V)	Minimum pulse width (µs)	Maximum frequency (kHz)
0	1	500	10	2.5	200
1	2.5	50	2	10	50
2	12.5	10	2	62	8
3	64	2	2	100	5

See also Period-averaging measurements (p. 111).

17.5.4 Current-loop measurement specifications

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

Terminals: **RG1** and **RG2**

Sustained Input Voltage without Damage: ±13.1 V

Resistance to Ground: 101 Ω

Current Measurement Shunt Resistance: 10 Ω

Maximum Current Measurement Range: ±80 mA

Sustained Maximum Current without Damage: ±130 mA

Resolution:

- **±1000 mV range**: ≤ 20 nA
- **±200 mV range**: ≤ 7.5 nA

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

See also Current-loop measurements (p. 101).

17.6 Pulse measurement specifications

Terminals individually configurable for switch closure, high-frequency pulse, or low-level AC measurements. See also Digital input/output specifications (p. 262). Each terminal has its own independent 24-bit counter.

> **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()`.

Sustained Input Voltage without Damage: (P1-P2): ±20 VDC

Sustained Logic Input Voltage without Damage: (C1-C8): +16/-12 VDC

Maximum Counts Per Scan: 2^{24}

Input Resistance: 5 kΩ

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

17.6.1 Low-level AC input

Terminals: P1-P2

Minimum Pull-Down Resistance: 10 kΩ to ground

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

17.6.2 Switch closure input

Terminals: C1-C8, P1-P2

Pull-Up Resistance: 100 kΩ to 5 V

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

Maximum Input Frequency: 100 Hz

Minimum Switch Closed Time: 5 ms

Minimum Switch Open Time: 5 ms

Maximum Bounce Time: 1 ms open without being counted

17.6.3 High-frequency input

Terminals: C1-C8, P1-P2

Pull-Up Resistance: 100 kΩ to 5 V

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

Maximum Input Frequency: 250 kHz

17.7 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, plus UART, RS-232, RS-422, RS-485, SDM, SDI-12, I2C, and SPI serial-communications functions. 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

Sustained Logic Input Voltage without Damage: +16/-12 VDC

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
C1 - C8	10 mA @ 3.5V	10 mA @ 1.85V

17.7.1 Switch closure input

Terminals: C1-C8, P1-P2

Pull-Up Resistance: 100 kΩ to 5 V

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

Maximum Input Frequency: 150 Hz

Minimum Switch Closed Time: 5 ms

Minimum Switch Open Time: 6 ms

Maximum Bounce Time: 1 ms open without being counted

17.7.2 High-frequency input

Terminals: C1-C8, P1-P2

Pull-Up Resistance: 100 kΩ to 5 V

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

Maximum Input Frequency: 250 kHz

17.7.3 Edge timing

Terminals: C1-C8

Maximum Input Frequency: ≤ 1 kHz

Resolution: 500 ns

17.7.4 Edge counting

Terminals: C1-C8

Maximum Input Frequency: ≤ 2.3 kHz

17.7.5 Quadrature input

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

Maximum Frequency: 2.5 kHz

Minimum Pulse Width: 10 µs

17.7.6 Pulse-width modulation

Terminals: C1-C8

Maximum Period: 128 seconds

Resolution:

- **0 – 5 ms**: 83.33 ns
- **5 – 300 ms**: 5.33 µs
- **> 300 ms**: 1.95 ms

See also Pulse measurements (p. 112) and Pulse measurement specifications (p. 260).

17.8 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, 10/100Base Mbps, full and half duplex, Auto-MDIX, magnetic isolation, and TVS surge protection. See also Ethernet communications option (p. 35).

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, MQTT

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

USB: Type C 2.0. Full speed: 12 Mbps. Operates as:

- Device for computer communications

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

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**): Up to two full duplex or four half duplex

RS-422 (**C1** to **C8**): Up to two full duplex or four half duplex

RS-232/CPI: Single RJ45 module port 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.

CPI: One CPI bus. Up to 1 Mbps data rate. Synchronization of devices to 5 µS. Total cable length up to 610 m (2000 ft). Up to 20 devices. 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.

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

Satellite: GOES, Argos, Inmarsat Hughes, Irridium

17.9 Standards compliance specifications

View compliance and conformity documents at www.campbellsci.com/cr1000x.

