INSTRUCTION MANUAL

# *PC400 User's Manual*

Version 4.4

Revision: 4/16

Copyright © 1999-2016
Campbell Scientific, Inc.

# LIMITED WARRANTY

CAMPBELL SCIENTIFIC (CANADA) CORPORATION warrants that the installation media on which the accompanying computer software is recorded and the documentation provided with it are free from physical defects in materials and workmanship under normal use. CAMPBELL SCIENTIFIC (CANADA) CORPORATION warrants that the computer software itself will perform substantially in accordance with the specifications set forth in the instruction manual published by CAMPBELL SCIENTIFIC (CANADA) CORPORATION. CAMPBELL SCIENTIFIC (CANADA) CORPORATION warrants that the software is compatible with computers running Microsoft Windows 3.1, 95, 98, NT and 2000.

CAMPBELL SCIENTIFIC (CANADA) CORPORATION will either replace or correct any software that does not perform substantially according to the specifications set forth in the instruction manual with a corrected copy of the software or corrective code. In the case of significant error in the installation media or documentation, CAMPBELL SCIENTIFIC (CANADA) CORPORATION will correct errors without charge by providing new media, addenda or substitute pages.

If CAMPBELL SCIENTIFIC (CANADA) CORPORATION is unable to replace defective media or documentation, or if CAMPBELL SCIENTIFIC (CANADA) CORPORATION is unable to provide corrected software or corrected documentation within a reasonable time, CAMPBELL SCIENTIFIC (CANADA) CORPORATION will either replace the software with a functionally similar program or refund the purchase price paid for the software.

The above warranties are made for ninety (90) days from the date of original shipment.

CAMPBELL SCIENTIFIC (CANADA) CORPORATION does not warrant that the software will meet licensee's requirements or that the software or documentation are error free or that the operation of the software will be uninterrupted. The warranty does not cover any diskette or documentation that has been damaged or abused. The software warranty does not cover any software that has been altered or changed in any way by anyone other than CAMPBELL SCIENTIFIC (CANADA) CORPORATION. CAMPBELL SCIENTIFIC (CANADA) CORPORATION is not responsible for problems caused by computer hardware, computer operating systems or the use of CAMPBELL SCIENTIFIC (CANADA) CORPORATION's software with non-CAMPBELL SCIENTIFIC (CANADA) CORPORATION software.

**All warranties of merchantability and fitness for a particular purpose are disclaimed and excluded. CAMPBELL SCIENTIFIC (CANADA) CORPORATION shall not in any case be liable for special, incidental, consequential, indirect, or other similar damages even if CAMPBELL SCIENTIFIC (CANADA) CORPORATION has been advised of the possibility of such damages.**

**CAMPBELL SCIENTIFIC (CANADA) CORPORATION is not responsible for any costs incurred as a result of lost profits or revenue, loss of use of the software, loss of data, cost of re-creating lost data, the cost of any substitute program, claims by any party other than licensee, or for other similar costs.**

**Licensee's sole and exclusive remedy is set forth in this limited warranty. CAMPBELL SCIENTIFIC (CANADA) CORPORATION's aggregate liability arising from or relating to this agreement or the software or documentation (regardless of the form of action; e.g., Contract, tort, computer malpractice, fraud and/or otherwise) is limited to the purchase price paid by the licensee.**

# LICENSE FOR USE

This software is protected by both United States copyright law and international copyright treaty provisions. The installation and use of this software constitutes an agreement to abide by the provisions of this license agreement.

You may copy this software onto a computer to be used and you may make archival copies of the software for the sole purpose of backing-up CAMPBELL SCIENTIFIC (CANADA) CORPORATION software and protecting your investment from loss. This software may not be sold, included or redistributed in any other software, or altered in any way without prior written permission from CAMPBELL SCIENTIFIC (CANADA) CORPORATION. All copyright notices and labelling must be left intact.

This software may be used by any number of people, and may be freely moved from one computer location to another, so long as there is not a possibility of it being used at one location while it's being used at another. The software, under the terms of this license, cannot be used by two different people in two different places at the same time.

CAMPBELL® SCIENTIFIC

Campbell Scientific (Canada) Corp.
14532 131 Avenue NW | Edmonton AB T5L 4X4
780.454.2505 | fax 780.454.2655 | campbellsci.ca

# 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.*


## 1. Introduction ..... 1-1

1.1 PC400 Overview ..... 1-1
1.1.1 Main Screen ..... 1-1
1.1.2 Clock/Program and the EZSetup Wizard ..... 1-1
1.1.3 Monitor Data ..... 1-2
1.1.4 Collect Data ..... 1-2
1.1.5 View ..... 1-2
1.1.6 Split ..... 1-2
1.1.7 CardConvert ..... 1-2
1.1.8 Short Cut ..... 1-3
1.1.9 CRBasic Editor ..... 1-3
1.1.10 Edlog ..... 1-3
1.2 What's New in Version 4? ..... 1-3
1.3 Getting Help for PC400 Applications ..... 1-5
1.4 Windows Conventions ..... 1-5

## 2. System Requirements ..... 2-1

2.1 Hardware and Software ..... 2-1

## 3. Installation, Operation and Backup Procedures ... 3-1

3.1 CD-ROM Installation ..... 3-1
3.2 PC400 Operations and Backup Procedures ..... 3-1
3.2.1 PC400 Directory Structure and File Descriptions ..... 3-1
3.2.1.1 Program Directory ..... 3-1
3.2.1.2 Working Directories ..... 3-2
3.2.2 Backing up the Network Map and Data Files ..... 3-2
3.2.2.1 Performing a Backup ..... 3-3
3.2.2.2 Restoring the Network from a Backup File ..... 3-3
3.2.3 Loss of Computer Power ..... 3-3
3.2.4 Program Crashes ..... 3-4

## 4. The PC400 Main Screen ..... 4-1

4.1 Overview ..... 4-1
4.2 Clock/Program Tab and EZSetup Wizard ..... 4-2
4.2.1 EZSetup Wizard ..... 4-2
4.2.2 Clock/Program Tab ..... 4-4
4.3 Monitor Data Tab ..... 4-4
4.4 Collect Data Tab ..... 4-8
4.5 Pull-down Menus ..... 4-9
4.5.1 View Menu ..... 4-9
4.5.2 Datalogger Menu ..... 4-10
4.5.2.1 Connect/Disconnect ..... 4-10
4.5.2.2 Update Table Definitions ..... 4-10
4.5.2.3 Station Status ..... 4-10

4.5.2.4 File Control for CR5000, CR800, CR1000, CR6 Series, CR300 Series, CR3000, and CR9000 Dataloggers ..... 4-11
4.5.2.5 Terminal Emulator ..... 4-16
4.5.3 Network Menu ..... 4-17
4.5.3.1 Add/Delete/Edit/Rename Datalogger ..... 4-17
4.5.3.2 Backup/Restore Network ..... 4-18
4.5.3.3 Computer's Global PakBus Address ..... 4-18
4.5.4 Tools Menu ..... 4-18
4.5.4.1 Stand-alone Applications ..... 4-18
4.5.4.2 Options ..... 4-19
4.5.4.3 LogTool ..... 4-19
4.5.4.4 PakBus Graph ..... 4-20
4.5.4.4.1 Selecting the PakBus Network to View ..... 4-21
4.5.4.4.2 Dynamic and Static Links ..... 4-21
4.5.4.4.3 Viewing/Changing Settings in a PakBus Datalogger ..... 4-21
4.5.4.4.4 Right-Click Functionality ..... 4-22
4.5.4.4.5 Discovering Probable Routes between Devices. 4-22

**5. Split ..... 5-1**

5.1 Functional Overview ..... 5-1
5.2 Getting Started ..... 5-2
5.3 Split Parameter File Entries ..... 5-8
5.3.1 Input Files ..... 5-8
5.3.1.1 File Info ..... 5-9
5.3.1.2 File Offset/Options ..... 5-10
5.3.1.3 Start Condition ..... 5-13
5.3.1.3.1 Starting Relative to PC Time ..... 5-14
5.3.1.4 Stop Condition ..... 5-16
5.3.1.4.1 "C" Option: Formatting Event Tests Containing Conditional Output Arrays ..... 5-16
5.3.1.4.2 Trigger on Stop Condition (F Option) Output of Time Series ..... 5-18
5.3.1.5 Copy ..... 5-19
5.3.1.6 Select ..... 5-20
5.3.1.7 Ranges ..... 5-20
5.3.1.8 Variables ..... 5-21
5.3.1.9 Numerical Limitations ..... 5-22
5.3.1.10 Mathematical Functions, Details, and Examples ..... 5-22
5.3.1.11 Time Series Functions, Details, and Examples ..... 5-24
5.3.1.12 Special Functions, Details, and Examples ..... 5-29
5.3.1.13 Split Functions Example ..... 5-33
5.3.1.14 Summary of Select Line Syntax Rules ..... 5-36
5.3.1.15 Time Synchronization ..... 5-36
5.3.1.15.1 Time Synchronization and the Copy Condition ..... 5-38
5.3.1.15.2 Using Time Synchronization While Starting Relative to PC Time ..... 5-38
5.3.2 Output Files ..... 5-39
5.3.2.1 Description of Output Option Commands ..... 5-40
5.3.2.2 Report Headings ..... 5-45
5.3.2.3 Column Headings ..... 5-45
5.4 Help Option ..... 5-45
5.5 Editing Commands ..... 5-46

5.6 Running Split from a Command Line ..... 5-46
5.6.1 Splitr Command Line Switches ..... 5-46
5.6.1.1 Closing the Splitr.exe Program After Execution (/R or /Q Switch) ..... 5-46
5.6.1.2 Running Splitr in a Hidden or Minimized State (/H Switch) ..... 5-46
5.6.1.3 Running Multiple Copies of Splitr (/M Switch) ..... 5-47
5.6.2 Using Splitr.exe in Batch Files ..... 5-47
5.6.3 Processing Alternate Files ..... 5-47
5.6.3.1 Input/Output File Command Line Switches for Processing Alternate Files ..... 5-48
5.6.4 Processing Multiple Parameter Files with One Command Line ..... 5-51
5.7 Log Files ..... 5-51

6. View Pro ..... 6-1

6.1 Overview ..... 6-1
6.2 The Toolbar ..... 6-1
6.3 Opening a File ..... 6-3
6.3.1 Opening a Data File ..... 6-3
6.3.2 Opening Other Types of Files ..... 6-4
6.3.3 Opening a File in Hexadecimal Format ..... 6-4
6.4 Viewing a LoggerNet Database Table ..... 6-4
6.4.1 Selecting a Database ..... 6-4
6.4.2 Selecting a Table ..... 6-6
6.5 Importing a CSV File ..... 6-7
6.6 Data View ..... 6-9
6.6.1 Column Size ..... 6-10
6.6.2 Header Information ..... 6-11
6.6.3 Row Shading ..... 6-11
6.6.4 Locking the TimeStamp Column ..... 6-11
6.6.5 File Information ..... 6-11
6.6.6 Background Color ..... 6-11
6.6.7 Font ..... 6-11
6.6.8 Window Arrangement ..... 6-11
6.7 Graphs ..... 6-12
6.7.1 Line Graph ..... 6-13
6.7.1.1 Selecting Data to be Graphed ..... 6-13
6.7.1.2 Graph Width ..... 6-14
6.7.1.3 Scrolling ..... 6-14
6.7.1.4 Graph Cursor ..... 6-15
6.7.1.5 Line Graph Toolbar ..... 6-15
6.7.2 Histogram ..... 6-17
6.7.2.1 Selecting Data to be Viewed ..... 6-18
6.7.2.2 Options ..... 6-20
6.7.2.3 Histogram Toolbar ..... 6-20
6.7.3 XY Plot ..... 6-21
6.7.3.1 Selecting Data to be Plotted ..... 6-21
6.7.3.2 XY Plot Toolbar ..... 6-21
6.7.4 Rainflow Histogram ..... 6-22
6.7.4.1 Selecting Data to be View ..... 6-23
6.7.4.2 Options ..... 6-25
6.7.4.3 Rainflow Histogram Toolbar ..... 6-25
6.7.5 FFT ..... 6-26
6.7.5.1 Selecting Data to be Graphed ..... 6-26

6.7.5.2 Options ..... 6-28
6.7.5.3 FFT Toolbar ..... 6-28
6.8 Right-click-Menus ..... 6-29
6.8.1 Data View ..... 6-29
6.8.2 Graphs ..... 6-31
6.8.3 Traces ..... 6-31
6.9 Printing Options ..... 6-31
6.9.1 Print Setup ..... 6-31
6.9.2 Printing Text ..... 6-31
6.9.3 Printing Graphs ..... 6-32
6.10 View Pro Online Help ..... 6-32
6.11 Assigning Data Files to View ..... 6-32

**7. Short Cut Program Generator ..... 7-1**

7.1 Overview ..... 7-1
7.2 Creating a Program Using Short Cut ..... 7-2
7.2.1 Step 1 – Create a New File or Open Existing File ..... 7-2
7.2.2 Step 2 – Select Datalogger and Specify Scan Interval ..... 7-2
7.2.3 Step 3 – Choose Sensors to Monitor ..... 7-5
7.2.4 Step 4 – Setup Output Tables ..... 7-12
7.2.5 Step 5 – Generate the Program in the Format Required by the Datalogger ..... 7-15
7.3 Short Cut Settings ..... 7-16
7.3.1 Program Security ..... 7-16
7.3.2 Datalogger ID ..... 7-16
7.3.3 Power-up Settings ..... 7-17
7.3.4 Select CR200 Compiler ..... 7-17
7.3.5 Sensor Support ..... 7-17
7.3.6 Integration/First Notch Frequency ($f_{N1}$) ..... 7-18
7.3.7 Font ..... 7-18
7.3.8 Set Working Directory ..... 7-18
7.3.9 Enable Creation of Custom Sensor Files ..... 7-18
7.4 Editing Programs Created by Short Cut ..... 7-18
7.5 New Sensor Files ..... 7-19
7.6 Custom Sensor Files ..... 7-19

**8. Datalogger Program Creation with Edlog ..... 8-1**

8.1 Overview ..... 8-1
8.1.1 Creating a New Edlog Program ..... 8-2
8.1.1.1 Program Structure ..... 8-4
8.1.1.2 Edlog File Types ..... 8-4
8.1.1.3 Inserting Instructions into the Program ..... 8-5
8.1.1.4 Entering Parameters for the Instructions ..... 8-6
8.1.1.5 Program Comments ..... 8-7
8.1.1.6 Expressions ..... 8-7
8.1.2 Editing an Existing Program ..... 8-12
8.1.2.1 Editing Comments, Instructions, and Expressions ..... 8-13
8.1.2.2 Cut, Copy, Paste, and Clipboard Options ..... 8-13
8.1.3 Library Files ..... 8-14
8.1.4 Documenting a DLD File ..... 8-14
8.1.5 Display Options ..... 8-14
8.1.5.1 Graphical Toolbar ..... 8-14
8.1.5.2 Renumbering the Instructions ..... 8-15
8.1.5.3 Compress VIEW ..... 8-16

8.1.5.4 Indention ..... 8-16
8.2 Input Locations ..... 8-16
8.2.1 Entering Input Locations ..... 8-16
8.2.2 Repetitions ..... 8-17
8.2.3 Input Location Editor ..... 8-17
8.2.4 Input Location Anomalies ..... 8-19
8.3 Final Storage Labels ..... 8-20
8.4 Datalogger Settings Stored in the DLD File ..... 8-21
8.4.1 Program Security ..... 8-22
8.4.1.1 Setting Passwords in the DLD ..... 8-22
8.4.1.2 Disabling Passwords ..... 8-22
8.4.2 Final Storage Area 2 ..... 8-22
8.4.3 DLD File Labels ..... 8-22
8.4.3.1 Mixed-array Dataloggers ..... 8-22
8.4.3.2 Table-Based Dataloggers ..... 8-23
8.4.4 Power Up Settings/Compile Settings ..... 8-23
8.4.5 Datalogger Serial Port Settings ..... 8-24
8.4.6 PakBus Settings ..... 8-24
8.4.6.1 Network ..... 8-24
8.4.6.2 Beacon Intervals ..... 8-25
8.4.6.3 Neighbor Filter ..... 8-25
8.4.6.4 Allocate General Purpose File Memory ..... 8-25

**9. Datalogger Program Creation with CRBasic Editor ..... 9-1**

9.1 Overview ..... 9-1
9.2 Inserting Instructions ..... 9-2
9.2.1 Parameter Dialog Box ..... 9-2
9.2.2 Right-Click Functionality ..... 9-4
9.3 Toolbar ..... 9-5
9.3.1 Compile ..... 9-7
9.3.2 Compile, Save, and Send ..... 9-7
9.3.3 Conditional Compile and Save ..... 9-11
9.3.4 Templates ..... 9-11
9.3.5 Program Navigation using BookMarks and GoTo ..... 9-12
9.3.6 CRBasic Editor File Menu ..... 9-12
9.3.7 CRBasic Editor Edit Menu ..... 9-13
9.3.7.1 Other Options ..... 9-13
9.3.8 CRBasic Editor View Menu ..... 9-13
9.3.8.1 Editor Preferences ..... 9-13
9.3.8.2 Instruction Panel Preferences ..... 9-15
9.3.8.3 Other Options ..... 9-15
9.3.9 CRBasic Editor Tools Menu ..... 9-16
9.3.9.1 Edit Instruction Categories ..... 9-16
9.3.9.2 Constant Customization ..... 9-17
9.3.9.3 Other Options ..... 9-19
9.3.10 Available Help Information ..... 9-20
9.4 CRBasic Programming ..... 9-20
9.4.1 Programming Sequence ..... 9-20
9.4.2 Program Declarations ..... 9-21
9.4.3 Mathematical Expressions ..... 9-22
9.4.4 Measurement and Output Processing Instructions ..... 9-22
9.4.5 Line Continuation ..... 9-23
9.4.6 Inserting Comments Into Program ..... 9-23

9.4.7 Example Program ..... 9-24
9.4.8 Data Tables ..... 9-24
9.4.9 The Scan — Measurement Timing and Processing ..... 9-26
9.4.10 Numerical Entries ..... 9-27
9.4.11 Logical Expression Evaluation ..... 9-27
9.4.11.1 What is True? ..... 9-27
9.4.11.2 Expression Evaluation ..... 9-28
9.4.11.3 Numeric Results of Expression Evaluation ..... 9-28
9.4.12 Flags ..... 9-29
9.4.13 Parameter Types ..... 9-29
9.4.13.1 Expressions in Parameters ..... 9-29
9.4.13.2 Arrays of Multipliers and Offsets for Sensor Calibration ..... 9-30
9.4.14 Program Access to Data Tables ..... 9-30

**10. Utilities ..... 10-1**

10.1 CardConvert ..... 10-1
10.1.1 Input/Output File Settings ..... 10-1
10.1.2 Destination File Options ..... 10-2
10.1.2.1 File Format ..... 10-2
10.1.2.2 File Processing ..... 10-3
10.1.2.3 File Naming ..... 10-4
10.1.2.4 TOA5/TOB1 Format ..... 10-4
10.1.3 Converting the File ..... 10-4
10.1.3.1 Repairing/Converting Corrupted Files ..... 10-5
10.1.4 Viewing a Converted File ..... 10-6
10.1.5 Running CardConvert from a Command Line ..... 10-6
10.2 Transformer Utility ..... 10-6
10.2.1 Transforming a File ..... 10-6
10.2.2 Controls ..... 10-9
10.3 Device Configuration Utility ..... 10-9
10.3.1 Overview ..... 10-9
10.3.2 Main DevConfig Screen ..... 10-10
10.3.3 Downloading an Operating System ..... 10-11
10.3.4 Terminal Tab ..... 10-14
10.3.5 The Unknown Device Type ..... 10-14
10.3.6 Off-line Mode ..... 10-15
10.4 File Format Convert ..... 10-16
10.4.1 Overview ..... 10-16
10.4.2 Options ..... 10-17

*Appendices*

**A. Glossary of Terms ..... A-1**

**B. Table-Based Dataloggers ..... B-1**

B.1 Memory Allocation for Final Storage ..... B-1
B.1.1 Edlog TD Family Dataloggers ..... B-1
B.1.2 CR800/CR1000/CR6 Series/CR300 Series/CR3000/ CR5000/CR9000 Memory for Programs and Data Storage .. B-2
B.1.3 CR200 Series Dataloggers ..... B-3
B.2 Converting a Mixed-array Program to a TD or PB Table-Based Program using Edlog ..... B-3

B.2.1 Steps for Program Conversion ..... B-3
B.2.2 Program Instruction Changes ..... B-4
B.3 Table Data Overview ..... B-5
B.4 Default Tables ..... B-6

C. Log Files and the LogTool Application ..... C-1

C.1 Event Logging ..... C-1
C.1.1 Log Categories ..... C-1
C.1.2 Log File Message Formats ..... C-2
C.1.2.1 General File Format Information ..... C-2
C.1.2.2 Transaction Log Format ..... C-2
C.1.2.3 Communications Status Log Format ..... C-15
C.1.2.4 Object State Log Format ..... C-17

## *Tables*

5-1. Comma Separated, Field Formatted, Printable ASCII, and Table Oriented ASCII Input File Format Types ..... 5-8
5-2. Example of Event Driven Test Data Set ..... 5-17
5-3. Processed Data File Using Option C ..... 5-18
5-4. Input File Entries to Process the First Data Point for each Test ..... 5-19
5-5. Effects of Out of Range Values for Given Output Options ..... 5-21
5-6. Split Operators and Math Functions ..... 5-22
5-7. Time Series Functions ..... 5-24
5-8. Split SPECIAL FUNCTIONS ..... 5-29
5-9. Definition of Blank or Bad Data for each Data File Format ..... 5-42
8-1. Operators and Functions ..... 8-8
8-2. Editor Keystrokes ..... 8-13
9-1. Formats for Output Data ..... 9-26
9-2. Formats for Entering Numbers in CRBasic ..... 9-27
9-3. Synonyms for True and False ..... 9-28
9-4. Rules for Names ..... 9-29
B-1. Example of Status Table Entries ..... B-8
C-1. Transaction Log Messages ..... C-3
C-2. Communications Status Log Messages ..... C-16

# Section 1. Introduction

PC400 is a software application used to set up, configure, and retrieve data from Campbell Scientific dataloggers. This software application runs on Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 10 platforms.

PC400 software supports programming, communication, and data collection for the "CRBasic dataloggers", including the CR200/205, CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, and CR9000. PC400 also supports the "Edlog dataloggers", such as the CR500, CR510, CR10, CR10X, 21X, CR23X, and CR7 dataloggers, with mixed-array, table-data, and PakBus operating systems where available for a particular datalogger. Communications technologies supported include "direct connect" (or "RS-232" via local serial cable, short haul modems, or other "transparent" links), telephone, TAPI, TCP/IP, VHF/UHF radios, RF400-series spread spectrum radios, and multidrop modems (MD9 and MD485).

PC400 is an ideal solution for users desiring easy-to-use reliable data collection software over a single telecommunications medium, who do not rely on scheduled data collection or graphical real-time displays.

## 1.1 PC400 Overview

### 1.1.1 Main Screen

The main screen for PC400 provides three tabs for communications functions (**Clock/Program**, **Monitor Data**, and **Collect Data**), as well as buttons to launch utilities for working with data files (**View**, **Split**, and **CardConvert**) and for generating or editing datalogger programs (**Short Cut**, **CRBasic**, and **Edlog**). There is also a button to launch the Device Configuration Utility (or DevConfig). DevConfig is used to send new operating systems to dataloggers and other devices, and to configure the settings in the dataloggers/devices. These screens and utilities are also accessible from the PC400 menu, as are other tools, such as a terminal emulator, PakBus Graph, and LogTool, a program to view and store communication logs. Each application has its own extensive help.

Two additional stand-alone utilities are installed with PC400 and can be opened from the Windows Start menu, Programs | Campbell Scientific | PC400 | Utilities. These utilities are Transformer and File Format Convert. (CardConvert and Device Configuration Utility can also be opened from this menu.)

Transformer is a utility to convert Edlog programs to CRBasic programs. Conversions from CR10X to CR800 or CR1000, CR510 to CR800 or CR1000, and CR23X to CR3000 are supported. File Format Convert is used to convert data files from one format to another.

### 1.1.2 Clock/Program and the EZSetup Wizard

Setting up the PC400 datalogger network is a relatively simple process with the EZSetup Wizard, which guides you through the steps necessary to add and enter settings for dataloggers. Once a datalogger is added to the list, you can

choose the Edit button to change those settings, again with the EZSetup Wizard. Progress through the wizard is shown on the left side of the screen, with steps for choosing a datalogger, defining the communications path between the computer and the datalogger, fine tuning settings for the datalogger (e.g., baud rate or security code), testing communications, checking or setting the clock, and finally sending a program or associating an already running program file.

### 1.1.3 Monitor Data

Once you've added and connected to a datalogger, the Monitor Data tab switches to a view that lets you monitor the latest values. These can either be the input locations or variables being updated each execution of the datalogger program or the latest data stored in the datalogger's final storage memory.

For table-based ("TD", "PB", or CRBasic) dataloggers, this screen displays the last record in final storage and updates it as new data is stored in the datalogger. ***Note that this data is only for display purposes; it is not automatically stored to a file on the PC.*** For mixed-array dataloggers, only the last array that was actually collected to a file is displayed and this data is not automatically updated; you must manually initiate another collection.

The Monitor Data tab also lets you edit input locations, public variables, ports, and flags.

### 1.1.4 Collect Data

Once a program is storing data in the datalogger you can collect a copy of that data to a file on the PC. The Collect Data tab shows a list of the final storage areas or tables in the datalogger. You can either retrieve the uncollected data, appending it to a file on the PC, or you can retrieve all of the data from the datalogger, overwriting the file on the PC if it already exists. (Any file in that directory with the same name will be moved to the Windows Recycle Bin.) A browse button lets you choose a folder and file name for the data.

### 1.1.5 View

View is also launched from a button on PC400's main screen. View lets you examine data files collected to the PC, and displays data in a tabular format by record or array. A graph can be displayed to show an unlimited number of traces.

### 1.1.6 Split

A button on PC400's main screen launches Split, a stand-alone application used to post process data files on the PC and generate reports. It can be used to separate mixed-array data files into individual files based on the array ID, merge data from multiple stations into one file, perform calculations, and change date/time formats. Split can create reports or new files for input to other data analysis and display applications, including html formats.

### 1.1.7 CardConvert

CardConvert is a utility to retrieve binary data from Compact Flash or microSD cards containing CR1000, CR3000, or CR6 Series data, and convert the data to an ASCII file.

### 1.1.8 Short Cut

Short Cut is a datalogger program "generator." You only need to select the datalogger type, sensors, and desired outputs, and then Short Cut creates the program file to send to the datalogger. Users don't need to learn about the individual programming instructions. Short Cut includes support for multiplexers and a limited number of other peripherals, and it provides a wiring diagram that you can print to leave in the field with the datalogger.

Short Cut can also be an excellent way to learn about new programming languages. Users familiar with programming for Edlog dataloggers can generate similar programs for CRBasic dataloggers to begin learning about programming in CRBasic, as the CRBasic programs created by Short Cut can be loaded directly into the CRBasic Editor for inspection or editing.

### 1.1.9 CRBasic Editor

The CRBasic Editor is also a program editor, but for the CRBasic dataloggers, including the CR200/205, CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, and CR9000. Instructions are included for declarations, data tables, measurements/control, processing/math, operators, output, and program control. Extensive assistance and program examples are provided in the Help.

### 1.1.10 Edlog

Edlog is a program editor, a software tool to create and modify datalogger programs at the instruction level. Edlog supports the CR500, CR510, CR10, CR10X, 21X, CR23X, and CR7 dataloggers, for any of their operating systems, including mixed-array, table-data, and PakBus where available for a particular datalogger. Instructions are included for sensor measurement, intermediate processing, program and peripheral control, and output processing. The built-in precompiler provides error checking and warns of potential problems in the program. Edlog also gives access to other datalogger settings, such as PakBus addresses, power-up settings, etc., that can be sent in the datalogger program instead of entered via a keyboard/display.

## 1.2 What's New in Version 4?

Version 4.0 of PC400 includes a new data file viewer, View. View maintains the ease of use of our previous data file viewer with enhanced capabilities, including the ability to view an unlimited number of traces on a graph.

You also now have the ability to launch the Device Configuration Utility from PC400's toolbar.

Starting in version 4.1 the CRBasic Editor has the capability to open a read-only copy of any file. This gives you the ability to open multiple copies of a program and examine multiple areas of a very large program at the same time. You can also now continue an instruction onto multiple lines by placing the line continuation indicator (a single space followed by an underscore "_") at the end of the each line that is to be continued. Also, bookmarks in a CRBasic program are now persistent from session to session.

The Device Configuration Utility has a new off-line mode which allows you to look at the settings for a certain device type without actually being connected to a device.

The ability to lock the timestamp column on the left of the data file has been added to View. This keeps the timestamp visible as you scroll through columns of data.

Split has a new "Time Sync to First Record" option that can be used with the time-sync function to avoid blank lines at the start of the output file. Also, a range of time values rather than a single time can now be entered in a Split Copy Condition.

CardConvert can now be run from a command line without user interaction.

An option to provide feedback on PC400 has been added to the PC400 Toolbar's Help menu.

Starting in version 4.2, PC400 now supports display and input of Unicode characters/strings in many areas of the product. Unicode is a universal system for encoding characters. It allows PC400 to display characters in the same way across multiple languages and countries. To support Unicode, an Insert Symbol dialog box has been added to the CRBasic Editor. This allows you to insert Unicode symbols into your CRBasic program for use in Strings and Units declarations.

PakBus Encryption is now supported for communication between PC400 and CR1000, CR3000, and CR800 series dataloggers. Note that the datalogger must be running OS 26 or later in order for PakBus Encryption to be used. A PakBus Encryption Key must be entered in both the datalogger's device settings and PC400's EZSetup Wizard. AES-128 encryption is used.

Boolean values displayed on the Monitor Data tab now have an LED icon next to them to allow for easy toggle.

A Constant Customization feature has been added to the CRBasic Editor. This allows you to define values for one or more constants in a program prior to performing a conditional compile. The constants can be set up with an edit box, a spin box field for selecting/entering a value, or with a list box. A step increase/decrease can be defined for the spin box, as well as maximum and minimum values.

The CRBasic Editor now allows you to Save and Open Display Settings. Display settings affect the look and feel of the CRBasic Editor. This includes font and background, as well as syntax highlighting.

View has a new View Record option in the right-click menu that can be used to view an entire record in a new window.

The main effort in the development of version 4.3 has been support for the new CR6 Series datalogger. Support was also added for the CRVW Series Vibrating Wire Recording Interface.

The main effort in the development of version 4.4 has been support for the new CR300 Series datalogger. The ability to change the number of rows and

columns was added to the Monitor Data tab. The data file viewer was also upgraded from View to View Pro.

## 1.3 Getting Help for PC400 Applications

Detailed descriptions of each application and tool are included in later sections of this manual. Each application also has its own built-in help system. Context sensitive help for an application can usually be accessed by moving the focus to (clicking on) a particular item and pressing the F1 key or by selecting Help from the application's menu.

If you are unable to resolve your problem after reviewing the above noted resources, contact your Campbell Scientific Representative.

## 1.4 Windows Conventions

With the introduction of Windows, Microsoft has been working to establish a standard for the operation of graphical user interfaces. There are numerous conventions and expectations about the way a software program looks and behaves running under Microsoft Windows. Campbell Scientific has adopted as many of these conventions in PC400 as reasonable.

This manual describes a collection of screens, dialogs and functions to work with dataloggers. As with most Windows based software there is almost always more than one way to access the function you want. We encourage you to look around and experiment with different options to find which methods work best for you.

To keep this manual as concise and readable as possible, we will not always list all of the methods for getting to every function. Typically each function will have two methods and some will have as many as four.

The most common methods for doing things are:

**Menus** – Text menus are displayed at the top of most windows. Menu items are accessed either by a left mouse click, or using a hot key combination (e.g., Alt+F opens the File menu). When the menu is opened, you can click on an item to select it, or use arrow keys to highlight it and press the Enter key, or just type the underlined letter.

By convention, menu items that bring up dialog boxes or new windows will be followed by an ellipsis (…). Other items execute functions directly or can be switched on or off. Some menu items show a check mark if a function is enabled and no check mark if disabled.

**Items with Program Focus** – On each screen one button, text area, or other control is selected at a time or "has the focus." Focus is usually indicated when the item is surrounded by a dotted line or is bolded. Pressing the tab key can move focus from item to item. Typing changes a selected text edit box. Pressing the space bar toggles a selected check box. A selected button can also be activated by pressing the Enter key.

**Buttons** – Buttons are an obvious way to access a function. They are normally used for the functions that need to be called frequently or are very important. Clicking a button executes that function or brings up another window. Button

functions can also be accessed from the keyboard using the tab key to move among items on a screen and pressing the Enter key to execute the button function.

**Right Click Menus** – Some areas have pop-up menus that bring up frequently used tasks or provide shortcuts. Just right click on an area and if a menu appears, left click the menu item you want.

**Hot Keys or Keyboard Shortcuts** – Many of the menus and buttons can be accessed using Hot Keys. An underlined letter identifies the hot key for a button or function. To get to a menu or execute a function on a button hold down the Alt key and type the underlined letter in the menu name or the button text. On Windows XP, the hot key letters may not appear until after you've pressed the Alt key.

**Pop-Up Hints** – Hints are available for many of the on-screen controls. Let the mouse pointer hover over a control, text box or other screen feature and the hint will appear automatically and remain visible for a few seconds. These hints will often explain the purpose of a control or a suggested action. For text boxes where some of the text is hidden, the full text will appear in the hint.

# Section 2. System Requirements

## 2.1 Hardware and Software

PC400 is an integrated application of 32-bit programs designed to run on Intel-based computers running Microsoft Windows operating systems. Recommended platforms for running PC400 include Windows XP, Windows Vista, Windows 7, Windows 8, or Windows 10 because they offer the most stable operating environments.

PC400 also requires that TCP/IP service be installed on the PC.

# Section 3. Installation, Operation and Backup Procedures

## 3.1 CD-ROM Installation

The following instructions assume that drive D: is a CD-ROM drive on the computer from which the software is being installed. If the drive letter is different, substitute the appropriate drive letter.

1. Put the installation CD in the CD-ROM drive. The install application should come up automatically. Skip to step 3.

2. If the install does not start, then from the Windows system menu, select Start | Run. Type "d:\setup.exe" in the Open field or use the Browse button to access the CD-ROM drive and select the setup executable. This activates the PC400 Installation Utility.

3. Follow the prompts on the screen to complete the installation. The installation will require a CD key. You will find this code printed on the back of the jewel case for the original installation CD.

A shortcut to launch PC400 is added to your computer's Start menu under Programs | Campbell Scientific | PC400. If the default directories are used, PC400 executable files and help files are placed in the C:\Program Files\CampbellSci\PC400 directory.

Working directories will also be created under C:\Campbellsci for PC400's configurations and data files, user programs, and settings for the accessory applications and utilities.

## 3.2 PC400 Operations and Backup Procedures

This section describes some of the concepts and procedures recommended for routine operation and security of the PC400 software. If software and computer systems were perfect, this section would not be necessary. However, since this software is required to run with predictable results in the real world on real computers, the following guidelines and procedures will be helpful in minimizing possible problems that may occur.

### 3.2.1 PC400 Directory Structure and File Descriptions

#### 3.2.1.1 Program Directory

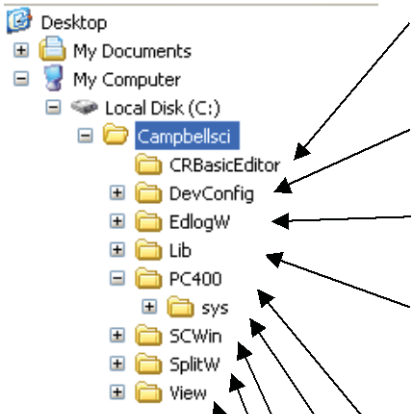
As described in the installation procedures, all of the files for program execution are stored in the C:\Program Files\Campbellsci\PC400 directory. This includes the executables, DLLs, and most of the application help files. This directory does not need to be included in back up efforts. PC400 and its applications rely on registry entries to run correctly; therefore, any restoration of the program should be done by reinstalling the software from the original CD.

### 3.2.1.2 Working Directories

In this version of PC400, each major application keeps its own working directory. The working directory holds the user files created by the application, as well as configuration and initialization (*.INI) files.

This scheme was implemented because we use the underlying tools and many of the applications (the server itself, library files, datalogger program editors, etc.) in a number of different products. By providing a common working directory for each major application, we hope to make it easier to keep track of files and information as you move from one product to another.

The following figure shows the typical working directories for PC400 if the default options were selected during installation.




**CRBasicEditor**
Default directory for user's CRBasic datalogger programs.

**DevConfig**
Default directory for Device Configuration Utility saved settings files.

**EdlogW**
Default directory for user's Edlog datalogger programs .

**Lib**
Libraries of pre-compilers, templates and help files used by CRBasic Editor, including the CR200/205 pre-compilers used by PC400's communications engine, and libraries of settings supported by devices configured by the DevConfig application.

**PC400**
Default directory for data files.

**PC400\SYS**
PC400 working directory – these files are controlled by the communications server and **should not be disturbed** by the user except when backing up the system. Tampering with these directories or files can cause unpleasant consequences.

**SCWin**
Default directory for user's datalogger programs created in the Short Cut program generator.

**SplitW**
Default directory for user's Split parameter files and resulting reports and output files.

**View**
Default directory for View's window settings.

## 3.2.2 Backing up the Network Map and Data Files

As with any computer system that contains important information, the data stored in the PC400 working directory should be backed up to a secure archive on a regular basis. This is a prudent measure in case the hard disk crashes or the computer suffers some other hardware failure that prevents access to the stored data on the disk.

The maximum interval for backing up data files depends primarily on the amount of data maintained in the datalogger memory. The datalogger's final storage is configured as ring memory that will overwrite itself once the storage area or table is full. If the data is backed up more often than the oldest records in the datalogger are overwritten, a complete data record can still be maintained by restoring the data from the backup and then re-collecting the newest records from the datalogger.

#### 3.2.2.1 Performing a Backup

PC400 provides a simple way to back up the network map, the PC400 data cache, and the initialization files for each of the applications. The network map will restore all settings and data collection pointers for the dataloggers and other devices in the network. The data cache is the binary database which contains the collected data from the datalogger. Initialization files store settings such as window size and position, configuration of the data display, etc.

From PC400's menu, choose Network | Backup/Restore Network, and then press **Backup**.

The backup file is named PC400.bkp and is stored in the C:\CampbellSci\PC400 directory (if you installed PC400 using the default directory structure). You can, however, provide a different file name if desired.

#### 3.2.2.2 Restoring the Network from a Backup File

To restore a network from a backup file, choose Network | Backup/Restore Network. Select the *.bkp file that contains the network configuration you want to restore, and press **Restore**. Note that this process DOES NOT append to the existing network — the existing network will be overwritten when the restore is performed.

### 3.2.3 Loss of Computer Power

The PC400 communications server writes to several files in the \SYS directory during normal operations. The most critical files are the data cache table files and the network configuration files. The data cache files contain all of the data that has been collected from the dataloggers by the PC400 server. These files are kept open (or active) as long as data is being stored to the file.

The configuration files contain information about each device in the datalogger network, including collection schedules, device settings, and other parameters. These files are written to frequently to make sure that they reflect the current state and configuration of each device. The configuration files are only opened as needed.

If computer system power is lost while the PC400 server is writing data to the active files, the files can become corrupted, making the files inaccessible to the server.

While loss of power won't always cause a file problem, having files backed up as described above will allow you to recover if a problem occurs. If a file does get corrupted, all of the server's working files need to be restored from backup to maintain the synchronization in the server state.

### 3.2.4 Program Crashes

If the communication server crashes, there is a possibility that files can be corrupted (note, however, that corruption is much less likely with a program crash than during a power loss, since the computer operating system remains in control and can close the files left open by the failed program). If, after a program crash, the server does not run properly, you may need to restore the data from backup.

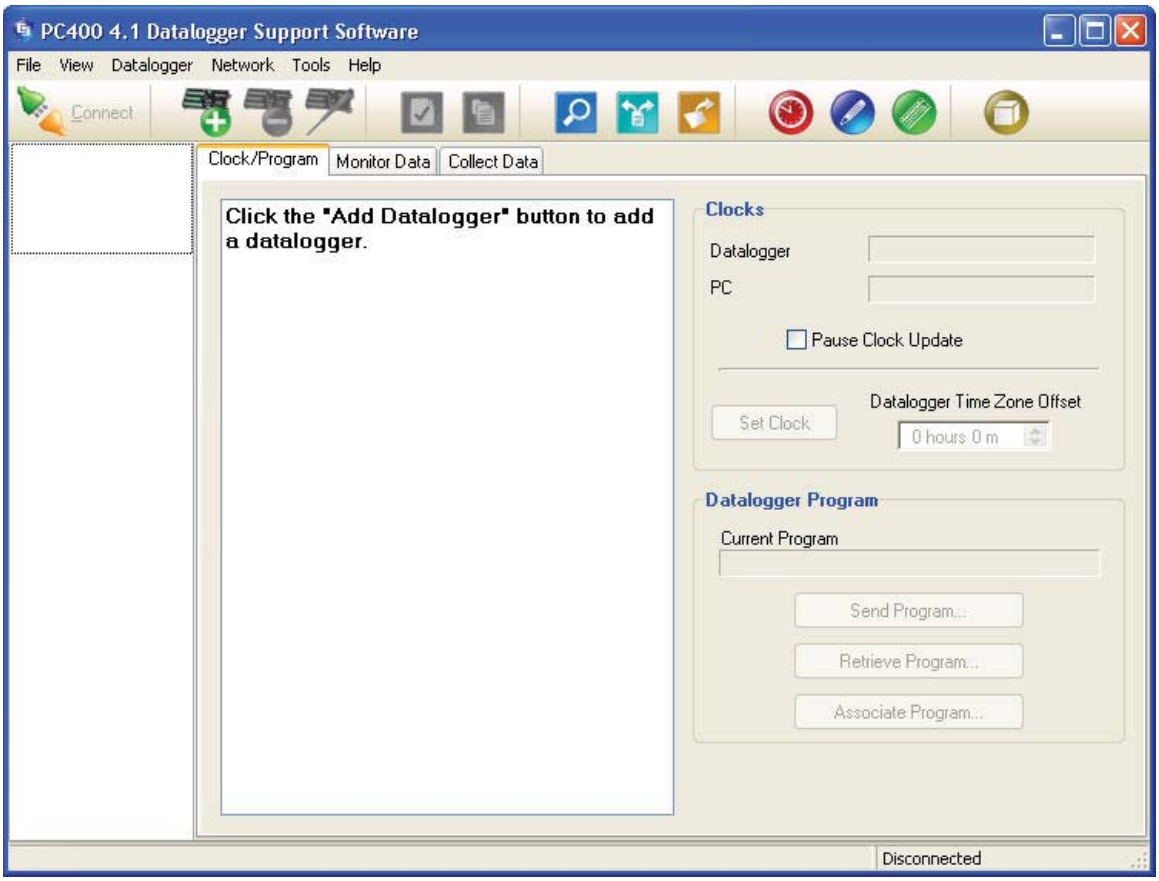
If you have problems restarting the PC400 server after a program crash or it crashes as soon as it starts, make sure that the PC400 server has not left a process running. You can check this by going to the Windows Task Manager and selecting the Process tab. In the list of processes, look for the Toolbar or one of the client applications. If one of these processes exists but the Toolbar is not running, select this process and click "End Process"; you will be asked to confirm the end process.

# Section 4. The PC400 Main Screen

*This section provides an overview of PC400, a detailed description of the communications tabs and pull-down menus of the PC400 Main Screen, and an overview of PC400's troubleshooting tools.*

## 4.1 Overview

To start PC400 go to the start menu of the computer and select the PC400 icon under Start | Programs | Campbell Scientific | PC400. You can alternatively use the shortcut on the desktop if you elected to create one during the installation process.




PC400 offers an integrated main screen, with three tabs for basic communications functions (Clock/Program, Monitor Data, and Collect Data), and buttons from which to launch standalone applications to work with data files or create datalogger programs.

Setting up and configuring PC400 to communicate with dataloggers is done with the EZSetup Wizard. This wizard appears automatically the first time you run PC400. To add additional dataloggers, click the Add button on the main toolbar to bring up the EZSetup Wizard again. Editing existing datalogger and communications settings is also done through this wizard, via the Edit button.

PC400 supports the CRBasic series of dataloggers, including the CR200/205, CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, and CR9000. PC400 also supports the Edlog dataloggers (CR500, CR510, CR10, CR10X, 21X, CR23X, and CR7 dataloggers) using any of their mixed-array, table-data, or PakBus operating systems.

PC400 supports one medium of communications for any given datalogger. These media include "direct connect" via serial communications (or "RS-232") via local serial cable, short haul modems, or other "transparent" links, telephone, TAPI, TCP/IP, VHF/UHF radios, RF400-series spread spectrum radios, and multidrop modems (MD9 and MD485).

PC400 does not support parallel port communications, RF95T modems, or multiple media (such as phone-to-RF). PC400 is designed to use PakBus dataloggers and other PakBus devices in their default configurations; they are not supported as routers.

In order to be easy to use, PC400 relies on user-attended communications. It does not provide for automated scheduled data collection or automated clock checks. It also doesn't support remote connections from other PCs.

In addition to the communications functions, PC400 provides Split, View Pro, and CardConvert for working with data files, Short Cut for generating programs, and Edlog and CRBasic Editor for more advanced datalogger program editing. These tools are accessed via the buttons on the main screen or the pull-down menu selections under the Tools menu item.

Help for each application is available from the Help menu item, or by moving the focus to a control (by clicking on or tabbing to a control) and pressing F1.

