INSTRUCTION MANUAL

SC115 CS I/O 2G Flash Memory Drive with USB Interface

Revision: 10/16

Guarantee

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

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

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

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

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

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

Campbell Scientific Ltd,
80 Hathern Road,
Shepshed, Loughborough, LE12 9GX, UK
Tel: +44 (0) 1509 601141
Fax: +44 (0) 1509 601091
Email: support@campbellsci.co.uk
www.campbellsci.co.uk

PLEASE READ FIRST

About this manual

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

Some useful conversion factors:

Area: 1 in^2 (square inch) = 645 mm^2

Length: 1 in. (inch) = 25.4 mm
1 ft (foot) = 304.8 mm
1 yard = 0.914 m
1 mile = 1.609 km

Mass: 1 oz. (ounce) = 28.35 g
1 lb (pound weight) = 0.454 kg

Pressure: 1 psi (lb/in^2) = 68.95 mb

Volume: 1 UK pint = 568.3 ml
1 UK gallon = 4.546 litres
1 US gallon = 3.785 litres

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

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

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

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

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

Recycling information

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

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

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

Campbell Scientific Ltd, 80 Hathern Road, Shepshed, Loughborough, LE12 9GX, UK
Tel: +44 (0) 1509 601141 Fax: +44 (0) 1509 601091
Email: support@campbellsci.co.uk
www.campbellsci.co.uk

Precautions

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

Use tripods, towers, and attachments to tripods and towers only for purposes for which they are designed. Do not exceed design limits. Be familiar and comply with all instructions provided in product manuals. Manuals are available at www.campbellsci.eu or by telephoning +44(0) 1509 828 888 (UK). You are responsible for conformance with governing codes and regulations, including safety regulations, and the integrity and location of structures or land to which towers, tripods, and any attachments are attached. Installation sites should be evaluated and approved by a qualified engineer. If questions or concerns arise regarding installation, use, or maintenance of tripods, towers, attachments, or electrical connections, consult with a licensed and qualified engineer or electrician.

General

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

Utility and Electrical

- **You can be killed** or sustain serious bodily injury if the tripod, tower, or attachments you are installing, constructing, using, or maintaining, or a tool, stake, or anchor, come in **contact with overhead or underground utility lines**.
- Maintain a distance of at least one-and-one-half times structure height, or 20 feet, or the distance required by applicable law, **whichever is greater**, between overhead utility lines and the structure (tripod, tower, attachments, or tools).
- Prior to performing site or installation work, inform all utility companies and have all underground utilities marked.
- Comply with all electrical codes. Electrical equipment and related grounding devices should be installed by a licensed and qualified electrician.

Elevated Work and Weather

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

Maintenance

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

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

Table of Contents

PDF viewers: These page numbers refer to the printed version of this document. Use the PDF reader bookmarks tab for links to specific sections.

Tables

SC115 CS I/O 2G Flash Memory Drive with USB Interface

1. Introduction

The SC115 is a portable 2-GB memory drive (thumb drive) compatible with Campbell Scientific CRBasic dataloggers that have a CS I/O port. It shuttles data, OS, and program files between Campbell Scientific dataloggers and a PC, or it remains connected to the datalogger to augment data memory. It can be used in tandem with an external memory card.

The SC115 can also be used as a CS I/O to USB communications interface.

2. Cautionary Statements

- Corruption of multiple data files may occur if the SC115 is removed from the host datalogger or PC during data transfer.
- Do not disconnect the SC115 from the host while the LED is flashing or lit. Always use the **Safely Remove Hardware** utility provided in the Windows® operating system prior to removal from a PC.
- Some data collection modes increase the risk of inadvertently disconnecting the SC115 from the datalogger during data transfer. See Section 12.1, *Preventing Data Corruption* *(p. 16)*, to assess the risk in a particular application.
- Always click **Disconnect** in the datalogger support software prior to removing the SC115 when it is a communications interface. See Section 11, *Serial Communications Interface Function* *(p. 15)*, for communication interface details and precautions.
- Before the SC115 can be used as a CS I/O to USB communications interface or be configured through the *Device Configuration Utility* (*DevConfig*), the device driver must be installed. *DevConfig* is used to install the device driver. Under **Device Type**, select **Peripheral | SC115**. Click the **install the device driver for the SC115** link and follow the prompts.

3. Initial Inspection

Upon receipt of the SC115, inspect the packaging and contents for damage. File damage claims with the shipping company. Immediately check package contents against the shipping documentation. Contact Campbell Scientific concerning discrepancies.

4. Quickstart

The primary function of the SC115 is as a portable memory drive (thumb drive) to collect data from compatible Campbell Scientific dataloggers (see Section 6, *Specifications* *(p. 3)*, for compatibility). Simply connect the CS I/O

connector of the SC115 to the CS I/O connector of a properly programmed datalogger, and data are transferred automatically.

Memory drive setup (see Section 7, *Setup (p. 4)*, for complete instructions):

- SC115 – shipped from the factory ready for use in most applications.
- Datalogger – as shown in the following code example, insert the **TableFile()** instruction into the datalogger CRBasic program immediately following the **DataInterval()** instruction. This configuration enables the SC115 to collect the newest data in the data collection mode (see Section 8.1, *Data Collection Modes (p. 4)*). Note that in data collection mode, data will only be written to the SC115 when it is first attached to a datalogger. While left connected, no new data will be written to the SC115. Additional data will not be written to the SC115 until it is disconnected and reconnected to the datalogger. Note that "USB:" is the correct drive name. For complete programming examples, see Section 10.5, *Example Programs (p. 12)*.

```
DataTable (Test,1,-1)
  DataInterval (0,60,Min,0)
  TableFile ("USB:"+Status.SerialNumber+"_Filename",8,-1,0,0,Min,0,0)
  Sample (1,PanelTempC,FP2)
  Minimum (1,BattVolt,FP2,0,False)
EndTable
```

5. Overview

The SC115 is a portable 2-GB memory drive (thumb drive) compatible with CRBasic dataloggers that have a CS I/O port. It can be used in tandem with an external memory card. It shuttles data, OS, and program files between Campbell Scientific dataloggers and a PC, or it remains connected to the datalogger to augment data memory.