Test	Applied standard	Description
Shock and vibration:	MIL-STD 810G methods 516.6 and 514.6	
Protection:		
Wiring panel	IP40	
Measurement module when connected to wiring panel	IP65	
EMI and ESD immunity:		
ESD	IEC 61000-4-2	±15 kV air, ±8 kV contact discharge
Radiated RF	IEC 61000-4-3	10 V/m, 80-1000 MHz
EFT	IEC 61000-4-4	4 kV power, 4 kV I/O
Surge	IEC 61000-4-5	4 kV power, 4kV I/O
Conducted RF	IEC 61000-4-6	10 V power, 10 V I/O
Emissions and immunity performance criteria available on request.		

18. CR1000X specifications

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

See CR1000Xe specifications (p. 238) for CR1000Xe specifications.

18.1 System specifications 252
18.2 Physical specifications 253
18.3 Power requirements 253
18.4 Power output specifications 254
18.5 Analog measurement specifications 256
18.6 Pulse measurement specifications 260
18.7 Digital input/output specifications 262
18.8 Communications specifications 264
18.9 Standards compliance specifications 265

18.1 System specifications

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

Memory (see Data memory (p. 88) for more information):

- Total onboard: 128 MB of flash + 4 MB battery-backed SRAM
  - Data storage: 4 MB SRAM + 72 MB flash (extended data storage automatically used for auto-allocated Data Tables not being written to a card)
  - CPU drive: 30 MB flash
  - OS load: 8 MB flash
  - Settings: 1 MB flash
  - Reserved (not accessible): 10 MB flash
- Data storage expansion: Removable microSD flash memory, up to 16 GB

Program Execution Period: 1 ms to 1 day

Real-Time Clock:

- Battery backed while external power is disconnected
- Resolution: 1 ms
- Accuracy: ±3 min. per year, optional GPS correction to ±10 µs

Wiring Panel Temperature: Measured using a 10K3A1A BetaTHERM thermistor, located between the two rows of analog input terminals.

18.2 Physical specifications

Dimensions: 23.8 x 10.1 x 6.2 cm (9.4 x 4.0 x 2.4 in); additional clearance required for cables and wires. For CAD files, see CR1000X Images and CAD 2D Drawings.

Weight/Mass: 0.86 kg (1.9 lb)

Case Material: Powder-coated aluminum

Operating Temperature Range:

- -40 °C to +70 °C (standard)
- -55°C to +85 °C (extended)

Storage Temperature Range: -60 to 85 °C

Operating Humidity: 0 to 100 %RH, non-condensing environment

18.3 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:

- Input Voltage: 10 to 18 VDC

  > NOTE: To prevent voltage input issues with sensors and peripherals, do not use more than 16 V when powering them through the 12V, SW12-1, SW12-2, or CS I/O port on the data logger.

- Input Current Limit at 12 VDC:
  - 4.35 A at -40 °C
  - 3 A at 20 °C
  - 1.56 A at 85 °C

- **Sustained Input Voltage without Damage**: 30 VDC

  Transient voltage suppressor (TVS) diodes at the **POWER IN** terminal clamps transients to 36 to 40 V. Input voltages greater than 18 V and less than 32 V are tolerated; however, the 12 V output **SW12-1** and **SW12-2** are disabled and will not function until the input voltage falls below 16 V. Sustained input voltages in excess of 32 V can damage the TVS diodes. If the voltage on the **POWER IN** terminals exceeds 19 V, power is shut off to certain parts of the data logger to prevent damaging connected sensors or peripherals.

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.

USB Power: Functions that will be active with USB 5 VDC applied include sending programs, adjusting data logger settings, and making some measurements. If USB is the only power source, then the **CS I/O** port and the **5V**, **12V**, and **SW12** terminals will not be operational. When powered by only USB (no other power supplies connected) **Status table** field **Battery** = **0**.

Internal Lithium Battery: AA, 2.4 Ah, 3.6 VDC (Tadiran TL 5903/S) for battery-backed SRAM and clock. 3-year life with no external power source. See also Internal battery (p. 154).

Average Current Drain:

Assumes 12 VDC on **POWER IN** terminals.

- **Idle**: <1 mA, typical
- **Active 1 Hz Scan**: 1 mA
- **Active 20 Hz Scan**: 55 mA
- **Serial** (RS-232/RS-485): Active + 25 mA
- **Ethernet Power Requirements**:
  - **Ethernet 1 Minute**: Active + 1 mA
  - **Ethernet Idle**: Active + 4 mA
  - **Ethernet Link**: Active + 47 mA

18.4 Power output specifications

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

Temperature (°C)	Current limit[1] (A)
–40°	4.53
20°	3.00

Temperature (°C)	Current limit[1] (A)
70°	1.83
85°	1.56
[1] Limited by self-resetting thermal fuse	

18.4.2 12 V and SW12 V power output terminals

12V, **SW12-1**, and **SW12-2:** Provide unregulated 12 VDC power with voltage equal to the Power Input supply voltage. These are disabled when operating on USB power only. The **12V** terminal is limited to the current shown in the previous table.