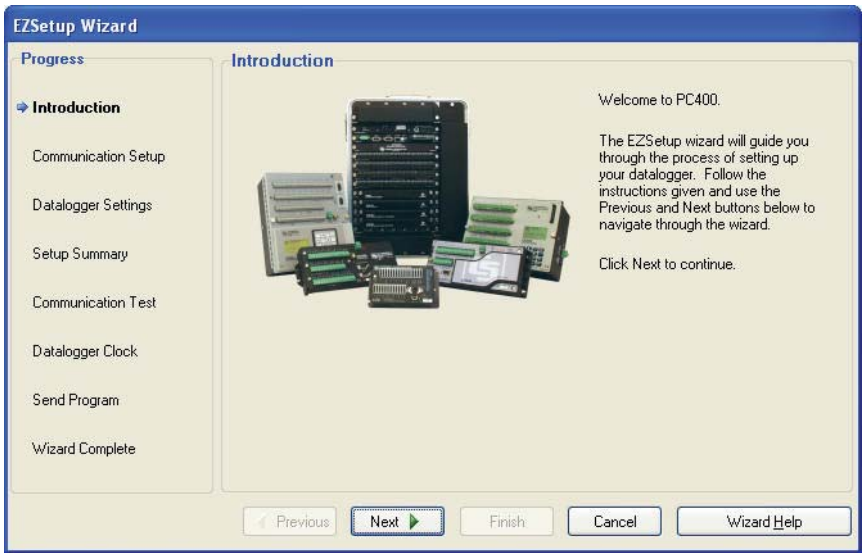
To exit PC400, either click the [X] in the upper right hand corner of the main screen, or select Exit under the File menu.

# 4.2 Clock/Program Tab and EZSetup Wizard

## 4.2.1 EZSetup Wizard

Dataloggers are added to the network with the EZSetup Wizard. This wizard is also used to edit the settings for a datalogger after it's been added. The EZSetup Wizard is automatically displayed when PC400 is run for the first time. It can also be opened by clicking the Add or Edit buttons.

The EZSetup Wizard starts as shown below.




Previous and Next buttons are provided to move through the wizard. Progress is shown by the blue arrow next to each step displayed at the left. Help is available from the Help button as well as the text displayed on the right side of the screen itself.

In Communication Setup you select the datalogger type and give it a name that will also become the default file name for data files collected from that datalogger. The next step allows you to choose from the possible communications media supported for that datalogger. PC400 will display the serial ports (COM ports) known to your Windows operating system. PC400 fills in as many communications settings as possible and in many cases you can use the default settings. It provides fields for user-entered communications settings such as phone numbers and RF radio addresses. Help for entering these settings is provided on the right side of each screen, by clicking the F1 key, or by pressing the Help button for each Wizard screen. You may also want to consult the manual for that particular communications hardware.

**NOTE**

If connecting via a datalogger USB port, the USB driver must be installed and a USB cable connected between the datalogger and computer before the COM part is available for selection. The **Install USB Driver** button can be used to select and install the USB drivers for dataloggers and peripherals that require them.

Datalogger Settings are provided for fine tuning the connection to the datalogger. The baud rate offered is typically the maximum baud rate supported by that datalogger and communications medium; lower rates may be required for cell phones or noisy telephone links. Enter a Security Code only if the datalogger is configured – via the keyboard/display or settings in the datalogger program – to use it. Enter a PakBus Encryption Key only for a CR1000, CR6 Series, CR300 Series, CR3000, or CR800 Series datalogger that has been configured to use it.

Setup Summary provides a list of the settings entered. You can use the Previous button to change these settings if necessary.

The Communications Test step allows you to test the communications link before going any further. If the datalogger is not installed, you can skip this and the next two steps.

If communication succeeds, you can move to the Datalogger Clock step where you can check or set the datalogger's clock to match the PC's system time. If the datalogger is in a different time zone, you can enter an offset in hour units.

The Send Program step allows you to send a program to the datalogger. This may be a program you created with Short Cut, Edlog or the CRBasic Editor or a program supplied by someone else. If it is a mixed-array datalogger, and the datalogger is already running a program when you first connect to it, you should associate the .DLD file so that PC400 will use the labels for input locations and final storage. Dataloggers with table-based operating systems (TD, PakBus, and CRx000) will know their program if one is running and will provide table definitions that contain the labels. If you don't have a program for the datalogger you can skip this step and send a program later from the Clock/Program tab.

If you connected to the datalogger during the EZSetup Wizard, when you leave the wizard it will ask if you wish to stay connected.

### 4.2.2 Clock/Program Tab

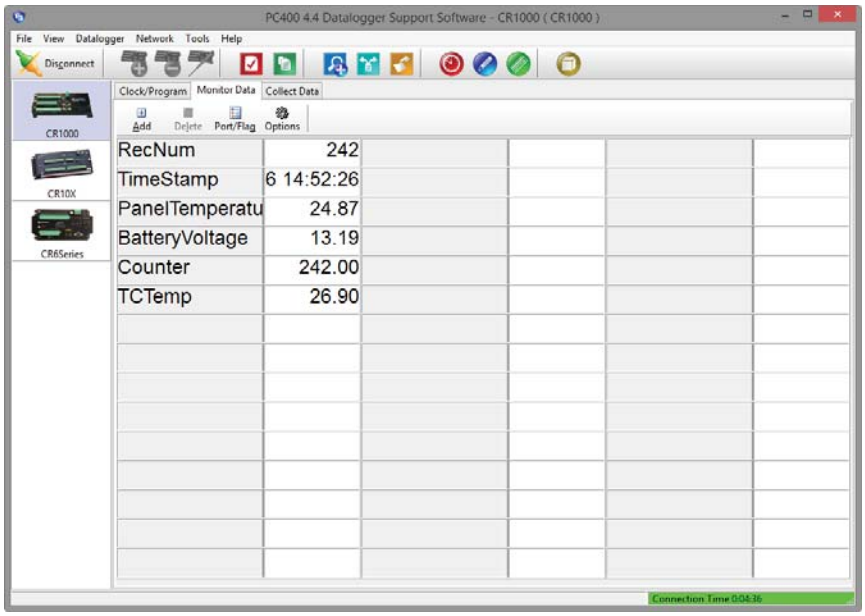
Once you've added a datalogger, you may use other buttons on the Clock/Program tab to delete it, edit its settings, or add another datalogger.

You must start with the Clock/Program tab to access either of the other two communication functions tabs. While connected on the Clock/Program tab, you may also change the datalogger clock or send or associate a new program.

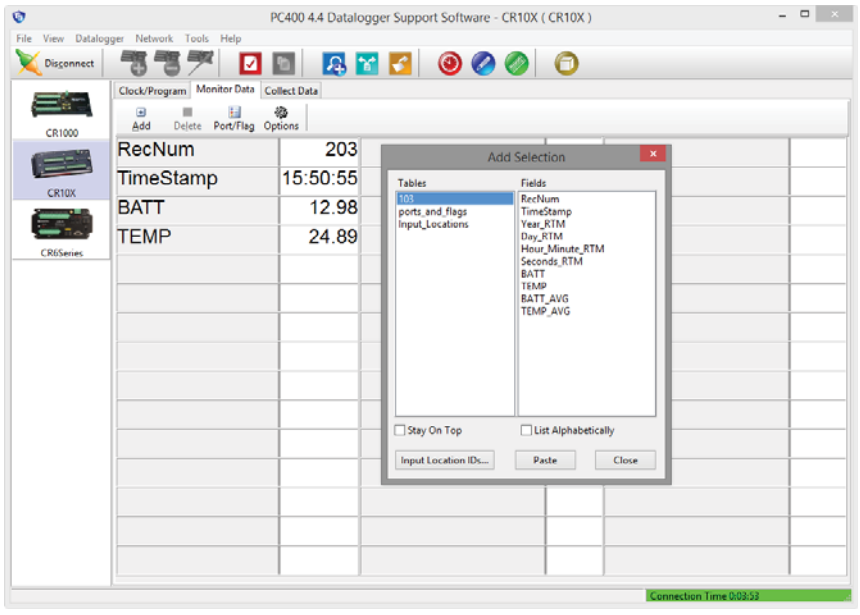
Note that the default Max Time On-Line setting for most communications links is zero ("0 d 00 h 00 m"), which means that PC400 will never hang up until you click Disconnect. For telephone links, the default Max Time On-Line setting is ten minutes in order to reduce the possibility of inadvertent and expensive long distance or cellular telephone charges. There are, however, other links that can result in expensive connection charges, such as digital cellular links using TCP/IP that charge by the byte. Leaving the datalogger connected also uses battery power, so if the datalogger power supply is not recharged from a reliable source, it may discharge its battery below safe levels. Be sure, therefore, that you do not leave the datalogger connected beyond the time necessary to do the tasks you need to do.

## 4.3 Monitor Data Tab

Once you've added and connected to a datalogger, you can monitor the values stored in the datalogger. If a program was sent to or associated with a datalogger, or if table definitions were obtained from a table-based datalogger, PC400 will, by default, show the input locations or public variables that have labels associated with them. An example of such a display from a CR1000 is below.




PC400 can also display final storage values. Press the Add button to add final storage values to the display. For mixed-array operating systems in the CR10X family of dataloggers, these values will be displayed only if the final storage data was collected from the datalogger, and only the most recent values actually collected will be displayed. An example from a CR10X-based station is below.




Note that, even though mixed-array dataloggers do not store record numbers internally, PC400 assigns one and displays it with a date/time stamp. PC400 computes these values from the labels and settings it finds in the .DLD program file for that array ID.

In the case of table-based dataloggers, PC400 will display the last record from a final storage table and will automatically update these records as they are

stored in the datalogger's memory. Note that PC400 does not automatically collect this data to the data file. If you want a permanent record of the data, you must collect it manually from the Collect Data tab.

Press the Options button to change the number of decimal places displayed, the interval at which the data is updated, or the number of rows and columns displayed.

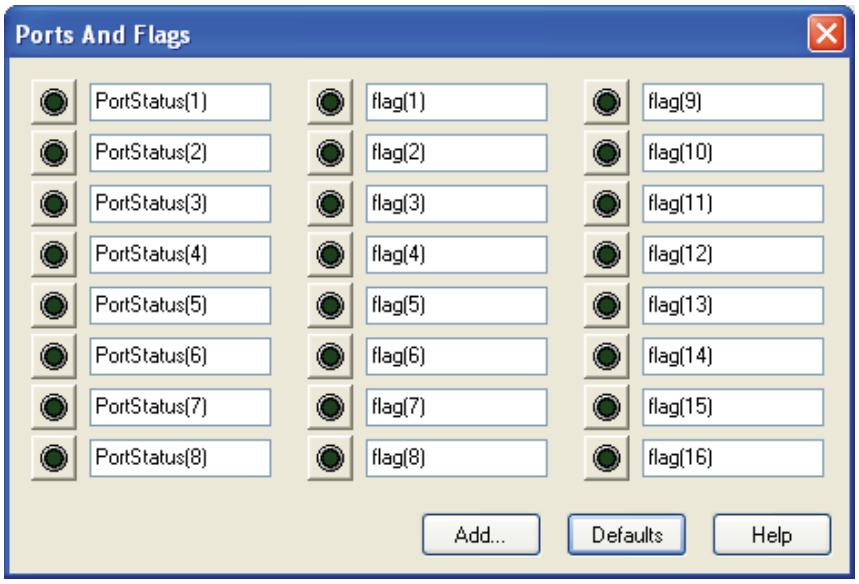
You may find that your labels or the number of digits displayed are too long to fit in the space provided. Using the mouse, you may put your cursor over the border between the columns for the labels and values and drag it left or right to make it easier to read. The column width is not preserved when PC400 is closed and restarted. The table name, variable name (or input location name), and data value also will be displayed momentarily as a popup hint if you hover your cursor over a value for a few seconds.

In some cases, you may wish to edit input locations or public variables, for example to change sensor offsets or control datalogger program execution. To change input location or public variable values, double click on the number itself until it turns yellow, then use the PC keyboard to enter a new value. (When a value is enabled for editing, if the ESC key is pressed, the change will be canceled.) Alternately, you can right-click the value, select View Value, and edit the value in the resulting dialog box.

The state of a port or flag can be changed by clicking the LED icon to the right of the field value. A black LED indicates that the port or flag is low; green indicates that it is high.

When the Port/Flag button is selected, a window appears displaying the ports and flags in the datalogger.

The state of a port or flag can be changed by clicking the LED icon to the left of the field label. A black LED indicates that the port or flag is low; green indicates that it is high. Custom labels can be assigned to the ports and flags by placing the cursor within the label field and typing in a new label.



Program variables that are declared as Boolean can also be placed on this display, for dataloggers that support data types. For these dataloggers, an Add button is available that, when pressed, lists all of the tables in the datalogger. When a table is highlighted on the left side of the window, any variables that are declared as Boolean in the program will be displayed on the right side of the window.

To return the Ports and Flags display to its original state, press the Defaults button. This will reset all labels to their original names, update the number of flags based on the currently running program (for CR6 Series and CRX000 dataloggers), and remove any Boolean values placed on the screen (for CR6 Series dataloggers and CRX000 dataloggers that support data types).

Different datalogger models have a different number of ports and flags. The Ports and Flags dialog box will display only those ports and flags available for the datalogger type. Behaviors for each datalogger type are shown below.

- Mixed array dataloggers have a fixed number of ports and user flags that are available. The ports and flags dialog box will display only those ports and flags supported by the datalogger; no additional values can be added. The Defaults button will reset any user-defined labels that have been typed in.

- CR1000, CR6 Series, CR300 Series, CR3000, CR800, CR5000, and CR9000X dataloggers do not have predefined flags. The first time a program is sent to the datalogger, PC400 will look for a Public array with the name of Flag in the program. If a Flag array is found, the declared flags will be added to the Ports and Flags dialog box. The number of flags that will be added is limited by the number of cells available on the Ports and Flags display. CR800, CR1000, CR6 Series, CR300 Series, and CR3000 dataloggers have ports that can be toggled from this display; they will be displayed in the first column and the remaining cells will be available to display flags and other Boolean values in the program. The CR5000's and CR9000X's ports cannot be controlled from this display, so all cells will be available for Flags and Boolean values. For these seven dataloggers, pressing Defaults will reset all labels to their original names, update the number of flags based on the currently running program, and remove any Boolean values placed on the screen.

- CR9000 and CR200 dataloggers do not have ports that can be toggled from this display. They also do not support the declaration of variables as Boolean. They also do not have any predefined flags. The Ports and Flags dialog will display one to three columns, depending upon the number of flags defined in a Public array with the name of Flag in the program. Pressing Defaults will reset all labels to their original names and update the number of flags based on the currently running program.

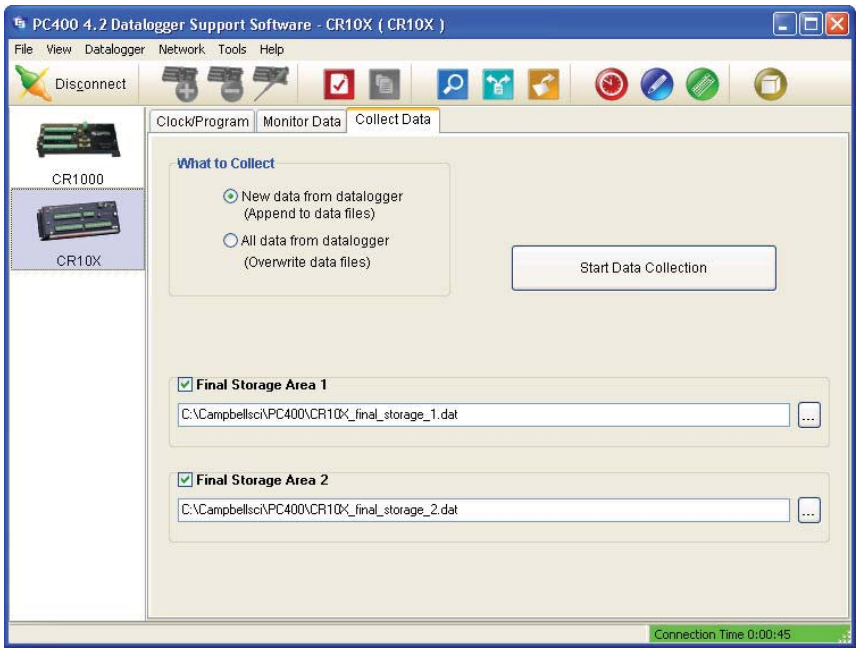
**NOTE** Note that a control port must first be configured for output in the datalogger program before it can be toggled on or off. Consequently, if you select a port and it doesn't appear to change, your program may not have the port configured for output (refer to your datalogger operator's manual).The CR500 and CR510 have two control ports, but only one of the ports, control port 1, can be configured for output. Therefore, control port 2 cannot be toggled on or off. It is included on the display so that you can monitor its status.

**NOTE** A Boolean variable is a variable that can have one of two states: high/low, off/on, -1/0, true/false. Variables for CRBasic dataloggers can be declared as Boolean with the Public or Dim statement.

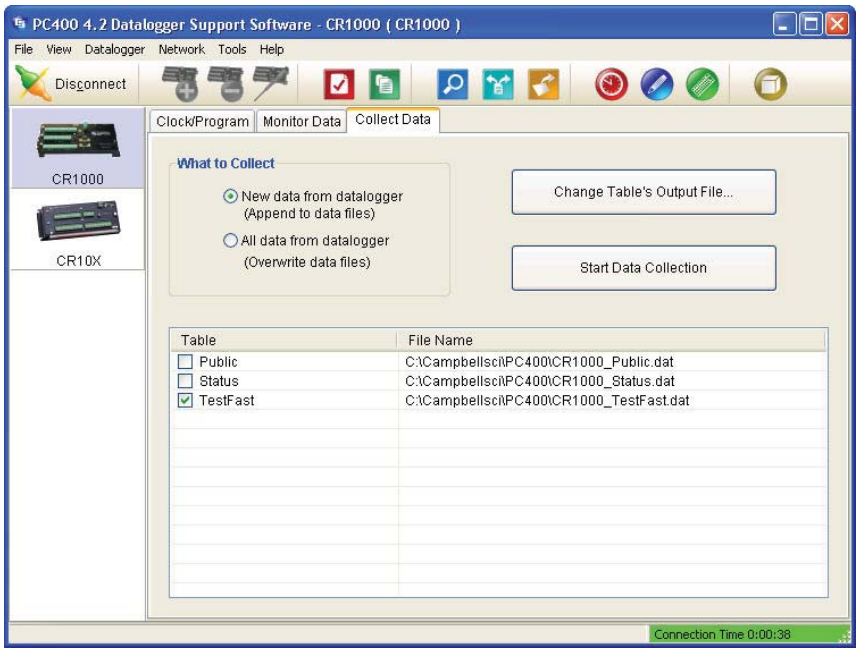
# 4.4 Collect Data Tab

If you have connected to a datalogger, PC400 enables the Collect Data tab to provide for manual data collection. PC400 shows the possible final storage areas for mixed-array dataloggers, and the tables stored in table-based dataloggers.

Example of Collect Data tab for a mixed-array CR10X:



Example of a Collect Data tab for a table-based CR1000:




In either case, once a final storage area or table is selected you may either collect the new data since the last time you collected it with PC400, in which case PC400 appends the new data to the file if it exists, or you may collect all of the data from that final storage area or table, in which case any file with the same name is replaced. (The previous file is moved to the Windows' Recycle Bin, in case you select this option by mistake.)

# 4.5 Pull-down Menus

Access to almost all of the buttons and tabs on PC400's screens is also available via pull-down menus. Many of these are described in detail in other sections.

## 4.5.1 View Menu

PC400 can display the user interface component text (for buttons, dialog boxes, etc.) in an alternate language if a separate language package has been installed. If a language package is installed on your machine, the View menu will have a Languages entry and you will see the language in the subsequent pick list. When a new language is chosen, the main PC400 user interface will immediately reflect the change. Opened auxiliary applications will not reflect the change until they are closed and reopened.

The main PC400 user interface and most auxiliary applications are capable of displaying text for alternate languages. However, Split and Edlog do not have alternate language support at this time.

**NOTE** Available language packages are provided by Campbell Scientific's international representatives or on the CSI website. They are not included in a standard PC400 installation.

## 4.5.2 Datalogger Menu

### 4.5.2.1 Connect/Disconnect

This option provides the same function as the Connect/Disconnect button on the main toolbar.

### 4.5.2.2 Update Table Definitions

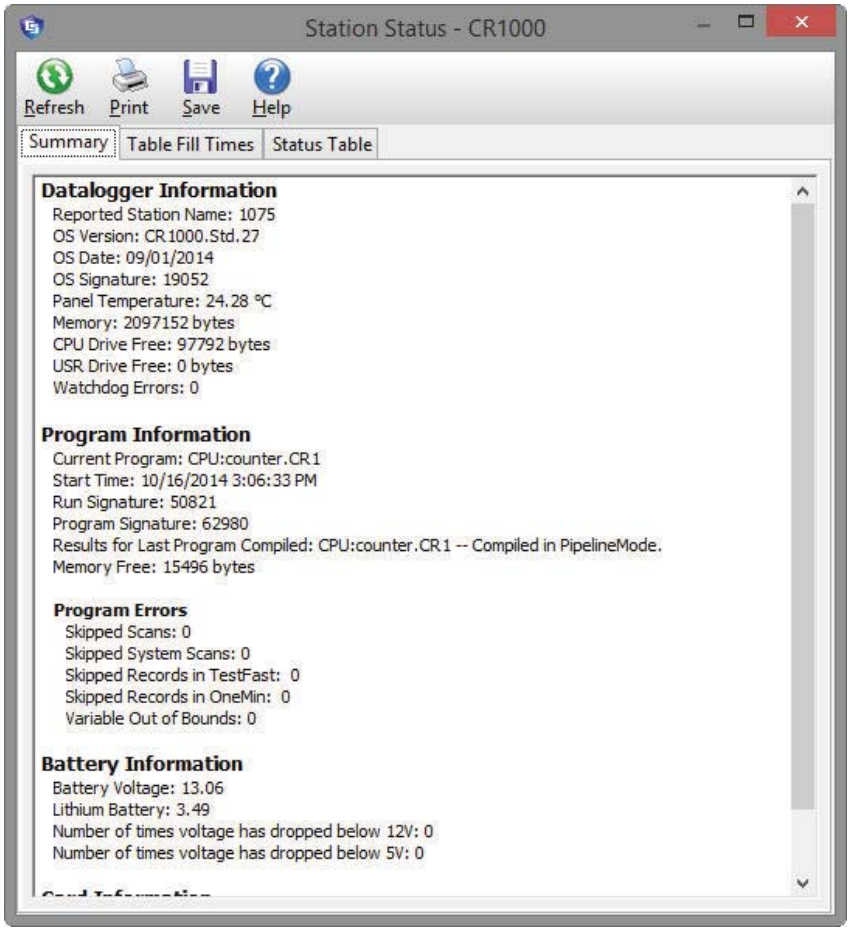
When a table-based datalogger is first set up or when a new program is sent, PC400 queries the datalogger for its table definitions. This information is used when displaying data in the Monitor Data tab and when collecting data.

If a new program is sent by another user or by using some other program, PC400 will not be aware of the changes in the datalogger's table definitions. To update the table definitions for a datalogger, connect to that datalogger and select **Update Table Definitions** from the Datalogger menu.

### 4.5.2.3 Station Status

Information about the datalogger program, the execution of the program, battery voltage, internal temperature, etc. can be viewed from the Datalogger | Station Status menu item.

The Station Status window is shown below:



The window has three tabs. The Summary tab provides an overview of important status information in the datalogger, including the information about the datalogger model and its firmware, program details, program errors, battery voltage levels, and card memory (if one is present).

**NOTE** Only the Summary tab is available for array-based dataloggers.

The Table Fill Times tab lists the tables in the datalogger, along with the maximum number of records the table can hold, and the estimated amount of time that it will take the table to fill. A data table can be reset from this window by pressing the Reset Tables button.

**NOTES** No Table Fill Time Statistics will be shown for a CR200-series datalogger, because they cannot be calculated for this datalogger.

For the CR10XTD, CR10XPB, CR510TD, CR510PB, CR23XTD, and CR23XPB, the Time of Fill will not be shown and you will not have the option to Reset Tables.

Resetting a table will erase the data in the datalogger and in the data cache.

The Status Table tab lists all of the status table fields in the datalogger along with their values. By default, all of the fields in the status table are displayed. To select only certain status data to be viewed, press the Select Fields button. This will display a list of the status data available in the datalogger. Select one or more of the fields and then press OK. The current values will be displayed in the table. If you select a cell within the status table and right click, a short cut menu will be displayed. From this menu you can select fields or view/modify a value (if it is a writable value).

Press Refresh to prompt PC400 to query the datalogger and update the values again, Print to print the information being displayed, or Save to save the information being displayed to a file. (Note that the Print and Save buttons are not enabled for the Table Fill Times tab.)

Refer to individual datalogger manuals for a list of fields included in the status table for each datalogger and a description of each.

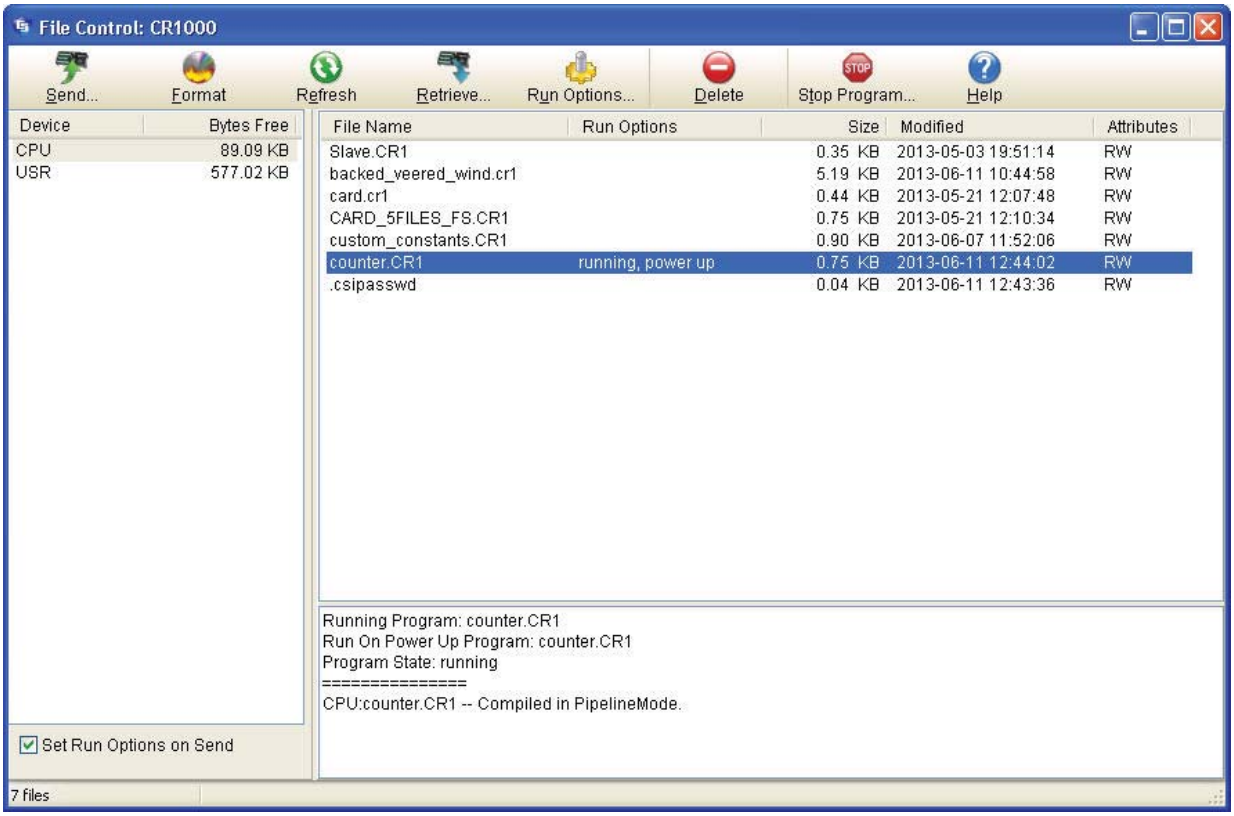
By default, certain values in the Status Table are checked each time a program is sent. If a problem is found during the status check, the Status Table will pop-up along with a warning indicating the problem that has been identified. Through the Tools | Options menu item, automatic status checks can also be generated each time a connection to the datalogger is made and/or on a specified interval. From this menu, the automatic status checks can also be turned off completely.

#### 4.5.2.4 File Control for CR5000, CR800, CR1000, CR6 Series, CR300 Series, CR3000, and CR9000 Dataloggers

CR5000, CR800, CR1000, CR6 Series, CR300 Series, CR3000, and CR9000 dataloggers have a built in file system much like a computer hard disk. Multiple files can be stored in the datalogger's memory or on a PC card,

including data files and datalogger programs. Note that unlike other dataloggers, these dataloggers retain in memory programs that have been downloaded to them unless the programs are specifically deleted or the datalogger memory is completely reset.

File Control is used to manage all the files on these dataloggers. File Control is opened from a button on the PC400 toolbar or from the Datalogger | File Control menu item.




The File Control window displays a list of files stored on the datalogger's CPU, PC card, SC115 USB drive, or USR drive. The window on the left lists all of the data storage devices available for the selected datalogger (CPU, CRD, USB, or USR). Selecting a device shows a list of the files stored there.

**NOTE** The USR drive is a user-created drive in the CR800, CR1000, CR6 Series, and CR3000 dataloggers. It can be set up by assigning a value to the datalogger's UsrDriveSize setting. This drive must be set to at least 8192 bytes, in 512 byte increments (if the value entered is not a multiple of 512 bytes, the size will be rounded up).

The Run Options for a file indicate whether it is set to running, power up, or running, power up. The currently executing program is indicated by running. The file size is displayed, as well as the last time the file was modified and whether or not the file is Read Only (R) or Read/Write (RW). Note that the Size, Modified date, and Attributes may not be available for all dataloggers.

At the bottom of the right-hand side of the window is a summary box that indicates the Running Program, the Run On Power Up Program, the current Program State (running, stopped, or no program), and the last compile results.

There are several options to work with the files and directories on the datalogger.

**Send** is used to transfer files from the computer to the datalogger. Clicking the Send button brings up a standard file selection dialog box. A new file can be chosen to send to the highlighted device. If the Set Run Options on Send check box is selected, you will be asked to specify the Run Options for the file being sent as described below under Run Options.

Datalogger programs, data files, and other ASCII files can be sent to the datalogger.

**Format** is used to format the selected device. Just like the formatting a disk on a computer, all of the files on the device are deleted and the device is initialized.

**Refresh** will update the list of files for the selected device. If a change is made via the keyboard or under program control, it will not be reflected in this window unless the screen is refreshed.

**Retrieve** will get the selected file from the datalogger and store it on the computer. A Save As dialog box comes up allowing you to specify the directory and file name for the saved file.

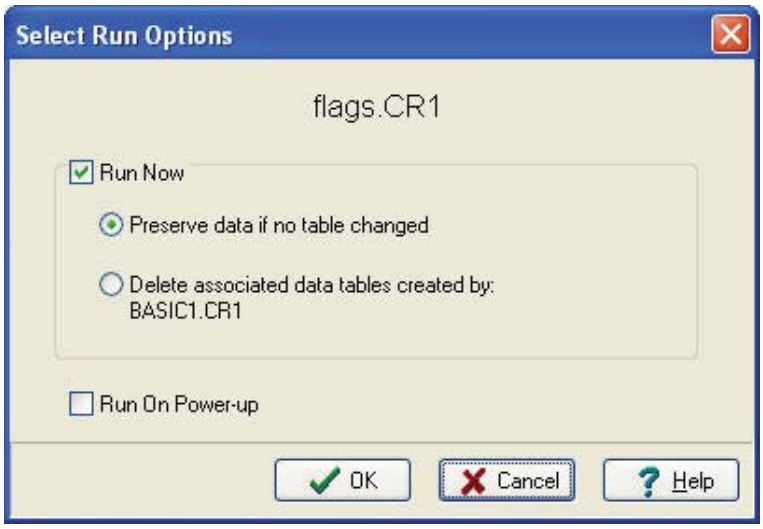
**NOTE** File Control should not be used to retrieve data from a CF card created using the CardOut instruction. Using File Control to retrieve the data file can result in a corrupted data file.

**Run Options** brings up a dialog box that is used to control what program will be run in the datalogger. Highlight a file, and then select the Run Options button. From the resulting dialog box, select the run options.

#### *Run Now*

The Run Now run options are different for the different datalogger types.

**CR1000/CR3000/CR800 Series/CR6 Series/CR300 Series Datalogger Run Now Options**




When Run Now is checked, the program is compiled and run in the datalogger. You may choose to preserve existing data tables on the datalogger's CPU if there has been no change to the data tables (**Preserve data if no table changed**) or to delete data tables on the CPU that have the same name as tables declared in the new program (**Delete associated data tables**).

**CAUTION**

Neither of these options affects existing data files on a card if one is being used. If a data table exists on the card that has the same name as one being output with the new program, the message will be returned "Data on Card is from a different program or corrupted". Data will not be written to the card until the existing table is deleted. Data tables on the card that have different names than those declared in the new program will be maintained and will not affect card data storage when the new program is running.

When using the **Preserve data if no table changed** option, existing data and data table structures are retained unless one of the following occurs:

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

To summarize, any change in data table structure will delete all tables on the datalogger's CPU, regardless of whether or not the Preserve Data option was chosen. If the Preserve Data option was chosen but the datalogger was unable to retain the existing data, the following message will appear in the Compile Results: Warning: Internal Data Storage Memory was re-initialized.

**CR9000(X)/CR5000 Datalogger Run Now Options**

The Run Now options and behavior for the CR9000(X) and CR5000 dataloggers are different from the CR1000, CR3000, CR6 Series, CR300 Series, and CR800 Series dataloggers. Below is a dialog box for a CR5000 datalogger.

When Run Now is checked, the program is compiled and run in the datalogger. All data tables on the CPU are erased. You have the option of whether or not to erase data files stored on a card.

### *Run On Power-up*

The file will be sent with the Run On Power-up attribute set. The program will be run if the datalogger loses power and then powers back up.

### *Run Always*

Run Now and Run On Power-up can both be selected. This sets the program's file attribute in the datalogger as Run Always. The program will be compiled and run immediately and it will also be the program that runs if the datalogger is powered down and powered back up.

Pressing **Run Options** to restart a stopped program in the CR1000, CR3000, CR6 Series, CR300 Series, or CR800 displays a different dialog box. From the dialog box you choose **Restart Program** to begin running the selected program immediately. You can also select an option button to determine whether or not the data tables previously created by the program are erased or retained. Note that you cannot change the **Run on Power-up** option when restarting a program.

**NOTE** CR1000, CR3000, CR6 Series, CR300 Series, and CR800 Series Dataloggers — A program marked as "Run on power up" can be disabled when power is first applied to the datalogger by pressing and holding the DEL key.

**Delete** – Highlight a file and press the delete button to remove the file from the datalogger's memory.

**Stop Program** halts execution of the currently running datalogger program.

Select the option to stop the program and retain the data files, or to stop the program and delete data files. The option to retain or delete data files includes those in internal datalogger memory and those on a card written by the CardOut instruction. In most cases, data written to a card using the TableFile instruction will not be erased. However, when writing data to the card using the TableFile instruction with option 64, the file currently in use will be erased while other files will be maintained.

If you select the option to Delete Data, you also have the option of whether or not to clear the Run On Power-up option for the file. Select the check box to clear the Run On Power-up option. Clear the check box to leave the Run On Power-up option of the file unchanged.

If a CR800, CR1000, CR6 Series, CR300 Series, or CR3000 program has been stopped, use the Run Options to restart it. If data files were not deleted when the program was stopped, you will once again be able to choose whether to retain or erase the data files. Instead of restarting the stopped program, you can choose a new program file to run by selecting a different file in the File Name field before pressing Run Options.

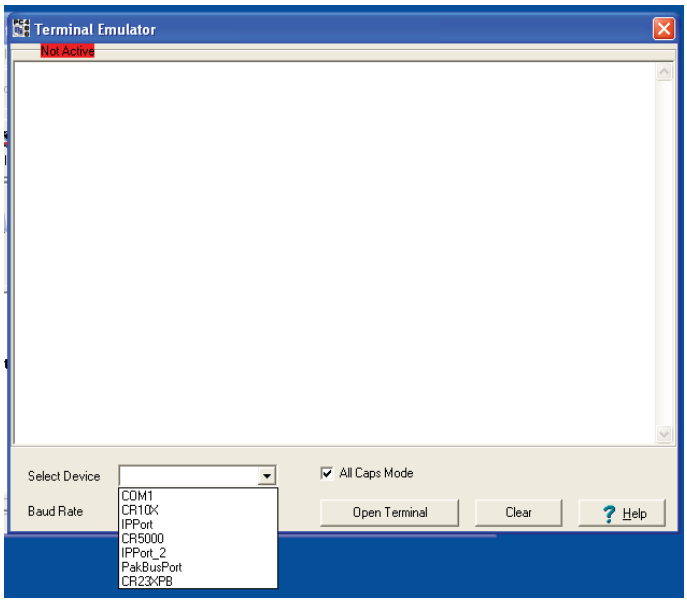
#### *Right Click Menu Options*

When a file name is selected, pressing the right mouse button displays a menu with the Retrieve File, Delete File, Rename File, View File (retrieves the file and opens it in the CRBasic Editor), Run Options, and Stop Program options.

**NOTE** The View File option can be used to edit a program on your datalogger. After making the desired edits and saving it to your computer, you will need to send the edited program to the datalogger.

### 4.5.2.5 Terminal Emulator

Terminal Emulator does just that – it emulates a terminal connected to a datalogger or communications device. Terminal emulator comes up showing a blank screen. Clicking on Select Device shows a list of devices known to PC400.




Selecting a device and baud rate, and then clicking Open Terminal causes PC400 to attempt to connect with that device. If the device is a datalogger, PC400 will call the datalogger over whatever communications link has been established and will attempt to get a prompt from that datalogger. If the device is a root device, such as a serial or COM port, PC400 simply opens that port at the specified baud rate.

Terminal Emulator has several uses. If you open a COM port, you can use it to set up devices that present their own menus, such as RF400 radios or NL100s. See the manual for that particular device for information on how to navigate the menus and what settings to choose.

Another potential use is to communicate with smart devices attached to the datalogger. Some dataloggers allow a “pass-through” mode whereby you can communicate through them to an attached SDI-12 sensor to set its address or other parameters. See the manuals for the datalogger and sensor for the relevant commands.

Troubleshooting communications devices is another use for Terminal Emulator. When calling for support on these devices, an applications engineer may ask you to use this method to manually dial through phone or RF modems, for example.

You can close a connection to start another one by clicking the Close Terminal button. You can close the Terminal Emulator itself by clicking the [X] button in the upper right corner of the screen.

## 4.5.3 Network Menu

### 4.5.3.1 Add/Delete/Edit/Rename Datalogger

The Add, Delete, and Edit options perform the same functions as the buttons on the main toolbar. The Rename Datalogger can be used to change the name of a datalogger.

#### 4.5.3.2 Backup/Restore Network

This function can be used to save a copy of the network map to a file, and then to restore the network if necessary. The settings for all the devices in the network will be saved.

A default is given for the directory and file name to be used for the backup or restore. This can be changed by typing over the default directory and/or file name or selecting the button to the right of the field containing the file name and browsing to the desired directory and file name.

To back up the current network map, select the file name to which the backup will be stored, and then press the Backup button. The network map will be saved to the chosen file, and a message will appear indicating that the network has been backed up.

To regenerate the network map from a backup file, select the name of the backup file to restore from the dialog box, and then press the Restore button. Note that this backup will replace the existing network (it does not add to the existing network).

The backup/restore option will be disabled, if you are currently connected to a datalogger. You must disconnect from the datalogger, before performing a backup or restoring the network.

#### 4.5.3.3 Computer's Global PakBus Address

All nodes and routers, including dataloggers and the computer itself, in a PakBus network must have a PakBus address no higher than 4094. Setting up a PakBus network that provides peer-to-peer communications between the nodes can be a complex task, so PC400 avoids this by setting up each PakBus datalogger within its own PakBus subnetwork. Hence, each datalogger in a PC400 network could, in fact, use the same PakBus address. However, even though each PakBus datalogger has its own subnetwork, these dataloggers do keep track of the PakBus addresses of other devices that communicate with them, including PCs. Therefore, if more than one PC is communicating with a single PakBus datalogger, it may be helpful to assign each PC its own PakBus address. Keep in mind that, although PakBus devices keep their own routing tables, they will communicate with any other PakBus device that has an address >3999. Therefore, you may want to use different PakBus addresses for each PC communicating with your PakBus dataloggers. This menu item provides that access.

### 4.5.4 Tools Menu

#### 4.5.4.1 Stand-alone Applications

Use these menu items to launch the stand-alone tools, including Split, View Pro, CardConvert, Short Cut, Edlog, and CRBasic Editor. Some users prefer using the keyboard to the mouse, and the Alt-T keystroke provides this access. These stand-alone applications are described in more detail in their own sections of this manual.

### 4.5.4.2 Options

The following two options can be set:

1) Automatically Check Datalogger Status – This option allows you to determine when the datalogger status is automatically checked. The options include After Connection, After Program Send, and/or On Interval.

   The status values checked include the following: PakBus Address, WatchDog Errors, Skipped Scans, Skipped Slow Scans, Skipped Records, Variable Out of Bounds, Program Errors, Battery Voltage, Lithium Battery, Number of times voltage has dropped below 12V, Number of times voltage has dropped below 5V, Stack Errors, and Calibration Errors. Note that not all status values are applicable to all dataloggers. Only those status values applicable to the current datalogger will be checked.

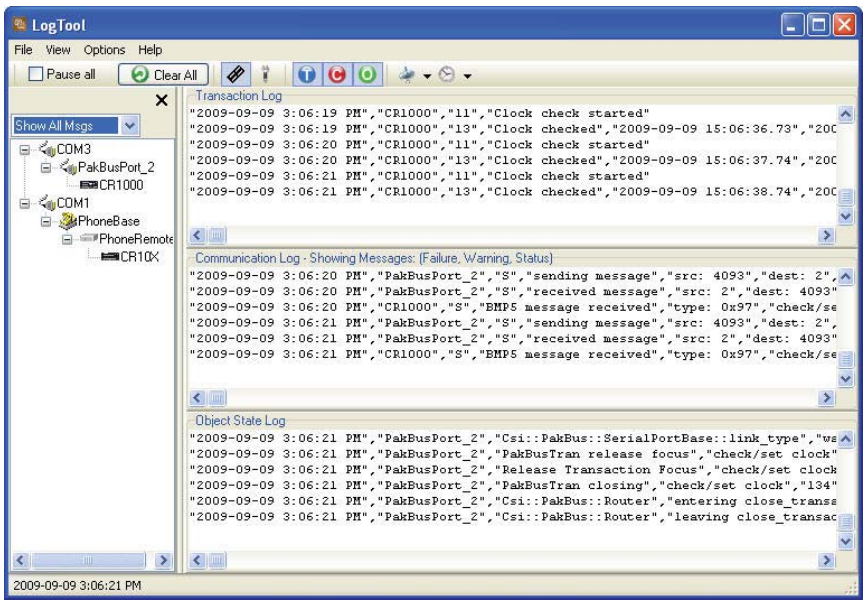
   If a problem is found during a datalogger status check, the status table will pop-up along with a warning indicating the problem that has been identified.

2) Show Hints – The Show Hints option is used to turn on/off the tool tips that appear when your mouse hovers over a control (button or field). When a check mark appears beside the menu item, the tool tips will be displayed. When the check mark does not appear beside the menu item, no tool tips will be displayed. This menu item is a toggle; select it to change its state.

### 4.5.4.3 LogTool

Sooner or later in the real world, every communications link has some problem. Sometimes these problems resolve themselves, such as when phone lines are busy at first, but become available as other users complete their tasks.

When resolving communications problems is not straightforward, the LogTool can let someone analyze the details of communications activity. The LogTool is accessed from the Tools | LogTool menu item.



On the left side of the LogTool window, is a display of all devices known to PC400. You can choose to show all messages or filter them to show only certain devices or dataloggers. PC400 will also store these messages to log files on the PC's hard drive and will eventually overwrite these files to keep them from growing forever. You can control the number and size of each log file type with the Options | Log File Settings menu and dialog.

PC400's communications engine creates four different types of message logs. Transaction messages are the highest level, and show every action undertaken by PC400. For example, if you're monitoring a datalogger every second, PC400 will show clock check messages each second. Many of the messages displayed on this log are understandable and may provide some insight to the processes going on behind the scenes when you monitor or collect data.

The other three logs are more technical in nature, but can be very useful to Campbell Scientific support engineers. Communications messages show when devices are activated, the settings passed to those devices, and their responses. This log may include "status", "warning", and "fault" messages. The Object State messages record the state of each software object behind the scenes in PC400. The most detailed log is the low level log. A separate log is stored for each root level device (each COM port, IP port or TAPI port). These logs record every byte sent or received through that port. Interpreting all of these logs requires knowledge of both Campbell Scientific protocols as well as other, lower level protocols, and requires detailed knowledge of what is supposed to be happening. Most users will only need to access these logs when requested to do so by Campbell Scientific support personnel.

For more information on these logs see Appendix C, *Log Files and the LogTool Application (p. C-1)*.

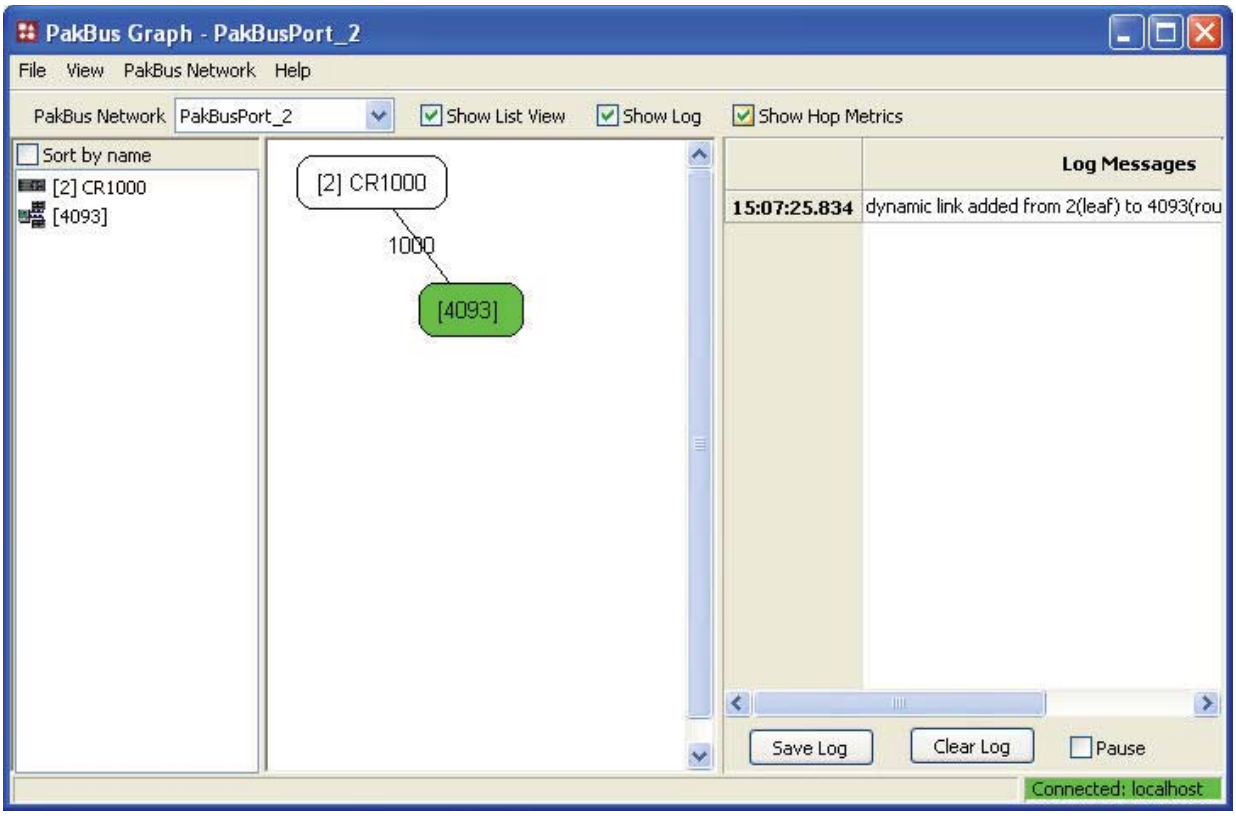
#### 4.5.4.4 PakBus Graph

PakBus Graph is a utility that graphically depicts the connections in a PakBus datalogger network. It provides a look at PC400's PakBus routing table. In addition, the utility can be used to change the settings of a PakBus device.