The SC115 can also be used as a CS I/O to USB communications interface.

The SC115 has a USB 2.0 compatible connector used to attach the SC115 to a PC USB port to allow stored data files to be copies to the PC. It also has a CS I/O 9-pin connector that attaches the SC115 to the CS I/O port of compatible Campbell Scientific dataloggers. The SC115 may be connected directly to a PC or datalogger, or connected through the supplied extension cables.

The 5 Vdc (pin 1) of the CS I/O interface determines whether the SC115 is used as a memory drive or as a CS I/O-to-USB communications interface. If the 5 Vdc pin is not driven high, it acts as a memory drive. If the 5 Vdc pin is driven high by the datalogger, the SC115 will serve as a communications interface.

5.1 Memory Drive Function

As a memory drive, when the SC115 is attached to the CS I/O port of a datalogger, the LED (Light Emitting Diode) first flickers as the SC115 and datalogger negotiate communications. The datalogger determines if the **TableFile()** instruction is set to write to an SC115. The LED flickers as the datalogger copies data files to the SC115. The LED stops flickering when data transfer is complete, and the SC115 can safely be disconnected.

As a memory drive, when the SC115 is attached to a PC, the LED first lights solidly as it communicates with the PC to obtain a unique address. Once negotiations are finished, the LED extinguishes and the SC115 appears as a removable drive in Windows® Explorer. Data files on the SC115 can be copied or moved to other drives. Files, such as CRBasic program files or operating system files, can be copied or moved from PC drives to the SC115.

5.2 Communications Interface Function

As a communication device, the SC115 provides pass-through communications.

6. Specifications

Features

- Portable 2-GB memory drive
- Shuttles data, OS, and program files or remains connected to datalogger to augment memory
- Can be used as a CS I/O to USB communications interface
- Mean time between failure (MTBF): >1,000,000 hrs
- Sealed, over-moulded case. No serviceable parts.

Compatibility

Dataloggers: CR800 / 850, CR6
CR1000, CR3000
CR5000, CR9000X

NOTE A datalogger can have only one SC115 connected.

Temperature

Operating: –25 to 50 °C

Power requirement

Source: 12 Vdc from datalogger via SC12 cable, or 5 Vdc through PC USB port

Load

Inactive: 200 µA
Active: 35 mA

Memory

Capacity: 2 GB
Media: Flash

Transfer Speed over CS I/O: Approximately 4 to 8 kBps[1]

[1] This transfer speed is suitable for most environmental applications. However, high frequency applications (for example, sub-minute **TableFile()** output intervals) are best served by a CFM100 or NL116 with a compact flash card due to faster transfer speeds across the peripheral bus compared to the CS I/O bus.

Weight:	0.64 kg (0.14 lbs)
Dimensions:	105 x 43 x 21 mm (4.2 x 1.7 x 0.8 in)
Compliance information:	View the EU Declaration of Conformity at: *www.campbellsci.com/sc115*

7. Setup

The datalogger requires inclusion of specific code in its CRBasic program to store data to the SC115. In rare instances, the SC115 may require a change to its default configuration through Campbell Scientific's *DevConfig* software v. 1.5 or higher.

7.1 Datalogger CRBasic Programming

The datalogger CRBasic program needs to include a **TableFile()** instruction to store datalogger data on the SC115. The **TableFile()** instruction is entered in the declaration of the data table after the **DataInterval()** instruction. The parameters entered for the **TableFile()** instruction depend on the data storage mode used (refer to Section 8, *Data Storage Modes* *(p. 4)*). Section 10, *Datalogger Programming* *(p. 6)*, discusses, in detail, datalogger programming, and provides several datalogger programming examples.

7.2 SC115 Settings (Optional)

The default SDC addressing in the SC115 is adequate for most applications. For exceptional circumstances, the CS I/O SDC address can be altered using *DevConfig* v. 1.5 or higher (see Section 14.1, *SDC Addressing* *(p. 19)*).

NOTE Before the SC115 can be configured through *DevConfig*, the device driver must be installed. *DevConfig* is used to install the device driver. Under **Device Type**, select **Peripheral** | **SC115**. Click the **install the device driver for the SC115** link and follow the prompts.

8. Data Storage Modes

The datalogger CRBasic program determines the SC115 data storage mode through the **TableFile()** instruction. The **TableFile()** instruction in the datalogger CRBasic program determines what data the SC115 will receive from the datalogger. Data collection modes and residential modes are available.

8.1 Data Collection Modes

With these modes, data collection is automatically initiated by connecting the SC115 to the datalogger. The SC115 only collects data already contained in datalogger memory at the time of connection. Additional data will not be collected until the SC115 is disconnected and then reconnected to the datalogger.

Both standard and enhanced data-collection modes are available. With the standard mode, the SC115 collects the newest data (data written to the datalogger memory since the last connection). With the enhanced mode, the

SC115 collects all of the data stored in the datalogger's memory every time the SC115 is reconnected.

NOTE To use the enhanced data collection mode, the datalogger must have a newer operating system (OS 22 or higher).

8.1.1 Programming for the Data-Collection Modes

The standard mode is enabled in the **TableFile()** instruction by entering *0* for both the *NumRecs* (*Num*ber of *Rec*ord*s*) parameter and the *Interval* parameter. The enhanced mode is enabled by entering *0* for the *NumRecs* parameter and entering –1 for the *Interval* parameter.