SW12 current limits	
Temperature (°C)	Current limit [1] (mA)
–40°	1310
0°	1004
20°	900
50°	690
70°	550
80°	470
[1] Thermal fuse hold current. Overload causes voltage drop. Disconnect and let cool to reset. Operate at limit if the application can tolerate some fluctuation.	

18.4.3 5 V fixed output

5V: One regulated 5 V output. Supply is shared between the **5V** terminal and **CS I/O** DB9 5 V output.

- **Voltage Output**: Regulated 5 V output (±5%)
- **Current Limit**: 230 mA

18.4.4 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: $V_O = 5\ V - (R_O \bullet I_O)$, 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 10 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 V - (150 Ω •I_o)
  - **3.3 V Logic Level Drive Capacity**: 10 mA @ 1.8 VDC; V_o = 3.3 V - (150 Ω •I_o)

18.4.5 CS I/O pin 1

5 V Logic Level Max Current: 200 mA

18.4.6 Voltage excitation

VX: Four independently configurable voltage terminals (**VX1**-**VX4**). When providing voltage excitation, a single 16-bit DAC shared by all **VX** outputs produces a user-specified voltage during measurement only. In this case, these terminals are regularly used with resistive-bridge measurements (see Resistance measurements [p. 103] for more information). **VX** terminals can also be used to supply a selectable, switched, regulated 3.3 or 5 VDC power source to power digital sensors and toggle control lines.

	Range	Resolution	Accuracy	Maximum source/sink current[1]
Voltage Excitation	±4 V	0.06 mV	±(0.1% of setting + 2 mV)	±40 mA
Switched, Regulated	+3.3 or 5 V	3.3 or 5 V	±5%	50 mA

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

18.5 Analog measurement specifications

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

18.5.1 Voltage measurements

Terminals:

- **Differential Configuration**: DIFF 1H/1L – 8H/8L
- **Single-Ended Configuration**: SE1 – SE16

Input Resistance: 20 GΩ typical

Input Voltage Limits: ±5 V

Sustained Input Voltage without Damage: ±20 VDC

DC Common Mode Rejection:

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

Normal Mode Rejection: > 70 dB @ 60 Hz

Input Current @ 25 °C: ±1 nA typical

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

Analog Range and Resolution:

		Differential with input reversal		Single-ended and differential without input reversal	
Notch frequency (f_{N1}) (Hz)	Range[1] (mV)	RMS (µV)	Bits[2]	RMS (µV)	Bits[2]
15000	±5000	8.2	20	11.8	19
	±1000	1.9	20	2.6	19
	±200	0.75	19	1.0	18
50/60[3]	±5000	0.6	24	0.88	23
	±1000	0.14	23	0.2	23
	±200	0.05	22	0.08	22

		Differential with input reversal		Single-ended and differential without input reversal	
Notch frequency (f_{N1}) (Hz)	Range[1] (mV)	RMS (µV)	Bits[2]	RMS (µV)	Bits[2]
5	±5000	0.18	25	0.28	25
	±1000	0.04	25	0.07	24
	±200	0.02	24	0.03	23

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

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

[3] 50/60 corresponds to rejection of 50 and 60 Hz ac power mains noise.

Accuracy (does not include sensor or measurement noise):

- 0 to 40 °C: ±(0.04% of measurement + offset)
- –40 to 70 °C: ±(0.06% of measurement + offset)

Voltage Measurement Accuracy Offsets:

	Typical offset (µV RMS)	
Range (mV)	Differential with input reversal	Single-ended or differential without input reversal
±5000	±0.5	±2
±1000	±0.25	±1
±200	±0.15	±0.5

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

Multiplexed Measurement Time:

These are not maximum speeds. Multiplexed denotes circuitry inside the data logger that switches signals into the ADC.

$$Measurement\ Time = Setup\ Time + ((Settling\ Time + 1/fN1) \times M \times Repetitions$$

Where:

M = 1 (default)
M = 2 if reverse differential or measurement offset is used
Setup Time = 150 µs

	Differential with input reversal	Single-ended or differential without input reversal
Example fN1[1] (Hz)	Time[2] (ms)	Time[2] (ms)
15000	2.04	1.02
60	35.24	17.62
50	41.9	20.95
5	401.9	200.95

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

[2] Default settling time of 500 µs used.