The window for PakBus graph is divided into three sections: the list of PakBus devices, a graphical depiction of the PakBus network, and the log messages for PakBus communication. The list of devices and the log can be toggled off by clearing the Show List View and Show Log options, respectively.

Software servers are identified in PakBus Graph by the color green. Other devices remain colorless unless they have been selected with the mouse cursor. When selected, they are colored cyan.

The default PakBus address for PC400 is 4093. Other PakBus devices will be shown by name and address, if known.




#### 4.5.4.4.1 Selecting the PakBus Network to View

When PakBus Graph is opened, it is set to view the first PakBus network on the computer on which the datalogger support software is running. If more than one PakBus network is set up on the computer, the different networks can be viewed individually by selecting a port name from the PakBus Network drop-down list.

#### 4.5.4.4.2 Dynamic and Static Links

There are two types of links to PakBus dataloggers that the server recognizes: static links and dynamic links. Static links (depicted using red lines) are the communication links to dataloggers that have been set up in the software, but which have not been confirmed by communicating with the datalogger(s). You will see these dataloggers listed in the software's network map. Dynamic links (black lines) are communication links to dataloggers that have been confirmed. You may also see links to leaf node dataloggers that have not been set up in the software, but which the server has "learned about" by querying the PakBus network.

#### 4.5.4.4.3 Viewing/Changing Settings in a PakBus Datalogger

If you right-click a device in PakBus graph, you will be presented with a floating menu. From this menu, select **Edit Settings** to display a list of the PakBus settings for the datalogger. Some of these settings are read-only, but other settings can be changed. Click within the cell for a setting, enter a new value, and then press return to make a change. If the change is accepted, the cell will appear green. If the change was denied (which likely means the setting is read-only), the cell will appear red.

#### 4.5.4.4.4 Right-Click Functionality

There are several options available from the floating menu that is displayed when you right-click a device (not all devices will have all settings):

**Edit Settings –** This option shows the PakBus settings of a device (see above).

**Ping Node –** This option will send a packet to the selected device to determine if it is reachable in the PakBus network. The results of the ping will be displayed in the Log Messages. Each ping message will include the size of the packet sent, and the time of response from the pinged device. The last message recorded will include summary information from the ping.

**Verify Routing Table** – This option will request the routing table from a PakBus device.

**Reset Node –** This option will reset the routing table in a PakBus device.

**Change PakBus Address** (server only) – By default, the PakBus address of the software server is 4093 (PC400) or 4094 (LoggerNet). This option lets you change this default.

**Search for Neighbors** (server only) – When this option is selected, the software server will broadcast a Hello Request every 5 seconds to search for PakBus neighbors with which it can communicate. During this time, the PakBus port is kept on-line.

**Broadcast Reset** (server only) **–** This option will reset the routing table in the selected PakBus device, as well as any neighbors of the selected device that are acting as routers.

**Unlock Position** – This option will unlock a device that has been locked into position in PakBus Graph by dragging it to a new position on the screen. All devices can be unlocked by selecting View | Unlock All Positions from the menu.

#### 4.5.4.4.5 Discovering Probable Routes between Devices

You can view the probable route that communication will take between two PakBus devices by sequentially clicking on the two devices in PakBus Graph. The probable communication route will be highlighted in cyan. If the Show Hop Metrics check box is selected, the graph will include the time, in milliseconds, that communication takes between the two devices. The results are also displayed in the Log Messages portion of the window.

# Section 5. Split

*Split is a tool that works with output data files (*.dat) collected from Campbell Scientific dataloggers. It is used to post-process data from either mixed-array or table-based dataloggers.*

*Split can create reports based on collected data by filtering data based on time or conditions. It can generate statistics, perform calculations, reformat files, check for data quality (limit testing), and generate tables with report and column headings. It can also handle the time synchronization necessary to merge up to eight data files.*

## 5.1 Functional Overview

Split is a tool to analyze data collected from Campbell Scientific dataloggers. Its name comes from its function of splitting out specific data from a larger data file. Originally, Split could only process mixed-array files, and it was used to "split" the different arrays – typically different time intervals – of a file into separate files (e.g., for hourly versus daily data).

In addition to splitting out mixed-array data, Split can filter output data based on time or conditions, calculate statistics and new values, reformat files, or check data quality (limit testing). Split can generate tables with report and column headings, as well as time synchronize and merge up to eight data files.

Input Files (maximum of eight) are read by Split, specific operations are performed on the data, and the results are output to a new Output File or a printer. Split creates a parameter file (*filename*.PAR) that saves all of your settings such as which data files are read, what operations are performed on the data set, and where the final results will be saved. The parameter file may be saved and used again.

Input Files must be formatted in Printable ASCII, Comma Separated ASCII, Field Formatted ASCII, Final Storage (Binary) Format, Table Oriented ASCII (TOACI1 or TOA5), Table Oriented Binary (TOB), or Raw A/D data (such as the results of a burst measurement).

Split can be used to convert a file of one format to a different format. For example, a Table Oriented ASCII file can be converted to the Comma Separated ASCII format used in mixed-array datalogger data files. This is useful to convert table-based data files to work with applications that were written to work with mixed-array files.

Output files generated by Split can be Field Formatted (default), Comma Separated ASCII, or Printable ASCII. Split can also create reports in ASCII as well as html formats, or send them directly to a printer.

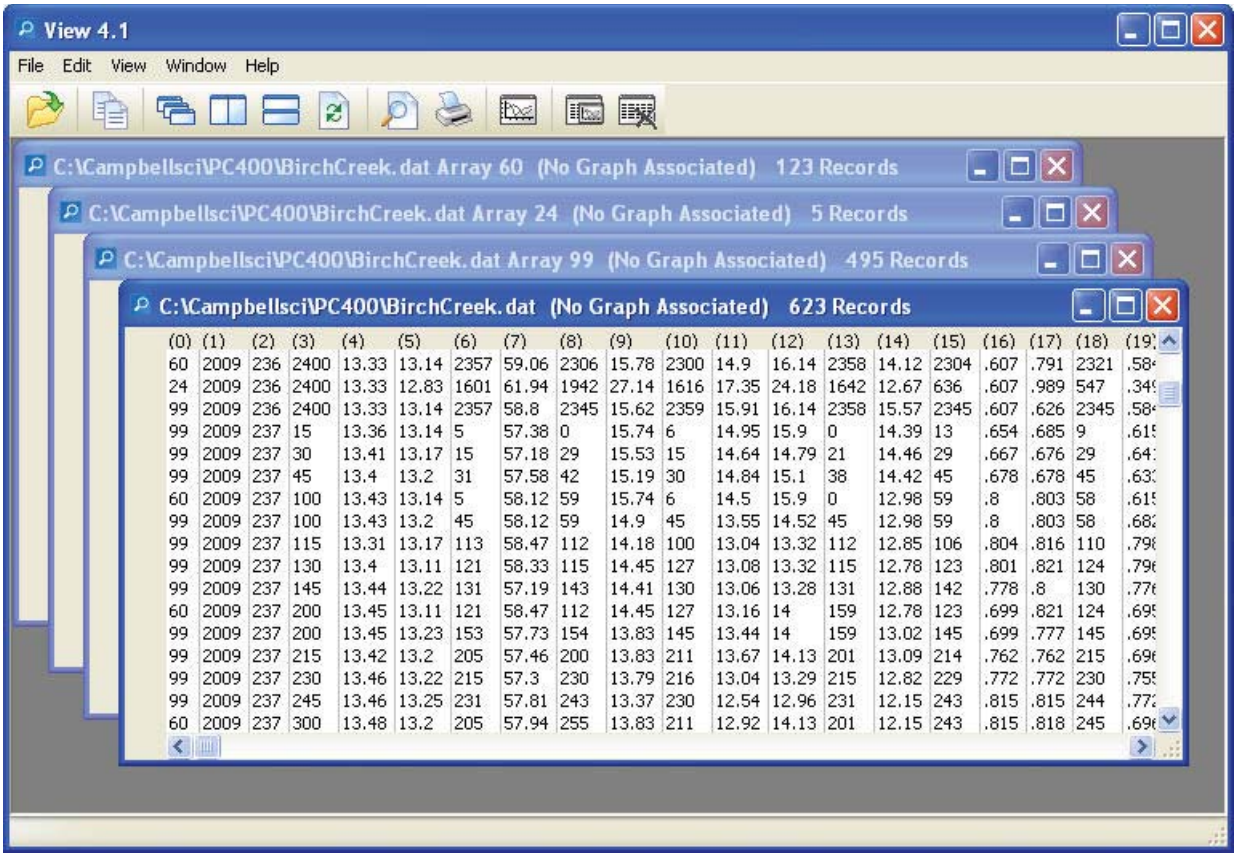
Split lends itself to experimentation. The processed data are displayed on the screen, giving immediate feedback as to the effect of changes or new entries to the parameter file. Split does not modify the original Input File.

# 5.2 Getting Started

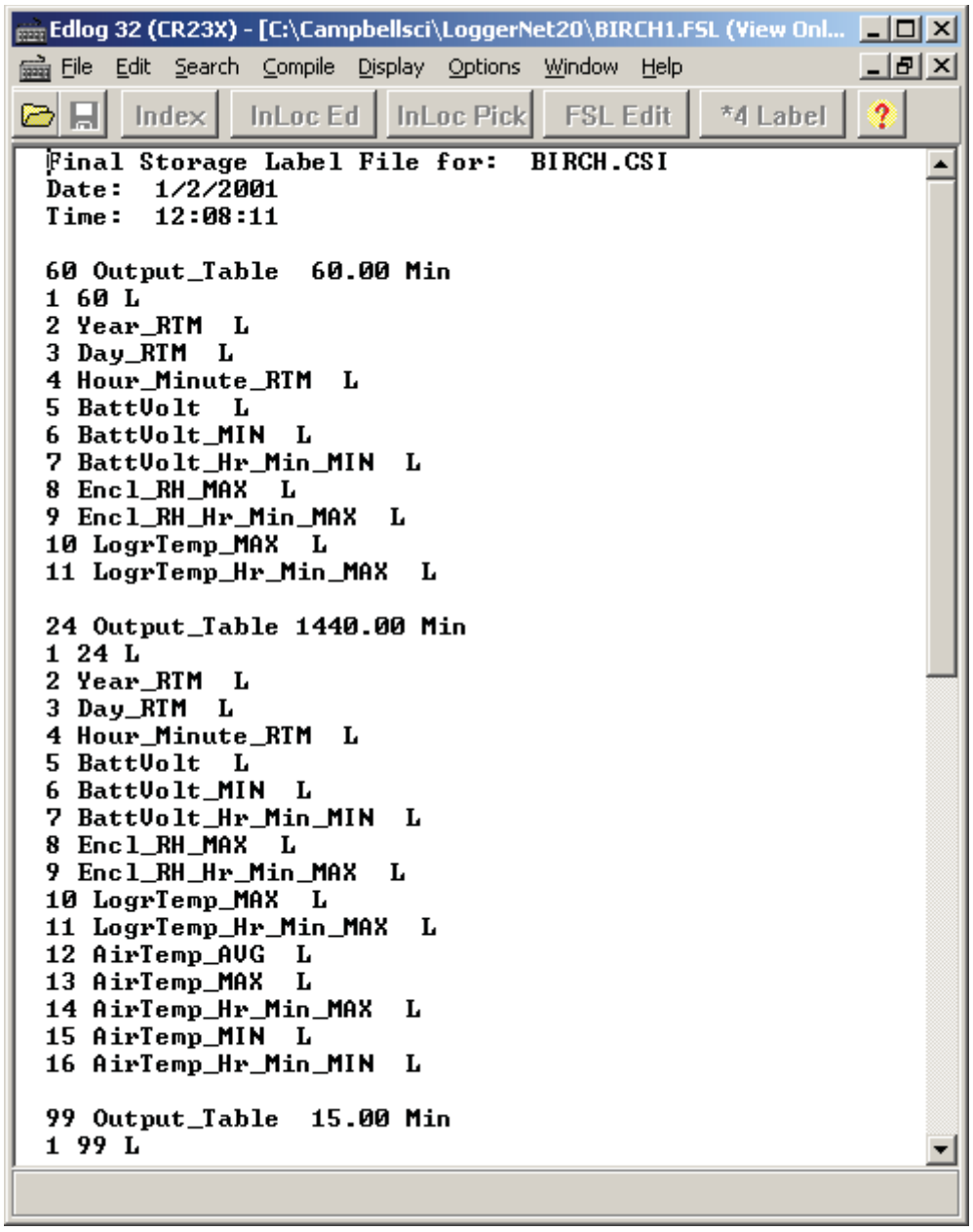
The most common use of Split is to separate array data collected on a particular interval from a data file containing data output at several different intervals.

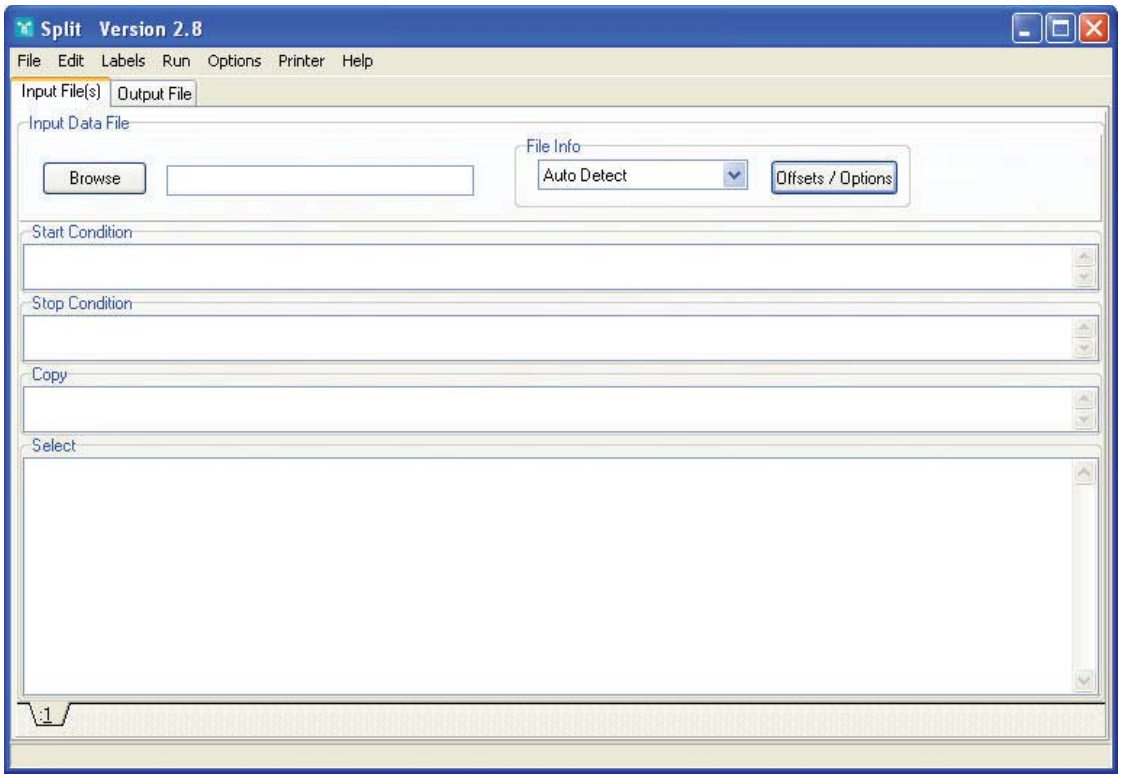
In the following example, hourly data are split from a data set that contains 15 minute, hourly and daily data. The data was collected from BirchCreek, a CR10X datalogger. The CR10X was loaded with a program created by Edlog named Birch.dld.

The 15 minute data, array 99, the hourly data, array 60, and the daily data, array 24, are intermixed in the data file.




When Edlog compiled Birch.dld, it also created the Final Storage Label file, Birch.fsl that lists the final storage locations for each data element.


Edlog 32 (CR23X) - [C:\Campbellsci\LoggerNet20\BIRCH1.FSL (View Onl...
File Edit Search Compile Display Options Window Help
Index InLoc Ed InLoc Pick FSL Edit *4 Label
Final Storage Label File for: BIRCH.CSI
Date: 1/2/2001
Time: 12:08:11
60 Output_Table 60.00 Min
1 60 L
2 Year_RTM L
3 Day_RTM L
4 Hour_Minute_RTM L
5 BattVolt L
6 BattVolt_MIN L
7 BattVolt_Hr_Min_MIN L
8 Encl_RH_MAX L
9 Encl_RH_Hr_Min_MAX L
10 LogrTemp_MAX L
11 LogrTemp_Hr_Min_MAX L
24 Output_Table 1440.00 Min
1 24 L
2 Year_RTM L
3 Day_RTM L
4 Hour_Minute_RTM L
5 BattVolt L
6 BattVolt_MIN L
7 BattVolt_Hr_Min_MIN L
8 Encl_RH_MAX L
9 Encl_RH_Hr_Min_MAX L
10 LogrTemp_MAX L
11 LogrTemp_Hr_Min_MAX L
12 AirTemp_AVG L
13 AirTemp_MAX L
14 AirTemp_Hr_Min_MAX L
15 AirTemp_MIN L
16 AirTemp_Hr_Min_MIN L
99 Output_Table 15.00 Min
1 99 L



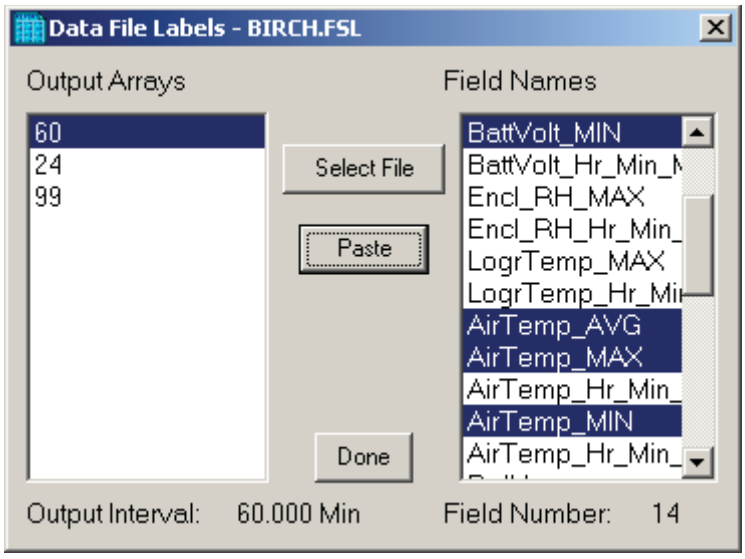

When you start Split a blank template similar to the one above is shown. This template is used to enter the parameters that will define what data from the input file to include in the output file. The parameters entered on this template can be saved as a parameter file (*.PAR) and reused for other data.

On the **INPUT FILE** tab you only need to specify the input file name, copy condition, and the data to select. Split allows start and stop conditions to be specified but if they are left blank, the entire file will be read.

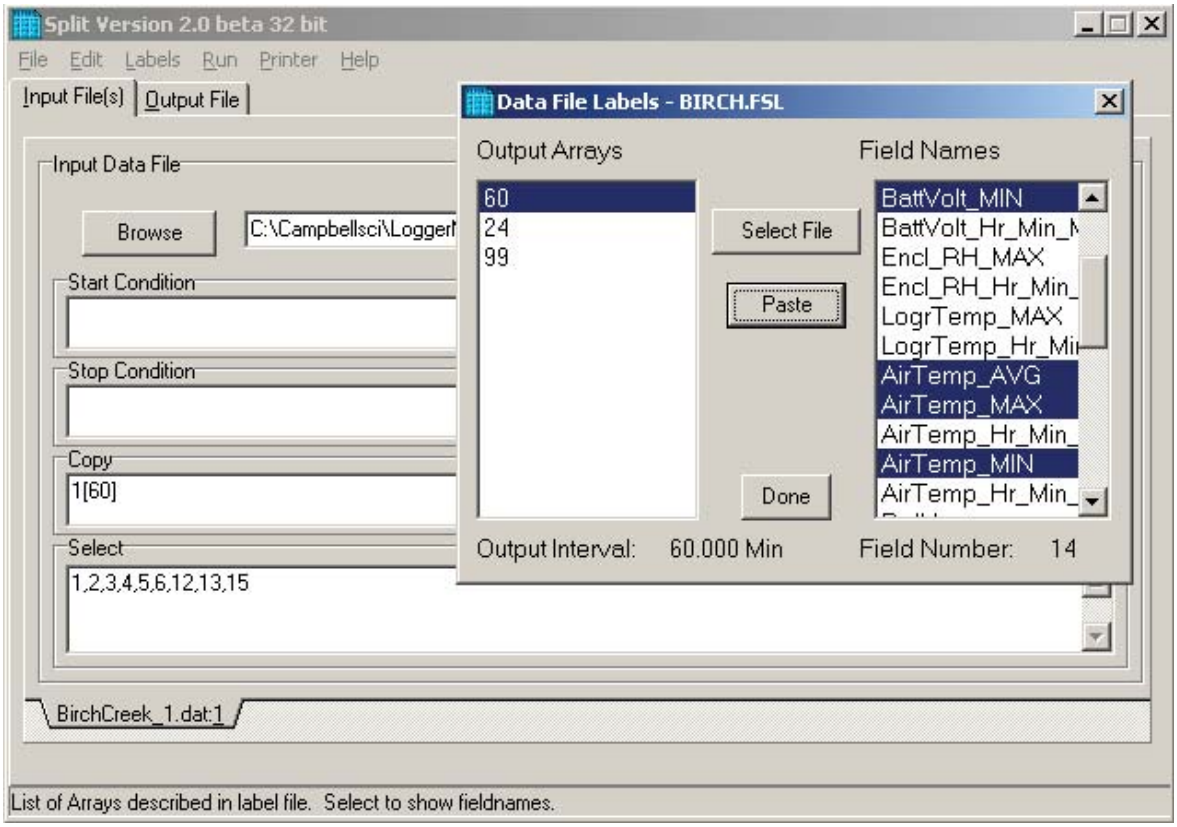
The name of the Input Data File can be typed in or the **Browse** button can be used to select from available files. In this example BirchCreek.dat will be selected as the input data file.

Selecting the data to copy is simplified by the use of the Birch.fsl file. From the toolbar menu, click Labels | Use Data Labels. From the Data File Labels pop-up, Select File is used to find Birch.fsl. When one of the Output Arrays is highlighted, the Field Names of the data in that array are displayed.

**NOTE** In this example, a mixed array data file is processed and the Use Data Labels feature uses an FSL file. When processing a table-based datalogger file, change the file type to "Table-based data file to use for labels" and select the table-based DAT file. Split will use the header information from this file for its labels.




In this example we want the hourly data (note the Output Interval at the bottom of the Data File Label window), so click array 60. To paste the desired values from this array into the Select box, select the field names while holding down the <ctrl> key. All of the values could be selected by clicking the first one and holding the mouse button down, and dragging to the end. Once the values you want have been selected click Paste.

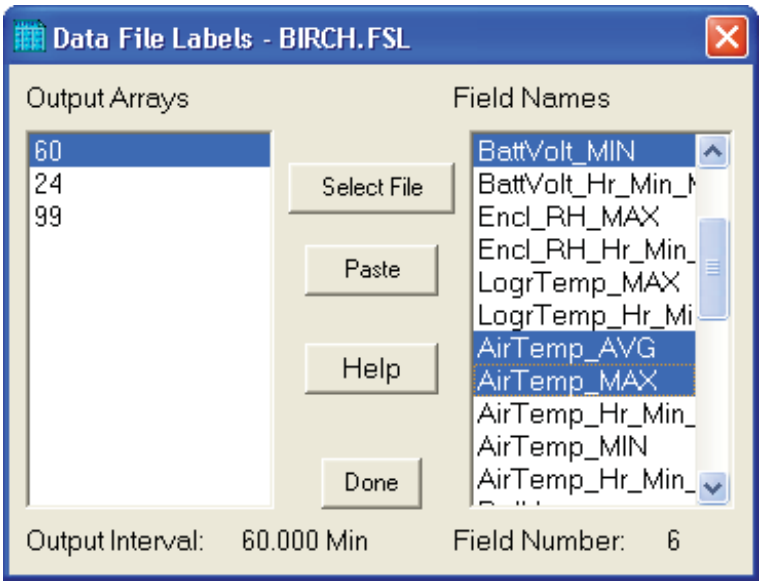


Note that the cursor in the **INPUT FILE(S)** screen must be in valid paste area (Copy or Select). If the cursor is in the File name box or in Start/Stop condition, you will get the error message "Cannot Paste There".

The Paste operation copied the numbers of each of the fields into the Select box. Notice also that it pasted the Array ID into the copy condition: 1[60] tells Split that in order to copy a line of data, the first value in that line must be 60. Split uses the Array ID to discriminate between the hourly and daily data.

Now specify the Output File name. (Without one specified, Split will run and display results but no output file will be created.) Click the **OUTPUT FILE** tab. Type in "hourly" for the name of the output file. By default, Split will use the file extension "PRN", creating the output file: hourly.prn. Depending upon the option chosen in the "If File Exists then" list box, an existing PRN file may be overwritten, appended to, or saved under a new name.

The Labels option from the toolbar can also assist in labeling the output values. Once again, choose LABELS | USE FINAL STORAGE LABELS and select array 60 and all the field names. This time move the cursor to Line 1 of the first column of labels on the **OUTPUT FILE** tab and press Paste. The labels from the final storage file will be pasted into each of the columns. Split will automatically break a label name into multiple rows at the "_" in a label name.




Maximum column heading width is one less than the number entered in the Default Column Width field. However, entering a number in the Width row for the column will set the column width for an individual column. Any FSL labels that are too long for Split column headings will be shown in red. They should be edited before running Split. To edit one of the labels, press the <Enter> key or use a mouse to copy, cut, and paste. A Report Heading can also be entered using the same editing technique.

Report and Column Headings

Report Heading: Hourly Data

| Column# | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|
| Element/Field# | 4 | 5 | 6 | 12 | 13 | |
| Filename | BirchCreek_1.d | BirchCreek_1.d | BirchCreek_1.d | BirchCreek_1.d | BirchCreek_1.d | |
| Line 1 | Hour_Minute | Battery | BattVolt | AirTemp | AirTemp | AirTemp_MIN |
| Line 2 | | Volts | Min | Average | Max | |
| Line 3 | | | | C | C | |
| Decimal | | | | | | |
| Width | 12 | | 12 | | | 12 |

For table based data files the timestamp is normally the first column and is a quoted text string ("2002-02-26 10:30:00"). To display these timestamps in the output you will need to change the column width for the first column to at least 24. If the column width is too small to accommodate the value output, the string will be highlighted in red and preceded by an asterisk, with the words "Bad Data" in the lower right corner when the file is processed.

To run Split, select RUN | GO. The hourly data will be split out and stored in hourly.prn. The results are displayed on the screen as shown below.

**NOTE** When Split is running on large files, the line counters will update only every 1000 lines.

Close the Run window. If you wish to save this parameter file for future reports, choose FILE | SAVE. The file will be saved with a .PAR extension.

Split Run 2.0beta 32bit

| 60 | Year_RTM | Day_RTM | Hour_Minute | Battery Volts | BattVolt Min | AirTemp Average C | AirTemp Max C |
|---|---|---|---|---|---|---|---|
| 60 | 2002 | 42 | 1000 | 13.9 | 13.79 | -12.51 | -7.43 |
| 60 | 2002 | 42 | 1100 | 13.78 | 13.78 | -10.87 | -6.153 |
| 60 | 2002 | 42 | 1200 | 13.79 | 13.59 | -3.309 | -1.261 |
| 60 | 2002 | 42 | 1300 | 13.71 | 13.48 | -1.099 | 2.493 |
| 60 | 2002 | 42 | 1400 | 13.82 | 13.49 | -1.355 | 4.375 |
| 60 | 2002 | 42 | 1500 | 13.61 | 13.52 | -2.932 | .077 |
| 60 | 2002 | 42 | 1600 | 13.59 | 13.53 | -1.673 | .067 |
| 60 | 2002 | 42 | 1700 | 13.8 | 13.56 | .724 | 2.963 |
| 60 | 2002 | 42 | 1800 | 13.88 | 13.59 | -2.677 | .405 |
| 60 | 2002 | 42 | 1900 | 13.99 | 13.75 | -4.235 | -2.941 |
| 60 | 2002 | 42 | 2000 | 13.87 | 13.8 | -6.903 | -5.224 |
| 60 | 2002 | 42 | 2100 | 14.05 | 13.82 | -8.06 | -7.3 |
| 60 | 2002 | 42 | 2200 | 14.07 | 13.85 | -7.37 | -6.424 |
| 60 | 2002 | 42 | 2300 | 14.09 | 13.86 | -7.1 | -6.189 |
| 60 | 2002 | 42 | 2400 | 14.12 | 13.88 | -9.6 | -8.58 |
| 60 | 2002 | 43 | 100 | 14.16 | 13.92 | -12.5 | -11.13 |
| 60 | 2002 | 43 | 200 | 14.16 | 13.96 | -12.09 | -10.89 |
| 60 | 2002 | 43 | 300 | 14.13 | 13.93 | -11.14 | -9.74 |
| 60 | 2002 | 43 | 400 | 14.12 | 13.92 | -9.69 | -8.87 |

Lines Read 953

Lines Written 189

Pause Stop Close

# 5.3 Split Parameter File Entries

## 5.3.1 Input Files

The name of the Input File is entered in the space to the right of the **Browse** button. The default directory is the working directory for Split (if the default installation directories were chosen, this will be c:\campbellsci\splitw). If the input file is not in the default directory, use the **Browse** button to find the input file.

In LoggerNet or PC400, mixed array datalogger files are stored in a simple comma separated ASCII format; tabled-based datalogger files are stored in TOA5 (a comma separated format with headers). Split can process Input files from other software, but they must be formatted in Comma Separated ASCII, Final Storage (Binary) Format, Field Formatted ASCII (Split default output format), Printable ASCII, Table Oriented ASCII (TOACI1 or TOA5) or Raw A/D data (refer to special Burst Mode instruction in your Campbell Scientific datalogger manual).

Files stored in Table Oriented Binary (TOB) format are converted to Table Oriented ASCII files when Split uses them. The converter runs in the background when you run Split to create the output file. You cannot use the Data Label browser to select the columns of data from a binary file. If you want to use the Data Label browser you can open the file first using View Pro, which converts the binary file to ASCII and saves it under a new name, prior to processing it with Split.

Split's default output file, a field-separated ASCII format with a *.PRN file extension, can be processed a second time if desired.

TABLE 5-1 provides an example of Comma Separated, Field Formatted, Printable ASCII, and Table Oriented ASCII input file types. The data in the various formats are identical. Each line of data represents an "Output Array", starting with an Output Array ID (in this case 115). Each data point in the Output Array is referred to as an "element". The element number is given in the Printable ASCII format, and implied in the other formats. Data presented in TABLE 5-1 is used for example purposes in the following sections.

**TABLE 5-1. Comma Separated, Field Formatted, Printable ASCII, and Table Oriented ASCII Input File Format Types**

COMMA SEPARATED

115,189,1200,89.6,55.3,25.36,270
115,189,1300,91.3,61.5,27.25,255.4
115,189,1400,92.7,67.7,15.15,220.1
115,189,1500,94.1,69,20.35,260.6

FIELD FORMATTED

| | | | | | | |
|---|---|---|---|---|---|---|
| 115 | 189 | 1200 | 89.6 | 55.3 | 25.36 | 270 |
| 115 | 189 | 1300 | 91.3 | 61.5 | 27.25 | 255.4 |
| 115 | 189 | 1400 | 92.7 | 67.7 | 15.15 | 220.1 |
| 115 | 189 | 1500 | 94.1 | 69 | 20.35 | 260.6 |

```
PRINTABLE ASCII

01+0115  02+0189  03+1200  04+089.6  05+055.3  06+25.36  07+270.0
01+0115  02+0189  03+1300  04+091.3  05+061.5  06+27.25  07+255.4
01+0115  02+0189  03+1400  04+092.7  05+067.7  06+15.15  07+220.1
01+0115  02+0189  03+1500  04+094.1  05+069.0  06+20.35  07+260.6

Element 1    =    Output Array ID# (115)
Element 2    =    Julian day (189)
Element 3    =    hour, minute
Element 4    =    average temperature in deg.  F
Element 5    =    average soil temperature in deg.  F
Element 6    =    average wind speed in mph
Element 7    =    wind direction in degrees

TABLE ORIENTED ASCII

"TOACI1","CR10T","15Minute"
"TMSTAMP","RECNBR","TCTempF_MAX","BattVolt_MIN"
"2002-02-26 10:30:00",0,73.97,13.99
"2002-02-26 10:45:00",1,74.03,13.98
"2002-02-26 11:00:00",2,74.53,13.98
"2002-02-26 11:15:00",3,74.82,13.98
"2002-02-26 11:30:00",4,75.23,13.98

Element 1    =    Timestamp
Element 2    =    Record Number
Element 3    =    temperature in degrees F
Element 4    =    minimum battery voltage
```

A maximum of eight input files may be processed by Split at one time. Additional input files are added using the EDIT | ADD DATA FILE menu option. Split looks for a file extension of .DAT if no extension is specified. If the Input File does not exist, an error message is displayed when RUN | GO is selected from the menu options.

For instance, to process two files named TEST.DAT and TEST_1.DAT the user would select TEST.DAT and TEST_1.DAT as Input Files. Two blank input file templates will be generated. To change from one template to the other, click the appropriate tab on the bottom of the screen. Both templates must be completed before Split will process the data. To merge different output arrays from the same input file into one array, open the data file once for each different array.

#### 5.3.1.1 File Info

In most instances, Split automatically recognizes the type of data file it is reading when using Auto Detect in the File Info field. However, there are two exceptions for which you should choose the appropriate option manually:

- **Reading Raw A/D Data from Burst Measurements**

  To read this type of data and convert it to ASCII, select Burst Format in the File Info box. Once Burst Format is selected, the Number of Values in Each Burst window in the Offset Menu will become accessible. Enter the number of elements in each Burst. This number does not include the array ID number or calibration data.

- **Reading Data in Final Storage (Binary) Format**

  If the data is in binary format and Start and Stop Offsets are used, Final Storage (Binary) Format must be selected in the File Info field. This tells Split that the file must be decoded as Final Storage before counting the bytes. If Offsets are not used, Auto Detect may be chosen and the file will be processed correctly.

### 5.3.1.2 File Offset/Options

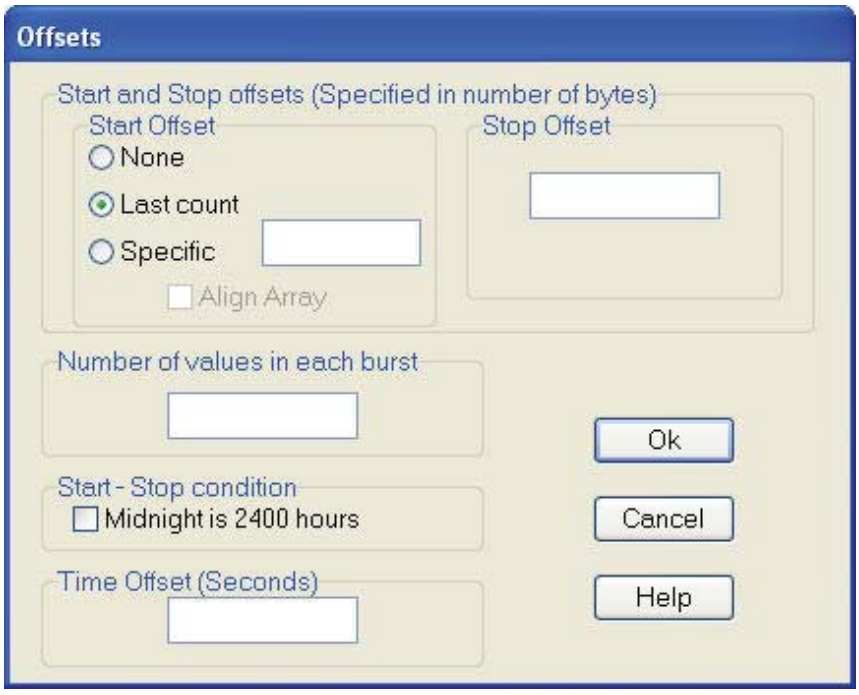
***Start Offset***

***None***

Select his check box to start reading the input file from the beginning.

***Last Count***

Each time Split runs a parameter file, it keeps track of the number of bytes it read from the input file and saves this information in the parameter file. Split can then start where it last left off. This is done by clicking the **Offsets** button and selecting the Last Count option. This feature may be used to process only the new data from a file in which new data are being appended periodically to the data file.




**CAUTION**

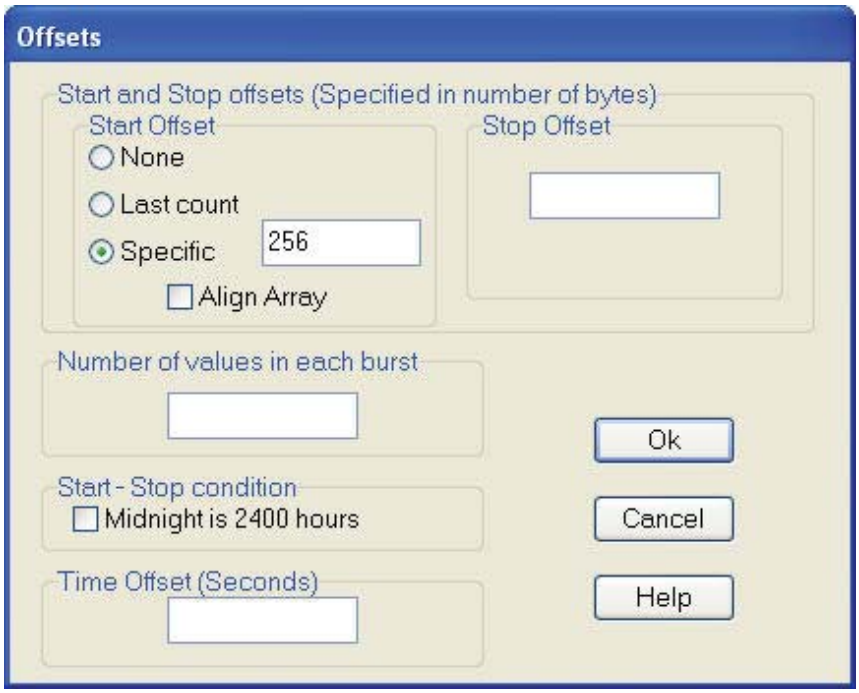
When using the Last Count option, if the Start and Stop Conditions are specified, they must exist in the newly appended data or Split will never begin execution.

Because Last Count keeps track of the number of bytes in the file, if you delete data from the beginning of a file, Last Count will not work properly.

### *Specific*

By selecting the Specific option and entering a number, Split will "seek" that position in the file. This option saves time by starting (or stopping) part way through a large data file. The number specifies the number of bytes into the file to seek before processing data. A positive or negative number can be entered. If the number is positive, Split will start reading from the beginning of a file; if the number is negative, Split will start reading from the end of a file. All characters, including spaces, carriage returns, and line feeds, are counted.

In the following figure, Split will skip the first 256 bytes of data before it begins processing the data in Input File.



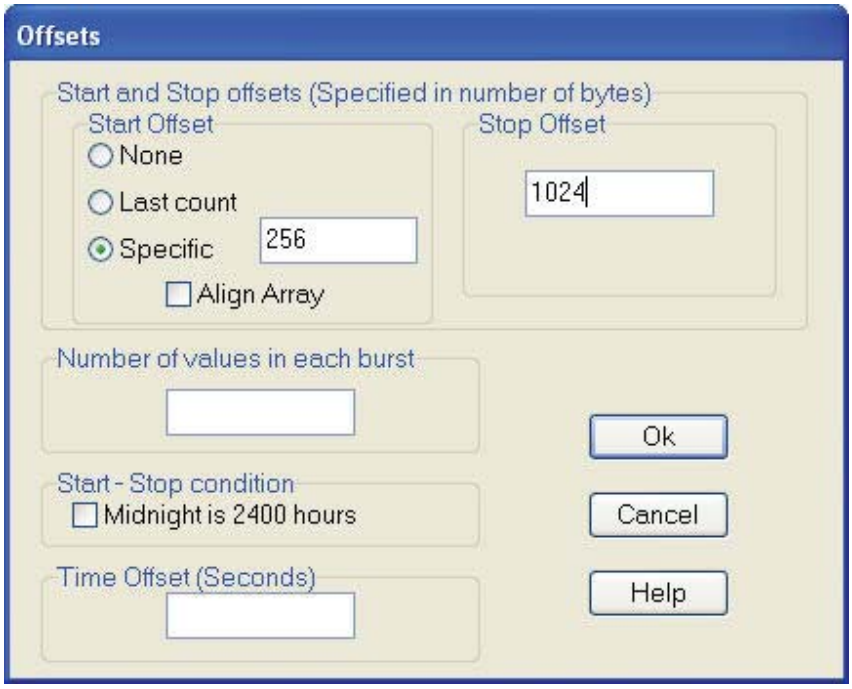

### *Align Array*

When using a specific start offset, the number of bytes specified may cause Split to seek to the middle of a row. Selecting the Align Array check box will cause Split to begin processing at the beginning of the next row.

### Stop Offset

This number specifies the number of bytes from the beginning of the file that Split should stop processing the data file.

In the following figure, Split will skip the first 256 bytes of data before beginning and stop execution on byte 1024.




### Number of Values in Each Burst

When processing a burst data file, enter the total number of values recorded for each Burst (this is the number of burst scans multiplied by the number of channels per scan). This number does not include the array ID or calibration data.

To break the results into a column for each channel, enter the number of channels for the Break Arrays value (Output File Tab, Other button).

### Midnight is 2400 hours

When programming mixed-array dataloggers, the Real Time instruction (P77) has two different options for the midnight time stamp: midnight = 2400 of the day just ending or midnight = 0000 of the day just beginning.

When processing mixed-array data files using time synchronization, select this check box if the time stamp is midnight at 2400 of the day just ending. This will ensure that Split processes the data file correctly.

### Time Offset

This field specifies a time offset, in seconds, that should be applied to each item on the Select line that uses the Date or Edate function to output a date. The offset can be positive or negative. Each input file can have its own offset (or no offset) for its Select line.

For example, with an input timestamp of "2008-10-09 10:25" and an offset of 3600, the timestamp output by Date ("yyyy-mm-dd hh:nn";1;1;1;1) would be "2008-10-09 11:25".

This may be useful when adjusting for different time zones.

**NOTE** The offset will not be applied to Date and Edate functions with only two parameters. (The two-parameter mode is backwards compatible with the original Date and Edate functions used in older versions of Split.)

### 5.3.1.3 Start Condition

A starting point may be specified to begin processing data. If the Start Condition field is left blank, Split will start processing data at the beginning of the data file. The starting point can be any element within the array or a combination of elements within an array.

**NOTE** The font for Start Condition, Stop Condition, Copy, and Select can be changed from the Options Menu.

The syntax can be expressed as:

$$e_i[val_i]$$

where $e_i$ = the position number of the element within the array

$val_i$ = the value of that element.

For example, the data in TABLE 5-1 contains seven elements per Output Array, representing hourly data. Assume that this data file contains one month of hourly data. To start processing data at 1500 hours on the first day, the Start Condition is expressed as 3[1500], where 3 means the third element within the array and 1500 is the value of that third element.

The element must match this start value exactly to trigger the start condition. However, when starting based on time, you can enable the "Start-Stop On/After Time" function to trigger the start of processing when the exact time is found or at the first instance of data after that time has occurred. This option is found on the Output tab, Other button.

**NOTE** Table data files contain the time and date as a single quoted string at the beginning of each data record. Split handles the dates as long as you include a colon separator as a placeholder for each of the fields in the timestamp. 1[Year]:1[Day of Year]:1[Time of Day]:1[Seconds]

See the examples below:

:1[60]:: Day of Year 60

1[2002]:1[60]:1[1250]: Year 2002, Day of Year 60, Time of Day 12:50

::1[1445]:1[30] Time of Day 14:45, Seconds 30

Logical “and” and “or” statements can be used when specifying the Start Condition. A logical “and” statement means that all conditions must be true for the statement to be true. Up to three conditions can be connected with “and” statements. If too many “and” statements are used, an error message will be displayed when you run Split.

The logical “or” statement means that if *any* of the conditions are true, then the statement is true. Split allows up to six conditions to be connected with “or” statements. Additionally, each “or” statement can contain up to three “and” conditions. As with the “and” statements, if the maximum number of valid statements is exceeded, an error message will be displayed.

These rules for logical statements also apply to the Stop and Copy Conditions.

An example of a simple logical “and” statement follows:

2[189]and3[1200]

Element two (the Julian day) must equal 189, and element three (the time in hours/minutes) must equal 1200.

If the following “and” statement was used:

2[189]and3[1200]and4[92]and5[67]

an error would be returned because the maximum number of allowable “and” statements has been exceeded.