Below are **TableFile()** instructions that enable data collection modes. The first **TableFile()** instruction enables the standard data collection mode. The second **TableFile()** instruction enables the enhanced data collection mode. See Section 10.5.1, *Data Collection Mode Examples* *(p. 12)*, for more detailed program code.

```
TableFile("USB:"+Status.SerialNumber+"_FileName",8,-1,0,0,Hr,0,0)

TableFile("USB:"+Status.SerialNumber+"_FileName",8,-1,0,-1,Hr,0,0)
```

8.1.2 Multiple Dataloggers

The data-collection modes allow one SC115 to be used with multiple dataloggers. When using multiple dataloggers, specify a unique file name for each instance of the **TableFile()** instruction (see Section 10.1.1, *FileName* *(p. 8)*). This avoids overwriting data files from other dataloggers.

8.2 Resident Mode

NOTE Resident modes increase the risk of data corruption. Review Section 12.1, *Preventing Data Corruption* *(p. 16)*, before employing a resident mode.

In the resident modes, the SC115 remains attached to a single datalogger allowing it to be used as resident external memory. The datalogger can be programmed to bale data to the SC115 at regular intervals or at uniform bale sizes.

When data retrieval is required, 1) a second SC115 is swapped with the first, or 2) the SC115 is removed, milked of data (perhaps by copying or moving data to a laptop computer), and then reconnected to the datalogger.

NOTE To avoid losing data when the SC115 is disconnected from the datalogger, the datalogger must have sufficient storage memory allocated in **DataTable()** / **EndTable** declarations in the form of internal or external card memory to cover the period the SC115 is not present.

8.2.1 Programming for the Resident Modes

A resident mode that collects data at regular intervals is enabled in the **TableFile()** instruction by entering zero or a positive value for the

NumRecs/TimeIntoInterval parameter and entering a non-zero, positive value for the *Interval* parameter. To collect data at uniform bale sizes, enter a non-zero, positive value for the *NumRecs* parameter and enter zero for the *Interval* parameter.

Below are **TableFile()** instruction examples that enable resident modes. The first **TableFile()** instruction sets the datalogger to bale data to the SC115 at midnight each day. The second **TableFile()** instruction sets the datalogger to send bales of 24 records to the SC115. See Section 10.5.2, *Resident Interval Mode Example* *(p. 13)*, for more detailed program code.

```
TableFile("USB:FileName",8,-1,0,24,Hr,0,0) 'Interval trigger

TableFile("USB:FileName",8,-1,24,0,Hr,0,0) 'Number of records
trigger
```

9. Retrieving Data from SC115

To collect data from the SC115, plug it into a PC USB port. The SC115 becomes a memory drive on the PC, and data files can be copied, moved, or deleted with the common file handling functions of Windows®. Multiple SC115s can be connected to a PC simultaneously and data passed between them. When used as a communications interface, however, only one SC115 should be connected to a PC.

Before disconnecting the SC115 from the PC, run the **Safely Remove Hardware** utility in Windows®.

10. Datalogger Programming

When used as a memory drive, the SC115 requires a **TableFile()** instruction in the datalogger CRBasic program. **TableFile()** creates a file from data table records, and writes the file to the SC115. The **TableFile()** instruction must be within the **DataTable()** / **EndTable** declaration.

10.1 DataTable() and TableFile() Instructions

DataTable() and **TableFile()** instructions work together. The data table size declared in **DataTable()** determines the size of the **TableFile()** buffer. For instance, a programmer may attempt to conserve datalogger memory by setting **DataTable()** *Size* to *0* with the thought that doing so will direct data to only be written to the SC115 when it is attached for milking. However, such a programming tactic will result in failure since, in setting the **DataTable()** size to zero, the programmer also sets the **TableFile()** data buffer to zero.

At the opposing extreme, if the same memory-conscious programmer sets **DataTable()** *Size* to auto-allocate (*–1*), there is a high probability that the datalogger will allocate far more memory for **TableFile()** buffering than is needed, which will consume memory that may be better used elsewhere.

CAUTION Memory allocated by the **DataTable**() instruction acts as the buffer for **TableFile()** data, so the **DataTable()** *Size* parameter must be declared large enough to buffer the **TableFile()** output between collection visits with the SC115.

TableFile() syntax is reviewed below with comments specific to SC115 applications. For more detail, consult CRBasic Editor Help. **TableFile()** is included in the **DataTable()** / **EndTable** declaration as shown in the programming examples in Section 10.5, *Example Programs* *(p. 12)*.

```
TableFile(a,b,c,d,e,f,g,h)
```

- *a* = "*Drive:FileName*." Drive always = USB: for SC115. Set a filename unique to each instance of TableFile() instruction. See Section 10.1.1, *FileName* *(p. 8)*, for required syntax.
- *b* = *Format*. Most common format is code *8* (TOA5). See CRBasic Help for details.
- *c* = *MaxFiles*. *–1* invokes auto allocation.
- *d* = *NumRec / Time Into Interval*. Data collection modes: enter *0*. Resident modes: enter number of records (integer > 0) or enter time into interval (integer ≥ 0). See following TABLE 10-1 and Section 10.1.4, *NumRecs/TimeIntoInterval Parameter* *(p. 10)*, for explanation.
- *e* = *TFInterval*. Data collection modes: enter *0* to collect newest data or enter *–1* to collect all of the datalogger's data. Resident mode: enter *0* if parameter *d* is set to number of records, otherwise enter time interval (integer > 0). See following TABLE 10-1 and Section 10.1.5, *Interval Parameter* *(p. 10)*, for detail.
- *f* = *TFUnits*. Enter units for time interval. Ignored at compilation if parameter *e* is 0 or –1.
- *g* = *OutStat*. Optional. Set to 0 or see CRBasic Help.
- *h* = *LastFileName*. Optional. Set to 0 or see CRBasic Help.