See also Voltage measurements (p. 99).

18.5.2 Resistance measurement specifications

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

Accuracy:

Assumes input reversal for differential measurements `RevDiff` and excitation reversal `RevEx` for excitation voltage <1000 mV. Does not include bridge resistor errors or sensor and measurement noise.

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

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

18.5.3 Period-averaging measurement specifications

Use `PeriodAvg()` to measure the period (in microseconds) or the frequency (in Hz) of a signal on a single-ended channel.

Terminals: SE1-SE16

Accuracy: ±(0.01% of measurement + resolution), where resolution is 0.13 µs divided by the number of cycles to be measured

Ranges:

- Minimum signal centered around specified period average threshold.
- Maximum signal centered around data logger ground.
- Maximum frequency = 1/(2 * [minimum pulse width]) for 50% duty cycle signals

Gain code option	Voltage gain	Minimum peak to peak signal (mV)	Maximum peak to peak signal (V)	Minimum pulse width (µs)	Maximum frequency (kHz)
0	1	500	10	2.5	200
1	2.5	50	2	10	50
2	12.5	10	2	62	8
3	64	2	2	100	5

See also Period-averaging measurements (p. 111).

18.5.4 Current-loop measurement specifications

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

Terminals: **RG1** and **RG2**

Maximum Input Voltage: ±16 V

Resistance to Ground: 101 Ω

Current Measurement Shunt Resistance: 10 Ω

Maximum Current Measurement Range: ±80 mA

Absolute Maximum Current: ±160 mA

Resolution: ≤ 20 nA

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

See also Current-loop measurements (p. 101).

18.6 Pulse measurement specifications

Terminals individually configurable for switch closure, high-frequency pulse, or low-level AC measurements. See also Digital input/output specifications (p. 262). Each terminal has its own independent 24-bit counter.

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().

Sustained Input Voltage without Damage: ±20 VDC

Maximum Counts Per Scan: 2^{24}

Input Resistance: 5 kΩ

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

18.6.1 Low-level AC input

Terminals: P1-P2

Minimum Pull-Down Resistance: 10 kΩ to ground

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

18.6.2 Switch closure input

Terminals: C1-C8, P1-P2

Pull-Up Resistance: 100 kΩ to 5 V

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

Maximum Input Frequency: 150 Hz

Minimum Switch Closed Time: 5 ms

Minimum Switch Open Time: 6 ms

Maximum Bounce Time: 1 ms open without being counted

18.6.3 High-frequency input

Terminals: C1-C8, P1-P2

Pull-Up Resistance: 100 kΩ to 5 V

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

Maximum Input Frequency: 250 kHz

18.7 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, plus UART, RS-232, RS-422, RS-485, SDM, SDI-12, I2C, and SPI serial-communications functions. 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

Sustained Logic Input Voltage without Damage: ±20 VDC

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
C1 - C8	10 mA @ 3.5V	10 mA @ 1.85V

18.7.1 Switch closure input

Terminals: C1-C8, P1-P2

Pull-Up Resistance: 100 kΩ to 5 V

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

Maximum Input Frequency: 150 Hz

Minimum Switch Closed Time: 5 ms

Minimum Switch Open Time: 6 ms

Maximum Bounce Time: 1 ms open without being counted

18.7.2 High-frequency input

Terminals: C1-C8, P1-P2

Pull-Up Resistance: 100 kΩ to 5 V

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

Maximum Input Frequency: 250 kHz

18.7.3 Edge timing

Terminals: C1-C8

Maximum Input Frequency: ≤ 1 kHz

Resolution: 500 ns

18.7.4 Edge counting

Terminals: C1-C8

Maximum Input Frequency: ≤ 2.3 kHz

18.7.5 Quadrature input

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

Maximum Frequency: 2.5 kHz

Minimum Pulse Width: 10 µs

18.7.6 Pulse-width modulation

Terminals: C1-C8

Maximum Period: 36.4 seconds

Resolution:

- **0 – 5 ms**: 83.33 ns
- **5 – 325 ms**: 5.33 µs
- **> 325 ms**: 31.25 µs

See also Pulse measurements (p. 112) and Pulse measurement specifications (p. 260).