A range can be specified for $val_i$ by putting “..” between the lower and upper limit. For example:

2[189]and7[200..275]

In this example two conditions must be satisfied to start processing data. First, the day of year must be 189, and second, element 7 must be between 200 to 275 degrees, inclusive.

#### *5.3.1.3.1 Starting Relative to PC Time*

Split has the ability to start relative to the current PC TIME (computer time). This feature allows a .PAR file to be run on new data files without changing the Start Conditions, provided the Input Data File is collected at a fixed interval and Split is run at a fixed interval. For example, the same PAR file could be run every day to display the last 48 hours of data without changing the start conditions. For example, using a table based data file:

Start Condition = 1:1[-1]:1[1200]:1:

In this instance, Split will begin processing data when the date for both files is one less than the current date (1:**1[-1]**:1[1200]:1:)and the time is 1200 (1:1[-1]:**1[1200]**:1:).

As an expanded example, assume that LoggerNet or PC400 is used to append data to an archive file. SplitR is executed using a desktop shortcut. In this case the frequency of data collection and data reduction is the same. Time values in the data file (day, hrmn, sec.) are different each time the data are collected, but

by telling Split where to Start reading relative to the PC clock, the Start Conditions do not need to be changed. To accommodate variations in the data collection and reduction frequencies, an interval in minutes or seconds may be specified as shown in the examples below.

**2[-0]:3[-60,5]** tells Split to start at a timestamp in the data that is between 55 and 65 minutes prior to the current PC time (the closest 5 minute interval of the current day that is less than the PC time minus 60 minutes). If you are processing data stored at the top of the hour and the PC time is 1404, Split calculates 1304 and looks for hour 1300 to start reading.

**2[-3]:3[-120,60]** tells Split to find the closest 60 minute interval that is less than the PC time minus 3 days and 2 hours. If the PC time is the day of year 159, hour 0017, Split will start reading on data output at 2200 hours on day 155.

**2[-3]:3[-120]:4[20,5]** tells Split to find the closest 5 second interval that is less than the PC time minus 3 days, 2 hours and 20 seconds. If the PC time is 27 seconds after noon on day 30, Split will begin reading on data output at 1000 hours and 05 seconds on day 27.

Split can also begin processing a file on a particular month and day. Use the syntax :E[Month%Day]::, where E is the element that contains the Julian Day, and Month and Day are either constants or a value related to PC time. For example:

**:2[-1%1]::** tells Split to begin processing on the first day of the previous month.
**:2[-0%15]::** tells Split to begin processing on the fifteenth day of the current month.
**:2[5%1]::** tells Split to begin processing on May 1.

This function can be used in both the Start and Stop conditions. It provides a simple way to create a monthly report. For additional information, refer also to Section 5.3.1.15.2, *Using Time Synchronization While Starting Relative to PC Time* *(p. 5-38)*.

**CAUTION**

Split will not start reading if the exact specified starting time cannot be found, unless you enable the "Start-Stop On/After Time" feature. The interval (5 minutes, 60 minutes, and 5 seconds in the examples above) must be evenly divisible into 60 minutes.

**NOTE**

• If the start time is a certain number of days prior to the PC time, the file will be processed beginning at midnight of the day specified.

• To specify a start time in minutes from the current PC time, you must also specify a day parameter of [-0]. Otherwise, processing will begin at the first instance in the data file that the minutes parameter equals the current minutes.

### 5.3.1.4 Stop Condition

The Stop Condition specifies when to stop processing data. This feature allows segments of data to be removed from large data files. For instance, if a data file contains one month of data and just one day is desired, the start and stop values allow the user to get just that day's data.

The Stop Condition is expressed with the same syntax as the Start Condition. If the Stop Condition parameter is left blank, Split will execute until the end of the file. As with the Start Condition, logical "and" and "or" statements can be used when specifying the Stop Condition (Section 5.3.1.3, *Start Condition* *(p. 5-13)*), as well as stopping based on PC time.

The array or record containing the Stop Condition is not included in the output file. If the stop value is not found, Split will display a dialog box that gives the option to select a new file and continue processing the data. This feature is useful when data are contained in more than one data file.

The "Start-Stop On/After Time" function can be used with a Stop Condition. This will stop processing of the file when the exact time is found or at the first instance of data after that time has occurred. This option is found on the Output tab, Other button.

The C and F commands alter the meaning of the Stop Condition.

#### *5.3.1.4.1 "C" Option: Formatting Event Tests Containing Conditional Output Arrays*

The C option is used to combine data from two or more conditional arrays onto one Split output line. A conditional array is one that is only output when a defined event occurs.

Assume that two or more conditional Output Arrays with unique Output Array IDs compose a test period, followed by an unconditional Output Array that defines the end of a test. The unconditional "end of test" Output Array is at the end of each test, but the conditional Output Arrays may or may not be present. The data file is comprised of several of these tests.

As an example, let's look at a vehicle test application. The start of the test is when the vehicle is turned on, and the end of the test is when the vehicle is turned off. The conditional output arrays could be:

- monitoring the engine temperature and outputting data to a unique array when the temperature exceeds a limit
- outputting data to a unique array when the brakes are applied
- outputting data when engine RPM exceeds a limit

The unconditional array data (the stop condition) would be output to a unique array when the engine is turned off. By processing the data with Split using the C option, the data collected during each test could be merged on to one line, with blanks inserted if a set of data didn't exist (e.g., if the engine temperature never exceeded the defined limit).

- An Input File must be set up for each array ID in the test. The first Input File is configured on the Input File tab that appears when you open Split. Additional Input Files are added by choosing Edit | Add Data File from the

Split menu. The same data file will be used as the Input File for each array.

- Type in the array ID in the Copy field of the Input File tab for each array. The array ID is the first element of a data file, so the line should read 1[*123*], where *123* is the actual array ID you want to process.

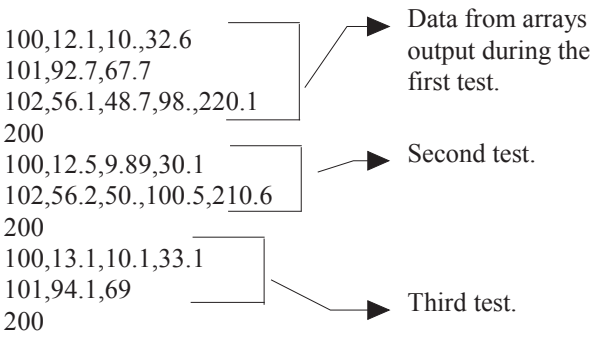
- In the Select field, type in the number for each element (data value) you want to be output in the report.

- In the Stop Condition field, type in a "C," followed by the ID of your stop condition array. If your "end of test" array was array ID 200, the Stop Condition field would read: C,1[200]. This should be typed into the Stop Condition fields of each array, including the "end of test" array.

Set up the Output File as you would for any Split process. If you are including column headings, the arrays and elements will appear in the order they are listed on the Input File tabs. That is, the first column will be Input File number 1, element number 1; the next column is Input File number 1, element number 2… Input File number 2, element number 1 follows in the column immediately after the last element of Input File number 1.

Consider TABLE 5-2 below:

**TABLE 5-2. Example of Event Driven Test Data Set**




This table contains four different output arrays: 100, 101, 102, and 200. During the first test, data was output from all three conditional arrays (100, 101, and 102), with 200 signaling the end of the test. During the second test, data was output from arrays 100 and 102. During the third test, data was output from arrays 100 and 101.

To process these files using the C option, the parameter file would be set up as follows (assuming the name of our data file is Data_1.DAT):

First Input File = Data_1.DAT:1
Stop condition = C,1[200]
Copy = 1[100]
Select = 1,2,3,4

Second Input File = Data_1.DAT:2
Stop condition = C,1[200]

Copy = 1[101]
Select = 1,2,3

Third Input File = Data_1.DAT:3
Stop condition = C,1[200]
Copy = 1[102]
Select = 1,2,3,4,5

Fourth ("end of test") Input File = Data_1.DAT:4
Stop condition = C,1[200]
Copy = 1[200]
Select = (leave blank)

**NOTE** The *:(number)* after the data file name is inserted automatically by Split.

**TABLE 5-3. Processed Data File Using Option C**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 12.1 | 10 | 32.6 | 101 | 92.7 | 67.7 | 102 | 56.1 | 48.7 | 98 | 220.1 |
| 100 | 12.5 | 9.89 | 30.1 | | | | 102 | 56.2 | 50 | 100.5 | 210.6 |
| 100 | 13.1 | 10.1 | 33.1 | 101 | 94.1 | 69 | | | | | |

When Split is run, the resulting data file will look similar to TABLE 5-3. Each line of data represents one test. Notice that blanks were inserted if the data set (conditional array) did not exist.

#### 5.3.1.4.2 Trigger on Stop Condition (F Option) Output of Time Series

The Trigger on Stop Condition, or F option, changes the function of the Stop Condition when one or more Time Series functions (Section 5.3.1.11, *Time Series Functions, Details, and Examples* *(p. 5-24)*) are contained in the Select field. When a Stop Condition is met, the time series data is calculated and written to the output file. However, instead of stopping at this point, processing resumes and time series data is output the next time the Stop Condition is met. This continues until the end of file or until the user stops Split manually.

The Trigger on Stop Condition is enabled by clicking **Other…** on the Output Tab and checking the box next to the Trigger on Stop Condition field. When the Trigger on Stop Condition is enabled, the function affects all files being processed that have a Stop Condition specified. If multiple files are being processed but it is desired that the function affect one or more—but not all—of the files, the F option is used in the Stop Condition field of the files that you want processed using the function. The syntax for the F option is: $F,e_i[val_i]$.

A typical application for the Trigger on Stop Condition is to reduce days of hourly data into daily summaries. A logical element to use for the Stop Condition is time (hrmn). Assuming the third element of the hourly Output Array is hrmn, and midnight is output as 0, the Stop Condition is entered as 3[0] (or F,3[0] if the F option is used). The Time Series processing is performed over a day defined by midnight to midnight.

If only hourly Output Arrays were contained in the Input File, the Copy line could be left blank. If other Output Arrays are present which need not be

included in the Time Series processing, a logical Copy condition would be the Output Array ID of the hourly output.

The Trigger on Stop Condition functions the same for multiple Input files as it does for a single Input File. If the option is enabled on several Input Files, and the Stop Conditions do not occur at the same point in each file, when a file's Stop Condition is met, its time series data are output and blanks are output for data selected from the other Input Files.

Say, for example, that you were interested in the average value of the first data point (element 2) for each test, in the data set listed in TABLE 5-2. The Input File template would look like that shown in TABLE 5-4.

**TABLE 5-4. Input File Entries to Process the First Data Point for each Test**

| | |
|---|---|
| First Input File = | DATA_1.DAT:1 |
| Stop Condition = | F,1[200] |
| Select = | AVG(2) |

### 5.3.1.5 Copy

The Copy Condition tells Split which arrays should be used for the output data. After the Start Condition is satisfied, and before the Stop Condition is met, the Copy condition must be satisfied before any data will be processed according to Select line instructions. If the Copy condition is left blank, all arrays are processed between the Start and Stop values. Syntax for the Copy condition is similar to the Start and Stop values mentioned above. Logical "and" and "or" statements (see Section 5.3.1.3, *Start Condition* (p. 5-13)) can be used when specifying the Copy condition.

For example, referring to TABLE 5-1, if only those hours during day 189 when the temperature was above 90 and the soil temperature was below 62 is desired, or, during day 189 when the average wind speed was below 21 while the wind direction was between 255 to 265 is desired, the Copy condition would be:

1[189]and4[90..150]and5[0..61.99]or1[189]and6[0..20.99]and7[255..265]

Only Output Arrays with hours 1300 and 1500, TABLE 5-1, conform to the above Copy conditions.

**NOTE**

The Copy Condition is used almost exclusively for mixed-array dataloggers, except when time-syncing two or more data files. See Section 5.3.1.15, *Time Synchronization* (p. 5-36), for additional information.

**Time Ranges**

When specifying a Copy condition, a range of time values can be specified instead of a single time. If the element being tested falls within the range, the Copy condition is satisfied and the data is processed. A range is indicated by entering two periods between the first and last values of the range.

***Examples:***

**Table-based**

With an entry of **1:1:1[600..1200]:1** in the Copy condition, Split will only process the data file when the time is between 6:00 a.m. and 12:00 p.m.

(Since the timestamp for table-based dataloggers is all one string, each portion of the timestamp (year, day, hour/minute, seconds) will use the same element number. Colons are used to separate each portion. The format is 1[year]:1[day]:1[hhmm]:1[seconds] (the number 1 was used since, typically, the timestamp is the first element in the data string). In this format, hhmm is the four-digit hour/minute.)

**Array-based**

With an entry of **1[30] and 2:3:4[600..1200]:** in the Copy condition, Split will only process the data file when the time is between 6:00 a.m. and 12:00 p.m.

(This assumes 2 is the year element, 3 is the day element, and 4 is the hour/minute element.)

**NOTE** Time ranges cannot be used with the time-sync function.

### 5.3.1.6 Select

The Select line specifies which elements of an Output Array are selected for processing and/or output to the specified Output File. The Select line becomes operable only after the Start Condition and Copy condition are met, and before the Stop Condition is satisfied. If the Select line is left blank, all elements in output arrays meeting the Start Condition and Copy conditions are output to the Output File.

Processing is accomplished through arithmetic operators, math functions, spatial functions, and time series functions.

### 5.3.1.7 Ranges

Element numbers may be entered individually (e.g., 2,3,4,5,6,7), or, in groups (e.g., 2..7) if sequential. Range limits (lower to upper boundary conditions) may be placed on elements or groups of elements specified in the Select or Copy lines. For example, 3[3.7..5],4..7[5..10] implies that element 3 is selected only if it is between 3.7 and 5, inclusive, and elements 4,5,6, and 7 must be between 5 and 10, inclusive.

If range limits are used in the Select condition, when Split is run, any data which are outside of the specified range will be highlighted according to the options chosen for the output file. TABLE 5-5 summarizes what each option produces on the screen and in the output file if out of range data are encountered. This type of range testing is a quick way to identify data problems.

**TABLE 5-5. Effects of Out of Range Values for Given Output Options**

| Output Option | Screen Display* | PRN File | RPT File or Printer Output |
|---|---|---|---|
| Report = None; No other options defined (default) | bad values displayed in red and preceded by asterisk; the text "bad data" highlighted in a red box at bottom right of screen | blanks inserted for bad values | N/A |
| Report = File or Printer; no other options defined | bad values displayed in red and preceded by asterisk; the text "bad data" highlighted in a red box at bottom right of screen | blanks inserted for bad values | bad values preceded by asterisk |
| Report = None; replacement text (abc) in "Replace bad data with" field | bad values displayed in red and preceded by asterisk; the text "bad data" highlighted in a red box at bottom right of screen | abc inserted in place of bad values | N/A |
| Report = File or Printer; comment in "Replace bad data with" field | bad values displayed in red and preceded by asterisk; the text "bad data" highlighted in a red box at bottom right of screen | comment inserted in place of bad values | bad values preceded by asterisk |
| Report = None; "Display only bad data" option enabled | only lines with bad data are displayed; bad values displayed in red and preceded by asterisk; the text "bad data" highlighted in a red box at bottom right of screen | only lines with bad data output; blanks inserted for bad values | N/A |
| Report = File or Printer; "Display only bad data" option enabled | only lines with bad data are displayed; bad values displayed in red and preceded by asterisk; the text "bad data" highlighted in a red box at bottom right of screen | only lines with bad data output; blanks inserted for bad values | only lines with bad data output; bad values preceded by asterisk |

*The Screen Display box must be checked; if not, no data will be displayed on the Split Run screen.

**NOTE** In this instance, out of range data refers to data outside of the specified output range. It is not to be confused with out of range data generated by the logger.

### 5.3.1.8 Variables

Variables can be assigned names in the Select line. For example, x = 4-5∗(6∗ 3.0) means that x is equal to element 6, times the number 3, times element 5, subtracted from element 4. A numeric value is distinguished from an array element by the inclusion of a decimal point. Variables must be declared before they can be used in the Select line. A variable name must start with an alpha character, can include numbers and must not exceed eight characters. Variable names can start with the same character but they must not start with another complete variable name (e.g., the variable XY is not valid if there is also the variable X). A comma must follow each variable statement, as with all

parameters in the Select line. Once the variables have been declared they can be used later in the Select line (i.e., x=4-5∗(6∗3.0), y=6/3,2,3,6,7,7∗x,6+y).

**NOTE** Variables can be defined in the **first four Input File's Select lines** only, but may be used in subsequent Input File's Select lines.

Illegal operations (e.g., logarithm of a negative number) will cause Split to store blanks for the Output. It is possible to get a run time error (error 0/1) if the floating point math exceeds the limits of the PC.

### 5.3.1.9 Numerical Limitations

The greatest number that can be output is determined by the field width (Output File tab). If the width is eleven or greater, the maximum number is 99,999,999; for widths from eight through ten the maximum is 99,999; for widths less than eight the maximum is 9999. If a column is not large enough for a value, it will be stored as a 9,999, 99,999 or 99,999,999 based on the column width. In some instances, such as when a column is not large enough for the date function, you will see the text "bad data" on the Split Runtime window.

### 5.3.1.10 Mathematical Functions, Details, and Examples

**TABLE 5-6. Split Operators and Math Functions**

| OPERATORS | | OPERATOR PRECEDENCE ORDER (3 = high, 1 = low) |
|---|---|---|
| ^ | = raise to the power | 3 |
| x Mod y | = Modulo divide of x by y | 2 |
| ∗ / | = multiplication, division | 2 |
| + - | = addition, subtraction | 1 |

**EXAMPLES OF SYNTAX FOR MATHEMATICAL OPERATORS**

| | |
|---|---|
| 3∗5 | multiply element 3 by element 5 |
| 3/5 | divide element 3 by element 5 |
| (3..5)/(8..10) | same as 3/8, 4/9, 5/10 |
| 3+5 | add element 3 to element 5 |
| 3-5 | subtract element 5 from element 3 |
| (3,9,5)-(8,7,10) | same as 3-8, 9-7, 5-10 |
| 3∗2.0 | multiply element 3 by a fixed number 2 |
| 2^3.0 | raise element 2 to the third power |

**MATH FUNCTIONS**

| | |
|---|---|
| Abs(x) | = Absolute value of x |
| Arctan(x) | = Arc tangent of x (in degrees) |
| Cos(x) | = Cosine of x (in degrees) |
| Exp(x) | = Natural Exponent function ($e^x$) |
| Frac(x) | = Fractional portion of x |
| Int(x) | = Integer portion of x |
| Ln(x) | = Natural logarithm of x |
| Sin(x) | = Sine of x (in degrees) |
| SpaAvg(x..y) | = Spatial average of elements x through y |
| SpaMax(x..y) | = Spatial maximum of elements x through y |
| SpaMin(x..y) | = Spatial minimum of elements x through y |
| SpaSd(x..y) | = Spatial standard deviation of elements x through y |
| Sqrt(x) | = Square root of x |

The following array of ASCII data will be used for all Mathematical function examples.

0105 0176 1200 -07.89 55.10 12.45 270.5

**Abs(x)** returns the absolute, or positive value of element x.
Examples:
Abs(4) = 7.89
Abs(4∗5) = 434.74

**Arctan(x)** returns the arc tangent of element x in degrees.
Examples:
Arctan(7) = 89.788
Arctan(7/6) = 87.365

**Cos(x)** returns the cosine of element x in degrees.
Examples:
Cos(5) = .57215
Cos(5-6) = .73551

**Exp(x)** returns the exponential base e to the power of element x.
Example:
Exp(4) = .00037

**Frac(x)** returns the fractional value of the element x.
Examples:
Frac(4) = -.89
Frac(6+7) = .95

**Int(x)** returns the integer portion of the element x.
Examples:
Int(7) = 270
Int(5∗6) = 685

**Ln(x)** returns the natural log of element x.
Examples:
Ln(6) = 2.5217
Ln(7/6∗5/1) = 2.4337

**Sin(x)** returns the sine of element x in degrees.
Examples:
Sin(7) = -.99996
Sin(7-2+5) = .50603

Spatial functions, included under Mathematical functions, operate on a per Output Array basis. The average, maximum, minimum, and standard deviation of a specified group of elements within an array are calculated.

**SpaAvg(x..y)** returns the spatial average of elements x through y.
Examples:
SpaAvg(1..7) = 258.74
SpaAvg(1,4,7) = 122.54

**SpaMax(x..y)** returns the maximum value of elements x through y.
Examples:
SpaMax(1..7) = 1200
SpaMax(1,2,5) = 176

**SpaMin(x..y)** returns the minimum value of elements x through y.
Examples:
SpaMin(1..7) = -7.89
SpaMin(1,2,5) = 55.1

**SpaSd(x..y)** returns the standard deviation of elements x through y.
Examples:
SpaSd(1..7) = 394.57
SpaSd(5,2,1) = 49.607

**Sqrt(x)** returns the square root of element x.
Examples:
Sqrt(3) = 34.641
Sqrt(3^ 2.0) = 1200

### 5.3.1.11 Time Series Functions, Details, and Examples

**TABLE 5-7. Time Series Functions**

| **TIME SERIES FUNCTIONS** | |
|---|---|
| Avg(x;n) | = Average |
| Blanks(x;n) | = Number of blanks in element |
| Count(x;n) | = Number of data points in element |
| Max(x;n) | = Maximum |
| Min(x;n) | = Minimum |
| RunTotal(x;n) | = Running total |
| Sd(x;n) | = Standard deviation |
| Smpl(x;n) | = Sample raw value |
| SmplMax(x;y;n) | = Sample (y) on a maximum (x) |
| SmplMin(x;y;n) | = Sample (y) on a minimum (x) |
| Total(x;n) | = Totalize |
| WAvg(x;n) | = Unit vector mean wind direction (in degrees) |

**NOTE:** x can be an element or a valid expression. n is optional and is the number of arrays to include in the function. Date and Edate can be used for the "n" in the Time Series functions to produce monthly output (see TABLE 5-8 Special Functions).

Time Series functions are used to perform vertical processing on selected elements, such as calculating the average of an element over a specified range of data. Time Series results are output in three instances:

1. when a Trigger on Stop Condition (F option) is met

2. at the end of a data file (or within a range specified by Start and Stop Conditions)

3. when an interval count is met

When the Trigger on Stop Condition (or F option) is used, any time series data defined in the Select line is output each time the Stop Condition is met. Refer

to Section 5.3.1.4.2, *Trigger on Stop Condition (F Option) Output of Time Series* *(p. 5-18)*, for more information on the Trigger on Stop Condition.

Results which are output at the end of a file or a range of data are referred to as Final Summaries. A typical select line that would produce a Final Summary is:

1,2,3,4,Avg(4)

This line would output values for elements 1 through 4 each time an array was output. Additionally, an average value for element 4 would be calculated for the entire file and output as the last line of data in the output file.

1,2,3,4,Avg(4;24)

This line would output values for elements 1 through 4 each time an array was output, and an average value for element 4 would be calculated every 24$^{th}$ array and output as an additional column in the file. An additional summary would occur for an Interval Count if the count was not evenly divisible into the number of output arrays present in the Input File. The summary, in this case, is calculated from an incomplete interval count.

The date( ) function can be used for the interval in a time series function to produce monthly output. Refer to the Monthly summary example in Section 5.3.1.12, *Special Functions, Details, and Examples* *(p. 5-29)*.

**NOTE**

When Date and Edate are used within other functions they must be used with the older format Date(doy;y) and Edate(doy;y) instead of using the extended date functions. For example AVG(1;Date(2;2002.0)). The decimal is needed to indicate a fixed number. Numbers without the decimal are interpreted as element IDs.

The interval count in a Time Series Function is optional and does not require a decimal point. To determine the interval, Split counts the number of arrays which meet the specified conditions (Stop, Start, and Copy). If the time synchronize function is enabled, the Time Series functions remain synchronized to the starting time even if a complete array is missing from the input data. When elements are missing, the Time Series calculations are based on the actual number of elements found.

Semicolons are used in Time Series functions to separate the elements or expressions from the count which determines the interval. SmplMax and SmplMin require two elements separated by a semicolon. The first is checked for a maximum or minimum, while the second is sampled on the maximum or minimum.

The following set of weather data from Mt. Logan in northern Utah gives a total of seven elements each hour. This Field Formatted output, with title and column headers, was generated by Split. These data are used in the following examples of Time Series functions.

**Mt. Logan Weather Data**

| Day | Time | Airtemp deg F | RH | Mean Wind Speed mph | Mean Wind Direction | Std Dev of Direction |
|---|---|---|---|---|---|---|
| 178 | 100 | 58.56 | 17.42 | 5.855 | 338.3 | 6.562 |
| 178 | 200 | 57.48 | 17.65 | 8.27 | 344.8 | 7.51 |
| 178 | 300 | 56.85 | 17.76 | 7.75 | 330.8 | 5.065 |
| 178 | 400 | 56.55 | 18.89 | 7.6 | 319.7 | 10.93 |
| 178 | 500 | 56.57 | 19.6 | 10.41 | 307.3 | 4.23 |
| 178 | 600 | 55.33 | 23.32 | 8.99 | 317.7 | 6.258 |
| 178 | 700 | 55.95 | 24.79 | 9.52 | 322.3 | 4.609 |
| 178 | 800 | 58.12 | 23.98 | 6.588 | 315.6 | 9.43 |
| 178 | 900 | 59.79 | 23.46 | 5.458 | 312 | 15.32 |
| 178 | 1000 | 61.09 | 24.12 | 4.622 | 299.3 | 18.3 |
| 178 | 1100 | 61.34 | 25.03 | 5.926 | 303 | 17.26 |
| 178 | 1200 | 60.61 | 27.46 | 6.815 | 309.7 | 18.71 |
| 178 | 1300 | 61.01 | 25.44 | 8.35 | 310.2 | 18.37 |
| 178 | 1400 | 60.93 | 25.48 | 10.92 | 317.5 | 12.68 |
| 178 | 1500 | 62.3 | 23.79 | 8.43 | 310.6 | 19.21 |
| 178 | 1600 | 63.75 | 24.31 | 8.88 | 321.4 | 15.22 |
| 178 | 1700 | 66.15 | 22.45 | 7.97 | 341 | 17.77 |
| 178 | 1800 | 67.33 | 23.06 | 6.758 | 344.1 | 20.74 |
| 178 | 1900 | 66.59 | 24.75 | 7.08 | 341.8 | 16.09 |
| 178 | 2000 | 64.52 | 26.03 | 8.76 | 337.2 | 14.91 |
| 178 | 2100 | 59.84 | 27.45 | 11.81 | 305.4 | 12.36 |
| 178 | 2200 | 56.19 | 35.46 | 15.62 | 316.7 | 19.01 |
| 178 | 2300 | 55.48 | 38.8 | 17.12 | 338.7 | 11.41 |
| 179 | 0 | 55.22 | 37.13 | 11.86 | 351.6 | 8.22 |

**Avg(x;n)** returns the average of element x over a full data set or every $n^{th}$ value.
Examples:
Avg(3) = 59.898 (average daily temp)
Avg(3;4) = 57.36 (average 4 hour temp)
56.493 (average 4 hour temp)
60.708 (average 4 hour temp)
61.998 (average 4 hour temp)
66.148 (average 4 hour temp)
56.683 (average 4 hour temp)

**Blanks(x;n)** returns the number of blanks or bad data in element x over a full data set or every nth value. Refer to TABLE 5-9 for definition of blank or bad data. Example:
Blanks(3) = 0 (no holes in data set).

**Count(x;n)** returns the number of data points (non blanks) in element x over a full data set or every $n^{th}$ value.
Example:
Count(1) = 24 (24 data points in data set).

**NOTE** Blanks and Count are functions designed for checking the integrity of the data file. A common use for these two functions is "100.*BLANKS(x;n)/BLANKS(x;n)+COUNT(x;n)" which gives the percentage of holes (bad data) in the file.

**Max(x;n)** returns the maximum value of element x over a full data set or every $n^{th}$ value.

Examples:

Max(5) = 17.12 (max WS for day)

Max(5;12) = 10.41 (max WS for 12 hours)
17.12 (max WS for 12 hours)

**Min(x;n)** returns the minimum value of element x over a full data set or every $n^{th}$ value.

Examples:

Min(7) = 4.23 (min std. dev. of WS for day)

Min(3;8) = 55.33 (min temp for 8 hours)
59.79 (min temp for 8 hours)
55.22 (min temp for 8 hours)

**RunTotal(x;n)** returns a running total of element x for every line in the data set. If an $n^{th}$ value is specified, a running total will be output every $n^{th}$ value.

Example: RunTotal(5) =

5.85
14.12
21.87
29.47
39.88
48.87
:
:
:
166.76
182.38
199.50
211.36
211.36

Running total of hourly average wind speed provides up-to-the-hour wind run for that day. Because an $n^{th}$ value was not specified, the Final Summary output, which is daily wind, is the same as the "total" output.

**Sd(x;n)** returns the standard deviation of element x over a full data set or every $n^{th}$ value.

Examples:

Sd(3) = 3.6593 (std. dev. temp for day)

Sd(3;8) = 1.011 (Sd temp for 8 hours)
1.1182 (Sd temp for 8 hours)
4.965 (Sd temp for 8 hours)

**Smpl(x;n)** returns a sample of element x every $n^{th}$ value.

Examples:

Smpl(4;8) = 23.98 (RH every 8 hours)
24.31 (RH every 8 hours)
37.13 (RH every 8 hours)

**SmplMax(x;y;n)**

looks for a maximum value in element x and samples element y when the maximum is found. If an $n^{th}$ value is specified then it outputs the sample on a maximum every $n^{th}$ value, otherwise it outputs the sample on a maximum at the end of file.
Examples:
SmplMax(5;(3)) = 55.48 (on max wind speed sample temperature)
SmplMax(5;(3,6);8) = 56.57 307.3
60.93 317.5
55.48 338.7
(on max wind speed sample temperature and wind direction every 8 hours)

**SmplMin(x;y;n)** looks for a minimum value in element x and samples element y when the minimum is found. If an $n^{th}$ value is specified then it outputs the sample on a minimum every $n^{th}$ value, otherwise it outputs the sample on a minimum at the end of file. Examples:
SmplMin(3;5) = 11.86 (on min temp sample wind speed)
SmplMin(3; (5,6);8) = 8.99 317.7
5.458 312
11.86 351.6
(on min temperature sample wind speed and wind direction every 8 hours)

**Total(x;n)** returns the total of element x over a data set or every $n^{th}$ value.
Examples: Total(5) = 211.36 (daily wind run)

**WAvg(x;n)** Returns the unit vector mean wind direction in degrees of element x (wind direction in degrees) over a full data set or every nth value.
Example:
WAvg(6) =
323.14 (mean wind direction for the day)
WAvg(6;4) =
333.41 (mean wind direction for 4 hours)
315.73 (mean wind direction for 4 hours)
306 (mean wind direction for 4 hours)
314.92 (mean wind direction for 4 hours)
341.03 (mean wind direction for 4 hours)
328.09 (mean wind direction for 4 hours)

### 5.3.1.12 Special Functions, Details, and Examples

**TABLE 5-8. Split SPECIAL FUNCTIONS**

| | |
|---|---|
| **Crlf** | = Insert carriage return line feed in Output File. |
| **Date("format"S;H;D;Y)** | = Convert day of year and time to a timestamp with calendar date and time, where format uses Windows conventions to specify output format. S=seconds, H = HoursMinutes, D = Day, Y = year. The output timestamp is quoted text. Date can be used to create monthly time series summaries. |
| **Edate("format"S;H;D;Y)** | = The same as the Date function except that the output text is not quoted. EDate can be used to create monthly time series summaries. |
| **"Label"** | = Insert Comment in Output file. (Label is anything within the quote marks.) |
| **Line** | = Number of lines written to Output file. |
| **smpl(.pa;n)** | = Page break such that n is the number of lines per page for the printer or the .RPT file. |
| **PCdate or PCEdate** | = Used in a report header to print the current date. |
| **WDQ(n)** | = Outputs the wind direction using an alphabetical abbreviation, based on 8 quadrants . |
| **WDQS(n)** | = Outputs the wind direction using an alphabetical abbreviation, based on 16 quadrants . |

The Mt. Logan data set is used for the Special Function examples. These functions are helpful in converting time fields to formatted timestamps and formatting the output. Since one of the main differences between mixed-array data files and table based data files is the time format, these functions can be used to convert between file types.

**NOTE** If you are processing the data file in multiple passes including formatting of the date and time fields, you should put the date processing in the final pass. Split cannot read all of the timestamp formats that it can produce. For example, the quoted timestamp in table based data files has a specific structure. Any changes to the structure will make the timestamp unreadable for Split.

**Crlf** returns a carriage return and line feed where the Crlf is placed in the parameter file.

Examples:

```
Smpl("Max Temp";24),Max(3;24),
Smpl(Crlf;24),Smpl("Max RH";24),Max(4;24)
= Max Temp    67.33
  Max RH      38.8
```

The Crlf is placed after the maximum temperature 67.33 so that the maximum RH is on the next line.

**NOTE** A carriage return/line feed is recognized by Split as an element, and may throw the column headers off in the output file.

**"Label"** returns a comment in the output file. This is a useful formatting function when labels are desired on the same line as the data. The label includes anything within the quote marks, the quote marks are not output but must be in the parameter file. The label cannot exceed the width of the output column (default is eight characters). A maximum of thirty (30) labels are allowed per Select line.

Make sure that the column widths are big enough for the label to fit. Otherwise the output will indicate Bad Data.

Examples:

"Max Temp" =
Max Temp (outputs Max Temp
Max Temp 24 times)
.
.
.
Max Temp

Smpl("8 hour ";8),Smpl("Max Temp";8), Max(3;8) = 8 hour
Max Temp 58.56
8 hour Max Temp 63.75
8 hour Max Temp 67.33

This example samples the labels called "8 hour" and "Max Temp" and looks for a Maximum temp for every 8 hour interval.

**Line** numbers each line written to the report file or printer. This differs from the Count function in that Count looks at how many lines were read.
Examples:
Line, 4, 5 =

| | | |
|---|---|---|
| 1 | 17.42 | 5.855 |
| 2 | 17.65 | 8.27 |
| 3 | 17.76 | 7.75 |
| 4 | 18.89 | 7.6 |
| 5 | 19.6 | 10.41 |
| 6 | 23.32 | 8.99 |
| 7 | 24.79 | 9.52 |
| | . | . |
| | . | . |
| | . | . |
| 19 | 24.75 | 7.08 |
| 20 | 26.03 | 8.76 |
| 21 | 27.45 | 11.81 |
| 22 | 35.46 | 15.62 |
| 23 | 38.8 | 17.12 |
| 24 | 37.13 | 11.86 |

Smpl (Line;8), Smpl (4;8), Smpl (5;8)

| | | |
|---|---|---|
| 1 | 23.98 | 6.588 |
| 2 | 24.31 | 8.88 |
| 3 | 37.13 | 11.86 |

**smpl(.PA,n)** Outputs the data to the printer or .RPT file with **n** lines per page.

Examples:

2, 3, Smpl (.PA;12) =

| | |
|---|---|
| 100 | 58.56 |
| 200 | 57.48 |
| . | . |
| . | . |
| . | . |
| 1100 | 61.34 |
| 1200 | 60.61 |
| 1300 | 61.01 |
| 1400 | 60.93 |
| . | . |
| . | . |
| . | . |
| 2300 | 55.48 |
| 0 | 55.22 |

**WDQ(n)** Outputs the wind direction using an alphabetical abbreviation, based on 8 quadrants (N, S, E, W, NE, NW, SE, SW). n is an element containing wind direction. For example, if n = 182, S would be returned in the output file.

**WDQS(n)** Outputs the wind direction using an alphabetical abbreviation, based on 16 quadrants (N, S, E, W, NE, NW, SE, SW, NNE, ENE, ESE, SSE, SSW, WSW, WNW, NNW). n is an element containing wind direction. For example, if n = 111, ESE would be returned in the output file.

**Date("format"; S; H; D; Y)** Converts a datalogger's time stamp to a different format and encloses it in double-quotes (edate will produce a date without quotes). "Format" is a string which identifies how the date should be output. The "format" string is similar to the date format used by Windows. See the online help in Split to get a complete list of the format parameters.

S is the element number that contains seconds; H is the element number that contains hours/minutes; D is the element number that contains day; and Y is the element number that contains the year. A constant can be used in place of any of the element numbers (the constant must be a valid value for the type of date field and include a decimal point; e.g., 2000.0 for the year). If only three elements are specified, these will be assumed to be hour/minute, day, and year.

When using the Date function for a table-based datalogger (e.g., a time stamp in the format "2002-02-03 21:16:00"), if

the time stamp is the first element in the array, a 1 is used for all of the time stamp elements (S; H; D; Y).

If "serial" is entered for the "format" string, a serial date will be output. Other special functions are "hourarray" and "dayofyear". Both of these are used when processing data from table-based dataloggers so that the timestamps are similar to that of mixed array dataloggers. Hourarray changes a 0000 hourly timestamp to 2400, and dayofyear produces a Julian Day.

In older versions of Split, the date( ) and edate( ) functions were limited to converting the Julian day to a MM-DD format, with a syntax of date(doy;y) where doy = the element number for the day of the year; y = the element number for the year. This older format is still supported.

**NOTE**

Split will mark the date as Bad Data if the time and date resulting from the conversion will not fit in the specified column width. The on-screen display and the report file will precede the date with asterisks. In the .PRN output file, Split uses the Bad Data string.

When Date and Edate are used within other functions they must be used with the older format Date(doy;y) and Edate(doy;y) instead of using the extended date functions as shown in the table. For example AVG(1;Date(2;2002.0)). The decimal is needed to indicate a fixed number. Numbers without the decimal are interpreted as element IDs.

**Date Format Examples**

Assume that in a mixed array data file, element 2 is Year, element 3 is Day of Year, element 4 is Hour/Minute, and element 5 is Seconds.

| String Entered | Output |
|---|---|
| date("mm/dd/yy, h:nn";5;4;3;2) | "02/25/02, 4:10" |
| edate("mm/dd/yy, hh:nn";5;4;3;2) | 02/25/02, 04:10 |
| edate("dddd, mmmm d, yyyy";5;4;3;2) | Monday, February 25, 2002 |
| edate("'Date:' mmm d, yyyy";5;4;3;2) | Date: Feb 25 02 |

If a time element is missing from a mixed array data file, use a valid constant instead.

If processing a table-based data file, use a 1 for all time elements (assuming the time stamp is the first element in the data file). For the examples above:

| | |
|---|---|
| date("mm/dd/yy, h:nn";1;1;1;1) | "02/25/02, 4:10" |
| edate("mm/dd/yy, hh:nn";1;1;1;1) | 02/25/02, 04:10 |

edate("yyyy", "dayofyear", "hhnn";1;1;1;1) 2002, 56, 0410

Notice that this last example essentially creates an array-type of timestamp.

**NOTE** When processing a data file from a mixed array datalogger, if the time stamp uses midnight as 2400 with "today's" date, the date function will convert that time stamp to 0000 hours with "tomorrow's" date. The "No Date Advance" function can be used to stop the date from rolling forward (Other button, No Date Advance check box).

**edate("format"; S; H; D; Y)** edate( ) functions identically to date( ) above, except that the time stamp is not surrounded by quotes.

**Monthly Summary Example**

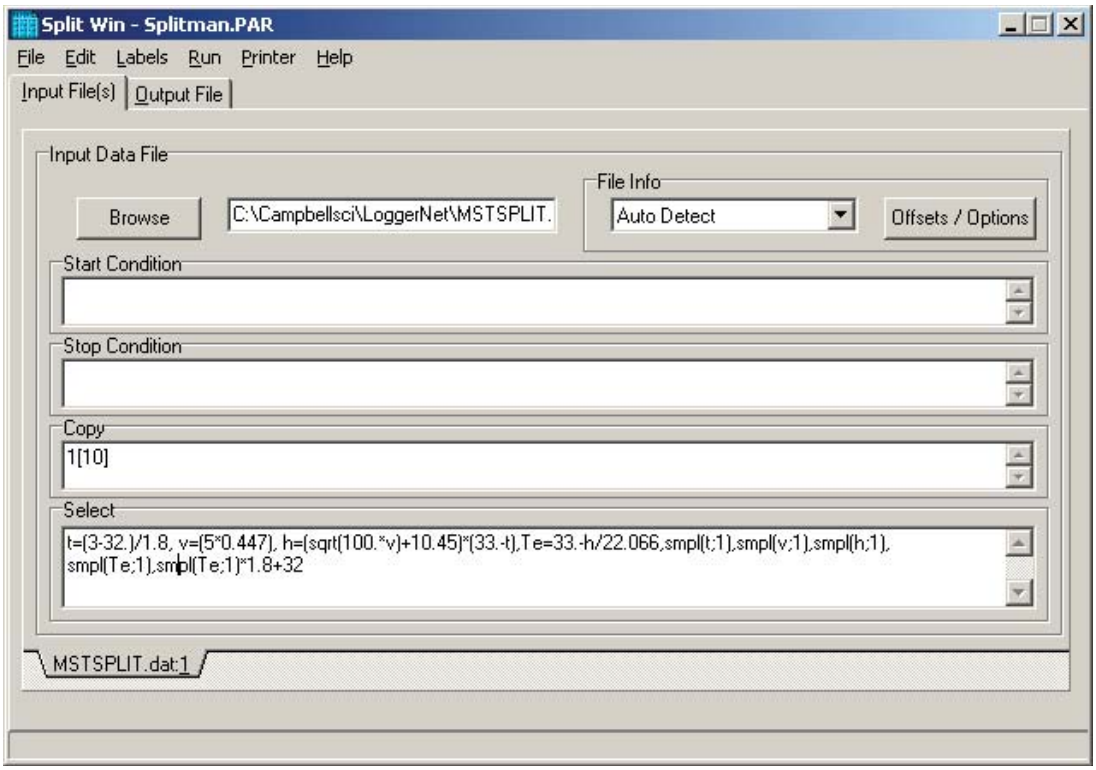
The Date function can be used to produce a monthly summary of daily time series data by using Date( ) for the interval in the time series function. This will trigger time series output for the first day of each month. The syntax is avg(7;date(3;2)), where you want to take a monthly average of element 7, and the day of year is contained in element 3 and the year in element 2. If you have data recorded on a once per minute or once per hour basis, it must first be processed into a 24 hour summary for this function to produce the output expected.

**NOTE** When Date and Edate are used within other functions they must be used with the older format Date(doy;y) and Edate(doy;y) instead of using the extended date functions. For example AVG(1;Date(3;2)). When used with table based data files the format would be AVG(1;Date(1;1)).

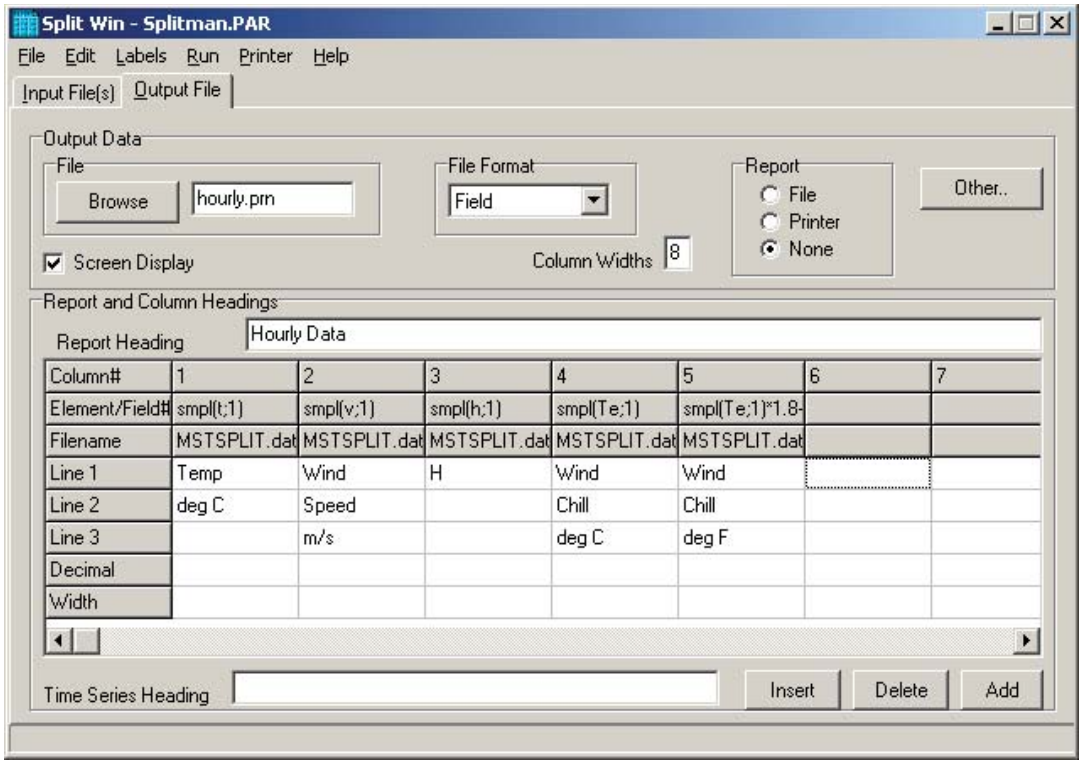
When producing a monthly summary and outputting the month along with the data, you might want to set up the value for the month as "month -1", to correctly reflect the month that the data actually represents.

### 5.3.1.13 Split Functions Example

The following is a parameter file that operates on the Mt. Logan data with several of the Split features being utilized. This first screen shows the input file and the select criteria that were programmed. This example does calculations based on temperature and wind speed to determine the wind chill.




The following screen shows the output file setup including the column headings and the units.



This .PAR file produces a wind chill summary of the Mt. Logan Peak data set. The formula for calculating wind chill is given as follows:

$$Te = 33-(h/22.066)$$

where

Te = Wind Chill equivalent temperature, degrees C

$h = ((100V)^{0.5} + 10.45-V)(33-T)$

where

h = Kcal $m^{-2}$ $hr^{-1}$ wind chill index

v = wind speed in meters/second

T = temperature in degrees C

Note that at wind speeds between 0 to 4 mph (0 to 1.8 m/s), the wind chill should be ignored because this formula results in wind chill temperatures that are greater than the ambient temperature. The National Weather Service includes wind chill in reports only when temperatures drop below 35°F (1.7° C).[1] The formula is for example purposes and is not endorsed by Campbell Scientific as a standard.

When this .PAR file is executed, the following output is displayed on the screen.