TABLE 10-1. Function of *NumRec*, *TFInterval*, and *TFUnits* Parameters

***NumRec/ Time Into Interval* (d) Entry**	***TFInterval* (e) Entry**	***TFUnits* (f) Entry**	**Function**
0	0	N/A (compiler ignores)	When the SC115 is connected to the datalogger, the datalogger automatically sends to the SC115 only the data collected after the SC115 was last connected. The datalogger will not send additional data if the SC115 remains connected to the datalogger.
0	–1	N/A (compiler ignores)	When the SC115 is connected to the datalogger, the datalogger automatically sends to the SC115 all of the data contained in the datalogger's memory. The datalogger will not send additional data if the SC115 remains connected to the datalogger.

TABLE 10-1. Function of *NumRec*, *TFInterval*, and *TFUnits* Parameters

***NumRec/ Time Into Interval* (d) Entry**	***TFInterval* (e) Entry**	***TFUnits* (f) Entry**	**Function**
number of data records (value > 0)	0	N/A (compiler ignores)	The datalogger sends the specified number of data records to the SC115 when the data records become available. For example, if the *NumRec* parameter is set to *20* and *TFInterval* is set to *0*, the datalogger will write twenty records to the SC115 when twenty records become available.
time into interval (value ≥ 0)	interval (value > 0)	*mSec*, *uSec*, *Sec*, *Min*, *Hr*, or *Day*	The datalogger sends data to the SC115 at the specified interval. For example, when the *NumRec/Time Into Interval* parameter is set to *0*, the *TFInterval* parameter is set to *60* and the *TFUnit* parameter is set to *Min*, the datalogger will write data to the SC115 every 60 minutes.

The following sections explain each **TableFile()** parameter in detail.

10.1.1 FileName

The *FileName* parameter must be a string declared as Const, such as

```
Const FileName = "USB:FileName",
```

or as an expression that evaluates to a constant, such as

```
"USB:"+Variable+"FileName"
```

Quotation marks are required. The created file will have a suffix of **X.dat**, where **X** is a number that increments each time a new file is written.

When using a single SC115 to collect data from several dataloggers, the **TableFile()** *FileName* parameter must be unique for each data table being collected. Otherwise, data may be overwritten on the SC115. Using a **TableFile()** instruction with parameters as shown below creates a naming scheme wherein the station name or serial number are part of the file name. This allows the source of data to be easily identified during post-processing.

```
TableFile("USB:"+Status.SerialNumber+"Filename",8,-1,0,0,Min,0,0)
```

When a program is compiled with "**USB:**" in the file path, it searches the attached SC115 for any file names in the series. If it finds any, it uses the

highest numbered file name, increments it by one, uses the result as the name for a new file, and writes the new data to the new file.

When multiple SC115s are used to retrieve data from multiple dataloggers, it is often desirable to set the interval parameter to **–1**. If the interval parameter is set to **0** instead of **–1**, the complete set of records from a single datalogger will likely be spread across all SC115s used to retrieve data.

CAUTION If data are retrieved from a datalogger using an SC115 that already contains files whose names match those created by the current datalogger's **TableFile()**, the old files are likely to be overwritten. Using unique filenames for each datalogger, such as including the station name or serial number, is the best practice. See example in Section 10.5.1.1, *Standard Data Collection Mode* *(p. 12)*. A rarely used alternative is to recompile the datalogger program with the second SC115 connected to retrieve its directory information. This is done by powering down the datalogger, connecting the SC115, and then powering up the datalogger.

10.1.2 Options

The *Options* parameter specifies the type of file to be saved and whether to include the header information, timestamp, and/or record number. Options *0*, *8*, *16*, and *20* correspond to Campbell Scientific formats for TOB1, TOA5, CSIXML, and CSIJSON, respectively. Choosing an option not among these four may make the file incompatible with Campbell Scientific software that reads or writes data files. For example, Option *7* results in a TOB1 file that cannot be read by *CardConvert* software or *View Pro* software. Refer to the CRBasic Editor Help system for a complete listing of option codes.

10.1.3 MaxFiles

The *MaxFiles* parameter specifies the maximum number of files to retain on an SC115. TABLE 10-2 reviews the options.

TABLE 10-2. Synopsis of MaxFiles Parameter Options

***MaxFiles* Entry**	**Function**
X	Create Maximum of X files, ring memory (overwrite oldest file with newest file when full).
0	The filename will remain fixed with no number appended. The old file if it exists will be overwritten at each output.
–1	No limit to number of files, ring memory (overwrite oldest file with newest file when full)
–2	No limit to number of files, fill and stop memory (datalogger stops writing to SC115 when SC115 is full)

When *MaxFiles* is *X*, SC115 memory may fill before X number of files is reached. If this occurs, the datalogger internally reduces X to the current number of files and overwrites the oldest file with new data.

Refer to the CRBasic Help system for detailed information on this parameter.

10.1.4 NumRecs/TimeIntoInterval Parameter

Data Collection Modes

For Data Collection Modes, the *NumRecs/TimeIntoInterval* parameter is set to zero, and the *Interval* parameter is set to either *0* or *–1* (see Section 10.1.5, *Interval Parameter* *(p. 10)*). With these modes, the datalogger begins writing to the SC115 as soon as it is connected to the datalogger. All new records are written to a single file. See previous TABLE 10-1 and Section 10.5.1, *Data Collection Mode Examples* *(p. 12)*.

Resident Modes

NOTE Resident Mode increases the risk of data corruption. Review Section 12.1, *Preventing Data Corruption* *(p. 16)*, before employing a resident mode.

For resident modes, the *NumRecs/TimeIntoInterval* parameter determines when files are written to the SC115. The function of *NumRecs/TimeIntoInterval* is linked to the *Interval* parameter. If *Interval* is set to *0*, enter the number of records to be included in each new file sent to the SC115. The program will create a new file each time *NumRecs* is reached. If *Interval* is set to non-zero, *NumRecs/TimeIntoInterval* becomes the time into *Interval* that the program writes the next file. Previous TABLE 10-1 and CRBasic Editor Help provide further explanation of the interplay between *TimeIntoInterval* and *Interval*.