18.8 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, 10/100Base Mbps, full and half duplex, Auto-MDIX, magnetic isolation, and TVS surge protection. See also Ethernet communications option (p. 35).

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, MQTT

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

USB Device: Micro-B device for computer connectivity

USB: Type C 2.0. Full speed: 12 Mbps. Operates as:

- Device for computer communications

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

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 (**C5** to **C8**): One full duplex or two half duplex

RS-422 (**C5** to **C8**): One full duplex or two half duplex

RS-232/CPI: Single RJ45 module port 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.

CPI: One CPI bus. Up to 1 Mbps data rate. Synchronization of devices to 5 µS. Total cable length up to 610 m (2000 ft). Up to 20 devices. 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.

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

Satellite: GOES, Argos, Inmarsat Hughes, Irridium

18.9 Standards compliance specifications

View compliance and conformity documents at www.campbellsci.com/cr1000x.

Test	Applied standard	Description
Shock and vibration:	MIL-STD 810G methods 516.6 and 514.6	
Protection:		
Wiring panel	IP40	
Measurement module when connected to wiring panel	IP65	
EMI and ESD immunity:		
ESD	IEC 61000-4-2	±15 kV air, ±8 kV contact discharge
Radiated RF	IEC 61000-4-3	10 V/m, 80-1000 MHz
EFT	IEC 61000-4-4	4 kV power, 4 kV I/O
Surge	IEC 61000-4-5	4 kV power, 4kV I/O
Conducted RF	IEC 61000-4-6	10 V power, 10 V I/O
Emissions and immunity performance criteria available on request.		

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

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

Aggregate Type

Denotes the aggregate type for the incoming measurement into CampbellCloud. For example, if the incoming measurement is a minimum value (for example, minimum battery voltage), set aggregate type to minimum.

alias

A second name assigned to variable in CRBasic.

allowed neighbor list

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

amperes (A)

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

analog

Data presented as continuously variable electrical signals.

API

Application Programming Interface

application

Also called app for short. A group of functions for related tasks.

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.

asset

Primarily this is a data source such as a data logger or CR1000X/CR1000Xe. It can also be another piece of hardware.

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 communications, this coordination is accomplished by having each character surrounded by one or more start and stop bits which designate the beginning and ending points of the information. Also indicates the sending and receiving devices are not synchronized using a clock signal.

AWG

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

B

baud rate

The rate at which data is transmitted.

beacon

A signal broadcasted to other devices in a PakBus network to identify "neighbor" devices. A beacon in a PakBus network ensures that all devices in the network are aware of other devices that are viable.

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."

CampbellGo

A companion mobile field app for CampbellCloud, available for iOS and Android

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®

Classification

Refers to the primary classification of a measurement, such as tempreature, relative humidity, or precipitation.

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 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 PC400, LoggerNet, and RTDAQ software.

COM port

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

COM1

When configured as a communications 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 analog 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.

CS I/O

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

CVI

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

D

DAC

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

data bits

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

data cache

The data cache is a set of binary files kept on the hard disk of the computer running the 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, and PC400 - 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

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 source

An asset that sends data to CampbellCloud. This includes data loggers and the Aspen 10 edge device.

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 memory. The data table structure resides in the data cache, in discrete data files, and in files that result from collecting final-data memory with data logger support software.

DC

Direct current.

DCE

Data Communications 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. Also called DevConfig.

DHCP

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

differential

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

Dim

A CRBasic command for declaring and dimensioning variables. Variables declared with Dim remain hidden during 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 communications 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 analog voltage measurements. Note that most objects have a "an electrical potential" and the potential at different places on the earth - even a few meters away - may be different.

endian

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

engineering units

Units that explicitly describe phenomena, as opposed to, for example, the data logger base analog-measurement unit of millivolts.

ESD

Electrostatic discharge.

ESS

Environmental sensor station.

Etc/UTC

UTC time zone refers to Coordinated Universal Time (UTC). The "Etc" prefix is used in certain systems (like UNIX/Linux) to provide a standardized label for time zones. Etc/UTC is always at the same time, regardless of the time of year.

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, 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 memory allocated for storing data tables. 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 gage thermocouple at 1 kHz will give a high sample rate but does not ensure the measurement has a high frequency response. A fine-wire thermocouple, which changes output quickly with changes in temperature, is more likely to have a high frequency response.

FTP

File Transfer Protocol. A TCP/IP application protocol.

full-duplex

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

G

garbage

The refuse of the data communications 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 navigation satellite system

A satellite navigation system with global coverage such as GPS (North America), Galileo (Europe), and BeiDou (China).