**Wind Chill Report from Mt. Logan**

| Temp deg C | Wind Speed m/s | H | Wind Chill deg C | Wind Chill deg F |
|---|---|---|---|---|
| 14.756 | 2.6172 | 438.06 | 13.148 | 55.666 |
| 14.156 | 3.6967 | 489.58 | 10.813 | 51.463 |
| 13.806 | 3.4643 | 491.34 | 10.733 | 51.319 |
| 13.639 | 3.3972 | 493.4 | 10.64 | 51.151 |
| 13.65 | 4.6533 | 529.57 | 9.0005 | 48.201 |
| 12.961 | 4.0185 | 530.58 | 8.9547 | 48.118 |
| 13.306 | 4.2554 | 528.27 | 9.0596 | 48.307 |
| 14.511 | 2.9448 | 456.04 | 12.333 | 54.199 |
| 15.439 | 2.4397 | 414.97 | 14.194 | 57.55 |
| 16.161 | 2.066 | 383.21 | 15.633 | 60.14 |
| 16.3 | 2.6489 | 402.08 | 14.778 | 58.601 |
| 15.894 | 3.0463 | 425.2 | 13.731 | 56.715 |
| 16.117 | 3.7325 | 439.59 | 13.078 | 55.541 |
| 16.072 | 4.8812 | 468.26 | 11.779 | 53.202 |
| 16.833 | 3.7682 | 421.85 | 13.882 | 56.988 |
| 17.639 | 3.9694 | 405.59 | 14.619 | 58.314 |
| 18.972 | 3.5626 | 361.39 | 16.622 | 61.92 |
| 19.628 | 3.0208 | 331.76 | 17.965 | 64.337 |
| 19.217 | 3.1648 | 345.62 | 17.337 | 63.207 |
| 18.067 | 3.9157 | 393.08 | 15.186 | 59.335 |
| 15.467 | 5.2791 | 493.51 | 10.635 | 51.142 |
| 13.439 | 6.9821 | 584.71 | 6.5016 | 43.703 |
| 13.044 | 7.6526 | 607.86 | 5.4526 | 41.815 |
| 12.9 | 5.3014 | 566.29 | 7.3368 | 45.206 |

Reference

[1] "Wind Chill Errors", Edwin Kessler, Bulletin of the American Meteorology Society, Vol. 74, No. 9, September 1993, pp 1743-1744.

### 5.3.1.14 Summary of Select Line Syntax Rules

- A fixed numeric value must include a decimal point "." or be in scientific notation. There are some exceptions to this as noted below.
- Scientific notation has the format "mantissa E power of ten" (e.g., 3E5 = 3 x $10^5$).
- Element numbers are entered without a decimal point.
- Commas separate Select line parameters (e.g., 2,3,(3+4)/3.2,6).
- Two decimal points are used to select consecutive elements between starting and ending elements (e.g., 3..6, refers to the elements 3,4,5, and 6).
- A set is a group of two or more elements and/or expressions separated by commas and enclosed by parentheses. No member of a set can include parentheses. Therefore, a set cannot include a set or a function as one of its members. For example:

| VALID EXPRESSION | INVALID EXPRESSION |
|---|---|
| Arctan (2/3) | Arctan ((2/3)) |
| Arctan (2/3, 3/4, 4/5) | Arctan ((2/3, 3/4), 4/5) |
| Arctan (COS(2)) | Arctan (COS(2), COS(3)) |

- A single expression can operate on a set of elements. For example, the expression (3..6,8)/2.0 is the same as 3/2.0, 4/2.0, 5/2.0, 6/2.0, 8/2.0; (3..6)/(2..5) is the same as 3/2, 4/3, 5/4, 6/5.
- The element or expression that is the argument of a math or Time Series function, must be enclosed in parentheses. A range of elements can be specified, resulting in as many outputs as elements (e.g., Avg(3..5,7) will output 4 averages).
- Square brackets are used to enclose an allowable range for a value (e.g., 3[3.6..12] ) to indicate that the allowable range for element 3 is from 3.6 to 12. Whole numbers within brackets do not require a decimal point. TABLE 5-5 explains how values outside the specified range are treated.
- The interval in a Time Series function is optional and does not require a decimal point.
- Semicolons are used in Time Series functions to separate the elements or expressions from the number that determines the interval. Sample on maximum and sample on minimum require two elements or expressions also separated by a semicolon.

### 5.3.1.15 Time Synchronization

The time synchronize function is useful when data is missing from a file or if several files of data need to be merged together. The files are synchronized according to time; any missing data in the file (or files) will be replaced with blank data.

This function synchronizes according to day, hrmn (hour-minute), and/or seconds. The syntax used to identify the time elements for array data is:

$e_i$[day]:$e_i$[hrmn]:$e_i$[seconds]

Referring to TABLE 5-1, to identify the day of year for a mixed-array data file, type:

2[189]::

for hrmn type:

:3[1200]:

and seconds are expressed as:

::4[5]

A single colon is assumed to be between day and hrmn (e.g., 2[189]: means day, :3[1200] means hours, and 2[189]:3[1200] means day and hour-minute). When the time synchronize function is used, a time interval must be specified in the Copy line of the first data file. For example, 4[60] in the Copy line will create a synchronized file containing the data from the input files that occurred every 60 minutes. If no time interval is specified in the Copy line then the time specified in the Start Condition becomes simply a starting time with no time synchronization.

Typically, the starting time specified must actually be found in the input file before the Start Condition is satisfied (e.g., if the input file starts at 1100 hrs and 1000 hrs is entered for the starting time, with no day specified, Split will skip over arrays until it reaches 1000 hrs the next day). However, the Start-Stop On/After Time function can be enabled (Output tab, Other button) to trigger the start of processing when the exact time is found or at the first instance of data after that time has occurred.

*Table-based dataloggers*

Because the time stamp for a table-based datalogger is all one string, and therefore read by Split as one element, the syntax is somewhat different. All elements in the time stamp are specified by a 1 (if the time stamp is the first item in each row of data).

The 1s in the string identify the position of the time stamp in the line of data. Each colon represents a portion of the time stamp. The format is 1[year]:1[day]:1[hour/minute]:1[seconds]. The colons in the time stamp must be present or the function will not work correctly.

**NOTE** Time synchronization can only be done for data from a single year. It will not work over a year boundary.

Time elements can be identified without specifying a starting time (e.g., 2:3). If you are working with only one file, Split will begin processing that file at the first record in the file. If any gap in the data is found, blank data (or the "Replace Bad Data With" text) and a carriage return line feed will be inserted for each line of missing data. Note that Split will also detect a gap in data if, for instance, you specify a start time of 2[92]:3 (start at Julian day 92) and your

hour/minute for day 92 starts at 9:30 a.m. The time between the start of the day (0000) and 9:30 a.m. will be considered missing data. Blanks (or the "Replace bad data with" text) and a carriage return line feed will be inserted at the beginning of the PRN file for each "missed" output interval.

If you are working with two or more files, once Split starts processing the files (based on the time of the first record of the first file), if no data exists for the other file(s), blank data will be inserted.

If multiple input files are given specific starting times, Split starts the output at the earliest specified starting time. In a PRN file, Blanks or the comment entered in the "Replace bad data with" field are inserted for values from other input files until their starting times are reached. In a RPT file only blanks are used.

**NOTE** When using time synchronization with a mixed array data file, with a midnight time stamp of 2400, you will need to select the Other button, "Midnight at 2400 hours" checkbox.

#### 5.3.1.15.1 Time Synchronization and the Copy Condition

To use the time synchronize function, time element(s) must be specified in the Start Condition. The user must also specify a time interval in the Copy condition. For instance, if the original data had 15 minute outputs and you only want hourly outputs, then an interval of 60 minutes must be specified following the element number. This is entered as (assuming hrmn is element number 3) "3[60]". If time synchronization is specified in the Start Condition, Split looks for the interval in a time element in the Copy condition. Only one time interval is specified. This interval is the unit of time to synchronize each file.

The interval can be given tolerance limits by following the interval with a comma and the tolerance. For example, if 3 is the hrmn element, and the time interval is 60 minutes +/-2 minutes, the syntax is 3[60,2].

Table based data files need to use the same time format as described in Section 5.3.1.3, *Start Condition* *(p. 5-13)*. You can specify the interval for time synchronization on table files as ::1[60]: which will give you an output interval of 60 minutes.

If the time synchronize function is enabled and data are missing at one or more of the time intervals specified, then a blank (or the comment entered in the "Replace bad data with" field) is output to the Output File. See TABLE 5-5.

#### 5.3.1.15.2 Using Time Synchronization While Starting Relative to PC Time

Split tries to time-sync files to the top of the hour when starting relative to PC time. If you are synchronizing files where the data output interval is not at the top of the hour, you will need to specify an interval in the Copy Condition that represents a window of time in which Split should look for the hour/minute. For instance, if your data is output 50 minutes into a 60 minute interval (and therefore, your time stamps are 50, 150, 250, 350...2350) your Start Condition and Copy Conditions for the first file might look like the following:

Start Condition
2[-1]:3[50]:

Copy Condition
1[106]and3[60,10]

Where:
element 1 is the array ID
element 2 is the Julian day
element 3 is the hour/minute

The Start Condition directs Split to begin processing data when the time is one day prior to the current PC time and when the hour/minute value is equal to 50. The 1[106] in the Copy Condition specifies the array from which the data should be copied. The 3[60,10] indicates that the interval for the time stamp is 60 minutes and designates a 10 minute time window on each side of the top of the hour in which Split should look for the hour/minute data (10 minutes before the hour, 10 minutes after the hour).

The second file's Copy Condition should include only the array from which to copy the data. No interval is necessary.

## 5.3.2 Output Files

To create an Output File, click the **OUTPUT FILE** tab. The file is created on the default drive or directory unless the file name is preceded with an alternative drive or directory. Use the Browse button to change directories.

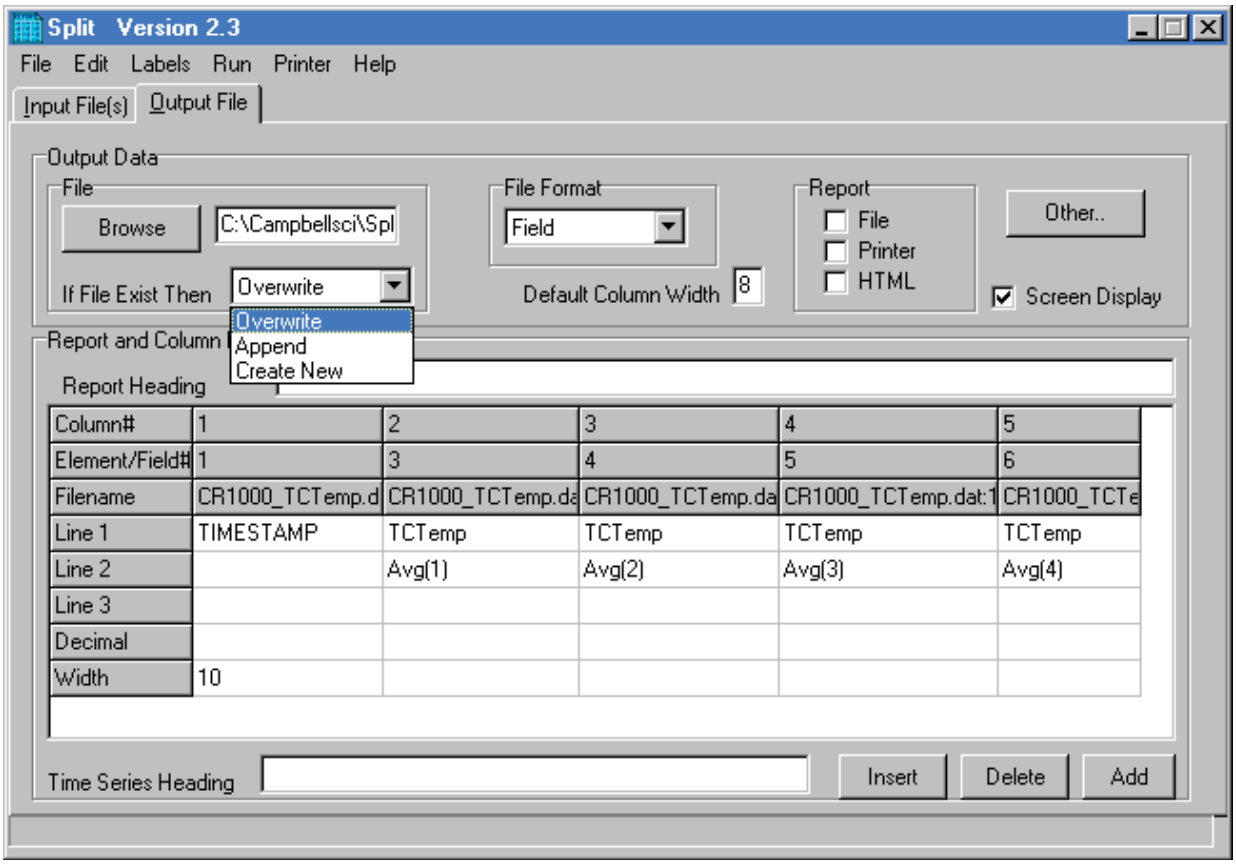
Split will assign this file an extension of .PRN if an extension is not specified by the user. Whenever an Output file name is entered, regardless of extension, an Output file is created only when the RUN | GO menu option is selected.

If the file name you have selected already exists, you can use the "If File Exists Then" drop-down list box to determine what action Split will take. By default, each time a PAR file is run the existing output files (PRN, RPT, and HTM) are overwritten (Overwrite option). When Append is selected, the PRN file will not be overwritten — the new data will be added to the end of the existing file. However, the RPT and HTM files will be overwritten. If Create New is selected, Split will create all new files using the original file name and appending an _0, _1, and so on to each subsequent run.

In Append mode, if an HTM or RPT file is needed with all the data, you will need to run the PRN created by Split through the program a second time. If the Output File name is left blank, Split does not write data to an Output File on disk; rather, it will display the processed values on the screen if the Screen Display box is checked. If Screen Display is not enabled, no data will be displayed on the Split RUN screen.

**CAUTION** The Output file name cannot be the same as the Input file name. Split will display an error message if this condition occurs.

Several output options may be specified to alter the default output to the file. Some are located on the main **OUTPUT FILE** screen and some are made available by pressing the **Other** button.




### 5.3.2.1 Description of Output Option Commands

**File Format**

There are five File Format options to choose from: No File, Field, Comma, Printable, and Custom. If No File is chosen, then only the .PRN file is saved to disk. The Field, Comma, and Printable options produce files formatted as Field Formatted, Comma Separated, and Printable ASCII, respectively. An example of each of these file types is given in TABLE 5-1 in the Input Files section.

The Custom file format uses the regional settings in the Windows operating system to determine the decimal symbol and the separator used with data values. In the Regional Settings for Numbers, the decimal symbol uses the character specified in the Decimal Symbol field; the separator uses the character specified in the List Separator field. These settings are typically found in Control Panel | Regional Settings (or Options), Numbers tab. This allows users who are used to the comma "," as the decimal and the period "." as a data separator to see the output data in that format.

**Default Column Widths**

The Default Column Widths field is used to set the default width of the columns. Valid entries are 6,7,8, and 9. The initial width is 8. High Resolution Final Storage data requires a minimum column width of 8. Entering a number in the Width row for each column overrides the default

settings and sets the width of individual columns. If this field is left blank, the Default Column Widths field is used.

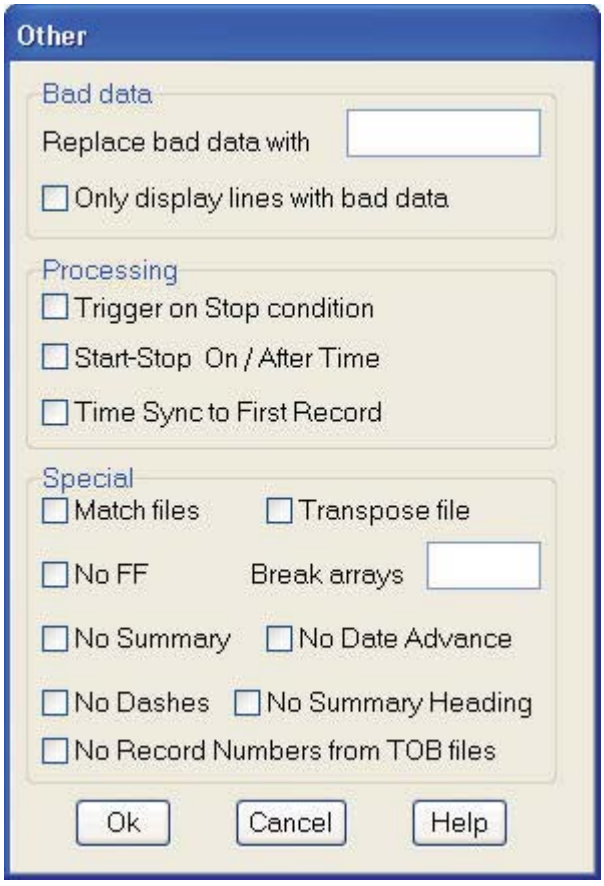
**Screen Display**
The Screen Display field controls writing the processed data to the screen. To write to the screen, check the box. For faster execution, clear the box to omit writing to screen. The data will then be written to the file only.

**Report**
A report, with page and column headings, can be sent to a file or printer. There are three report options: File, Printer, HTML. One or more can be selected. A report sent to a file has the extension of .RPT. If the report is sent to a printer, the printer must be on-line. In all cases a .PRN output file is created. A basic HTML file can be created containing the formatted report data. The HTML file can be used as a display of the formatted data output in a web browser.

**NOTE** To remove page breaks in the HTML file, enable the "No FF" option.

**Other**

The **Other** button provides access to the dialog box shown below.



It allows the following settings to be modified:

**Replace bad data with –** The text in the field, to the right of this option, is entered into the .PRN output file data set if data are blank, bad, or out of range. See TABLE 5-9 for definition of blank or bad data. Whatever text string the user enters in the field will be entered if a blank or question mark is in the data or if data are out of range. This option is useful when the Output file is imported into a spreadsheet program, such as Excel.

**TABLE 5-9. Definition of Blank or Bad Data for each Data File Format**

| File Format | Definition of Blank or Bad Data |
|---|---|
| Printable ASCII | ???? |
| Comma Separated ASCII | blank or any character except numeral or space |
| Field Formatted | blank or "" (double quotation marks) |

**Only display lines with bad data** – Outputs only those arrays containing one or more Out of Range elements. If a report is generated, an asterisk precedes the Out of Range value in the .RPT file.

**Trigger on Stop condition** – Changes the meaning of Stop Condition to trigger Time Series processing output. The Stop Condition is included in the Time Series processing if it satisfies the Copy line.

If the Trigger on Stop Condition is selected, a Time Series output will occur each time the Stop Condition is met.

**Start-Stop On/After Time** – In most instances, Split will not start or stop processing a file unless the exact start condition is found. However, when starting or stopping based on time, you can enable Split's **Start-Stop On/After Time** option. This will trigger the start (or end) of processing when the exact time is found or at the first instance of data after that time has occurred (which meets other defined criteria in the PAR file).

**Time Sync to First Record** – This option is used with the time-sync function. It allows you to set specific times in the Start Condition, but have synchronization start at the first record in the file that meets the Start Condition. This may avoid an output file that starts with blank lines.

For example, you have table-based data file(s) containing 15 minute data. Your first data file starts on Sept 9th at 12:15 p.m. You want to time sync the files and output only the data that occurs at midnight.

You need to specify '0' for the hour/minute field in the Start Condition or the output will contain the data that occurs each day at 12:15. Therefore, you would use:

Start Condition = 1:1:1[0]:1

The Copy Condition determines the interval of your data. Therefore, to output data that occurs every 24 hours, you would use:

Copy Condition = 1:1[1]:1:1

Because you have specified a time in the Start Condition, but not the day, Split assumes the first day of the year. Therefore, by default, you will have blank lines in your output file for each day from Jan 1st to Sept 9th. Using the **Time Sync to First Record** option will avoid these blank lines.

**Match files** – This option compares two files of the same data. If good data exists in one and not the other (question marks), then Split will fill the OUTPUT file with the good data. This is used to get a more complete record from an error ridden file (e.g., one recorded at freezing temperatures by reading a tape twice and running both files through Split).

**CAUTION**

For the Match files option to produce a correct Output File, the differences between the two Input Files can only be question marks. Both files must have the same Start Condition or the beginning of both files must be the same.

**Transpose file** – Transposes the rows and columns of the input file. Only one Input File can be transposed at a time and no Select options can be specified. A maximum of 26 arrays are transposed per pass of Split.

To transpose a file containing more than 26 arrays, several passes are required. Change the Output file name and Start Condition for each pass. Split may then be used to merge the multiple files.

**No FF –** Suppresses form feeds and page breaks in RPT and HTML files. When this option is selected, a header appears on the first page only. This option is used for printing reports on continuous feed paper or for displaying HTM files in a browser.

**Break arrays –** This option breaks up the Output Array into new arrays that are #+1 elements in each new array. Split automatically assigns an array ID number equal to the first element in the first array. Only one Input File may be specified. Start, Stop, and Copy Conditions may be specified, but the Select line must be left blank.

**NOTE**

The Break Arrays function works only for mixed array data. It is typically used when processing data from burst measurements.

**No Summary –** When producing reports that include time series processing based on an interval, sometimes that interval will not divide evenly into the number of lines in the data file that is being processed. For example, you may be processing one-minute data on a five-minute interval, and the data file has 103 lines; thus, there are 3 lines of data "left over" at the end of the report. By default, the summary (average, total, maximum, etc., depending upon which time series function is being used) of the left over values is printed at the bottom of the report following the Time Series Heading. Enable the No Summary check box to omit the

summary of the left over values and the Time Series Heading from the report.

**No Date Advance –** When processing a data file from a mixed array datalogger, if the time stamp uses midnight as 2400 with "today's" date, the date function will convert that time stamp to 0000 hours with "tomorrow's" date. (This is because the algorithm used by the date function is based on Windows' time format, and it does not support a 2400 time stamp.) For example:

| Array ID | Year | Julian Day | Hour/Minute | Date Function | Data | Data |
|---|---|---|---|---|---|---|
| 10 | 2002 | 151 | 2200 | 05/31/02 22:00 | 1.701 | 193.6 |
| 10 | 2002 | 151 | 2300 | 05/31/02 23:00 | 1.476 | 31.99 |
| 10 | 2002 | 151 | 2400 | 06/01/02 00:00 | 1.123 | 106.2 |

At Julian Day 151 (May 31) 2400 hours, the date function produces an output of June 1 00:00 hours. The date can be stopped from rolling forward by using the No Date Advance check box. The output will then be similar to:

| Array ID | Year | Julian Day | Hour/Minute | Date Function | Data | Data |
|---|---|---|---|---|---|---|
| 10 | 2002 | 151 | 2200 | 05/31/02 22:00 | 1.701 | 193.6 |
| 10 | 2002 | 151 | 2300 | 05/31/02 23:00 | 1.476 | 31.99 |
| 10 | 2002 | 151 | 2400 | 05/31/02 00:00 | 1.123 | 106.2 |

Caution should be used when applying the date function and enabling or disabling No Date Advance, since it is possible to produce an incorrect date. For instance, using the above example if you were to enter the following into your select line:

3,edate("hh:mm";4;3;2)

with the No Date Advance enabled, you would get the output:

| | | | |
|---|---|---|---|
| 151 | 22:00 | 1.701 | 193.6 |
| 151 | 23:00 | 1.476 | 31.99 |
| 151 | 00:00 | 1.123 | 106.2 |

If you were to enter:

edate("mm/dd/yy";4;3;2),4,6,7

with the No Date Advance disabled, you would get the output:

| | | | |
|---|---|---|---|
| 05/31/02 | 2200 | 1.701 | 193.6 |
| 05/31/02 | 2300 | 1.476 | 31.99 |
| 06/01/02 | 2400 | 1.123 | 106.2 |

**No Dashes** – When the **No Dashes** check box is selected, the dashed line that typically appears under the column headings will not be displayed. This option affects all output types (PRN, RPT, HTM, and printed page).

**No Summary Heading** – When processing data using time series functions (see No Summary, above), select this option to prevent the Time Series

Heading and Column Headings from being printed at the bottom of the report. The "left over" summary data will still be printed.

**No Record Numbers from TOB Files** – Split automatically converts TOB (binary) files to ASCII prior to being processed. When this check box is selected, the record numbers will not be included in the converted file. This will affect the element numbers used for the Start, Stop, Copy, or Select fields of the PAR (e.g., if a file has a timestamp, record number, and data value, when this check box is selected the data value would be element 2. When the check box is cleared, the data value would be element 3).

#### 5.3.2.2 Report Headings

A report is output to a printer or file with the extension .RPT. Headings are not included in the standard output to disk (.PRN or user named extension output file). However, a report can be labeled with a header by entering text into the Report Heading field. A report heading can have several lines, but it is limited to a total of 253 characters including backslashes and carriage returns. "\" characters break the report heading into multiple lines.

When Time Series functions are used in the Select field without an interval, they appear as a final summary at the end of the report. They can be labeled by entering a title into the Time Series Heading field at the bottom of the Output File page. Time Series interval summaries cannot be assigned individual titles directly, but you can use special functions such as "Label" and "Crlf" to create column headings and special formatting.

"PCDATE" within the Report Heading inserts the computer's current date (Month-Day-Year). For the European format (Day-Month-Year), enter "PCEDATE".

#### 5.3.2.3 Column Headings

Up to three lines per column can be entered as column headings. These headings are limited to a length of one less than the Output field width.

Column headings associated with Time Series outputs are repeated for Final Summaries if a title for the Final Summary is requested on the headings for report line.

The number of digits to report to the right of the decimal point is entered in the Decimal field and can be set independently for each column. The value output will be rounded to the specified number of digits. Leave this field blank if you do not want to round the data to a specific number of digits.

Column headings can be entered using Split's Data Labels Function (Labels | Use Data Labels).

## 5.4 Help Option

On-line Help is available from any location in Split. Simply select the area of Split in question and press <F1>. Split also offers a brief on-the-fly Help. Place the cursor on the area of Split in question; after a moment a brief description is displayed in the hint line of the Split window (bottom left).

# 5.5 Editing Commands

Split supports the Windows Cut, Copy, and Paste commands. Text from any field in Split or other Windows applications can be Cut, Copied, or Pasted.

# 5.6 Running Split from a Command Line

Existing parameter files can be executed using Splitr.exe which is a "run-time" version of the Split Report Generator. When Splitr.exe is run, the file is processed as if the user chose Run | Go from the Split menu. Splitr.exe can be executed by the Task Master, from a batch file, or from a Windows command line prompt or shortcut.

## 5.6.1 Splitr Command Line Switches

Splitr has four switches that can be used to control how the executable is run.

### 5.6.1.1 Closing the Splitr.exe Program After Execution (/R or /Q Switch)

Typically when Split is run, after the file is processed the user must close the Screen Display window. When Splitr.exe is run from a command line, the user must also close the Screen Display window unless the /R switch is used.

The syntax for this switch is:

SPLITR LOGAN/R

where LOGAN is the parameter file name.

The /R switch should follow immediately after the parameter file name with no space between the two. If a space is used, the following message will be displayed "There was a problem opening the input file. File could not be found or may be in use."

The /Q switch is similar in function and syntax to /R. However, if Split encounters an error when processing the file, no message box is displayed that requires user response (the exceptions are a disk space error or an internal error with the Split executable). This option should be used with caution, since there will be no indication of a problem if a file cannot be processed.

### 5.6.1.2 Running Splitr in a Hidden or Minimized State (/H Switch)

Splitr can be run in a minimized state, so that the Screen Display window does not interrupt other processes on the computer. The syntax for running Splitr minimized is:

SPLITR /H LOGAN

where LOGAN is the parameter file name.

The /H switch must be positioned after SPLITR but before the parameter file name, and a space is required between the executable name and the switch.

#### 5.6.1.3 Running Multiple Copies of Splitr (/M Switch)

Multiple copies of Splitr can be run at one time by using the /M switch. This switch must appear immediately after Splitr. For instance, a batch file containing the lines:

SPLITR /M Logan/R
SPLITR /M Sinks/R

will open two copies of Splitr and process the two files simultaneously.

**NOTE** When using the /M switch in a batch file, the behavior may depend on your Windows version. In some cases, the files will be processed simultaneously, while in other cases, the files will be processed sequentially. It may be possible to change this behavior using the Windows "start" command.

## 5.6.2 Using Splitr.exe in Batch Files

Batch files containing one or more Splitr command lines can be useful for automating data processing. Batch files can be executed manually or by setting them up in the Task Master.

Batch files process each command in succession, without waiting for execution of a command to be completed before proceeding to the next unless they are configured to do so. If multiple parameter files are being processed using Splitr in a batch file, there are no conflicts because only one copy of Splitr can be active at any one time (unless the /M switch is used. However, if other commands are used along with Splitr (such as opening the file in a spreadsheet, copying it to an archive directory, or appending it to an existing file) these commands might be executed before Splitr finishes processing data.

The Windows Start /w (wait) command can be added to a batch file command line to delay execution of the next command until the first command has finished. The Start command has different arguments depending upon the operating system you are using. Refer to your computer's on-line help for information on this command.

## 5.6.3 Processing Alternate Files

Splitr allows the user to select different input and/or output files for an existing parameter file by entering them on the command line after the parameter file name. For example:

"Splitr LOGAN.PAR/R TEST.DAT TEST.PRN"

Replaces the Input and Output file names in LOGAN.PAR, with TEST.DAT and TEST.PRN, respectively.

A space must be used to separate command line parameters. Splitr uses as many entries as exist on the command line. However, the command line has a limit to the number of characters it can accommodate—this limit is operating system dependent. The parameters must be in the following sequence: Input file name, Output file name, Start Condition, Stop Condition, Copy Condition, and Select.

If a parameter is to be left as it is in the parameter file, then space comma space ( , ) may be entered in the command line. For instance, if the parameter file LOGAN.PAR contained TEST1.DAT as an input file name, the following command line would leave the input file TEST1.DAT and change the output file to TEST.PRN.

"SPLITR LOGAN/R , TEST.PRN"

#### 5.6.3.1 Input/Output File Command Line Switches for Processing Alternate Files

The one caveat of using the command line to specify an alternate input and/or output file name is that Split's default options will be used with the alternate file. For instance, by default, output files are written with field-formatted columns. If the original PAR file specified a comma-separated output, that option would be ignored and the and defaults would be used.

Command line switches can be used to control these options for the output and input files. The switch is added immediately after the input or output file name.

**NOTE** In most instances, full path names to the Splitr executable and the input and output file names must be used. In addition, if long file names are used in the path, you may need to surround the path and file name by double quotes.

**Output File Options**

These switches are entered after the output file name; e.g., Splitr Test.par/r Input.dat **Output.prn/P**

/P Sends the output to a printer. This is the same as checking the Printer box for the Report type on the Output File tab.

/R Creates a formatted RPT file. This is the same as checking the File box for the Report type on the Output File tab.

/W Creates a simple HTML file. This is the same as checking the HTML box for the Report type on the Output File tab.

/A Appends the output to the end of an existing file. This is the same as selecting Append for the If File Exists option on the Output File tab.

/L Creates a new output file with a different name if a file exists. This is the same as selecting Create New for the If File Exists option on the Output File tab.

/O Turns the screen display off when Split is processing the PAR file. This is the same as clearing the Screen Display check box on the Output File tab.

/6..9 Sets the default width for all the columns in the report. This is the same as entering a value in the Default Column Width field on the Output File tab.

/[text] Sets the text that will be used in the place of bad data. This is the same as the text string used in the Replace Bad Data field that is found under the Other button of the Output File tab.

/M Compares two input files and creates an output file with a complete data set comprised of both files. This is the same at the Match Files option that is found under the Other button of the Output File tab. The two input file names are separated with a comma but no spaces. Example: Splitr Test.par/r Input1.dat,Input2.dat Output.prn/M

/S Writes the output file without a form feed command after each page. This is the same as the No FF (form feed) option that is found under the Other button of the Output File tab.

/G Outputs only the data marked as "bad" to the file. This is the same as the Only Display Lines with Bad Data check box that is found under the Other button of the Output File tab.

/0 Outputs the data in comma separated format. This is the same as choosing the Comma option for the File Format.

/1 Outputs the data in printable ASCII format. This is the same as choosing the Printable option for the File Format.

/2 Outputs the data using the Regional Settings of your Windows operating system for the decimal indicator and data value separator. This is the same as choosing the Custom option for the File Format (this is the default option for the File Format field).

/F Conditionally outputs the data using the Trigger On Stop Condition. This is the same as choosing the Trigger On Stop Condition option that is found under the Other button of the Output File tab. A stop condition must also be specified. The example below does not specify a start or copy condition. These two fields are indicated by the "space-comma-space" entries. Select line entries are also shown in this example.

Example: Test.par/r input1.dat Output.prn/F , 4[1450] , smpl(1..6),avg(7)

/T Transposes the rows and columns of a file. This is the same as choosing the Transpose File option that is found under the Other button of the Output File tab.

/D Enables the No Date Advance function, which keeps the date for midnight from rolling to the next day. This is the same as choosing the No Date Advance check box that is found under the Other button of the Output File tab.

/N Suppresses the summary information when processing time series data. This is the same as choosing the No Summary check box that is found under the Other button of the Output File tab.

/H Removes the dashed lines from the heading of the RPT file. This is the same as choosing the No Dashes check box that is found under the Other button of the Output File tab.

/U Removes the record number from TOB files that are processed with Split. This is the same as choosing the No Record Numbers from TOB Files check box that is found under the Other button of the Output File tab.

/E Begins processing the file, or stops processing the file, on or after the Start or Stop condition when starting or stopping based on time (the default is to start only if the exact start condition is found). This is the same as choosing the Start -Stop On/After Time option that is found under the Other button of the Output File tab.

Example: Splitr test.par input1.dat Output.prn/E 4[1450]: 4[1456]: (where 1450 and 1456 are the start and stop times, respectively. Colons are required to indicate a time value.)

/I Suppresses the time series heading and column heading information when processing time series data. This is the same as choosing the No Summary Heading check box that is found under the Other button of the Output File tab.

/Bnnn Breaks a long array into multiple lines, where nnn is the number of values to place on each line. This is the same as choosing the Break Arrays check box that is found under the Other button of the Output File tab.

**Input File Options**

These switches are entered after the input file name; e.g., Splitr Test.par/r **Input.dat/L** Output.prn

/nnn Begins processing nnn bytes into the file. If /nnn..mmm is used, then processing begins at nnn bytes into the file and stops at mmm bytes into the file. This is the same as setting a specific Start and Stop offset, which is found under the Offsets/Options button of the Input File tab.

/L Begins processing the file at the byte value where processing last stopped. If /L..mmm is used, then processing begins where it left off and stops at mmm bytes into the file. This is the same as enabling Last Count, which is found under the Offsets/Options button of the Input File tab.

/Bnnn Specifies the file type as Burst data. nnn indicates the size of the arrays. This is the same as selecting Burst Format for the File Info field on the Input File tab.

/F Specifies the file type as Final Storage (binary) data. This is the same as selecting Final Storage Format for the File Info field on the Input File tab.

/M Changes the value for midnight to 2400 instead of 0000. This is the same as selecting Midnight is 2400 Hours check box found under the Offsets/Options button of the Input File tab.

**Batch File Example**

"c:\Program Files\Campbellsci\SplitW\splitr.exe"
c:\Campbellsci\SplitW\switch-test.par input1a.dat Output.prn/E/H/W 4[1200]: ,
, 1..6

where
PAR file: switch-test.par
Input file: input1a.dat
Output file: output.prn
Other outputs: Output.HTML
Start condition: on or after 1200
Stop condition: end of file
Copy condition: none
Elements: 1 through 6

### 5.6.4 Processing Multiple Parameter Files with One Command Line

More than one .PAR file can be executed with a single Splitr command line. Each .PAR file and its associated parameters are separated from the next .PAR file by a semicolon with one space on each side ( ; ). For example:

“SPLITR LOGAN/R TEST.DAT TEST.PRN ; SINKS/R TEST1.DAT TEST2.DAT 1[189]”

executes the LOGAN.PAR file on TEST.DAT and outputs the results to TEST.PRN, then executes the SINKS.PAR file on TEST1.DAT and outputs the results to TEST2.DAT. Execution of SINKS.PAR starts when the first element in TEST1.DAT is 189.

## 5.7 Log Files

Split maintains a log file each time Splitr is run. The main purpose of this log file is to enable users running Splitr in command line mode to identify what happened with each execution of Splitr. The file is named splitr.log and is written to the Sys directory of the Split working directory. (By default, this is C:\Campbellsci\Splitw\sys.) The file will grow to approximately 4-5K in size and then be renamed to splitr.bak. (Any previous splitr.bak file will be overwritten. Therefore, only two log files will be retained.)

If a second instance of Splitr is started when one is already running, another log file, splitrunning.log, will be written. This file simply identifies the time that the second instance of Splitr was started and that Splitr was already running.

# Section 6. View Pro

*After data has been collected from the datalogger, you need a way to analyze that data. View Pro is a file viewer that provides a way to look at the collected data. It will open data files (*.DAT) saved in a variety of formats including files from mixed-array and table-based dataloggers. It can also be used to view data from a LoggerNet database table created with LNDB. View can also open other CSI file types (*.DLD, *.CSI, *.PTI, *.FSL, *.LOG, *.CR2, *.CR5, *.CR1, *.CR3, *.CR6, *.CR8, *.CR9). Once a data file or database table is opened, data values can be graphed in several different formats including Line Graphs, Histograms, XY Plots, FFTs, and Rainflow Histograms.*

## 6.1 Overview

The **View Pro** button on the PC400 toolbar brings up View Pro. This program can be used to open data files (*.DAT) or other CSI file types (*.DLD, *.CSI, *.FSL, *.LOG, *.CR1, *.CR3, etc.). View Pro can easily handle files up to 2 Gigabytes in size. View Pro can also be used to view data from a LoggerNet database table created with LNDB.

Once a data file or database table is opened, data can be printed or graphed in several different graph types including Line Graphs, Histograms, XY Plots, FFTs (Fast Fourier Transforms), or Rainflow Histograms as appropriate for the data type. (Note that these graphical windows are only available for *.DAT files and database tables. Other file types are viewable only as text.)

Since View Pro is primarily a file viewing utility, a file cannot be edited or saved using this program.

View Pro can also be run as a stand-alone program by using Windows Explorer and double-clicking on the ViewPro.exe file in the C:\Program Files\CampbellSci\View Pro folder. A desktop shortcut can be created by right-clicking on this file in Windows Explorer and choosing **Create Shortcut**.

View Pro is closed by selecting **File | Exit** from the menu or pressing the red **X** in the upper right-hand corner. When View Pro is closed, all open graphs and data files will also be closed.

## 6.2 The Toolbar

Many of View Pro's features can be accessed from the toolbar. The main View Pro toolbar includes the following icons:

**Open**. Brings up a dialog box from which you can choose a data file to open.

**Copy**. Copies selected text to the clipboard. Text is selected by dragging the mouse pointer across the desired selection. Multiple columns in a data file can be selected by dragging the mouse pointer across the column headings.

**Cascade**. Rearranges all open, non-minimized data file windows so that the title bar of each window is visible. Windows cascade down and to the right starting from the upper left corner.

**Tile Vertically**. Rearranges all open, non-minimized data file windows as non-overlapping vertical tiles. This makes them all visible at the same time.

**Tile Horizontally**. Rearranges all open, non-minimized data file windows as non-overlapping horizontal tiles. This makes them all visible at the same time.

**Refresh Current File**. Refreshes an open data file. This is useful if you are viewing a file, and additional data has been stored since the file was first opened.

**Print Preview**. Displays how the currently selected data file will appear when it is printed.

**Print**. Brings up a dialog box that allows you to print the currently selected data file.

**New Line Graph**. Brings up a Line Graph window from which you can graph data values on the ***y-axis*** against their timestamps on the ***x-axis***.

**New Histogram**. Brings up a Histogram window from which you can view Histogram data values.

**New XY Plot**. Brings up an XY Plot window from which you can plot data values on the ***y-axis*** against another specified data value on the ***x-axis***.

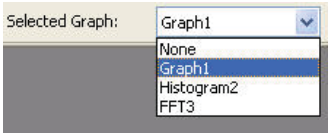
**New Rainflow Histogram**. Brings up a Rainflow Histogram window from which you can view Rainflow Histogram data values.

**New FFT**. Brings up an FFT window from which you can view FFT data values.

**Keep Selected Graph On Top**. When data is being graphed and this option is selected, the currently selected graph will always be at the forefront of the View Pro program.

This is a toggle button. The button icon will have a green check mark through it when the option is currently selected.

**Reset Grid for New Selections**. If no graphs are open, this button will clear all selections in all open files. If one or more graphs are open, this button will change the selected graph to "None" and clear all selections in all open files for the "None" selection set.




**Selected Graph**. Indicates which graph is currently selected. The drop-down list can be used to switch the currently selected graph to a different graph. Selected Graph can also be set to None in anticipation of making selections for a new graph.

This control is disabled if no graphs are opened.

**Bring Selected Graph to Front**. Brings the currently selected graph to the forefront of the View Pro program.

This control is disabled if no graphs are opened.

## 6.3 Opening a File

View Pro provides three ways to open a file. The one you use depends on the type of file being opened. Use **File | Open** to open a data file, **File | Open as Text** to open other types of files, or **File | Open as Hex** to view a binary or text file in hexadecimal format.

A file that has been recently opened, can be quickly opened again by selecting it from the **File | Recent Files** list. The file will be opened in the same format as it was originally opened (data, text, or hex).

No matter what option is used to open a file, it is closed by selecting **File | Close** from the menu or pressing the red **X** in the upper right-hand corner of the data file window.

### 6.3.1 Opening a Data File

To open a data file, click the File Open icon or select **File | Open** from the menu. (TOACI1, TOA5, TOB1, TOB2, and TOB3 data files can be opened with View Pro.) When a file is first opened, it is displayed in the data panel in a tabular format.

### 6.3.2 Opening Other Types of Files

To open a file that is not a data file (*.DLD, *.CSI, *.PTI, *.FSL, *.LOG, *.CRX) select **File | Open As Text** from the menu. Files opened in this mode cannot be graphed.

A file opened in this mode can be viewed only in its original format (i.e., as text). This mode is most often used to open files other than data files (or to quickly open data files, but without any of View Pro's graphing capabilities).

In text mode, data can be copied to the Windows clipboard and pasted into other applications. This is done by highlighting the text and choosing **Edit | Copy** from the menu.

### 6.3.3 Opening a File in Hexadecimal Format

To open a file in hexadecimal format select **File | Open As Hex** from the menu. This may be useful when viewing binary files.

## 6.4 Viewing a LoggerNet Database Table

A LoggerNet database table created using LNDB can be viewed in View Pro by selecting **View LoggerNet Database Table** from View Pro's File menu. You will need to select a database and then a table from that database.

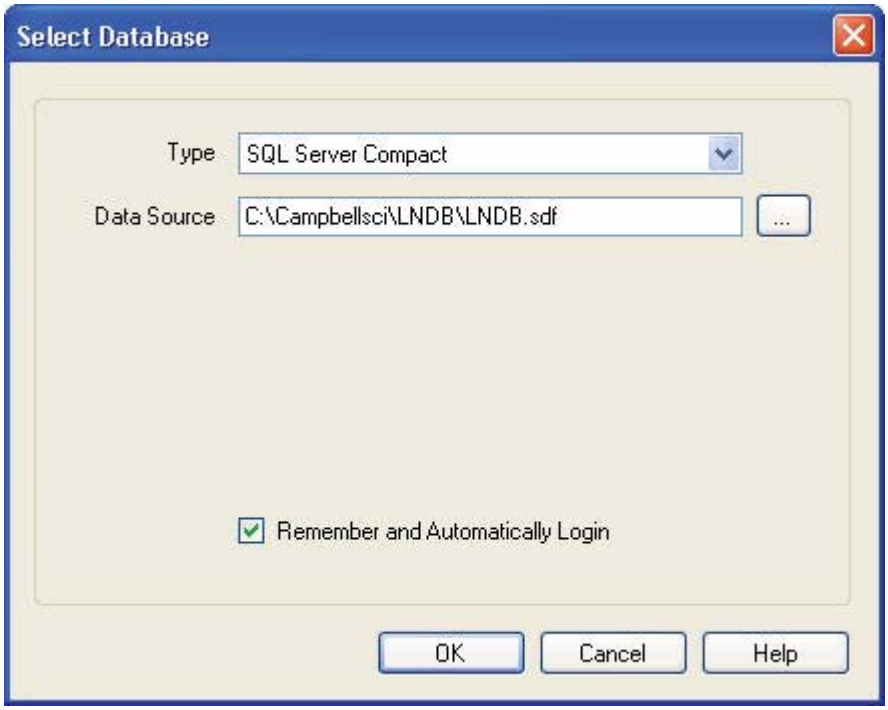
### 6.4.1 Selecting a Database

The Select Database dialog box comes up the first time you select **File | View LoggerNet Database Table**. (After a database has been selected, this menu item will bring up the Select Table dialog box. To view a table from a different database, you will need to press the **Change Database** button on the Select Table dialog box.)

View Pro supports SQL Server Compact, SQL Server, and MySQL databases.