10.1.5 Interval Parameter

Data Collection Modes

For data collection modes, the *Interval* parameter determines what data are written to the SC115. Set this parameter to *0* to collect only data written to datalogger memory since it was last connected. Set this parameter to *–1* to collect all of the data stored in the datalogger's memory. See previous TABLE 10-1.

Resident Mode

NOTE Resident Mode increases the risk of data corruption. Review Section 12.1, *Preventing Data Corruption* *(p. 16)*, before employing a resident mode.

For Resident Mode, the *Interval* parameter is used to determine how frequently data are written to the SC115. By setting this parameter to a non-zero number, the datalogger writes a new file to the SC115 at the interval based on this number and the *Units* parameter.

If *Interval* is set to *0*, but *NumRecs/TimeIntoInterval* is set to a non-zero value, the datalogger writes data to the SC115 whenever the number of new records matches the *NumRecs/TimeIntoInterval* value.

10.1.6 Units Parameter

The *Units* parameter specifies the units used by the Interval parameter. The compiler ignores this parameter if the interval parameter is set to *0* or *–1*.

10.1.7 OutStat Parameter

OutStat is inactive when set to *0*.

When set to a variable declared as **Public** or **Dim**, *OutStat* (output status) indicates whether or not a new data file was stored the last time **TableFile()** executed. If a new file is written, *–1* is stored in the variable the next time **CallTable()** is executed. If a new file is not written, *0* is stored the next time **CallTable()** is executed.

TableFile() has been designed to permit the datalogger to continue program execution while data are baled, by a separate process, into the table file. The up-side is that **TableFile()** does not slow program execution. The down-side is that notification that the **TableFile()** task is complete is delayed. This situation is manifest as follows.

If writing a new file takes longer than the **CallTable()** interval (the interval of the **Scan() / NextScan** construct wherein the **CallTable()** instruction resides), then the *OutStat* variable will not be updated. In other words, if the table file is large, and the scan rate is short, *OutStat* may not be updated.

10.1.8 LastFileName Parameter

Set to 0 (no last file name) if desired. The *LastFileName* parameter is a variable that contains the name of the last file written. *LastFileName* is updated the next time **CallTable()** instruction is executed.

TableFile() has been designed to permit the datalogger to continue program execution while data are baled by a separate process into the table file. The up-side is that **TableFile()** does not slow program execution. The down-side is that notification is delayed that the **TableFile()** task is complete. This situation is manifest as follows.

If writing a new file takes longer than the **CallTable()** interval (the interval of the **Scan() / NextScan** construct wherein the **CallTable()** instruction resides), then the *LastFileName* variable will not be updated. In other words, if the table file is large, and the scan rate is short, *LastFileName* may not be updated.

10.2 CardFlush Instruction

If a program **Scan()** / **NextScan** *Count* parameter is non-zero, the latest data may fail to write to the SC115 unless the **CardFlush** instruction is used. For example, with a scan count of 600, a scan interval of 100 ms, and writing a file with **TableFile()** after every 10 seconds (every 100 records), only 400–575 records may be written to the SC115 due to the program ending at the 600th scan prior to transferring all the records to the SC115. To transfer the remaining data to the SC115, place the **CardFlush** instruction between the **NextScan** and **EndProg** instructions. See Section 10.5.4, *CardFlush Example* ***(p. 15)***.

10.3 CardOut Instruction

A datalogger that supports external memory cards can be programmed to simultaneously store data to internal memory, an external memory card, and an SC115. The datalogger program needs to include a **CardOut()** instruction in the **DataTable()** declaration. By including **CardOut()**, the datalogger can write data to the SC115 that is present only on the external memory card. See Section 12.4.1, *External Memory Card and Collection* Speed *(p. 17)*, for more information on SC115 and external memory card interactions. See Section 10.5.5, *CardOut Example (p. 15)*.

10.4 Powerup.ini

The SC115 supports the use of a powerup.ini file, which allows the host datalogger to be loaded with a specific program or operating system at datalogger power-up. Consult the powerup.ini section in the datalogger manual for more information.

10.5 Example Programs

10.5.1 Data Collection Mode Examples

10.5.1.1 Standard Data Collection Mode

```
'Plug – Retrieve Data Since Last Plug - Pull

'In this example, the SC115 is connected to the datalogger to retrieve the data
'written to the datalogger memory after the last connection. It only retrieves
'data that were already in datalogger memory when it was connected. To retrieve
'subsequent data, the SC115 must be disconnected then re-connected to the
'datalogger. Note that the file name incorporates the system Status.SerialNumber
'variable, which inclusion helps avoid file overwrites.

Public PanelTempC, BattVolt

DataTable (Test,1,-1)
  DataInterval (0,60,Min,0)
  TableFile ("USB:"+Status.SerialNumber+"_Filename",8,-1,0,0,Min,0,0)
  Sample (1,PanelTempC,FP2)
  Minimum (1,BattVolt,FP2,0,False)
EndTable

BeginProg
  Scan (10,Sec,3,0)
    PanelTemp (PanelTempC,250)
    Battery (BattVolt)
    CallTable Test
  NextScan
EndProg
```

10.5.1.2 Enhanced Data Collection Mode Example

```
'Plug - Retrieve All Data - Pull

'In this example, the SC115 is connected to the datalogger to retrieve all of
'the data stored in datalogger memory. It only retrieves data that were already
'stored in the datalogger memory when it was connected. To retrieve subsequent
'data, the SC115 must be disconnected and then reconnected to the datalogger.

Public PanelTempC, BattVolt

DataTable (Test,1,-1)
  DataInterval (0,60,Min,0)
  TableFile ("USB:"+Status.SerialNumber+"_Filename",8,-1,0,-1,Min,0,0)
  Sample (1,PanelTempC,FP2)
  Minimum (1,BattVolt,FP2,0,False)
EndTable

BeginProg
  Scan (10,Sec,3,0)
    PanelTemp (PanelTempC,250)
    Battery (BattVolt)
    CallTable Test
  NextScan
EndProg
```