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 communications 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 analog voltages.

H

half-duplex

A serial communications 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.

hello exchange

In a PakBus network, this is the process of verifying a node as a neighbor.

hertz

SI unit of frequency. Cycles or pulses per second.

Hidden station

Stations that are missing location data. They cannot be geo-located on a map.

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-controller, multi-peripheral, 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

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 communications 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 communications signals to pass, but breaks alternate ground paths and may filter some electromagnetic noise.

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,

neighbor lists, routing table, and communications timeouts.

keyboard/display

The data logger has an optional external keyboard/display.

L

leaf

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 communications ports to another. It will not maintain a list of neighbors, 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.

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 communications ports to another. It will not maintain a list of neighbors, 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.

lf

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

linearity

The quality of delivering identical sensitivity throughout the measurement.

local variable

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

LoggerLink

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

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

modulo divide

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

MQTT

An open communications protocol for the Internet of Things (IoT). MQTT is not an acronym, it is simply the name of the protocol. Source: https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/

MSB

Most significant bit (the leading bit).

multimeter

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

multiplier

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

mV

The SI abbreviation for millivolts.

N

NAN

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

neighbor device

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

network

A group of one or more stations.

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.com/loggernet.

NFC

Near field communications

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 °F = °C*1.8 + 32. The factor 32 is the offset.

ohm

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

Ohm's Law

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

on-line data transfer

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

onboard

A collective term for the tasks that have to complete successfully in order for a data source asset to be correctly configured and send data to CampbellCloud. These tasks may be automated or require manual user input depending on the data source type. For data logger data sources, these tasks include asset claiming, automated sensor identification, cellular communications registration, secure Cloud communications, program retrieval, successful sensor measurement, and confirmation that Cloud received data.

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.com/downloads.

organization

An entity (individual, business, or group) that uses CampbellCloud services to manage a network of stations owned by the entity. Every user must be associated with an organization.

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

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

owner

An owner is a member of the Owners security group. By default, the Owners security group is created with a new CampbellCloud Organization account. The creator of the organization account is automatically added to the Owners security group and can then add other users to the group or create other security groups with different permissions. Owners have access to all functionality across the range of applications in CampbellCloud (i.e., users within the Owners security group have all permissions enabled).

P

PakBus

™ A proprietary communications 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 Encryption Key

A PakBus Encryption Key is a security feature used with Campbell Scientific data loggers that communicate via the PakBus protocol. It helps protect data and restrict unauthorized access to the data logger.

PakBus Graph

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

parameter

Part of a procedure (or command) definition.

PC200W

Retired basic data logger support software for direct connect.

PC400

Free entry-level data logger support software that supports a variety of communications options, manual data collection, and data monitoring displays. Short Cut and CRBasic Editor are included for creating data logger programs. PC400 does not support scheduled data collection or complex communications options such as phone-to-RF.

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 communications 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).

Precision

Specifies the number of decimal places shown for a measurement.

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.

Q

QR code

Quick response barcode

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.

recipe

A set of files that include the CR1000X/CR1000Xe program, settings and configuration for a specific sensor and application.

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 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 neighbors and sends this list to other routers in the PakBus network. It also obtains and receives neighbor lists from other routers. Routers maintain a routing table, which is a list of known nodes and routes. A router will only accept and forward packets that are destined for known devices. Routers pass their lists of known neighbors to other routers to build the network routing system.

RS-232

Recommended Standard 232. A loose standard defining how two computing devices can communicate with each other. The implementation of RS-232 in Campbell Scientific data loggers to computer communications is quite rigid, but transparent to most users. Features in the data logger that implement RS-232 communications 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 Unit

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 gage thermocouple at 1 kHz will give a high sample rate but does not ensure the measurement has a high frequency response. A fine-wire thermocouple, which changes output quickly with changes in temperature, is more likely to have a high frequency response.

SCADA

Supervisory Control And Data Acquisition

scan interval

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

scan time

When time functions are run inside the Scan() / NextScan construct, time stamps are based on when the scan was started according to the data logger clock. Resolution of scan time is equal to the length of the scan.

SDI-12

Serial Data Interface at 1200 baud. Communications 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.

security group

An application used to control user access to applications and their associated permissions. Users can be in more than one security group.

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 communications 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 analog voltage signal is carried on a single wire and measured with respect to ground (0 V).

skipped scans

Occur when the CRBasic program is too long for the scan interval. Skipped scans can cause errors in pulse measurements.

slow sequence

A usually slower secondary scan in the CRBasic program. The main scan has priority over a slow sequence.