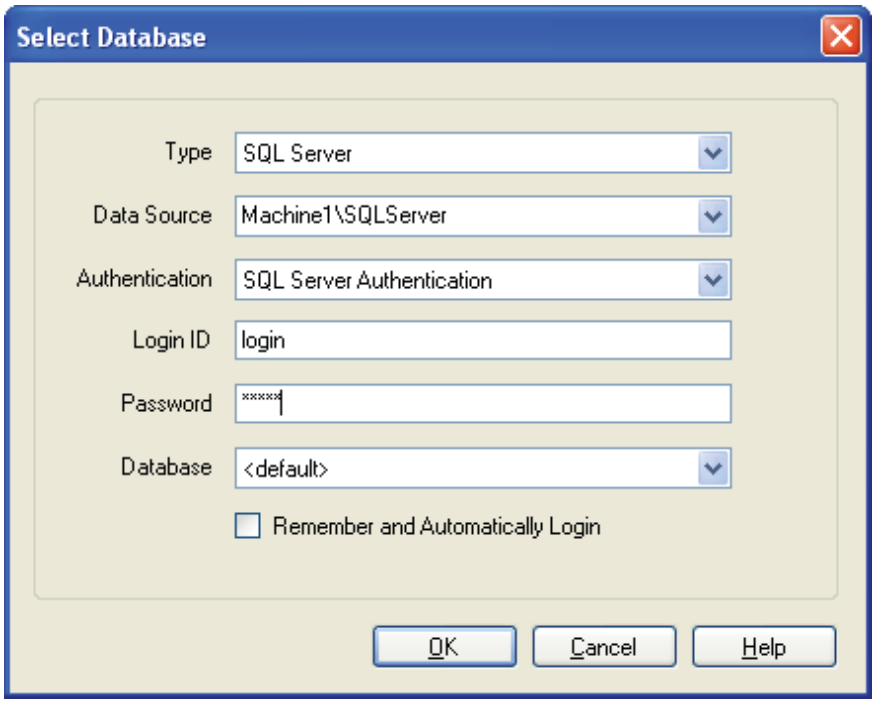
The information to enter changes depending on the database type as described below:

### SQL Server Compact



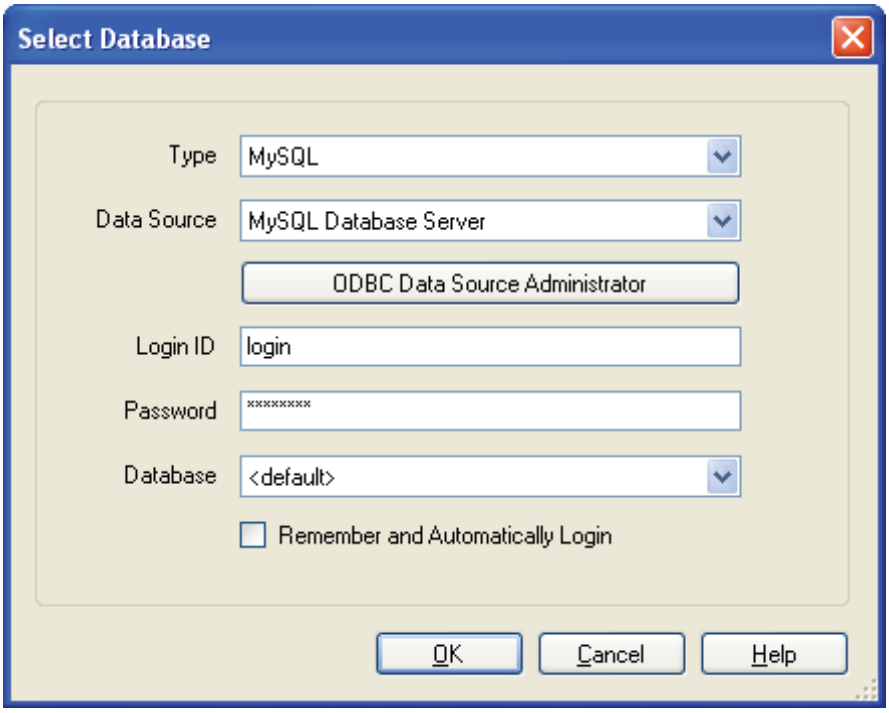

SQL Server Compact is an embedded database that just requires the selection of a filename. Press the **Browse** button to the right of the Data Source field to browse to the desired database.

### SQL Server



To configure a connection to SQL Server you must select a SQL Server instance. The list of published SQL Server instances is shown in the Data Source combo box. You can also type into the Data Source combo box, because the desired server might not be published. Windows Authentication or SQL Server Authentication can be selected. Windows Authentication does not require a username and password, but rather uses Windows user accounts to authenticate valid users. SQL Server Authentication requires a login ID and Password and is independent of Windows user accounts. You can select the <default> database or select a specific database from the Database combo box.

### *MySQL*



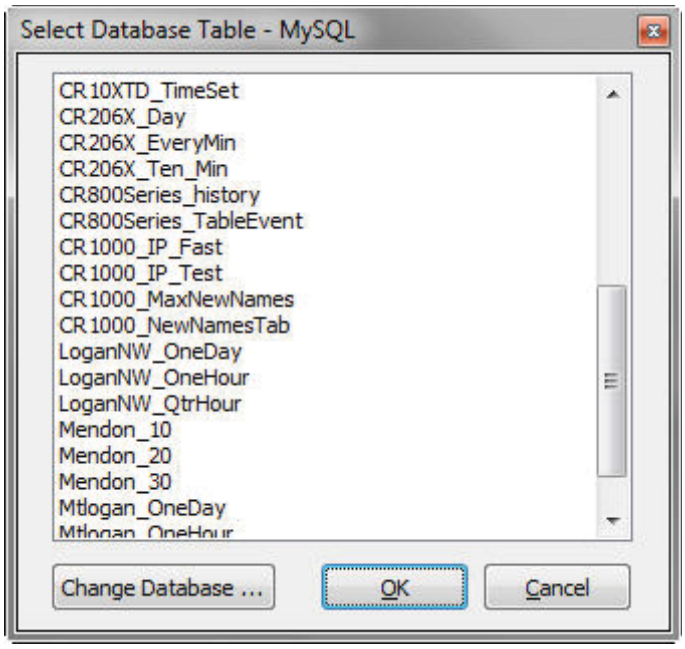

The MySQL connection is an ODBC connection. You must use the Windows ODBC Data Source Administrator to configure the database connection. Currently only system data sources are supported and show in the Data Source combo box. The Login ID and Password may be optional. They will be set to blank in the connection string. It has been found that when set to blank, the login id and password configured in the system data source are used. You can select the <default> database (default as configured in the data source) or select a different database.

### *Remember and Automatically Login*

If you select the **Remember and Automatically Login** check box, the Login ID and Password will be remembered and the next time the application starts an attempt will be made to login without showing the dialog again.

## 6.4.2 Selecting a Table

The Select Table dialog box comes up once a database has been selected.



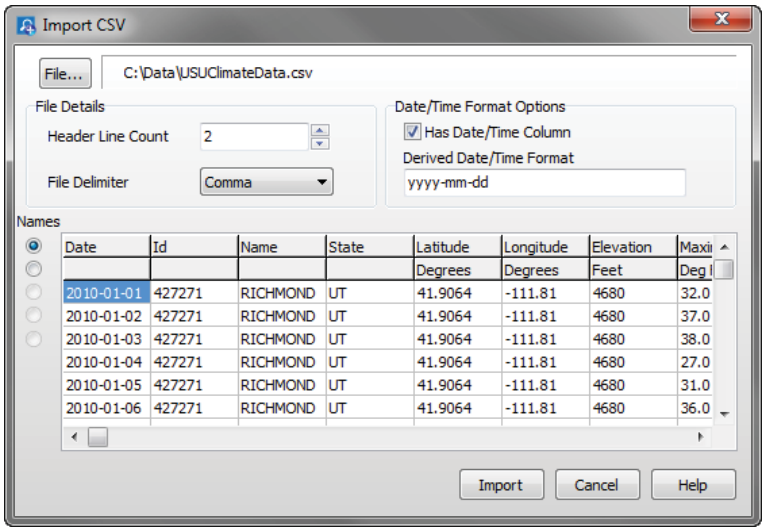

Select the database table that you wish to view and then press the **OK** button.

To select a table from a different database, press the **Change Database** button.

## 6.5 Importing a CSV File

The **File | Import CSV** menu item can be used to import A CSV (Comma Separated Value) file into View Pro.

When you select the **File | Import CSV** menu item, a browser will be displayed allowing you to browse to the CSV file to be imported. By default, only files with a .csv extension will be shown. If your file does not have a .csv extension, you will need to use the drop-down list box to select .txt files or all files.



**File**

The name of the file to be imported. Press the **File** button to bring up a browser to select the desired file.

**Header Line Count**

CSV files may have multiple header lines or no header line. Use the **Header Line Count** field to designate how many header lines your file contains before the data begins.

**File Delimiter**

Although CSV files are by definition comma delimited, other file delimiters (tab or space) can be selected in the **File Delimiter** drop-down list box.

**Date/Time Format Options**

When importing a CSV file, View Pro will attempt to derive a timestamp from data in the first column, if common timestamp delimiters exist in the data. If View Pro is able to derive the timestamp, the **Has Date/Time Column** check box will be checked and the derived timestamp format will be displayed in the **Derived Date/Time Format** field using the codes shown below. Any errors in the derived format can be corrected in this field.

If the first column contains a timestamp, but does not have the common timestamp delimiters that allow View Pro to determine that it is a timestamp, you can manually check the **Has Date/Time Column** check box and then input the appropriate codes in the **Derived Date/Time Format** field to designate the format of the timestamp.

If the first column of data does not contain a timestamp, leave the **Has Date/Time Column** check box unchecked. You will still be able to import the file into View Pro and view the data. However, you will not be able to graph the data.

*Date and Time Format Codes*

yy = Year last 2 digits
yyyy = Year as 4 digits
m = Month number no-leading 0
mm = Month number as 2 digits
mmm = Month using short form (Jan)
mmmm = Month using long form (January)
d = Day number no-leading 0
dd = Day number as 2 digits
ddd = Day using short day names (Sun)
dddd = Day using long day names (Sunday)
h = Hour with no leading 0's
hh = Hour as 2 digits
n = Minute with no leading 0's
nn = Minute as 2 digits
s = Seconds with no leading 0's
ss = Seconds as 2 digits
z = Milli-seconds with no leading 0's
zzz = Milli-seconds as 3 digits

**Names**

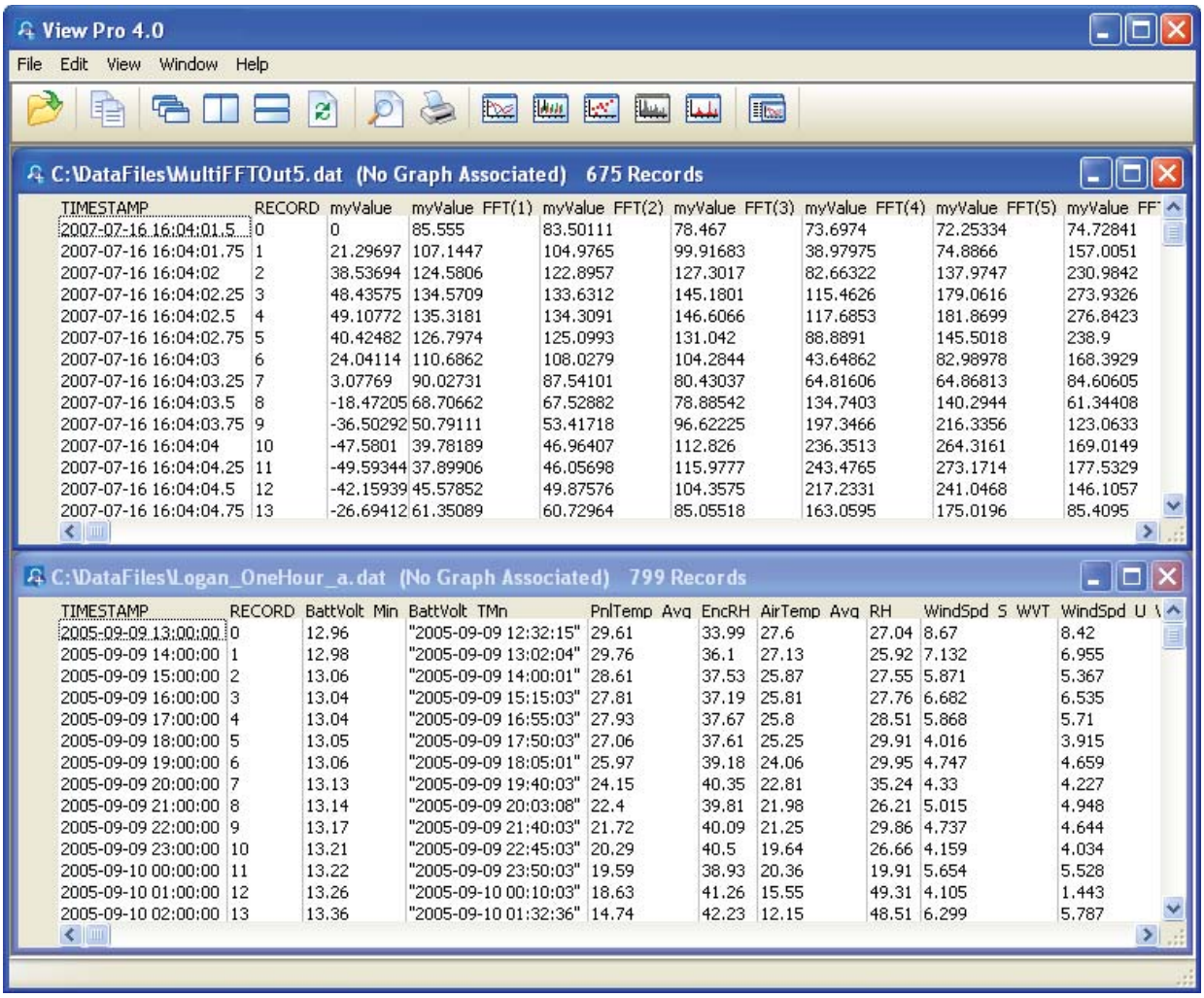
The header line that contains column names is designated by selecting an option button on the left of the preview grid under **Names**.

**Import**

After all of the settings have been specified, press the **Import** button to import the CSV file into View Pro.

## 6.6 Data View

The initial display for data files in View Pro is as normal text in a grid format. The following figure shows the View Pro main screen with two data files open. The data file windows have been tiled horizontally.




### Array-based Data Files

When opening a data file from an array-based datalogger, you will be given the option of loading an FSL (Final Storage Label) file. The FSL file will be used to provide column headings. (The *.FSL file is created when a datalogger program is compiled in Edlog or ShortCut.)

If a data file is opened that contains multiple arrays, the entire data file will be opened in one window. In addition, each array will be opened in a separate window. The window containing the entire data file is for viewing only. Data must be graphed from the individual array windows.

Array-based data files do not contain timestamps. If an FSL file is associated with the data file, View Pro will try to extract timestamps from the appropriate columns. You can select **Array Definitions** from View Pro's View menu to specify how the timestamps are created. Note that if no timestamps are used, data cannot be graphed.

## 6.6.1 Column Size

When a data file is opened, the columns are autosized to fit the data. Column sizes can be changed by dragging a column divider bar to the desired location.

If column sizes have been changed, they can be returned to the default sizes by selecting **View | Autosize Columns** from the menu.

## 6.6.2 Header Information

By default, only column names are shown for each column in the data file. Selecting **View | Show Full Header** will show full header information for each column in the data file, including units and field names, if available in the *.DAT file.

This is a toggle menu item. There will be a check mark next to the item, when it is active. Deactivate it by selecting it again.

## 6.6.3 Row Shading

Selecting **View | Row Shading** will shade every other row in the data file.

This is a toggle menu item. There will be a check mark next to the item, when it is active. Deactivate it by selecting it again.

## 6.6.4 Locking the TimeStamp Column

Selecting **View | Lock TimeStamp Column** will lock the timestamp column on the left of the data file so that it remains visible as you scroll through the columns in the data file.

This is a toggle menu item. There will be a check mark next to the item, when it is active. Deactivate it by selecting it again.

## 6.6.5 File Information

Selecting **View | View File Information** from the menu will display information on the currently selected data file including file name and file format, and information about the datalogger and program that generated the data file such as station name, table name, datalogger model, datalogger OS version, program name, and program signature.

## 6.6.6 Background Color

The background color used for the currently selected data panel can be changed by selecting **View | Background Color** from the menu.

## 6.6.7 Font

The font used for the data panel can be changed with the font selection dialog box. Select **View | Font** from the menu to change the font used for the printer and data panel. Normal font options such as color, bold, underline and italic are also available.

## 6.6.8 Window Arrangement

When multiple data files are opened, they are arranged so that the title bar of each window is visible and they cascade down and to the right starting from the upper left corner. Pressing the **Cascade** button or choosing Window | Cascade from the View Pro menu will return the data windows to this default arrangement at any time.

Windows can be tiled horizontally as non-overlapping horizontal tiles by pressing the **Tile Horizontally** button or choosing **Window | Tile Horizontally** from the menu. They can be tiled vertically as non-overlapping vertical tiles by pressing the **Tile Vertically** button or choosing **Window | Tile Vertically** from the menu.

A data window can be moved manually by clicking on the title bar and dragging it to the desired location.

The data view window can be kept in front of any opened graphs by selecting **View | Keep Data on Top** from the menu.

# 6.7 Graphs

Once a data file is opened, data values can be displayed in several different graphical forms including a Line Graph, Histogram, XY Plot, Rainflow Histogram, or FFT. Each of these are launched from a button on the View Pro toolbar. Graph buttons are enabled only if the graph type is appropriate for the opened data file. For instance, if there is no data appropriate for an FFT, the **FFT** button will be disabled on the View Pro toolbar. Multiple instances of each type of graph can be launched.

**Graphing Data from Multiple Data Files**

Data from multiple data files can be displayed in a single graph. This is done by opening multiple data files and selecting data to be graphed as described for each graph type below. This may be useful when comparing data from multiple datalogger stations.

**Options**

Each type of graph has several different options that can be set by the user. Options that apply to the entire graph are generally set from a dialog box that is opened by pressing the **Options** button at the lower-left of the graph window, by pressing the **Graph Options** button on the toolbar, or by right-clicking on the graph and choosing **Options**. Note that the XY Plot does not have an **Options** button, so the Options dialog box is opened by pressing the **Graph Options** button on the toolbar or right-clicking on the graph.

Options that apply to individual traces are set by selecting the trace in the list of data values being graphed, and then pressing the **Edit** button located below that list. The XY Plot does not have user-configurable trace options.

For information on graph options refer to the online help. Help for a graph can be accessed by pressing the **?** button in the upper-right corner of the graph.

**Zoom Feature**

You can zoom in on a particular area of a graph by holding the left mouse button and dragging the mouse cursor from top-left to bottom-right over the area to be zoomed. Dragging the mouse cursor from bottom-right to top-left will undo the zoom. This can also be accomplished by pressing the **Undo Zoom** button on the toolbar.

For a Rainflow Histogram or for a Histogram or FFT in 3D View, you can also zoom in and out by using the **Page Down** and **Page Up** buttons on your keyboard.

**Rotation**

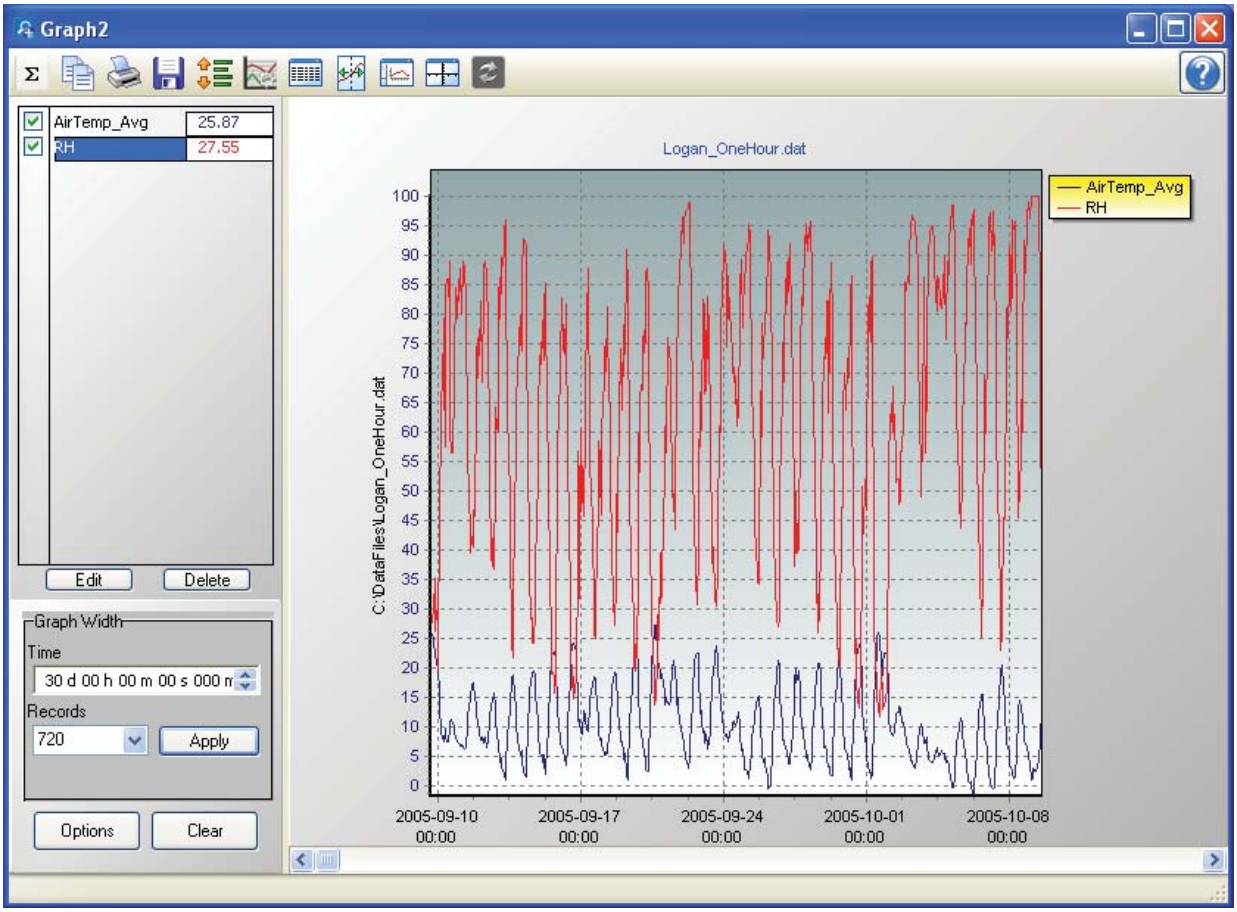
For a Rainflow Histogram or for a Histogram or FFT in 3D View, you can rotate the graph by using the scroll bars at the bottom and right of the graph.

**Keeping Graph on Top**

The currently selected graph can be kept in front of the data view window by selecting **View | Keep Graph on Top** from the menu or pressing this button on the main View Pro toolbar.

## 6.7.1 Line Graph

From the Line Graph screen, you can graph data values on the **y**-axis against their timestamps on the **x**-axis.




### 6.7.1.1 Selecting Data to be Graphed

Data value(s) are added to a graph by clicking the column heading(s) in the data panel with a single mouse click. The selection will be highlighted and will automatically be added to the currently selected graph. (The currently selected

graph is indicated on the main View Pro toolbar, and can be changed from the drop-down list.)

Multiple columns can be added by holding the Ctrl key and dragging the mouse pointer over the column headings. A partial column can be added by dragging the mouse pointer over the desired values. (Note that once a partial column is selected, it can be extended by holding the Shift key and clicking at a point below the current selection. The selection will be extended to that point.) Multiple partial columns can be added by holding the Ctrl key and dragging the mouse pointer over the desired values.

Data from multiple data files can be displayed in a single graph. This is done by opening multiple data files and selecting data to be graphed as described above. This may be useful when comparing data from multiple datalogger stations.

**Creating Multiple Line Graphs**

To open an additional Line Graph, select "**None**" from the **Selected Graph** drop-down menu on the main View Pro toolbar. The highlighting in the data file will be cleared. Select the data that you would like graphed as described above and then press the **Line Graph** button. A new Line Graph will be created with your selection(s) graphed. You may then continue adding selections to the Line Graph as described above.

An unlimited number of Line Graphs can be opened using this same process.

You can navigate between multiple graphs by clicking on a graph or by choosing a graph from the **Selected Graph** drop-down list on the View Pro toolbar and pressing the **Bring the selected graph to the front** button.

**NOTE** The highlighted selections in the data files will always indicate the values being graphed in the currently selected graph

**Deleting Traces from a Graph**

A data value can be deleted from a graph by selecting it in the list of values being graphed (on the left side of the graph window) and then pressing the **Delete** button, by right-clicking on this name and choosing **Delete Selection**, by left-clicking on the column heading in the data panel, or by right-clicking on the column in the data panel and choosing **Clear Selection**.

### 6.7.1.2 Graph Width

The **Graph Width** box is used to set the width of a Line Graph either as a function of time in days, hours, minutes, seconds, and milliseconds or by the number of records being displayed. If one of these two field is changed, the other field will automatically change accordingly when the **Apply** button is pressed.

### 6.7.1.3 Scrolling

You can scroll through the graph by using the scroll bar at the bottom of the graph window. Scrolling the graph will scroll the data on the data panel as

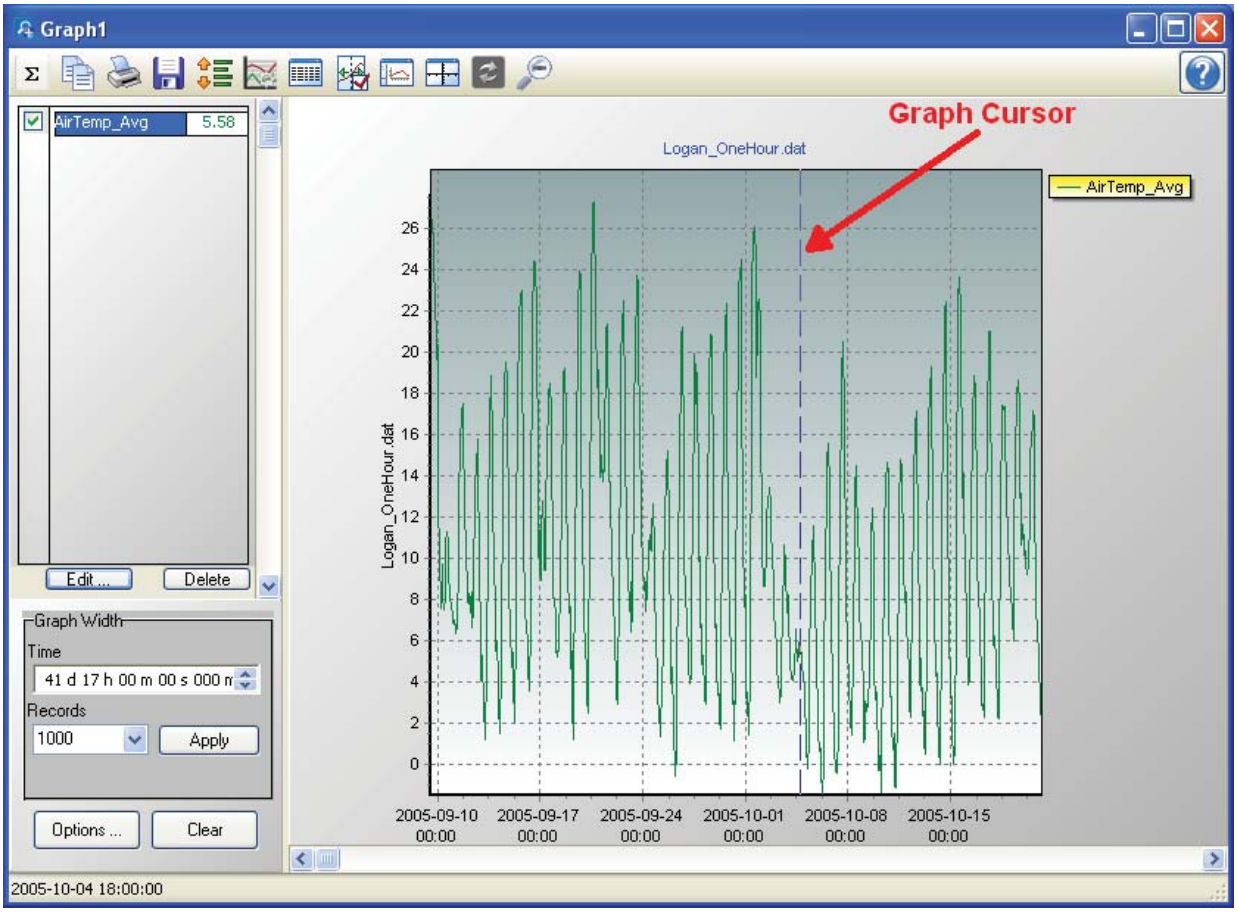
well. Conversely, scrolling through a data panel will also scroll the currently selected graph.

Graph scrolling can be disabled by pressing the Lock Scrolling icon on the toolbar. The icon will change to . Pressing the icon again will re-enable scrolling.

When locked, the graph will not scroll. However, moving the scroll bar on either the graph or the data file will still scroll the data file.

### 6.7.1.4 Graph Cursor

Pressing the Graph Cursor icon on the toolbar will show the graph cursor. As shown in the following figure, the graph cursor is a vertical line extending from the top to the bottom of the graph display. When visible, you can click and drag the cursor across the graph. Data values at the current cursor position will be shown in the table to the right of the graph.




### 6.7.1.5 Line Graph Toolbar

The Line Graph toolbar includes the following icons:

Σ **Statistics**. Displays statistics for each trace including Average, Standard Deviation, Minimum and Maximum. Note that these

statistics are for the data displayed in the graph. They are not statistics for the entire column(s) of data. Note that an asterisk next to a statistics value indicates that the trace contains one or more bad data values (i.e., NAN, INF, etc.).

The **Statistics** button is disabled when a graph is in a zoomed stated.

**Copy to Clipboard**. Places the graphic on the Windows clipboard. It can then be pasted into other applications.

**Print**. Prints the graph. Print options can be set before printing begins.

**Export**. Allows the graph to be exported in a choice of text or graphical formats.

**Lock Scrolling**. Locks and unlocks the scroll bar at the bottom of the graph.

When unlocked, moving the scroll bar on either the graph or the data file will scroll both the graph and the data file.

When locked, the graph will not scroll. However, moving the scroll bar on either the graph or the data file will still scroll the data file.

This is a toggle button. When the lock is currently enabled, there will be a lock on top of the icon.

**Graph Options**. Opens a dialog box from which you can set properties for the graph including colors, margins, titles, legend, etc. This dialog box can also be brought up by pressing the **Options** button.

**Show Table**. Brings the main View Pro window in front of other windows, making the data file(s) visible.

**Show/Hide Graph Cursor**. A toggle button that shows and hides the graph cursor. The graph cursor is a vertical line extending from the top to the bottom of the graph display. When visible, you can click and drag the cursor across the graph. Data values at the current cursor position will be shown in the table.

**Show/Hide Gradient**. A toggle button that turns on and off the gradient background of the graph. It may be useful hide the gradient, when printing the graph.

(Common)

**Common/Independent Axes**. When multiple data values are being graphed, determines whether they have common y-axes or independent y-axes.

When using common y-axes, one scale will apply to all traces assigned to the left y-axis and one scale will apply to all traces assigned to the right y-axis.

(Independent)

When using independent y-axes, the scale shown will apply only to the last selected trace assigned to that axis. (A trace is selected by clicking on its name in the list above the **Edit** and **Delete** buttons.)

**Synchronize Axes**. Only enabled when data is being graphed from multiple data files and all of the data files have an overlapping time period.

When a graph contains traces from multiple data files, a box with a drop-down list will appear in the Graph Width options box. The data file chosen from the drop-down list indicates which data file and graph will be scrolled by the graph scroll bar. After using the scroll bar to scroll the indicated graph, the **Synchronize Axes** button may be pressed to synchronize the timestamps of the remaining graph(s) and data file(s) so that they are all displaying data from the same time period.

When the data panels and graphs are currently synchronized, the button icon will have a check mark in the bottom right corner.

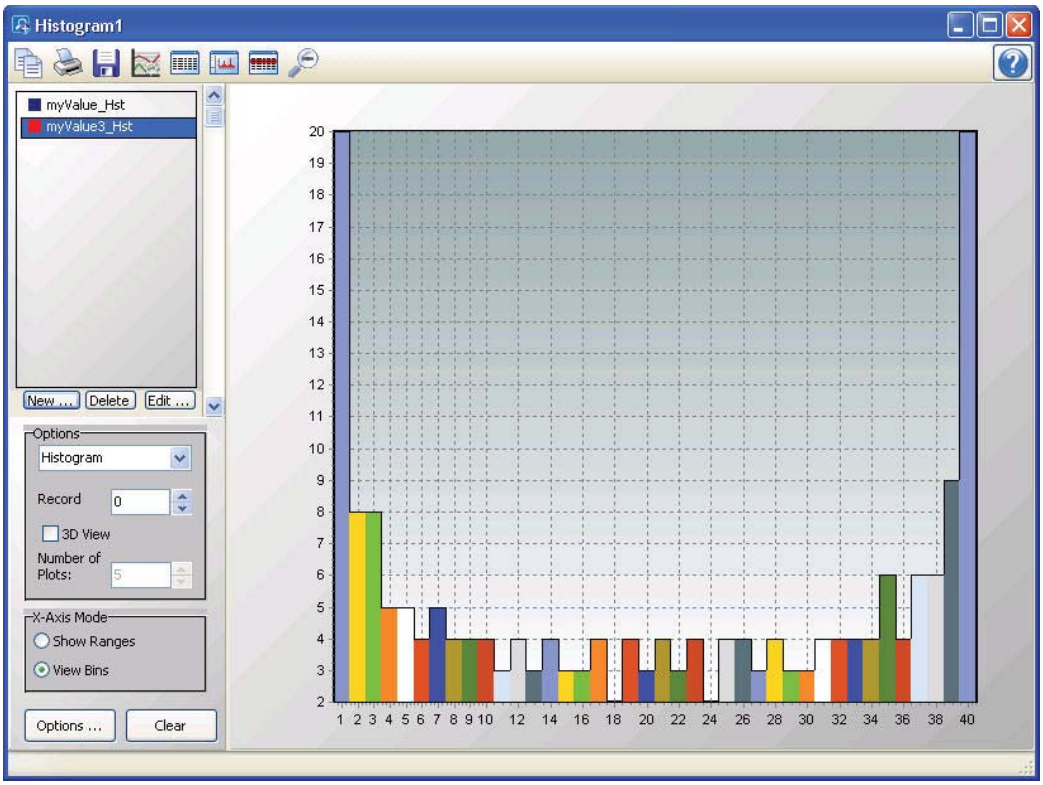
**Undo Zoom**. Returns the graph to its original state after zooming.

## 6.7.2 Histogram

From the Histogram screen, you can view histogram data. The **Histogram** button on the toolbar will be enabled if there is at least one valid histogram in the currently selected data file.

**NOTE** View Pro does not create histogram data from time series information. It only displays histogram data contained in a *.DAT file. Histogram data in a *.DAT file is created by using the CRBasic Histogram instruction in a CRBASIC program Data Table.

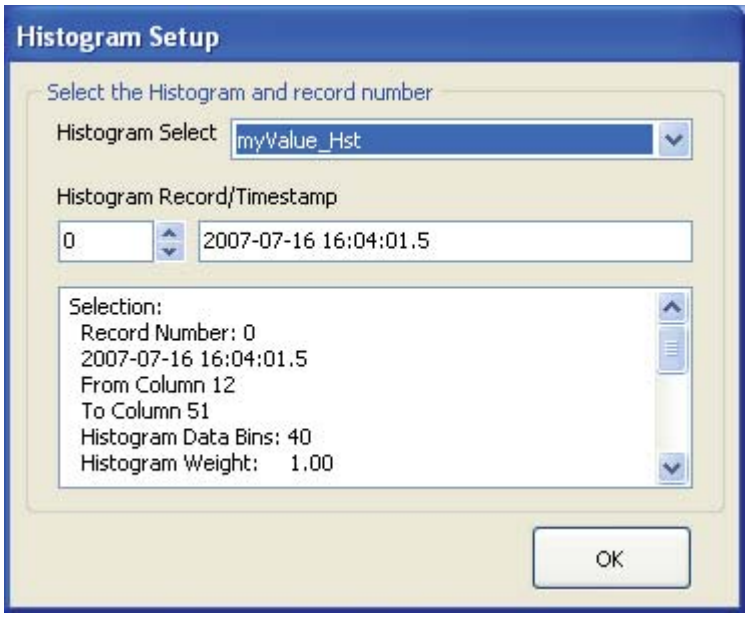
**NOTE** View Pro cannot display histograms from a TOACI1 file.




### 6.7.2.1 Selecting Data to be Viewed

**From a Histogram Screen**

When a Histogram screen is first opened with no histogram records selected in the data file, the Histogram Setup dialog box will open which allows you to set up the Histogram. The first option is a drop-down list that shows the available histograms in the currently selected data file. Select the histogram that you would like to view. The second option allows you to choose which record of the histogram you would like to view initially. Type in a number directly or use the arrow keys to the right of the box to change the value.




This dialog box can also be opened from a button, , on the Histogram toolbar. This allows you to change the options for the histogram record that is selected in the list on the left side of the Histogram screen.

Additional histogram records can be added by pressing the **New** button. (These additional records can be from either the same histogram or a different histogram in your data file.) You can then choose which histogram record is being displayed by selecting it in the list.

**From the Data Grid**

You can also select histogram records directly from a data file to be displayed on a Histogram screen. Clicking on any data value in a histogram record will select that histogram record. Histogram records can be selected before the Histogram screen is opened with the Selected Graph set to None. When the Histogram screen is opened, all selected histogram records will be listed on the left side of the Histogram screen. A histogram record can then be displayed by clicking on it in the list. Once the Histogram screen is opened, additional histogram records can be added to the Histogram screen by selecting them in the data file as described above.

**NOTE**

All histogram records from the same histogram will have the same default name in the list. They can be distinguished by the colored boxes next to their names. Each box is the same color with which that histogram record is highlighted in the data file. It is also the color with which that histogram record is displayed if the "Use Selection Color" option is chosen in the Selection Properties dialog box. The color associated with a histogram record can also be changed from this dialog box. (The Selection Properties dialog box is opened by clicking on the histogram record in the list and then pressing the **Edit** button.)

**Deleting Records from a Histogram Display**

A Histogram can be deleted from a graph by selecting it in the list of values being graphed (on the left side of the graph window) and then pressing the **Delete** button, by right-clicking on this name and choosing **Delete Selection**, or by right-clicking on the record in the data panel and choosing **Clear Selection**.

### 6.7.2.2 Options

From the main Histogram screen, you can set the Histogram type to Area, Histogram, Line, or Bar. You can use the arrow buttons to the right of **Record** to scroll through records of the Histogram.

You can also determine whether the Histogram is viewed in 2D or 3D. Selecting the **3D View** check box will enable 3D View. Clearing the check box will cause the Histogram to be viewed in 2D. When in 3D View, the **Number of Plots** field will determine how many records are viewed in the Histogram. In 3D View, scroll bars appear on the bottom and right of the screen which allow the Histogram to be rotated.

You can also determine how the labels on the X-Axis are displayed. Select **Show Ranges** to have ranges of data values shown on the X-Axis. Select **View Bins** to have bin numbers shown on the X-Axis.

### 6.7.2.3 Histogram Toolbar

The Histogram toolbar includes the following icons:

**Copy to Clipboard**. Places the Histogram graphic on the Windows clipboard. It can then be pasted into other applications.

**Print**. Prints the Histogram. Print options can be set before printing begins.

**Export**. Allows the Histogram to be exported in a choice of text or graphical formats.

**Histogram Options**. Opens a dialog box from which you can set properties for the Histogram including scaling, colors, margins, titles, etc. This dialog box can also be brought up by pressing the **Options** button.

**Show Table**. Brings the main View Pro window in front of other windows, making the data file(s) visible.

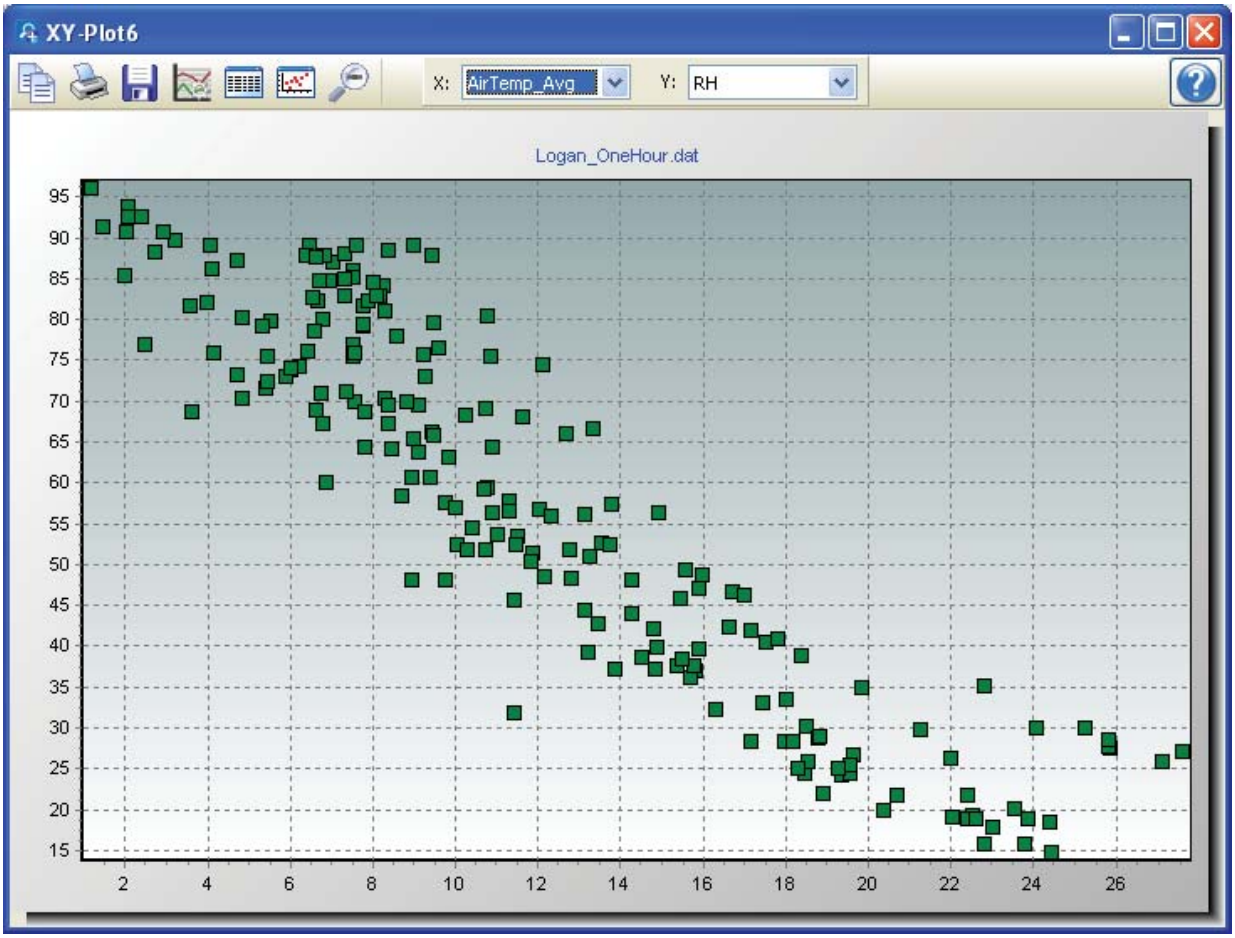
**Show/Hide Gradient**. A toggle button that turns on and off the gradient background of the Histogram. It may be useful to hide the gradient, when printing the Histogram.

**Modify Selection**. Brings up the Histogram Setup dialog box from which you can change the options for the selection.

**Undo Zoom**. Returns the Histogram to its original state after zooming.

## 6.7.3 XY Plot

From the XY Plot screen, you can graph a data value on the **y**-axis against a different data value on the **x**-axis. The user specifies what will be used for both the X axis data value and the Y axis data values. Each Y axis data value is plotted against the X axis data value with the identical timestamp.




### 6.7.3.1 Selecting Data to be Plotted

To select the data value(s) to be plotted, highlight the column headings in the data file with a single mouse click. Each column that is selected in the data file, will be added to the XY Plot in both the **X** and **Y** drop-down lists. Select from the **X** and **Y** drop-down lists the values to be used for the X axis and Y axis, respectively.

### 6.7.3.2 XY Plot Toolbar

The XY Plot toolbar includes the following icons:

**Copy to Clipboard**. Places the XY Plot graphic on the Windows clipboard. It can then be pasted into other applications.

**Print**. Prints the XY Plot. Print options can be set before printing begins.

**Export**. Allows the XY Plot to be exported in a choice of text or graphical formats.

**Graph Options**. Opens a dialog box from which you can set properties for the XY Plot including colors, margins, titles, symbols, marks, scaling, etc.

**Show Table**. Brings the main View Pro window in front of other windows, making the data file(s) visible.

**Show/Hide Gradient**. A toggle button that turns on and off the gradient background of the XY Plot. It may be useful to hide the gradient, when printing the XY Plot.

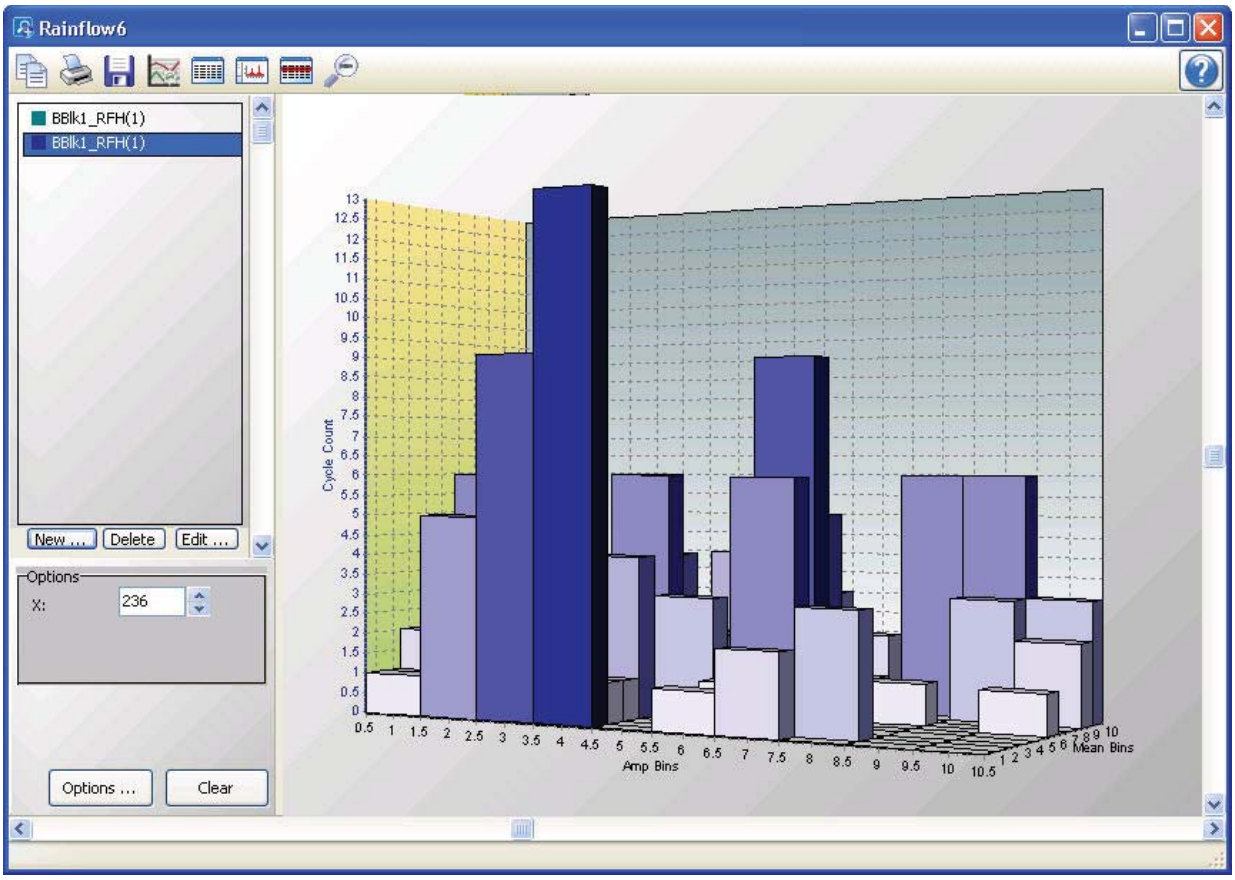
**Undo Zoom**. Returns the XY Plot to its original state after zooming.