10.5.2 Resident Interval Mode Example

```
'Plug In and Leave - data baled from buffer to SC115 every 60 minutes

'In this example, the SC115 remains at the datalogger as resident memory.
'This program avoids data corruption by setting TableFile() Interval to 60
'In this case, data are written only once an hour to the SC115.

Public PanelTempC, BattVolt

DataTable (Test,1,-1)
  DataInterval (0,1,Min,0)
  TableFile ("USB:Test",8,-1,0,60,Min,0,0)    'SC115 resident mode with interval
  Sample (1,PanelTempC,FP2)
  Minimum (1,BattVolt,FP2,0,False)
EndTable

BeginProg
  Scan (10,Sec,3,0)
    PanelTemp (PanelTempC,250)
    Battery (BattVolt)
    CallTable Test
  NextScan
EndProg
```

10.5.3 Resident Record Number Mode Examples

```
'Plug In and Leave – data baled from buffer to SC115 every 60 records

'In this example, the SC115 remains at the datalogger as resident memory.
'This program avoids data corruption by setting TableFile() NumRecs to 60
'In this case, data are written only once an hour to the SC115.

Public PanelTempC, BattVolt

DataTable (Test,1,-1)
  DataInterval (0,1,Min,0)
  TableFile ("USB:Test",8,-1,60,0,Min,0,0)    'SC115 resident mode with NumRec
  Sample (1,PanelTempC,FP2)
  Minimum (1,BattVolt,FP2,0,False)
EndTable

BeginProg
  Scan (10,Sec,3,0)
    PanelTemp (PanelTempC,250)
    Battery (BattVolt)
    CallTable Test
  NextScan
EndProg
```

```
'Plug In and Leave – data baled when buffer is full,
'i.e. DataTable() Size = TableFile() NumRecs (memory efficient!)

'In this example, the SC115 remains at the datalogger as resident memory.
'This program avoids data corruption by setting DataTable() Size to equal
'TableFile() NumRecs. In this case, data are written only once every two hours
'to the SC115.

Public PanelTempC, BattVolt

DataTable (Test,1,120)
  DataInterval (0,1,Min,0)
  TableFile ("USB:Test",8,-1,120,0,Min,0,0)    'SC115 resident mode with NumRec
  Sample (1,PanelTempC,FP2)
  Minimum (1,BattVolt,FP2,0,False)
EndTable

BeginProg
  Scan (10,Sec,3,0)
    PanelTemp (PanelTempC,250)
    Battery (BattVolt)
    CallTable Test
  NextScan
EndProg
```

10.5.4 CardFlush Example

```
'In this example, the SC115 remains at the datalogger as resident memory.
'The Scan count, usually left at 0, is set to 600 in this application.
'To ensure complete collection of data, CardFlush instruction is included.

Public PanelTempC, BattVolt

DataTable (Test,1,-1)
  DataInterval (0,10,Sec,0)
  TableFile ("USB:Test",8,-1,100,0,Hr,0,0)    'SC115 resident mode with NumRec
  Sample (1,PanelTempC,FP2)
  Minimum (1,BattVolt,FP2,0,False)
EndTable

BeginProg
   'Scan stops at count 600 - Not enough time to transfer all data!
  Scan (100,mSec,3,600)
    PanelTemp (PanelTempC,250)
    Battery (BattVolt)
    CallTable Test
  NextScan
  CardFlush    'Included to ensure complete data transfer to SC115
EndProg
```

10.5.5 CardOut Example

```
'In this example, the SC115 and an external memory card are used as external memory.

Public PanelTempC, BattVolt

DataTable (Test,1,-1)
  DataInterval (0,1,Min,0)
  CardOut (0,-1)
  TableFile ("USB:Test",8,-1,0,60,Min,0,0)    'SC115 resident mode with interval
  Sample (1,PanelTempC,FP2)
  Minimum (1,BattVolt,FP2,0,False)
EndTable

BeginProg
  Scan (10,Sec,3,0)
    PanelTemp (PanelTempC,250)
    Battery (BattVolt)
    CallTable Test
  NextScan
EndProg
```

11. Serial Communications Interface Function

NOTE

Before the SC115 can be used as a communications interface, the device driver must be installed. *DevConfig* is used to install the device driver. Under **Device Type**, select **Peripheral | SC115**. Click the **install the device driver for the SC115** link and follow the prompts.

The SC115, when used as a serial interface, uses the Campbell Scientific SDC protocol to facilitate communications between the datalogger and a PC. The SDC address can be set to **7**, **8**, **10**, or **11** (see Section 7.2, *SC115 Settings (Optional)* (p. 4)).

Consider the following points when using the SC115 as a communications interface:

- Use the provided extension cables to establish the physical connections between datalogger, SC115, and PC BEFORE opening *LoggerNet*.
- When selecting a port in the *LoggerNet* Network Map for USB to CS I/O communication, the COM port *SC115 (COM#)* must be selected. *SC115 MSD (COM#)* may also appear as a choice, but must not be selected as it does not support the communications interface mode.
- Always click **Disconnect** within the datalogger support software Connect window prior to breaking the physical connection between the PC, SC115 and datalogger. If this rule is not observed, and communication problems develop, follow the procedure in Section 13.1, *Breaking the Physical Comms Link* *(p. 19)*.
- Do not connect two SC115s to a PC at the same time if both are attached to powered dataloggers. This can cause confusion of virtual com ports on the PC (as memory devices, two SC115s may be attached to a PC at the same time).