SMS

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

SMTP

Simple Mail Transfer Protocol. A TCP/IP application protocol.

SNP

Snapshot file.

SP

Space.

SPI

Serial Peripheral Interface - a clocked synchronous interface, used for short distance communications, generally between embedded devices.

SRAM

Static Random-Access Memory

start bit

The bit used to indicate the beginning of data.

state

Whether a device is on or off.

station

A group of one or more assets

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.

Subclassification

Refers to the secondary classification of a measurement. For example, a temperature classification can have multiple subclassifications, such as air temperature, dew point temperature, or soil temperature

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 communications, this coordination is accomplished by synchronizing the transmitting and receiving devices to a common clock signal (see also asynchronous).

system time

When time functions are run outside the Scan() / NextScan construct, the time registered by the instruction will be based on the system clock, which has a 10 ms resolution.

T

table

See data table.

task

Grouping of CRBasic program instructions automatically by the 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.

thing

A thing resource is a digital representation of a physical device or logical entity in AWS IoT.

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 communications security protocol.

TOA5

Also called ASCII, Long Header. Data stored in a comma separated format. Header information for each column is included, along with field names and units of measure if they are available. Table output ascii version 5. See the LoggerNet manual appendix for details on differnet file formats.

TOACI1

Also called ASCII, Short Header. Data stored in a comma separated format. Header information for each of the columns is included. Table output ASCII version 1. See the LoggerNet manual appendix for details on differnet file formats.

TOB1

Binary. Data stored in a binary format. Though this format saves disk storage space, it must be converted before it is usable in other programs. Table output binary version 1. See the LoggerNet manual appendix for details on differnet file formats.

TOB3

Binary. Data stored to a card in a binary format. Table output binary version 1. See the LoggerNet manual appendix for details on differnet file formats.

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.

UID

Unique identifier

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.

Units

Specifies the unit type of the incoming measurement into CampbellCloud. For example, if the asset is sending a temperature measurement to CampbellCloud in degrees Celsius, Units must be set to degrees Celsius.

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

Individuals who have been added to an organization account. Users are assigned permissions via the Security Groups application.

user program

The CRBasic program written by you in Short Cut program wizard.

USR drive

A portion of memory dedicated to the storage of image or other files.

V

VAC

Volts alternating current.

variable

A packet of memory given an alphanumeric name.

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

volts

SI unit for electrical potential.

VSPECT™

™ A registered trademark for Campbell Scientific's proprietary spectral-analysis, frequency domain, vibrating wire measurement technique.

W

watchdog timer

An error-checking system that examines the processor state, software timers, and program-related counters when the CRBasic program is running. The following will cause watchdog timer resets, which reset the processor and CRBasic program execution: processor bombed, processor neglecting standard system updates, counters are outside the limits, voltage surges, and voltage transients. When a reset occurs, a counter is incremented in the WatchdogTimer entry of the Status table. A low number (1 to 10) of watchdog timer resets is of concern, but normally indicates that the situation should just be monitored. A large number of errors (>10) accumulating over a short period indicates a hardware or software problem. Consult with a Campbell Scientific support engineer.

weather-tight

Describes an instrumentation enclosure impenetrable by common environmental conditions. During extraordinary weather events, however, seals on the enclosure may be breached.

web API

Application Programming Interface

wild card

A character or expression that substitutes for any other character or expression.

X

XML

Extensible markup language.

T

τ

Time constant

Limited warranty

Covered equipment is warranted/guaranteed against defects in materials and workmanship under normal use and service for the period listed on your sales invoice or the product order information web page. The covered period begins on the date of shipment unless otherwise specified. For a repair to be covered under warranty, the following criteria must be met:

1. There must be a defect in materials or workmanship that affects form, fit, or function of the device.

2. The defect cannot be the result of misuse.

3. The defect must have occurred within a specified period of time; and

4. The determination must be made by a qualified technician at a Campbell Scientific Service Center/ repair facility.

The following is not covered:

1. Equipment which has been modified or altered in any way without the written permission of Campbell Scientific.

2. Batteries; and

3. Any equipment which has been subjected to misuse, neglect, acts of God or damage in transit.

Campbell Scientific regional offices handle repairs for customers within their territories. Please see the back page of the manual for a list of regional offices or visit www.campbellsci.com/contact to determine which Campbell Scientific office serves your country. For directions on how to return equipment, see Assistance.