## 6.7.4 Rainflow Histogram

From the Rainflow Histogram screen, you can view rainflow histogram data. The **Rainflow Histogram** button on the toolbar will be enabled if there is at least one valid rainflow histogram in the currently selected data file.

A Rainflow Histogram is a 3D representation based on the rainflow counting algorithm of Endo and Matsuishi which was first published in 1968. These diagrams can be used to monitor fatigue levels of structures under stress such as components of a large bridge.

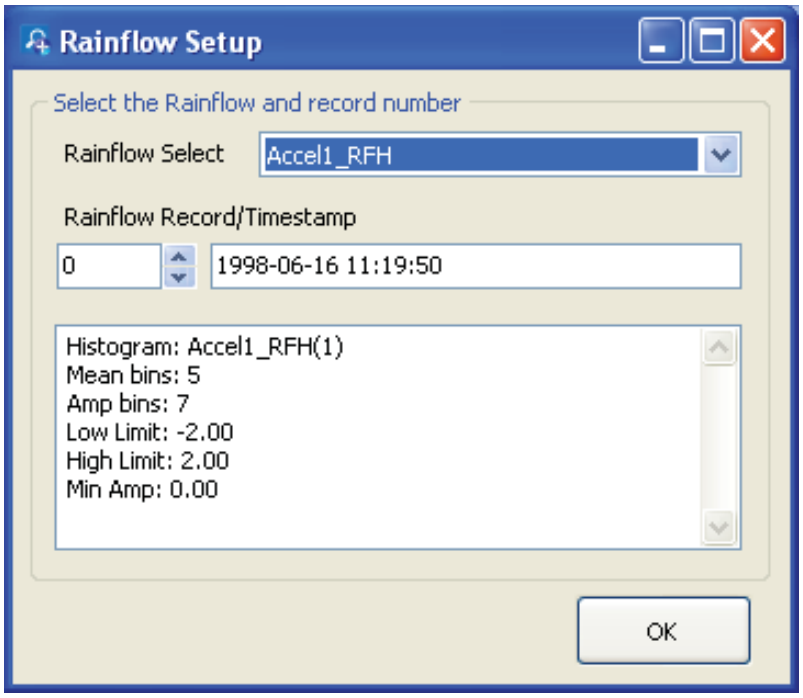
**NOTE** ViewPro does not create rainflow histogram data from time series information. It only displays rainflow histogram data contained in a *.DAT file. Rainflow Histogram data in the *.DAT file is created by using the CRBasic Rainflow instruction in a CRBasic program Data Table.




### 6.7.4.1 Selecting Data to be View

**From a Rainflow Histogram Screen**

When a Rainflow Histogram screen is first opened with no rainflow histogram records selected in the data file, the Rainflow Histogram Setup dialog box will open which allows you to set up the Rainflow Histogram. The first option is a drop-down list that shows the available rainflow histograms in the currently selected data file. Select the rainflow histogram that you would like to view. The second option allows you to choose which record of the rainflow histogram you would like to view initially. Type in a number directly or use the arrow keys to the right of the box to change the value.




This dialog box can also be opened from a button, , on the Rainflow Histogram toolbar. This allows you to change the options for the rainflow histogram record that is selected in the list on the left side of the Rainflow Histogram screen.

Additional rainflow histogram records can be added by pressing the **New** button. (These additional records can be from either the same rainflow histogram or a different rainflow histogram in your data file.) You can then choose which rainflow histogram record is being displayed by selecting it in the list.

**From the Data Grid**

You can also select rainflow histogram records directly from a data file to be displayed on a Rainflow Histogram screen. Clicking on any data value in a rainflow histogram record will select that rainflow histogram record. Rainflow histogram records can be selected before the Rainflow Histogram screen is opened with the Selected Graph set to None. When the Rainflow Histogram screen is opened, all selected rainflow histogram records will be listed on the left side of the Rainflow Histogram screen. A rainflow histogram record can then be displayed by clicking on it in the list. Once the Rainflow Histogram screen is opened, additional rainflow histogram records can be added to the Rainflow Histogram screen by selecting them in the data file as described above.

**NOTE** All rainflow histogram records from the same rainflow histogram will have the same default name in the list. They can be distinguished by the colored boxes next to their names. Each box is the same color with which that rainflow histogram record is highlighted in the data file. It is also the color with which that rainflow histogram record is displayed. The color associated with a histogram record can be changed from the Selection Properties dialog box. (The Selection Properties dialog box is opened by clicking on the rainflow histogram record in the list and then pressing the **Edit** button.)

**Deleting Records from a Rainflow Histogram Display**

A Rainflow Histogram can be deleted from a graph by selecting it in the list of values being graphed (on the left side of the graph window) and then pressing the **Delete** button, by right-clicking on this name and choosing **Delete Selection**, or by right-clicking on the record in the data panel and choosing **Clear Selection**.

### 6.7.4.2 Options

From the main Rainflow Histogram screen, you can use the arrow buttons to the right of **X** to scroll through records of the Rainflow Histogram.

### 6.7.4.3 Rainflow Histogram Toolbar

The Rainflow Histogram includes the following toolbar icons:

**Copy to Clipboard**. Places the Rainflow Histogram graphic on the Windows clipboard. It can then be pasted into other applications.

**Print**. Prints the Rainflow Histogram. Print options can be set before printing begins.

**Export**. Allows the Rainflow Histogram to be exported in a choice of text or graphical formats.

**Graph Options**. Opens a dialog box from which you can set properties for the Rainflow Histogram including scaling, colors, margins, titles, etc. This dialog box can also be brought up by pressing the **Options** button.

**Show Table**. Brings the main View Pro window in front of other windows, making the data file(s) visible.

**Show/Hide Gradient**. A toggle button that turns on and off the gradient background of the Rainflow Histogram. It may be useful hide the gradient, when printing the Rainflow Histogram.

**Modify Selection**. Brings up the Rainflow Histogram Setup dialog box from which you can change the options for the selection.

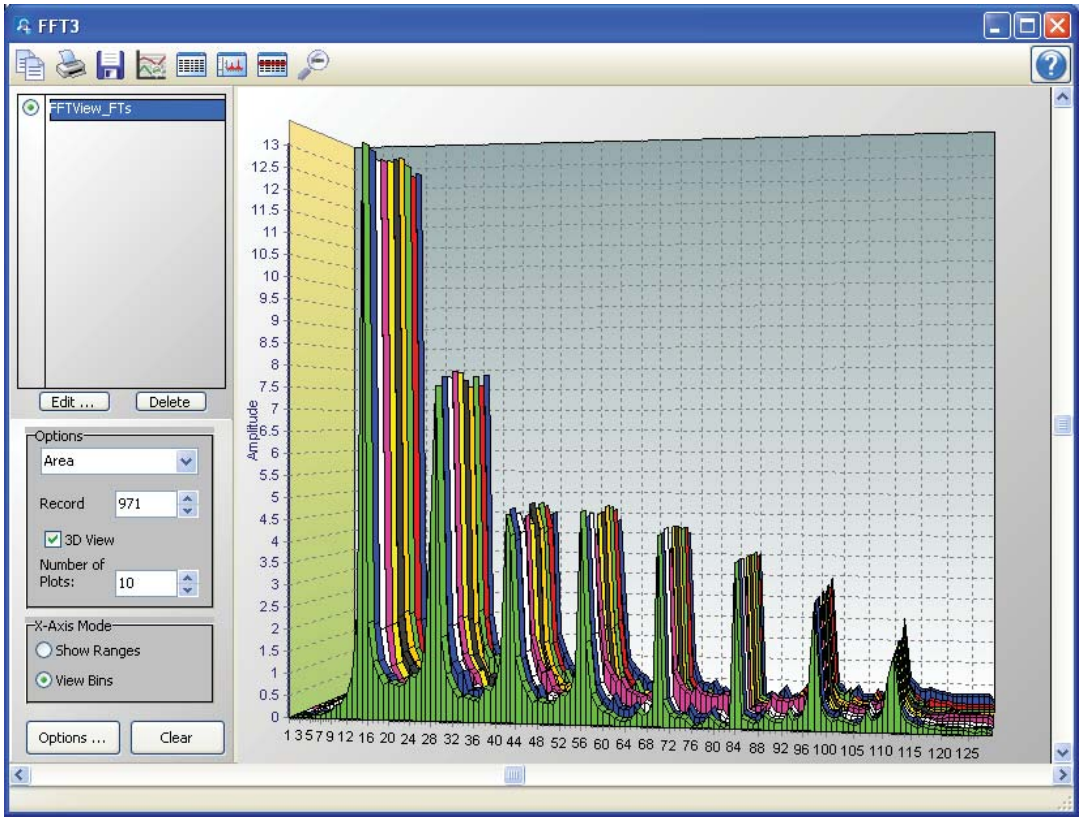
**Undo Zoom**. Returns the Rainflow Histogram to its original state after zooming.

## 6.7.5 FFT

From the FFT screen, you can view FFT data. The **FFT** button on the toolbar will be enabled if there is at least one valid FFT in the currently selected data file.

**NOTE** View Pro does not create FFT data from time series information. It only displays FFT data contained in a *.DAT file. FFT data in the *.DAT file is created by using the CRBasic FFT instruction in a CRBasic program Data Table.

**NOTE** View Pro cannot display FFTs from a TOACI1 data file.

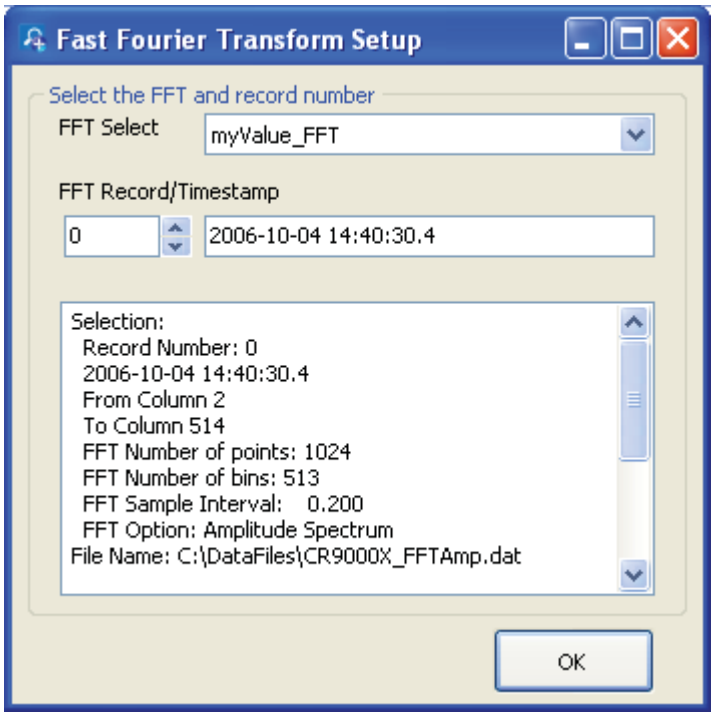



### 6.7.5.1 Selecting Data to be Graphed

**From an FFT Screen**

When an FFT screen is opened, a Fast Fourier Transform Setup dialog box will open which allows you to set up the FFT. The first option is a drop-down list that shows the available FFTs in the currently selected data file. Select the FFT that you would like to view. The second option allows you to choose which

record of the FFT you would like to view initially. Type in a number directly or use the arrow keys to the right of the box to change the value.




This dialog box can also be opened from a button, , on the FFT toolbar. This allows you to change the options for the FFT record that is selected in the list on the left side of the FFT screen.

Additional FFT records can be added by pressing the **New** button. (These additional records can be from either the same FFT or a different FFT in your data file.) You can then choose which FFT record is being displayed by selecting it in the list.

**From the Data Grid**

You can also select FFT records directly from a data file to be displayed on an FFT screen. Clicking on any data value in an FFT record will select that FFT record. FFT records can be selected before the FFT screen is opened with the Selected Graph set to None. When the FFT screen is opened, all selected FFT records will be listed on the left side of the FFT screen. An FFT record can then be displayed by clicking on it in the list. Once the FFT screen is opened, additional FFT records can be added to the FFT screen by selecting them in the data file as described above.

**NOTE** All FFT records from the same FFT will have the same default name in the list. They can be distinguished by the colored boxes next to their names. Each box is the same color with which that FFT record is highlighted in the data file. It is also the color with which that FFT record is displayed if the "Use Selection Color" option is chosen in the Selection Properties dialog box. The color associated with an FFT record can also be changed from this dialog box. (The Selection Properties dialog box is opened by clicking on the FFT record in the list and then pressing the **Edit** button.)

**Deleting Records from an FFT Display**

An FFT can be deleted from a graph by selecting it in the list of values being graphed (on the left side of the graph window) and then pressing the **Delete** button, by right-clicking on this name and choosing **Delete Selection**, or by right-clicking on the record in the data panel and choosing **Clear Selection**.

### 6.7.5.2 Options

From the main FFT screen, you can set the FFT type to Area, Histogram, Line, or Bar. You can use the arrow buttons to the right of **Record** to scroll through records of the FFT.

You can also determine whether the FFT is viewed in 2D or 3D. Selecting the **3D View** check box will enable 3D View. Clearing the check box will cause the FFT to be viewed in 2D. When in 3D View, the **Number of Plots** field will determine how many records are viewed in the FFT. In 3D View, scroll bars appear on the bottom and right of the screen which allow the FFT to be rotated.

You can also determine how the labels on the X-Axis are displayed. Select **Show Ranges** to have ranges of data values shown on the X-Axis. Select **View Bins** to have bin numbers shown on the X-Axis.

### 6.7.5.3 FFT Toolbar

The FFT includes the following toolbar icons:

**Copy to Clipboard**. Places the FFT graphic on the Windows clipboard. It can then be pasted into other applications.

**Print.** Prints the FFT. Print options can be set before printing begins.

**Export**. Allows the FFT to be exported in a choice of text or graphical formats.

**FFT Options**. Opens a dialog box from which you can set properties for the FFT including scaling, colors, margins, titles, etc. This dialog box can also be brought up by pressing the **Options** button.

**Show Table**. Brings the main View Pro window in front of other windows, making the data file(s) visible.

**Show/Hide Gradient**. A toggle button that turns on and off the gradient background of the FFT. It may be useful to hide the gradient, when printing the FFT.

**Modify Selection**. Brings up the FFT Setup dialog box from which you can change the options for the selection.

**Undo Zoom**. Returns the FFT to its original state after zooming.

# 6.8 Right-click-Menus

Right-clicking in View Pro will bring up a short cut menu. The options on this menu will vary depending on whether you right-click on a data panel, graph, or trace as described below.

## 6.8.1 Data View

Right-clicking in a data panel will bring up a menu with the following options:

**Define Selection**

Brings up a dialog box that allows you to define the records included and the color of the current data selection.

The records included can be defined by initial and ending time, time span, initial and ending record, or record span. If any of the date/time or record number options are changed, the other options will automatically adjust to reflect the change. Note that changes to the records included will not be reflected in the data panel, until the **Apply** button is pressed.

**Copy Selection**

Copies selected text to the Windows clipboard.

**Add Selections to Graph**

Adds the current selections to a graph. Click on or hover over this menu item to see a list of the opened graphs. Choose a graph to add the selections to. Note that this can be used to add selections made when Selected Graph is set to None to a graph. It can also be used to move selections from the currently-selected graph to a different graph.

**Clear All Selections**

Clears all selections in the currently selected data panel.

**Clear Selection**

In the currently selected data panel, clears the selection with focus. (This is the selection that has the dashed box around it. Left or right-click on a selection to give it focus.)

**Format Columns**

This option allows you to format a column in a binary (TOB) or CSIXML file. For a column containing a time, you can choose the date/time format option. For a column containing a floating point number, you can specify the number of decimal places, the number of leading zeroes, and whether the number is to be displayed in scientific notation. See ViewPro's online help for more information.

**Autosize Columns**

Returns columns to the default sizes. (This function can also be accomplished by selecting **View | Autosize Columns** from the menu.)

**Add Bookmark**

The user can quickly navigate to a bookmarked record by using the **Goto Bookmark** option. Choosing the **Add Bookmark** option will add a bookmark to a record. When a record is bookmarked, a numbered circle (beginning at 0) will appear to the left of the record.

A bookmark can also be added to the top visible record by typing Ctrl-Shift-n, where n is the number of the bookmark.

Note that bookmarks are not persistent and will be gone once the data panel or View Pro is closed.

**Goto Bookmark**

If the data panel contains one or more bookmarks, hovering over the **Goto Bookmark** menu item will bring up a list of the current bookmarks. Selecting a bookmark from the list will automatically move the data panel to that record.

The shortcut Ctrl-n, where n is the number of the bookmark, can also be used to move the data file to the desired bookmark.

**Delete Bookmark**

If the data panel contains one or more bookmarks, hovering over the **Delete Bookmark** menu item will bring up a list of the current bookmarks. Selecting a bookmark from the list will delete the bookmark.

**Delete All Bookmarks**

Deletes all bookmarks from the data panel.

**View Record**

Brings up the current record in a Record View window showing each column heading and the data value. (The Record View window can be brought up directly by clicking to the left of the record.)

### 6.8.2 Graphs

Right-clicking on a graph will bring up a menu from which you can choose **Export** to save the graph in a choice of formats, **Copy to Clipboard** to place the graph on the clipboard, **Print** to print the graph, or **Options** to bring up the graph's Options dialog box.

### 6.8.3 Traces

Right-clicking on a trace name in the list on the right side of a Line Graph, Histogram, Rainflow Histogram, or FFT brings up a menu from which you can choose **Edit Selection** to bring up the trace options dialog box, **Delete Selection** to delete the selection from the graph, or **Selection Summary** to see information about the trace, the data file, and the datalogger and program that generated the data file. For a Line Graph, this list also includes **Assign to Left Axis** to use the selected trace's scale on the left y-axis, **Assign to Right Axis** to use the selected trace's scale on the right y-axis, **Check All** to check all traces and make them visible on the graph, and **Uncheck All** to uncheck all traces and remove them from the graph.

## 6.9 Printing Options

### 6.9.1 Print Setup

Select **File | Print Setup** from the View Pro toolbar to set print options such as printer, paper size and source, orientation, duplex mode, pages per sheet, etc. Note that many of these options can also be set from the dialog box that is brought up when the **Print** button is pressed or from the Print Preview screen.

### 6.9.2 Printing Text

To print numerical data, press the **Print** button or select **File | Print** from the menu. A dialog box will appear allowing you to choose the printer, print range, number of copies, etc. After setting the properties, press **OK** to print the data.

To preview your data before printing, press the **Print Preview** button or select **File | Print Preview** from the menu. From Print Preview you can browse among the pages that will be printed and change the paper orientation if desired. You can zoom in on a particular area of the previewed page by left-clicking the page. You can zoom out by left-clicking with the **Shift** button pressed. You can pan across a page by right-clicking and dragging the page. To return to normal view, choose the Page Width or the Full Page icon. Simply press the **Print** button on the toolbar to print one or more pages. See the online help for details of the Print Preview options.

### 6.9.3 Printing Graphs

With a graph window opened, click the **Print** button to preview the printed page and set various printing options. Then select the **Print** button to print the graph. You can also right-click the graph to bring up a menu from which you can select **Print**.

## 6.10 View Pro Online Help

View Pro has an online help system that can be accessed by choosing **Help | View Pro** from the toolbar. Once the help file is opened, pressing the **Contents** tab will open the Table of Contents. Choosing the **Index** tab will bring up an index. Keywords can be typed in to search for a topic. An in-depth search can be performed by choosing the **Search** tab and typing in a word.

Help for any of the graphical windows can be accessed by pressing the **?** button in the upper right corner of the graph screen or by pressing **F1**. Help for dialog boxes can be accessed by pressing the **Help** button at the bottom of the dialog box or by pressing **F1** with the dialog box opened.

## 6.11 Assigning Data Files to View

Windows will let you assign the program with which a particular file type will be opened based on that file's extension. When a file with an assigned extension is double-clicked, it will be opened with the associated program. You may want to associate *.DAT files with the View Pro program for quick opening of data files. This association can be made by selecting **Tools | Folder Options | File Types** from the Windows Explorer menu.

# Section 7. Short Cut Program Generator

*Short Cut (also referred to as SCWIN) is an application for generating programs for Campbell Scientific's dataloggers and preconfigured weather stations except the CR7 and CR9000. Users do not have to know individual program instructions for each datalogger. Short Cut not only generates a program for the datalogger, but also a wiring diagram that can be left with the datalogger for field servicing.*

## 7.1 Overview

The Short Cut program generator creates programs for Campbell Scientific dataloggers in five easy-to-follow steps. Using a wizard-like interface, you create a new or open an existing program, select the datalogger and specify how often to make measurements, choose which sensors you wish to measure, select intervals or trigger conditions on which to store data and what processing to perform on the raw measurements for final storage, and finally generate the program. Short Cut also generates a wiring diagram for connecting your sensors to the datalogger.

Short Cut was designed to help the beginning datalogger programmer create datalogger programs quickly and easily. Short Cut effectively insulates the user from having to know the nuances of datalogger programming and the Edlog versus CRBasic programming languages. It supports the most commonly sold sensors from Campbell Scientific, as well as generic measurements (such as differential voltage, bridge, and pulse), commonly used calculation and control functions (such as heat index calculation, alarm conditions, and simple controls), and multiplexer analog channel expansion devices.

Short Cut cannot be used to edit existing Edlog, CRBasic, or Short Cut for DOS programs. Program editing and more complex datalogger programming functions should be accomplished using our Edlog or CRBasic Editor programming tools.

Short Cut was designed with extensive built-in help. Help can be accessed at any time by pressing the F1 key. There are also Help buttons on most screens. You can also open the Help by selecting Short Cut Help from Short Cut's Help menu. Help for each sensor can be accessed by searching the Help Index or pressing the Help button from the sensor form.

After generating the program, you can send it to the datalogger from the Results tab of Short Cut's Finish screen or from LoggerNet's Connect Screen or from PC400, PC200W, or RTDAQ's Clock/Program tab.

# 7.2 Creating a Program Using Short Cut

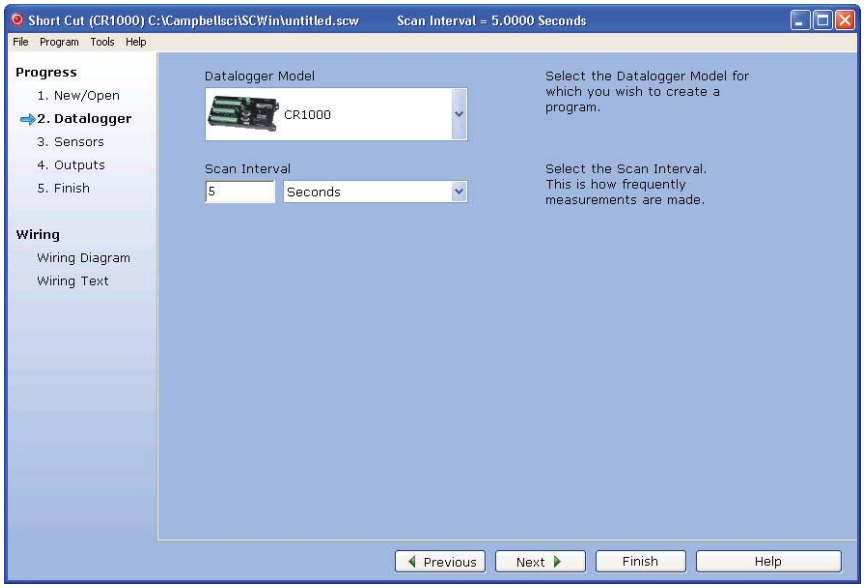
On opening, Short Cut presents a wizard that walks you through the steps of creating a datalogger program.

## 7.2.1 Step 1 – Create a New File or Open Existing File

To begin creating a new program, press the New Program button. To open an existing program, press the Open Program button and select a file from the resulting browser window.

## 7.2.2 Step 2 – Select Datalogger and Specify Scan Interval

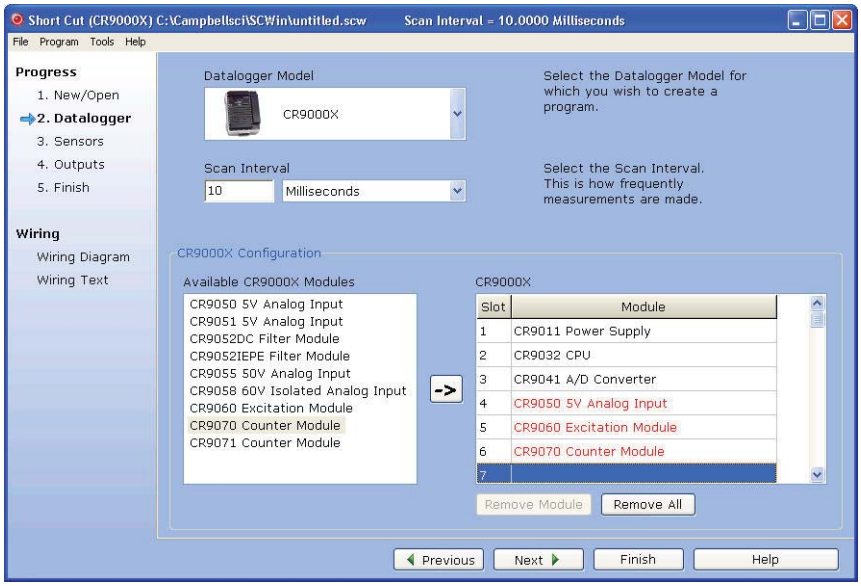
Click the arrow next to Datalogger Model to display a list of Campbell Scientific dataloggers. Select the datalogger type.




Set the Scan Interval, or period for each measurement cycle.

When choosing a scan interval, remember that faster scan intervals will use more power. For most applications, a 10 to 60 second scan interval is sufficient. If faster scan intervals are required for your application, make sure there is sufficient time for the execution of all instructions in the program (refer to the section in the datalogger manual on Execution Intervals for additional information).

If you are creating a program for a CR9000X, the CR9000X Configuration box will also appear on this screen.

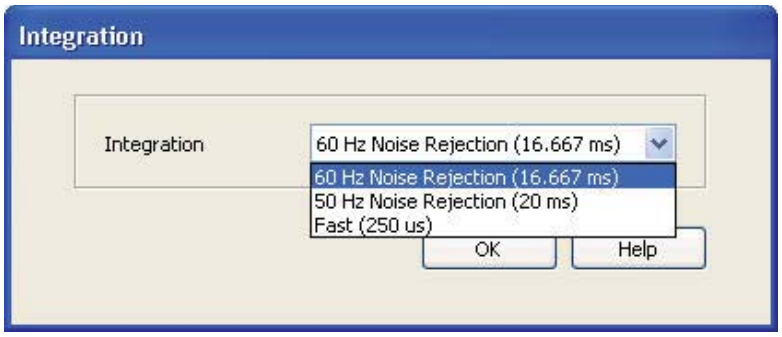



From this box, you indicate which CR9000X modules are inserted into which CR9000X slots. To add a module, select the module by clicking on it in the Available CR9000X Modules list, select the Slot by clicking on the slot number, then press the arrow key.

To remove a module, select the slot containing it and then press the Remove Module button.

**NOTE** Whenever you are working with a CR9000X program, this dialog box can be brought up by choosing Datalogger from the Progress panel. However, the Remove Module button is only available when a new program is being created. Once the Next button on the screen has been pressed, modules can be added but they cannot be removed.

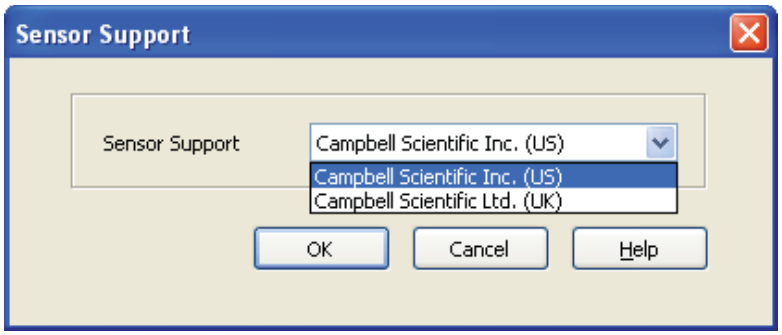
The next dialog box that is displayed is used to select the type of integration to apply to the measurements in the program. Integration can be used to filter out AC signals that might affect the accuracy of your measurements (such as noise from fluorescent lighting or a generator). Typically 60 Hz rejection is used for North America and 50 Hz rejection is used for countries following European standards. Fast (250 µs) integration should be used when you need an execution speed that cannot be accomplished using one of the other options.




This dialog box will be displayed the very first time you create a program for a specific datalogger type; it will not be displayed thereafter. With each subsequent program you create, the integration you chose when the datalogger was initialized in Short Cut will be used. However, you can change the integration from the Program menu. If you make this change, the setting will remain in effect for all programs for that datalogger type (whether they are new programs or edited programs) until it is changed again.

**NOTE** For the CR6 Series, the integration setting is named first notch frequency ($f_{N1}$).

The last dialog box displayed is the Sensor Support dialog box. (This dialog box will not be displayed when creating a CR9000X program.) This is used to select which group of sensor files will be displayed when creating a program: Campbell Scientific, Inc. (CSI, USA) or Campbell Scientific, Ltd. (CSL, UK). The standard set of Short Cut sensor files was created by CSI; however, CSL has created some additional files that are customized for their client base. When one option is selected, the sensor files developed specifically for the other are filtered out.




This setting is similar to the Integration setting in that the dialog box will be displayed only the first time you create a program for a specific datalogger type, and the setting will apply to all programs created or edited for that datalogger, unless it is changed via the Program menu. Note that programs containing sensor files that are filtered from the list of Available Sensors will still load and work correctly in Short Cut.

**NOTE**

The Integration and the Sensor Support settings are persistent settings for each datalogger model. The first time you create a program for a particular datalogger model, you will be presented with these two dialog boxes. The state of these settings is saved between Short Cut sessions. Any subsequent new or edited programs that are generated after a setting has been changed will reflect the change as well.

Each time you create the first program for a datalogger model you will be presented with these dialog boxes (e.g., the first time you create a CR10X program, you must initialize these settings; the first time you create a CR1000 program, you must initialize these settings).

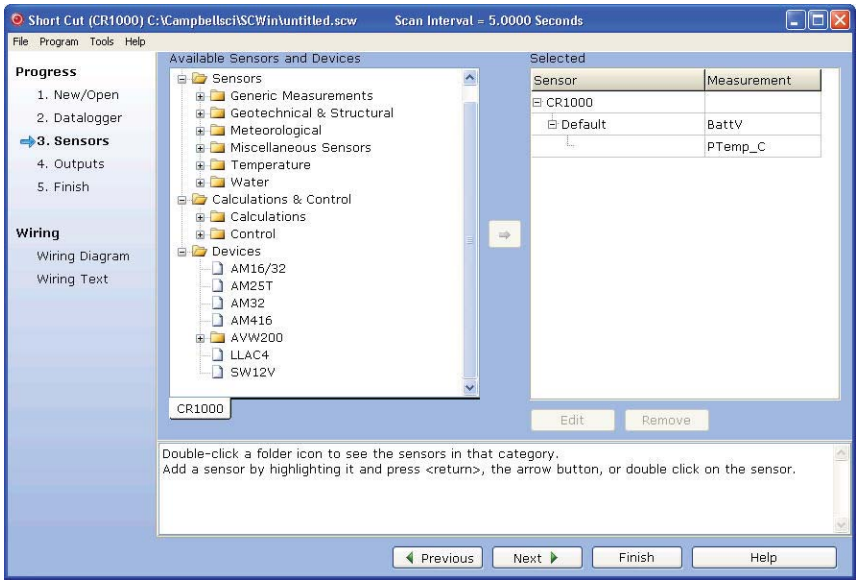
The settings can be changed at any time and the datalogger program will be regenerated to use the new setting when you click the Finish button on the Home screen.

After making your selections, note that the title bar shows the datalogger type and scan interval.

Once you have saved the file, the filename will replace "untitled.scw".

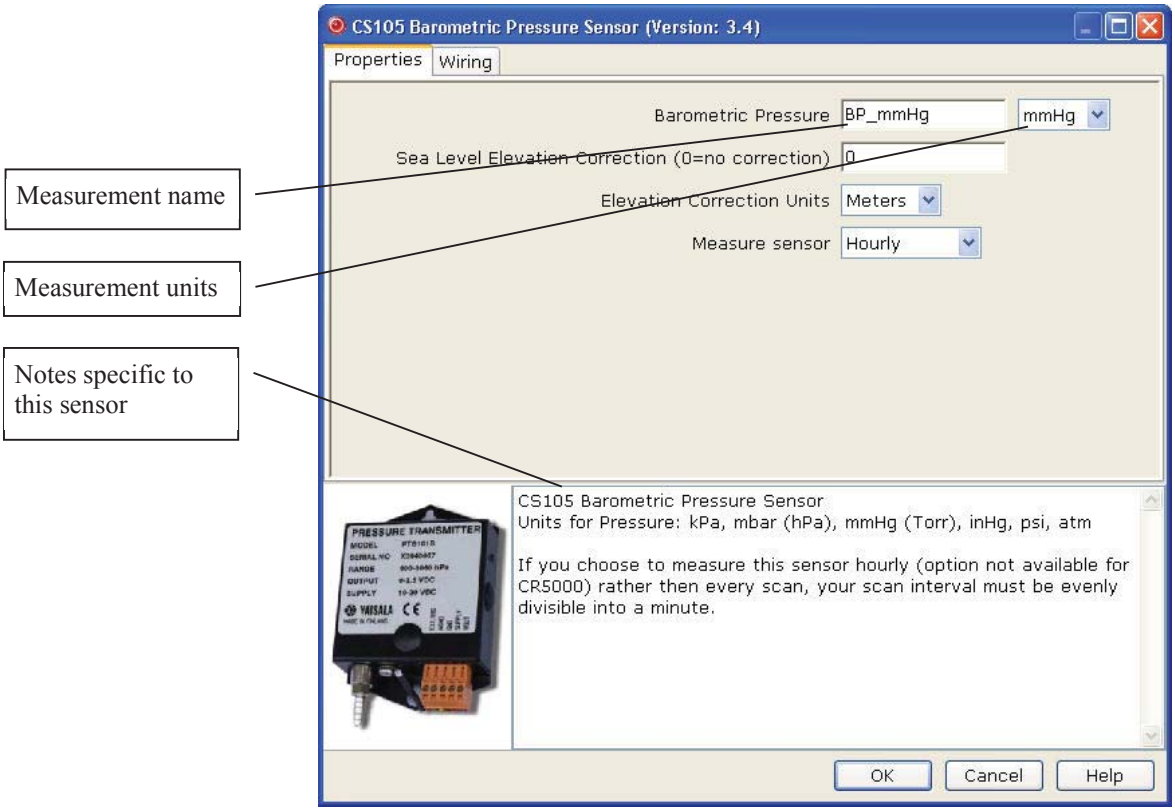
## 7.2.3 Step 3 – Choose Sensors to Monitor

In step 3, you tell Short Cut which sensors you'll be measuring. Short Cut organizes sensors into application groups:



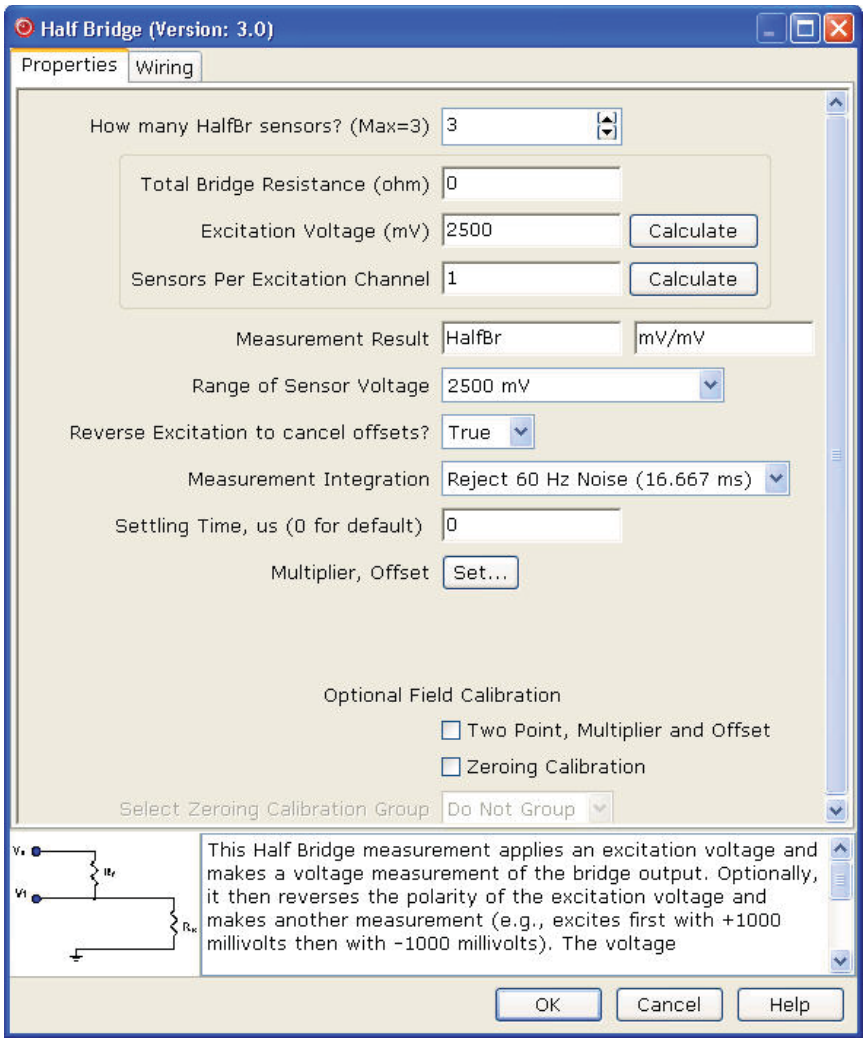

Some major groups have subgroups. Double-clicking the Meteorological group folder shows several subgroups of meteorological sensors. Double-click a subgroup to show the available sensors. Refer to the documentation for your sensors for the name of the sensors you have. If your sensor is not shown, you may be able to measure it with a generic measurement. Contact your Campbell Scientific application engineer for more assistance, if needed.

You "add" sensors to your program by double-clicking them or selecting them and clicking the arrow in the middle of the screen. Most sensors will require you to at least review the default settings for that measurement, including the measurement name, units, etc. An example of choosing the CS105 Barometric Pressure Sensor is below.

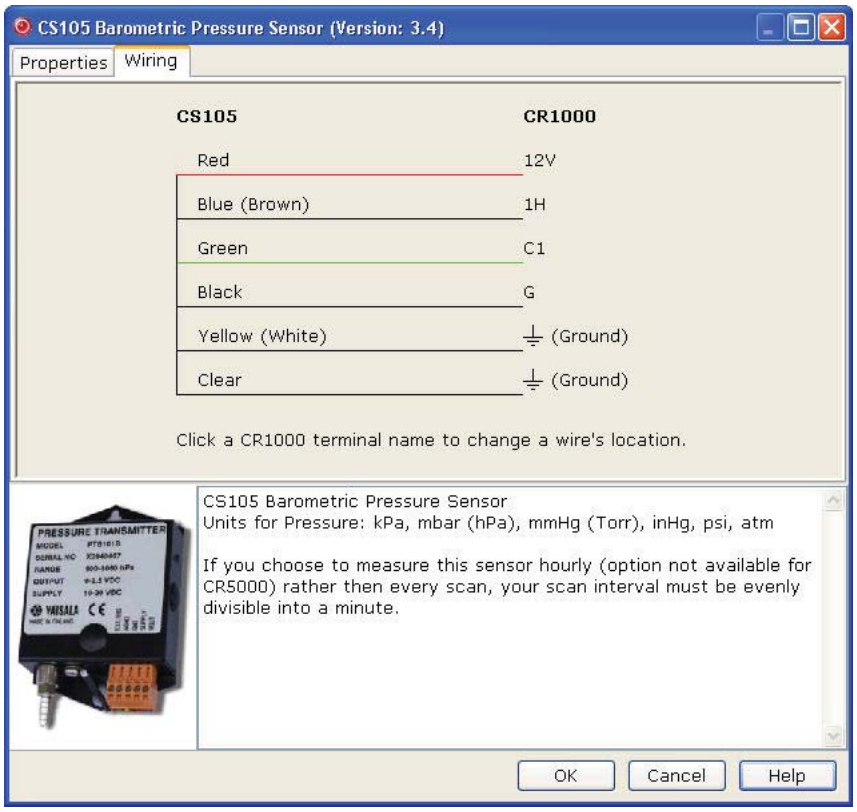



Note that this sensor not only offers a custom name field and units, but also allows you to correct for sea level, a common practice in measuring atmospheric pressure. In the middle of the screen, look over the notes (or refer to the Help for this sensor), for this sensor may require other sensors or have limitations. When you choose OK, Short Cut adds the necessary instructions with appropriate multipliers and offsets.

In some cases, multiple sensors of the same type can be added at one time. These sensors will have a How many sensors? parameter as the first parameter on the form as shown below. The maximum number of sensors that can be added will be indicated. The maximum will vary, depending upon the sensor and the number of other sensors already configured in the program. If the sensor form includes calibration and/or conversion parameters (e.g., multiplier, offset, gage factor), there will be a **Set** button next to these parameters. Pressing this button will allow you to set unique values for each sensor.


Half Bridge (Version: 3.0)
Properties
Wiring
How many HalfBr sensors? (Max=3)
3
Total Bridge Resistance (ohm)
0
Excitation Voltage (mV)
2500
Calculate
Sensors Per Excitation Channel
1
Calculate
Measurement Result
HalfBr
mV/mV
Range of Sensor Voltage
2500 mV
Reverse Excitation to cancel offsets?
True
Measurement Integration
Reject 60 Hz Noise (16.667 ms)
Settling Time, us (0 for default)
0
Multiplier, Offset
Set...
Optional Field Calibration
Two Point, Multiplier and Offset
Zeroing Calibration
Select Zeroing Calibration Group
Do Not Group
This Half Bridge measurement applies an excitation voltage and makes a voltage measurement of the bridge output. Optionally, it then reverses the polarity of the excitation voltage and makes another measurement (e.g., excites first with +1000 millivolts then with -1000 millivolts). The voltage
OK
Cancel
Help

Click on the Wiring tab of a sensor's parameter form to show the wiring for the sensor (or the first sensor in a sensor group).



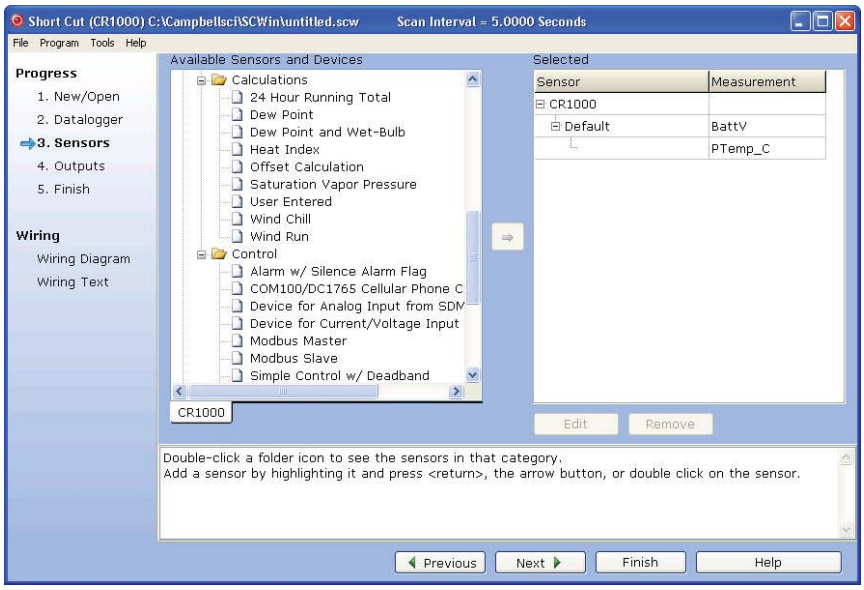

Each wire's caption/color is shown on the left side of the wire. The location where the wire will be connected to the device is shown on the right side (under the device). You can change a caption/color by clicking on the caption/color label. A wiring location can also be changed by clicking on the wiring location.

**NOTE** Changes to the wiring location for a sensor group can only be made when the group is first added. To make changes to a wiring location at a later time, you will need to change the number of sensors to one, press OK, reopen the parameter form, make the desired wiring location changes, and then change the number of sensors back to the desired number.

**NOTE** Not all sensors support changes to the wire caption/color and wiring location. When hovering over a wire caption/color or wiring location, the mouse cursor will change to indicate that the property can be changed. Changes are generally supported for generic sensors and other sensors that do not use special wiring connections.