12. Troubleshooting Memory Drive

12.1 Preventing Data Corruption

To avoid corruption of SC115 memory, never remove the SC115 from the datalogger during data transfer. The flashing LED indicates when data are being written to the SC115. Should the SC115 be removed while the LED is flashing, the most likely result is that the current data file will be corrupted. In addition, a FAT sector or the directory link sector may become corrupted. This condition requires that the SC115 memory be reformatted, which will result in the loss of all data.

When either the *NumRecs/TimeIntoInterval* or *Interval* parameters are set to a non-zero positive value (resident modes), there is a risk that the datalogger will begin writing data to the SC115 at the exact moment it is being removed from the datalogger, resulting in data corruption. To prevent this, set these parameters to values that allow the time between writing data to be easily discerned (such as an interval of 10 seconds) by watching the LED. The user then times the removal of the SC115 to occur when the datalogger is not writing data. See Section 10.5.2, *Resident Interval Mode Example* *(p. 13)*.

Another method is to set the *NumRecs* parameter in **TableFile()** to match the *Size* parameter in the associated **DataTable()** instruction. Depending on the rate at which records are written, this can cause data to be written to the SC115 at long intervals, greatly reducing the chance of removing the SC115 from the datalogger while transferring data. See Section 10.5.3, *Resident Record Number Mode Examples* *(p. 14)* (second example).

If SC115 data becomes corrupted, first attempt to retrieve all files from the SC115. Reformat the SC115 per Section 14.2, *Formatting Memory* *(p. 19)*.

12.2 Skipped Scans

To avoid skipped scans, ensure that the scan interval in the datalogger program is long enough to include writing to the SC115. For example, if the program has a single **TableFile()** instruction, add at least 100 ms to the scan rate to accommodate each added **TableFile()** instruction.

Compile datalogger programs in pipeline mode when possible. Datalogger programs compiled in sequential mode require a longer scan interval than programs compiled in pipeline mode to avoid skipped scans. In pipeline mode, the **Scan() / NextScan** instruction *BufferOption* parameter can be increased to prevent skipped scans. In sequential mode, the **TableFile()** instruction must finish before continuing to the next instruction, resulting in skipped scans unless the scan interval is long enough to handle all communication, measurement processing, and **TableFile()** tasks. Sequential mode ignores any scan buffers that may be assigned. Programs may run as much as three times faster in pipeline mode than in sequential mode.

12.3 SC115 with Large Data Compliment

An SC115 with a large compliment of data (either a large number of files, or a few very large files) may respond slower than an SC115 with less data. Compiling datalogger programs from a full SC115 may take longer than normal.

Opening the SC115 directory in support software **File Control** also takes longer if there are a large number of files within the SC115 directory. **File Control** can access approximately twenty files per second when opening the directory.

12.4 Data Collection Speed

Many factors affect collection speed in plug and pull data collection mode. As a rough estimate, collection speed is typically 3 to 4 kB of SC115 file size per second (TFOption = 8, SDC7 set to –115200 [autobaud], no card storage module present). Following are descriptions of two factors that contribute significantly to the slowing of collection speed.

12.4.1 External Memory Card and Collection Speed

Following is a discussion of SC115 performance expectations when it is used in tandem (plug and pull data collection mode) with a resident external memory card on a datalogger.

Background – When an SC115 is configured for data collection mode (plug and pull), and it is plugged into a datalogger with a resident external memory card, and the running program is using the **CardOut()** instruction, all data present since the last collection will be automatically transferred from the data table to the SC115 file.

How it works – When **CardOut** is used in a **DataTable()** / **EndTable** declaration, the file created on the external memory card becomes an extension of the table memory. The internal memory is used as a buffer to transfer data to the external memory card. When there are enough data to warrant a write to the external memory card, the data are flushed to the card. This results in duplicate data, up to the internal table size, that will exist on the internal memory and the

external memory card files. Any time the table data are retrieved via *LoggerNet* data collection, or by using the **TableFile()** instruction, the internal SRAM is searched first, then the card. So, when operating the SC115 in plug and pull mode, SRAM data are transferred first, then the external memory card is searched.

Effect – The rate of transfer is slowed, not because of the search of the card, but because of the serial communications and (even slower) if an ASCII output file is selected. This is compounded by large data files that can occur when the **CardOut** instruction is used. Searching the external memory card adds some overhead, but the vast majority of time is spent converting to ASCII and transferring across a slow serial link.

Take home – external memory card to SC115 transfer of data is about 30 to 50% slower than datalogger CPU to SC115 transfer of data. For example, in testing with a CR1000 datalogger with a 2 GB card attached, transfer of a 3.9 MB file (so no data were stored to the card) took aproximately16 minutes (about 250 kB per minute). However, the rate of transfer slowed to about 190 kB per minute when datalogger CPU memory was allowed to ring (fill and overwrite oldest data) many times, resulting in a large amount of data on the card that was no longer present on the CPU. In this example, it took 116 minutes to transfer a 22 MB data file from the external memory card to the SC115. In the case of very large data files stored to the external memory card, it makes more sense to swap external memory cards (see the *CFM100 CompactFlash Module* user manual, available at *www.campbellsci.com*, for details), or use a resident SC115, rather than milking the data with the SC115 in plug and pull mode.

12.4.2 Data Type Collection Speed

Data type declared in the **TableFile()** *TFOption* parameter can affect SC115 collection speed. Data are buffered in the datalogger as binary. While data are collected, the datalogger converts the binary to the declared data type. *TOB1* options are binary and require very little processor overhead to convert from the binary buffer. *TOA5*, *CSIXML*, and *JSON* are ASCII options and are far slower since they consume significant processor overhead to convert to ASCII from the binary buffer. *CSIXML* is especially slow. The effects on collection time will be particularly noticeable if the datalogger is running a long or complex program. In short, if large files from numerous dataloggers need collection, using the TOB1 format may save considerable time. Consult the *LoggerNet* Instruction Manual for options available to convert TOB1 data files on the PC to easier-to-read formats.