Other manufacturer's products, that are resold by Campbell Scientific, are warranted only to the limits extended by the original manufacturer.

CAMPBELL SCIENTIFIC EXPRESSLY DISCLAIMS AND EXCLUDES ANY IMPLIED WARRANTIES OF

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Campbell Scientific hereby disclaims, to the fullest extent allowed by applicable law, any and all warranties and conditions with respect to the products, whether express, implied, or statutory, other than those expressly provided herein.

Campbell Scientific will, as a default, return warranted equipment by surface carrier prepaid. However, the method of return shipment is at Campbell Scientific's sole discretion. Campbell Scientific will not reimburse the claimant for costs incurred in removing and/or reinstalling equipment. This warranty and the Company's obligation thereunder is in lieu of all other

warranties, expressed or implied, including those of suitability and fitness for a particular purpose. Campbell Scientific is not liable for consequential damage.

In the event of any conflict or inconsistency between the provisions of this Warranty and the provisions of Campbell Scientific's Terms, the provisions of Campbell Scientific's Terms shall prevail. Furthermore, Campbell Scientific's Terms are hereby incorporated by reference into this Warranty. To view Terms and conditions that apply to Campbell Scientific, Logan, UT, USA, see Terms and Conditions. To view terms and conditions that apply to Campbell Scientific offices outside of the United States, contact the regional office that serves your country.

Acknowledgements

lwIP v 2.1.1, LIBSSH2 v. 1.8.0, and Newlib

Copyright 2026 Campbell Scientific.

All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

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

Mbed TLS v. 3.1.0

Copyright 2026 Campbell Scientific.

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on

behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices

normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License.

You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility,

not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner] Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Assistance

Products may not be returned without prior authorization. Please inform us before returning equipment and obtain a **return material authorization (RMA) number** whether the repair is under warranty/guarantee or not. See Limited warranty for information on covered equipment.

Campbell Scientific regional offices handle repairs for customers within their territories. Please see the back page of the manual for a list of regional offices or visit www.campbellsci.com/contact to determine which Campbell Scientific office serves your country.

When returning equipment, a RMA number must be clearly marked on the outside of the package. Please state the faults as clearly as possible. 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, when equipment is returned to Campbell Scientific, Logan, UT, USA, it is mandatory that a "Declaration of Hazardous Material and Decontamination" form be received before the return can be processed. If the form is not

received within 5 working days of product receipt or is incomplete, the product will be returned to the customer at the customer's expense. For details on decontamination standards specific to your country, please reach out to your regional Campbell Scientific office.

> NOTE:
> All goods that cross trade boundaries may be subject to some form of fee (customs clearance, duties or import tax). Also, some regional offices require a purchase order upfront if a product is out of the warranty period. Please contact your regional Campbell Scientific office for details.

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.com 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

- Protect from over-voltage.
- Protect electrical equipment from water.
- Protect from electrostatic discharge (ESD).
- Protect from lightning.
- 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, 6 meters (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.
- Only use power sources approved for use in the country of installation to power Campbell Scientific devices.

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.

Internal Battery

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

Use and disposal of batteries

- Where batteries need to be transported to the installation site, ensure they are packed to prevent the battery terminals shorting which could cause a fire or explosion. Especially in the case of lithium batteries, ensure they are packed and transported in a way that complies with local shipping regulations and the safety requirements of the carriers involved.
- When installing the batteries follow the installation instructions very carefully. This is to avoid risk of damage to the equipment caused by installing the wrong type of battery or reverse connections.
- When disposing of used batteries, it is still important to avoid the risk of shorting. Do not dispose of the batteries in a fire as there is risk of explosion and leakage of harmful chemicals into the environment. Batteries should be disposed of at registered recycling facilities.

Avoiding unnecessary exposure to radio transmitter radiation

- Where the equipment includes a radio transmitter, precautions should be taken to avoid unnecessary exposure to radiation from the antenna. The degree of caution required varies with the power of the transmitter, but as a rule it is best to avoid getting closer to the antenna than 20 cm (8 inches) when the antenna is active. In particular keep your head away from the antenna. For higher power radios (in excess of 1 W ERP) turn the radio off when servicing the system, unless the antenna is installed away from the station, e.g. it is mounted above the system on an arm or pole.

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.

Global Sales and 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:	Montrouge, 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

Japan

Location:	Kawagishi, Toda City, Japan
Phone:	048.400.5001
Email:	jp-info@campbellsci.com
Website:	www.campbellsci.co.jp

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