12.5 Slow PC Boot-up

Leaving an SC115 attached to a PC while it is booting up may cause a delay of several minutes to the boot-up process on some computers. The PC will continue to boot normally if the SC115 is removed. The SC115 can then be inserted and used as normal.

12.6 Write Failure

A write failure (after retries) is interpreted to mean the memory is full. When a write failure occurs, *MaxFiles* parameter in the **TableFile()** instruction is adjusted to its current number. In addition, the datalogger enters ring mode,

overwriting the oldest file on the SC115 with the new data (as if the *MaxFile* parameter was set to *–1*).

The timeout period between a failed write command and a retry is 8 seconds.

13. Troubleshooting Communications Interface

13.1 Breaking the Physical Comms Link

To avoid a communications error, always click on **Disconnect** prior to physically breaking the SC115 interface link.

When the SC115 interface link between the PC and the datalogger is physically broken before **Disconnect** is clicked in the support software, the software requires about one minute to process the broken link. If the link is reconnected before the break is processed, a communications error may occur. If an error occurs, use the following procedure to re-establish communications.

1. Click on the support software **Cancel** button to stop the connection attempt.
2. Disconnect the SC115 from the PC for at least one minute.
3. After one minute, reconnect the SC115 to the PC and datalogger.
4. Close the support software, then reopen it, then click on **Connect**.

14. Technical Reference

14.1 SDC Addressing

SDC addressing suitable for most applications is set at the factory.

The SC115 responds to two SDC addresses. SDC address 0 is used when data files are transferred from the datalogger to the SC115. This setting cannot be changed.

When plugged into a USB port, addresses SDC7, SDC8, SDC10, and SDC11 are available to support Communication Interface Mode on the CS I/O port. This setting defaults to SDC7, but can be changed using *DevConfig*.

NOTE Before the SC115 can be configured through *DevConfig*, the device driver must be installed. *DevConfig* is used to install the device driver. Under **Device Type**, select **Peripheral | SC115**. Click the **install the device driver for the SC115** link and follow the prompts.

14.2 Formatting Memory

The SC115 is formatted at the factory and is ready to use out of the box. Should the SC115 require formatting, such as if the memory becomes corrupted, it can be formatted through datalogger **File Control** (*PC400*, *LoggerNet*, *RTDAQ* software), datalogger keypad (**File** | **Format** | **USB**), or

Windows® Explorer (**right click** | **Format**). When given an option between formatting as FAT or FAT32, always choose FAT32.

NOTE Formatting will erase all data files on the SC115.

14.3 Operating System

The SC115 is configured at the factory with an operating system. Unless notified by Campbell Scientific, the operating system does not need to be reloaded. Downloading a new OS to the SC115 does not affect files in memory.

15. Glossary

DevConfig – Device Configuration Utility. Campbell Scientific PC software used for configuring dataloggers and peripherals.

LED – Light Emitting Diode

LoggerNet – Campbell Scientific top-level datalogger support software.

PC400 – Campbell Scientific mid-level datalogger support software.

RTDAQ – Campbell Scientific datalogger support software for industrial applications.

SDC – Synchronous Devices Communication. A proprietary communications protocol between Campbell Scientific dataloggers and some peripherals.

16. Attributions

CompactFlash® is a registered trademark of SanDisk Corporation.

Windows® is a registered trademark of Microsoft Corporation.

Campbell Scientific Companies

Campbell Scientific, Inc.
815 West 1800 North
Logan, Utah 84321
UNITED STATES
www.campbellsci.com • info@campbellsci.com

Campbell Scientific Canada Corp.
14532 – 131 Avenue NW
Edmonton AB T5L 4X4
CANADA
www.campbellsci.ca • dataloggers@campbellsci.ca

Campbell Scientific Africa Pty. Ltd.
PO Box 2450
Somerset West 7129
SOUTH AFRICA
www.campbellsci.co.za • cleroux@csafrica.co.za

Campbell Scientific Centro Caribe S.A.
300 N Cementerio, Edificio Breller
Santo Domingo, Heredia 40305
COSTA RICA
www.campbellsci.cc • info@campbellsci.cc

Campbell Scientific Southeast Asia Co., Ltd.
877/22 Nirvana@Work, Rama 9 Road
Suan Luang Subdistrict, Suan Luang District
Bangkok 10250
THAILAND
www.campbellsci.asia • info@campbellsci.asia

Campbell Scientific Ltd.
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
UNITED KINGDOM
www.campbellsci.co.uk • sales@campbellsci.co.uk

Campbell Scientific Australia Pty. Ltd.
PO Box 8108
Garbutt Post Shop QLD 4814
AUSTRALIA
www.campbellsci.com.au • info@campbellsci.com.au

Campbell Scientific Ltd.
3 Avenue de la Division Leclerc
92160 ANTONY
FRANCE
www.campbellsci.fr • info@campbellsci.fr

Campbell Scientific (Beijing) Co., Ltd.
8B16, Floor 8 Tower B, Hanwei Plaza
7 Guanghua Road
Chaoyang, Beijing 100004
P.R. CHINA
www.campbellsci.com • info@campbellsci.com.cn

Campbell Scientific Ltda.
Fahrenheitstraße 13
28359 Bremen
GERMANY
www.campbellsci.de • info@campbellsci.de

Campbell Scientific do Brasil Ltda.
Rua Apinagés, nbr. 2018 ─ Perdizes
CEP: 01258-00 ─ São Paulo ─ SP
BRASIL
www.campbellsci.com.br • vendas@campbellsci.com.br

Campbell Scientific Spain, S. L.
Avda. Pompeu Fabra 7-9, local 1
08024 Barcelona
SPAIN
www.campbellsci.es • info@campbellsci.es

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