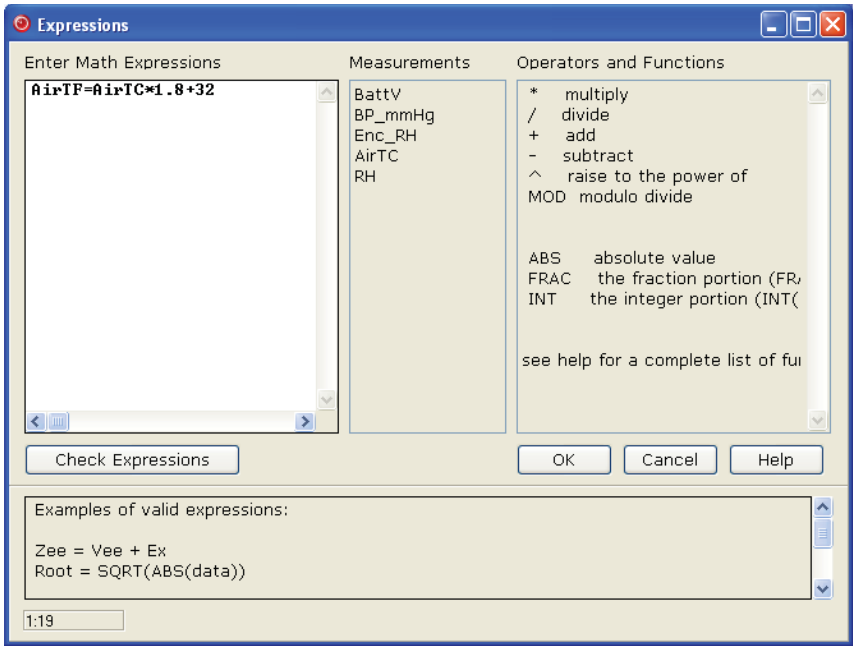
At any time, you may choose a measurement label on the right side of the Sensors screen and edit it or remove it.

In addition to actual sensors, Short Cut provides functionality to perform various calculations and effect some simple control:



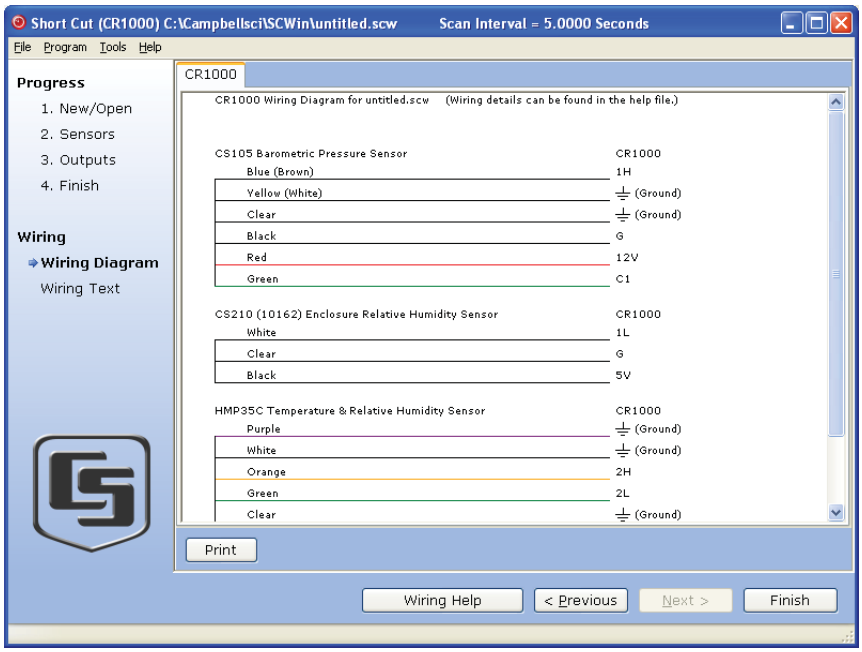

Some of these calculations may require additional sensors, or sensor measurements stored in particular units. See the help for each calculation to determine the necessary inputs. Note that there is also a User Entered calculation available in the Calculations folder. With it you can enter your own custom calculation.

In the example below, a new measurement, AirTF, is being created by performing calculations on an existing measurement, AirTC:

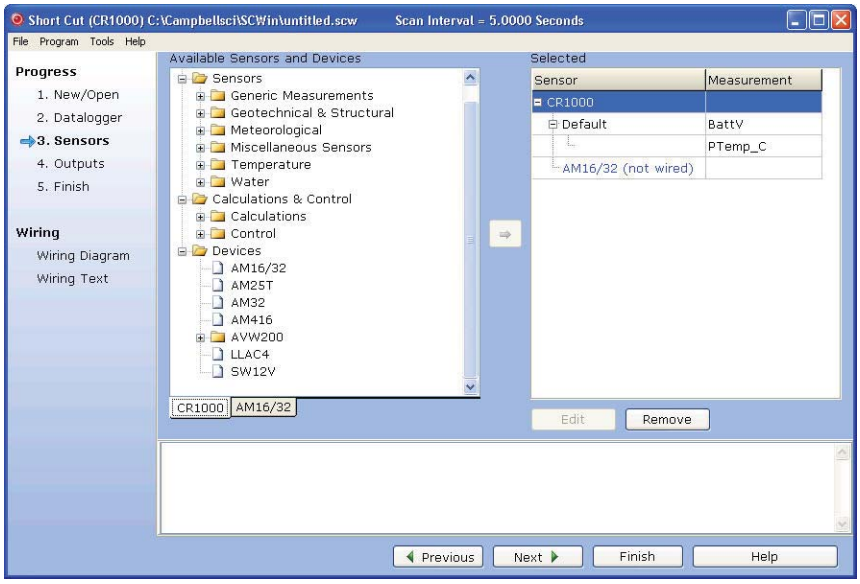



Refer to the online help for complete information on creating User Calculation.

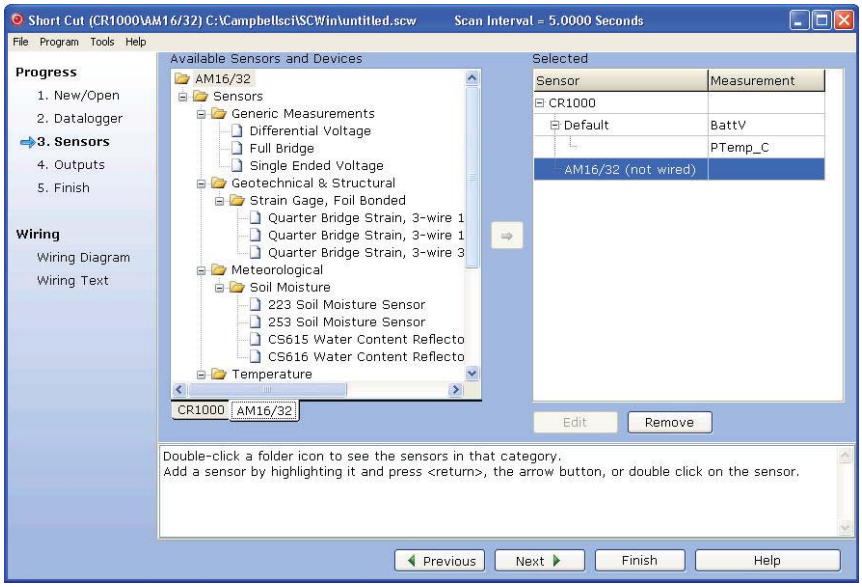
Short Cut provides you with a wiring diagram by clicking on Wiring Diagram on the left side of the Sensors window. In the example below, Short Cut was told to measure a CS105 Barometric Pressure sensor, an HMP45C Air Temperature and Relative Humidity sensor and a 05103 Wind Speed and Direction sensor. Each sensor was allocated the necessary terminals. Short Cut will not let you add more sensors than there are terminals on that datalogger or device. You can print this diagram (or the textual equivalent) by choosing the Print button. Many users find it handy to leave a printed wiring diagram in the enclosure with the datalogger in case a sensor has to be replaced.



Short Cut can also create programs for dataloggers using a variety of interface devices, including multiplexers and special interfaces for sensors. Add these devices by selecting them from the Devices folder in the Available Sensors and Devices tree.




Once you've added a device, such as the AM16/32 multiplexer, a tab is added to the screen for that device, and the sensors available for that device are shown:



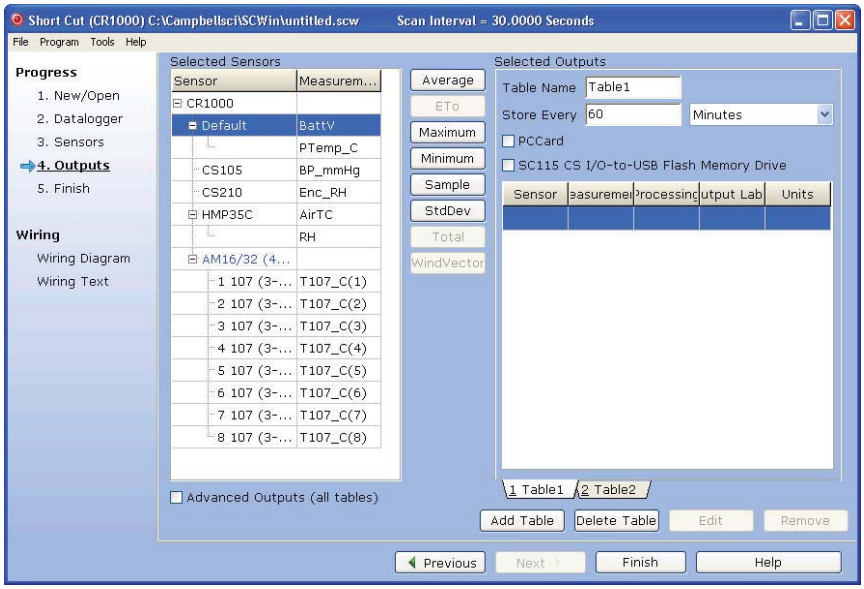

You can then add sensors to that device just as you would to the main datalogger.

Note that, once you add a sensor to a multiplexer, it may limit what kind of sensors can be added thereafter, as each sensor on the multiplexer must share the same wiring between the multiplexer and the datalogger.

After adding all the desired sensors, click Next.

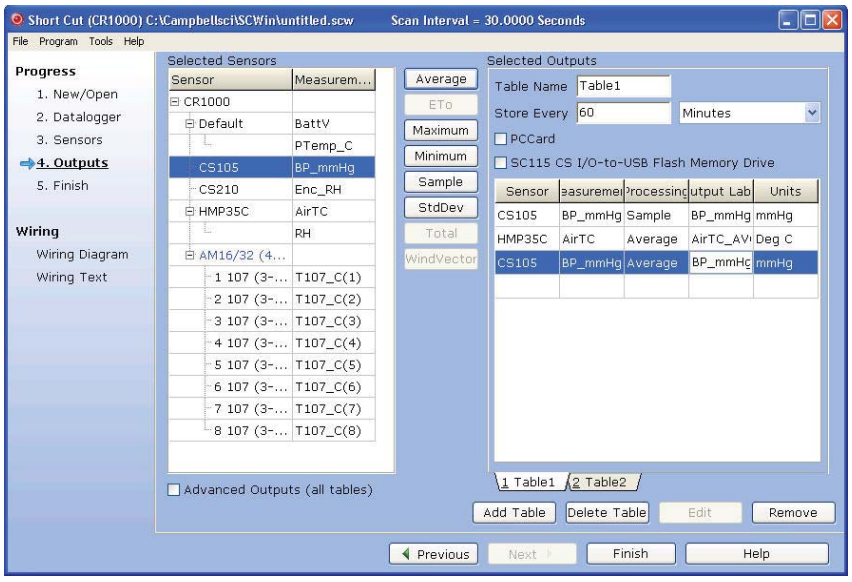
## 7.2.4 Step 4 – Setup Output Tables

After selecting the sensors to be measured, you can choose what data to store from the outputs step:




On the left, Short Cut will show the sensors you've added to be measured, with the measurement labels you've used. On the right is a multi-tabbed grid that shows the output tables. Most users store data as summaries over time – hourly and daily are common intervals for meteorological stations – but you can choose the table names (Array ID for Edlog mixed-array dataloggers) and intervals needed. You can even add additional tables as necessary.

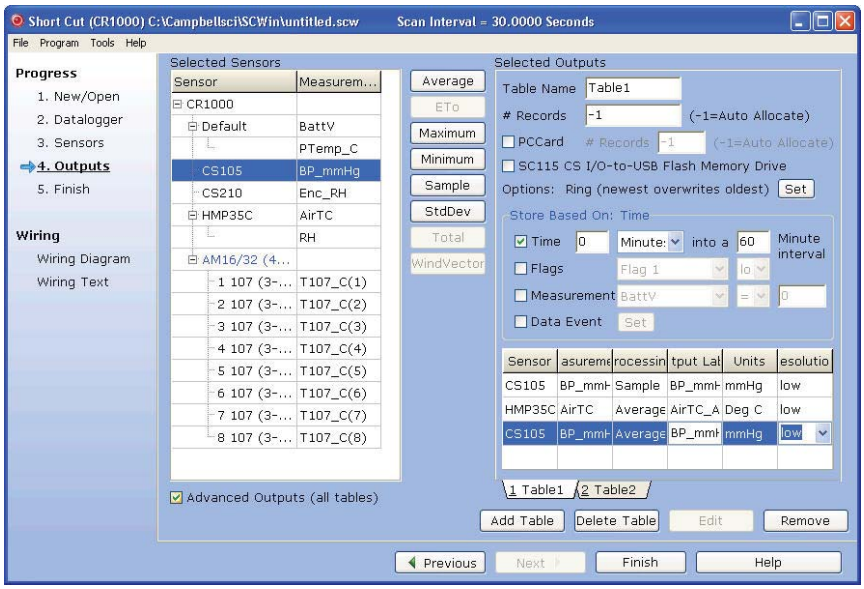
To store a measurement to final storage, simply click on a measurement label on the left, choose the data processing you want for that measurement by clicking one of the enabled buttons in the middle, and Short Cut adds the necessary instructions to save that data. In the example below, average air temperature, a sample of relative humidity, wind vector outputs, and average barometric pressure were selected for a 60-minute interval. Short Cut enables the most logical outputs for each measurement. If you require an output that is not enabled you can right-click on the measurement to get a pop-up menu containing all output options. You can also select a block of measurements (left-click+drag or left-click and shift+left-click) to do the same output on all of them. Note however that only output options common to all of the selected measurements will be enabled.




Note that outputs for a sensor don't have to be added in the same sequence as the measurement. You can even drag and drop the outputs to rearrange their order. Note also that multiple outputs can be added for any one sensor. For example, you may want to store the maximum and minimum air temperature as well as the average.

**NOTE** Table-based dataloggers have a TableName field as opposed to an array ID. There is also a check box used to store data to a PC card or SC115, if that functionality is supported by the datalogger model (see below for more information).

The Advanced Outputs check box in the lower left corner provides more output options:



The Advanced Output allows you to store data based on time, the state of a flag, the value of a measurement or a data event. Be cautious in using more than one check box, for the logic for the check boxes in the advanced mode are inclusive—that is, they must all be true in order for any output to be stored. If you were to choose to store rain only if both the time and the measurement logic were chosen, then the very last measurement of rain—in the last 10 seconds—would have to have been greater than zero RIGHT AT 6:00 a.m. Otherwise, both conditions wouldn't be true and the value wouldn't be stored.

The Output tab for dataloggers that can store data to a compact flash, microSD, or PCMCIA card has a check box which, when selected, will also store the data table to a card if one is inserted in the datalogger. In addition, the Advanced Output for these dataloggers allows for control of the number of records stored in the table.

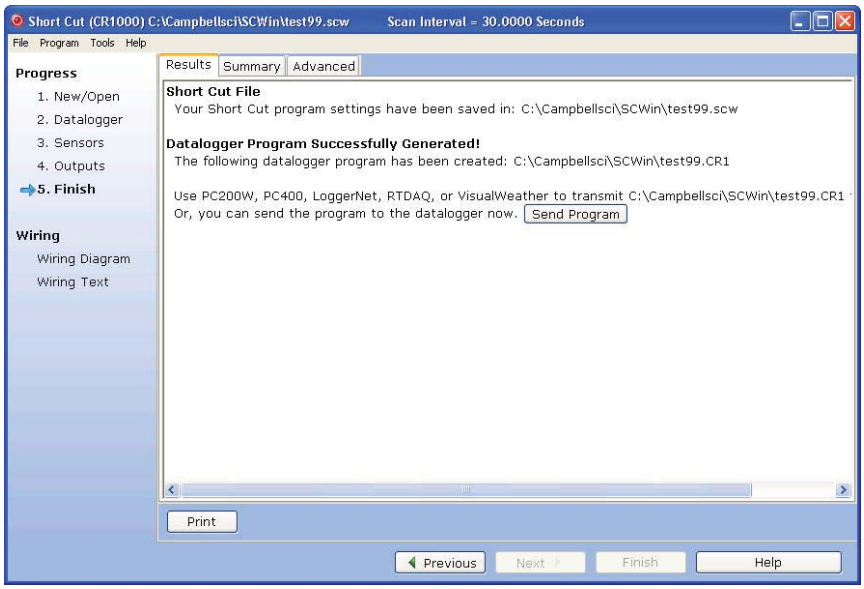
Table-based dataloggers that support output to an SC115 will have an SC115 CS I/O-to-USB Flash Memory Drive check box. When this check box is selected, new data will be copied to an SC115 when it is plugged into the CS I/O port of the datalogger.

The Table Name field is used to assign a unique name for the data table in the datalogger. The # Records field is used to set up the size of the data table; that is, the number of records that will be stored to the table before new records begin overwriting the oldest records. A fixed number can be entered into this field, or a -1 can be entered. If a -1 is entered, the table is set to auto-allocate, which means the datalogger will assign all available memory in the datalogger (or the card) to the table. If multiple tables are defined in the program, the datalogger will allocate memory to all fixed-size tables first, and then divide the remaining memory among any tables configured as auto-allocate. In doing so, the datalogger attempts to set the size of the auto-allocated tables so that they all fill at about the same time.

When you've configured all of your outputs, click Finish.

### 7.2.5 Step 5 – Generate the Program in the Format Required by the Datalogger

The Finish button completes the process. If you haven't yet saved the program, Short Cut asks for a program name and offers the default directory within its program working directory (default is C:\Campbellsci\SCWin). Short Cut also displays a Results, Summary, and Advanced window:




The Results tab provides information on the files that were created. If a program was created successfully, a Send Program button will also be displayed which allows you to send the program to the datalogger.

The files generated by Short Cut are as follows:

- *ProgramName*.SCW ("Example.SCW" in this example) at the top of the screen is the file in which Short Cut keeps all of your selections for datalogger, sensors, outputs, etc.

- For CR10, CR10X, CR500/510, CR23X, and 21X dataloggers (including mixed-array, table-data and PakBus operating systems), *ProgramName*.DLD is the ASCII text file that must be sent to the datalogger for it to make the measurements and store the data you want. For CR1000, CR6 Series, CR300 Series, CR3000, CR5000, CR800 Series, and CR9000X dataloggers, this file will be the .CR1, .CR6, .CR300, .CR3, .CR5, .CR8, or .C9X file. For CR200 dataloggers, this file will be a .BIN (binary image) file.

- *ProgramName*.DEF is the text file that describes the wiring for the sensors and devices to the datalogger, measurement labels, flag usage, and the output expected. You can view the contents of the DEF file by clicking the Summary button on the Results screen.

- For mixed-array dataloggers, *ProgramName*.FSL is a text file containing output labels (created for mixed-array dataloggers only). This file can be

used by Split or View or other software to provide column headers for the data file.

The Summary tab displays the information in the DEF file as described above.

The Advanced tab (for CRBasic dataloggers) displays the CRBasic program that was generated. It includes a CRBasic Editor button which opens the program for editing in the CRBasic Editor. Note that any changes made to the generated program in the CRBasic Editor will not be reflected in Short Cut or future programs generated by Short Cut.

Note that, while Short Cut can generate a program file for the datalogger, you must use datalogger communication software to transmit that program to the datalogger. (This is true even when pressing the "Send Program" button from Short Cut's Finish screen. Short Cut relies on the datalogger communication software to transmit the program.)

# 7.3 Short Cut Settings

The Program and Tools menus on the Short Cut menu offer several settings that may prove useful.

## 7.3.1 Program Security

Some dataloggers allow you to set security by entering one or more numbers into their security fields. You can allow different levels of access (e.g.; only allow data retrieval, or also allow monitoring of values, or also allow sending a new program or setting the clock) by entering multiple levels.

Datalogger security is not meant to be extremely tight. Rather, it is designed to prevent honest people from making mistakes.

Notwithstanding its intention, one mistake you can make is to set security and then forget the values. If you send a program with security set, you will then need to add that security setting to LoggerNet's Setup Screen or RTDAQ or PC400's EZSetup Wizard for that datalogger. If you don't, you may find that you can no longer communicate with the datalogger. Should this happen and you forget the security code and have lost the Short Cut program file, you may have to visit the datalogger site and cycle power on the datalogger to be able to communicate with it. Most dataloggers that offer security will communicate over their CS I/O port directly with a keyboard/display or PC in the first few seconds of powering up. See the datalogger manual for a full description of the security features.

## 7.3.2 Datalogger ID

Mixed-array dataloggers keep a memory location available for a datalogger ID value. This is typically an integer that you can read from within the program and store into final storage to keep track of the identity of the datalogger that created the data. Valid Datalogger IDs are 1 through 12 and 14 through 254. Use the Datalogger ID instruction in Short Cut (found under Miscellaneous Sensors) to use the ID in the datalogger program.

## 7.3.3 Power-up Settings

Some dataloggers offer the option to retain interim measurements or calculations or the states of flags or ports when they power-up from a low battery or loss of power condition. This may be useful when calculations are used to control devices. You may, for example, want to ensure that pumps or controls are off when a datalogger powers up so as to make the control decision based on a fresh measurement. See the datalogger manual for a full description of this feature.

## 7.3.4 Select CR200 Compiler

Use this setting to select the directory and executable name that will be used to pre-compile the CR200/205 program to check for errors.

Most Campbell Scientific dataloggers are sent an ASCII program file, which they then compile into machine code. The CR200/205 does not have enough memory and processing capability to do this compilation, so it's necessary to compile the program file into the binary version used by the datalogger itself. This compilation is done by Short Cut to check for errors in the program before sending it. It's done again by LoggerNet, RTDAQ, PC400, or PC200W when sending the program to the datalogger. Compilation is performed using a special executable that mimics the functions and capability in the datalogger's operating system. Therefore, the compiler executable must match the datalogger's operating system or the datalogger may fail to run the compiled binary (*.BIN) program. LoggerNet, RTDAQ, PC400, PC200W, and Short Cut are installed with precompilers for all of the released versions of the CR200/205 operating systems. If, at some time in the future, you acquire a newer CR200/205, or choose to install a later operating system, you must make sure you also have the compiler executable that matches. These compiler executables are typically installed in a library directory. By default, this directory would be installed as:

C:\Campbellsci\Lib\CR200Compilers

If you receive an operating system update, you should copy the compiler associated with it to this directory. If, for some reason, you put the compiler in a different directory, this menu item provides a way to choose that compiler executable.

## 7.3.5 Sensor Support

The Sensor Support option is used to select which group of sensor files will be displayed when creating a program: Campbell Scientific, Inc., (CSI) or Campbell Scientific, Ltd. (CSL). The standard set of Short Cut sensor files was created by CSI; however, CSL has created some additional files that are customized for their client base. When one option is selected, the sensor files developed specifically for the other are filtered out.

This dialog box is displayed the very first time you create a program for a specific datalogger type; it will not be displayed thereafter. With each subsequent program you create, the group of sensor files that you chose when the datalogger was initialized in Short Cut will be used. However, you can change this setting at any time. If you make a change, the setting will remain in effect for all programs for that datalogger type (whether they are new programs or edited programs) until it is changed again.

### 7.3.6 Integration/First Notch Frequency ($f_{N1}$)

Some dataloggers have parameters available in their measurement instructions to provide integration for rejection of noise due to AC electrical signals. These parameters will be used by Short Cut if possible, but the frequency of this noise varies. In most of North America, the AC frequency is 60 Hz. However, in many countries the frequency is 50 Hz. If you know the frequency of this AC noise, you can select one or the other frequency. Fast (250 µs) integration should be used when you need an execution speed that cannot be accomplished using one of the other options. This setting remains in effect for other programs generated by Short Cut until you change it.

**NOTE** For the CR6 Series, the integration setting is named first notch frequency ($f_{N1}$).

### 7.3.7 Font

This setting is accessed from the Options menu item of the Tools menu. Use this setting to change the appearance of the font used by Short Cut. Most windows other than the wiring descriptions (which require a non-proportional font to make sure wiring diagrams are aligned) will use this font.

### 7.3.8 Set Working Directory

This setting is accessed from the Options menu item of the Tools menu. This setting changes the directory that Short Cut offers as a default for your programs. Upon installation, the default is set to C:\CampbellSci\SCWIN.

### 7.3.9 Enable Creation of Custom Sensor Files

This setting is accessed from the Options menu item of the Tools menu. It allows the user to create custom sensor files as described in Section 7.6, *Custom Sensor Files* (*p. 7-19*).

## 7.4 Editing Programs Created by Short Cut

Short Cut is very flexible and has many features. It does not, however, support all of the functionality in Campbell Scientific dataloggers. Some users will need to develop programs with capabilities beyond that offered by Short Cut, but will want to take advantage of the library of instructions and settings known to a program generator in order to get a head start.

For Edlog dataloggers, the easiest method is to Document the DLD file from within Edlog (discussed later in this section). Short Cut creates a .DLD file to send to the datalogger that includes input location and final storage labels. Documenting a .DLD file causes Edlog to use the same labels and to show you the individual instructions being used to carry out the program. You can then add and delete instructions from within Edlog to add functionality to the program. Short Cut cannot import the files created by Edlog, however. Short Cut reads only its own SCW-formatted files.

For CRBasic dataloggers, you can use the CRBasic Editor to open the .CR# files directly. Again, Short Cut will not be able to open the files you've edited with the CRBasic Editor, since they are not an SCW file.

## 7.5 New Sensor Files

Short Cut was designed with future flexibility in mind. Datalogger and sensor support is provided as individual files and not part of the SCWIN executable. As new dataloggers and sensors become available, new definition files will be created to add and modify the necessary features known to Short Cut. To update these files, you can download the latest version of Short Cut from the Campbell Scientific website:

*www.campbellsci.com/downloads*

It is also possible to have custom sensor files created for sensors your organization uses that are not included with Short Cut. Contact your Campbell Scientific applications engineer for details.

## 7.6 Custom Sensor Files

The creation of custom sensor files can be enabled from Short Cut's Tools | Options menu item. Once enabled, custom sensor files can be created by right-clicking on a sensor in the Available Sensors and Devices list and choosing Create Custom Sensor.

The resulting dialog box will allow the user to make changes to the chosen sensor file and then save it with a new name. (See Short Cut's Online Help for additional information on changes that can be made.) By default, custom sensor files will be created in C:\CampbellSci\SCWin\SENSORS, which is a different location than that of Short Cut's included sensor files.

Once the custom sensor file has been saved, it will be added to the Available Sensors list.

# Section 8. Datalogger Program Creation with Edlog

*This section provides information on memory allocation and programming for Campbell Scientific's Edlog dataloggers, including the CR7, CR10, 21X, CR500, CR510, CR10X, and CR23X. Edlog also supports these same dataloggers configured with table-based operating systems, including the table-data or "TD" and PakBus or "PB" versions. See Section 9, Datalogger Program Creation with CRBasic Editor (p. 9-1), for information about programming the CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, CR9000, and CR200 dataloggers.*

*Programs can also be created with the Short Cut program generator, see Section 7, Short Cut Program Generator (p. 7-1).*

## 8.1 Overview

Edlog is a tool for creating, editing, and documenting programs for Campbell Scientific's mixed-array dataloggers: CR7x, CR500, CR510, CR10, CR10X, 21X, CR23X. Edlog also supports these same dataloggers configured with table-based operating systems, including the table-data or "TD" and PakBus or "PB" versions. It provides a dialog box from which to select instructions, with pick-lists and detailed help for completing the instructions' options (or parameters). Edlog checks for errors and potential problems in the program when pre-compiling the program. Some highlights of Edlog's features are listed below.

Precompiler – Edlog precompiles the program to check for errors and to create the file that is downloaded to the datalogger. The precompiler will catch most errors. Errors that the precompiler misses should be caught by the datalogger when the program is compiled. The download file (*.DLD) is stripped of comments to make it more compact. During the precompile step, a Program Trace Information file (*.PTI), that provides an estimate of program execution time, is also created (Section 8.1.1.2, *Edlog File Types (p. 8-4)*). For mixed-array dataloggers the precompiler also creates a Final Storage Label file (*.FSL) to supply labels for final storage values to be used by other software applications.

Context-sensitive Help – Pressing the right mouse button with the cursor on a parameter will provide a pick-list of options or pop-up help for that parameter. More help is available by pressing <F1> at any time or the Help button in various dialog boxes. Help, pick lists, and edit functions are also available from the menu bar or toolbar.

Cut and Paste – Several datalogger programs can be opened simultaneously, and instructions can be copied from one program into another. This simplifies writing different programs for the same sensor set, or similar programs for different sites. Edlog will also allow you to save sections of code as "Library Files" which can then be imported into other programs.

**NOTE** Be careful when copying instructions from a program written for one datalogger to a program for a different type of datalogger. Instructions may differ between dataloggers.

Input Location Labels – Though the datalogger uses a number to address input locations, Edlog allows you to assign labels to these locations for ease of use when programming and later when reviewing the data on-line. Edlog has several features that aid in the management of these labels. A new Input Location label is automatically assigned the next available Input Location number (address). That Input Location can be picked from a list when needed later in the program for further calculations or output. The Input Location Editor (Section 8.2.3, *Input Location Editor* (p. 8-17)) allows the Input Locations to be edited (moved, inserted, or deleted); the Input Location numbers are then automatically updated wherever the labels appear in the program. When a section of code is pasted into a program, Edlog will automatically use existing locations for matching labels and assign new locations to new labels. All location numbers in the pasted code are updated accordingly.

Final Storage Label Editor – The Final Storage Label Editor allows you to change the default labels assigned by Edlog. The labels are stored in the Final Storage Label file for mixed-array dataloggers and as part of the datalogger program for mixed-array and table-based (both table-data and PakBus) dataloggers.

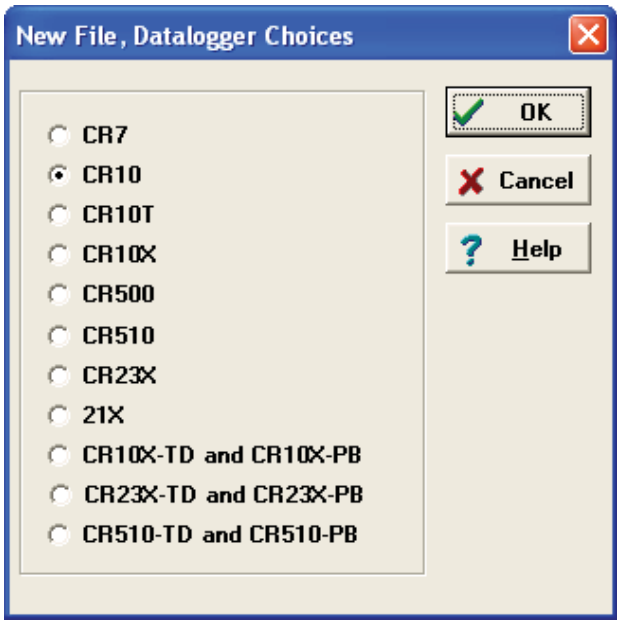
Expression Compiler – Mathematical calculations can be written algebraically using Input Location labels as variables. When the program is compiled, Edlog will convert the expressions to datalogger instructions.

For example, the following expression could be used to create a new input location for temperature in degrees Fahrenheit from an existing input location for temperatures in degrees Celsius.

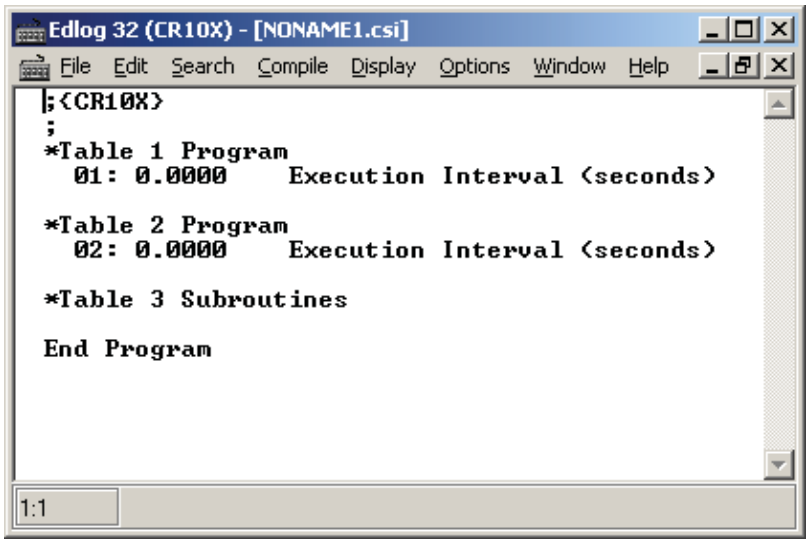
```
TempF=TempC*1.8+32
```

## 8.1.1 Creating a New Edlog Program

To create a new datalogger program, choose File | New from the Edlog menu and select the datalogger type from the dialog box. A window similar to the one shown below appears.



Select the datalogger you are using from the list and click OK. A blank program template will come up as shown below for a CR10X.




The first line of text identifies the type of datalogger program to be written. This is followed by a comment line and the Program Table Headers and Execution Interval fields. The Program Table Headers and Execution Interval fields are protected text that cannot be deleted or commented out. (The asterisk is used to identify the beginning of a program table in the datalogger.) When the cursor is moved to the Execution Interval line, the field for the execution interval is highlighted. A numeric value must be entered or the instructions in the table will never be executed.

Instructions inserted under the Program Table 1 header will be run based on the execution interval for that table. Likewise, instructions inserted under the Program Table 2 header will be run based on the execution interval for Program Table 2. Program Table 3 is reserved for subroutines that are called by either of the other tables. Most users find they can write the entire program in Program Table 1, avoiding complications associated with synchronizing two tables. Program Table 2 is normally used only when portions of the program require a different execution interval (placed in Program Table 2).

**NOTE**

Program tables in this section refer strictly to sections of the datalogger program. Do not confuse these program sections with the data tables created in table-based dataloggers using P84 to store output data.

When the program is complete, select File | Save from the Edlog menu. A standard file dialog box will appear in which to type a file name. Edlog supports long file names for the datalogger programs. Use descriptive names to help document the program's function. After saving the file, you will be prompted to compile the program. When a program is compiled the code will be checked for errors. After compiling, the datalogger program can be sent to the datalogger from the Clock/Program tab.

### 8.1.1.1 Program Structure

While Edlog is not a structured programming language there are some standard programming practices that will help you and others understand what the datalogger program is intended to do.

**Comments** – Edlog provides the ability to add comments on any blank line and to the right of all instructions. Liberal use of descriptive comments makes the program clearer and will help you remember what you were doing when you come back to it a year or two later. Especially useful are descriptions of what sensors are connected and how they are wired to the datalogger.

**Program Flow** – It is easier to follow a program that is written in distinct sections, each of which handles a specific type of instruction. The recommended sequence is:

– Measure Sensors – In this first section put all the instructions that get data from the sensors attached to the datalogger. The sensor readings are stored in input locations, ready for the next section.

– Process Measurements – In this section do all the calculations and data processing to prepare the data for output.

– Control – Do any control of external hardware or devices.

– Output Data – Check to see if it is time, or a condition exists, to trigger output data to be saved in final storage.

**Descriptive Labels** – Use input location and final storage labels that are meaningful for the data they contain.

### 8.1.1.2 Edlog File Types

When a program is saved and compiled, the following files are created:

- *.CSI – The CSI file is what the user actually edits. When an Edlog program is saved, Edlog automatically adds a CSI extension to the program's name. Existing CSI files can be edited by selecting File | Open. Although CSI files are ASCII files they require a particular format, so editing the *.CSI files with some other text editor can corrupt the Edlog programs so that they no longer load or compile.

- *.DLD – When Edlog compiles a program (*.CSI), a .DLD file is created. This is the file that is downloaded to the datalogger, (and also the type of file that is retrieved from the datalogger). If an existing program file is edited and compiled, the old DLD file will be overwritten by the new file. A CSI file can be created from a DLD by choosing File | Document DLD File.

- *.PTI – Program Trace Information files show the execution times for each instruction, block (e.g., subroutine), and program table, as well as the estimated number of final storage locations used per day. The execution times are estimates. PTI files do not account for If commands, Else commands, or repetitions of loops. For some instructions, the execution

times are listed as 0. This occurs when the execution time is unknown (e.g., P23 – Burst Measurement).

- *.FSL – Final Storage Label files contain the final storage labels for the data values in the output data records. This file is used by Split to show labels for data values in reports, and by View Pro for column headings. FSL files are not created for table-based dataloggers. Table-based datalogger program files contain the final storage labels.

Other files that are used in Edlog but are generated by other means than compiling the program include:

- *.LBR – Library files (*.LBR) are parts of a program that can be retrieved and used in other Edlog programs. If a programmer often uses an instruction set in his/her datalogger programs, this partial file can be saved to disk and inserted into a new program. For information about creating a library file, see Section 8.1.3, *Library Files* *(p. 8-14)*, or the help on Save to Library File.

**NOTE** Library files that are created for one type of datalogger should not be used in a different type of datalogger (e.g., do not use an LBR file created for a CR10X-TD in a CR10X or CR510-TD program). Instructions differ among dataloggers, and bringing in an invalid instruction to a datalogger will result in errors.

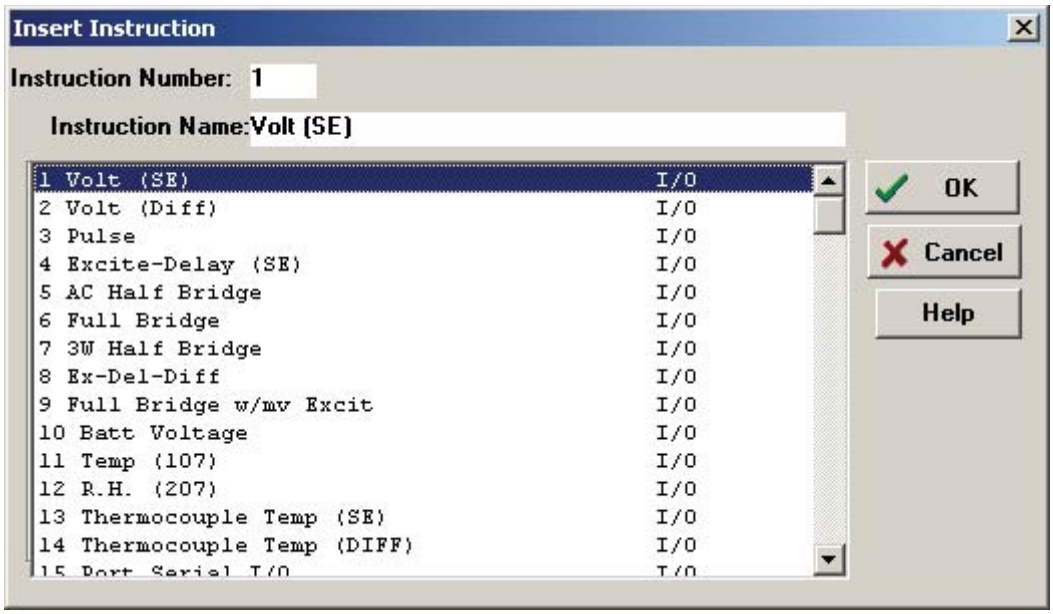
- *.TXT – Printer output files created by Edlog are saved with a TXT extension. These files can be sent to a printer or viewed with a text editor. A TXT file is created by selecting File | Print to File.

### 8.1.1.3 Inserting Instructions into the Program

Instructions are entered into the program table in the order that they should be executed in the program. There are four ways to insert an instruction:

- Select Edit | Insert Instruction from the Edlog menu.
- Press <Shift>+<Insert> on the keyboard.
- Right click a blank line and select Insert Instruction from the pop-up menu.
- Type the instruction number onto a blank line and press enter.

The first three options will invoke the Insert Instruction dialog box.




To insert an instruction into the program, select it and then choose OK, or double click the entry in the list. If you need more information on an instruction, select the instruction and click the Help button.

Note that to the right of each instruction name is a code for the instruction type: I/O for input/output, Process for instructions that calculate new values, Output for instructions that write to final storage, or Control for instructions that affect program flow.

### 8.1.1.4 Entering Parameters for the Instructions

When an instruction is inserted, the cursor moves to the first parameter. Type the parameter's value and press <Enter> to move to the next parameter. There are two ways to get help on a parameter:

- Select the parameter with your mouse and press the right mouse button. This brings up a dialog box from which to select a value or a pop-up description of what should be entered.

- With your cursor anywhere within the instruction, press <F1>. This opens the help system to a detailed description of the instruction and parameters.

Edlog provides hints for each parameter at the very bottom of the Edlog screen. These hints often display the valid entries for a field.

**NOTE** Many instructions are datalogger specific; refer to the specific datalogger manual for details on a particular instruction.

**Data Entry Warnings**

Edlog has a Data Entry Warning function that is accessed from the Options | Editor menu item. By default, the Data Entry Warning is enabled. When the Data Entry Warning is active, a warning is displayed immediately after an invalid input or potentially invalid input has been entered for an instruction's

parameter. The warning lists the valid inputs. A valid input must be entered before advancing to the next parameter.

#### 8.1.1.5 Program Comments

Comments can be entered to document the program for the programmer or future users. Comments are ignored by the compiler; they can be entered on any blank line or at the right of instruction or parameter text. A semicolon (;) is used to mark comments. Comments can also be used to temporarily remove instructions from a program for testing purposes.

In addition to typing a semicolon at the beginning of each line while entering comments, there are several ways to comment (or uncomment) lines, instructions, or blocks of code:

- Select a block of text, press the right mouse button, and select "comment" or "uncomment" from the right button pop-up menu.
- Select Edit | Comment or Edit | Uncomment from the Edlog menu.
- Select a block of text and press <Ctrl>+n to comment text (or <Shift><Ctrl>+n to uncomment text).
- Press <End> to automatically insert a semi-colon to the right of the protected text of an instruction or parameter, and type the desired comment.

Edlog will not allow a portion of an instruction or the table execution intervals to be commented out.

#### 8.1.1.6 Expressions

Algebraic expressions can be used in a program to easily perform processing on input locations. When a datalogger program that contains an expression is compiled, the appropriate instructions are automatically incorporated into the DLD file. As an example, the following expression could be used to convert temperature in degrees Celsius to temperatures in degrees Fahrenheit:

```
TempF=TempC*1.8+32
```

Following are rules for creating expressions:

- Expressions must be set equal to the label of the Input Location that will store the result. The result label must be to the left of the expression.
- Expressions can have both fixed numbers and Input Location labels. Input Locations can only be referenced by their label; each number in an expression is assumed to be a constant.
- Floating-point numbers are limited to six digits plus the decimal point and sign.
- The operator(s) and/or function(s) used in the expression are limited to those in the Operator and Function list (TABLE 8-1 below).

- Numbers and labels that appear immediately after a function must be enclosed in parentheses.

- Several operators and/or functions can be used in one expression. Operations and functions that are enclosed in parentheses are calculated first; the innermost parentheses are evaluated first.

- To continue an expression to the next line, end the first line with an underscore ( _ ).

**TABLE 8-1. Operators and Functions**

| | |
|---|---|
| Operators | |
| * | multiply |
| / | divide |
| + | add |
| - | subtract |
| ^ | raise to the power of; enclose negative values in parentheses |
| @ | modulo divide |
| E | scientific notation; 6e-1=0.6 |
| Functions | |
| COS | cosine; angle in degrees |
| SIN | sine; angle in degrees |
| TAN | tangent; angle in degrees |
| COTAN | cotangent; angle in degrees |
| ARCTAN | arctangent; angle in degrees |
| ARCSIN | arcsine; angle in degrees |
| ARCCOS | arccosine; angle in degrees |
| ARCCOT | arccotangent; angle in degrees |
| SQRT | square root |
| LN | natural logarithm |
| EXP | exponent of e; EXP(2) = $e^2$ |
| RCP | reciprocal; RCP(4) = 1/4 = 0.25 |
| ABS | absolute value |
| FRAC | takes the fraction portion; FRAC(2.78)=.78 |
| INT | takes the integer portion; INT(2.78)=2 |

Below are examples of valid expressions:

```
Zee    =  Vee+Ex
es     =  tee^(-2)
Root   =  SQRT(ABS(data))
avg    =  (data1+data2+data3+data4+data5)/5
length =  SQRT((adj^2)+(opp^2))
TempF  =  (TempC*1.8)+32
```

The following section of an Edlog program uses an expression to convert temperature from Celsius to Fahrenheit:

```
Execution Interval = 10 sec

;this instruction reads the temperature probe
;the output is in degrees C

1:  Temperature (107) (P11)
   1:   1              REPS
   2:   2              Channel
   3:   1              Excitation Channel
   4:   2              Loc [TempC]
   5:   1              Mult
   6:   0              Offset
;the following expression converts TempC to
;a temperature in degrees Fahrenheit

TempF = (TempC*1.8)+32
```

When this program is compiled, the DLD file contains the following instructions. The last 5 instructions calculate the expression.

```
1:  Temperature, 107 (P11)
   1:   1
   2:   2
   3:   1
   4:   2
   5:   1.0
   6:   0.0

2:  Z=X (P31)
   1:   2
   2:   5

3:  Z=F (P30)
   1:   1.8
   2:   0
   3:   3

4:  Z=X*Y (P36)
   1:   3
   2:   5
   3:   5

5:  Z=F (P30)
   1:   32
   2:   0
   3:   3

6:  Z=X+Y (P33)
   1:   3
   2:   5
   3:   6
```

**Errors That Can Occur With Expressions**

Some of the error messages that occur when using expressions need no further explanation:

Missing left parenthesis
Missing right parenthesis
Variable name expected
Number expected
Floating point numbers limited to 5 digits
Function expected
New line expected
Equal sign expected

Other errors are explained below.

**Variable Name Expected**

This message occurs when the expression is not set equal to an Input Location label. The label must be to the left of the expression and not enclosed in parentheses. An expression that contains no equal sign causes compiler error 202, "unrecognized text".

For Example:

"Variable name expected" is displayed when a program contains any of these expressions:

5=el*(Vee+en)
(lambda) = COS(theta)
10-(zee/2)=bee

These are correct ways of entering the above expressions:

five=el*(Vee+en)
lambda = COS(theta)
bee=10-(zee/2)

**Number Expected**

Indicates one of the following situations:

(1) An expression with a /, *, or ^ operator is missing a number or label before and/or after the operator.

(2) An expression with a + or - operator does not have a number or label after the operator.

(3) An expression with an @ operator does not have a number after the @; only a fixed number is allowed immediately after the @ operator.

(4) An expression with an @ operator does not have either a number or label before the @.

(5) There is nothing between a pair of parentheses (e.g., the expression contains this "()").

(6) A number is immediately followed by a label or function without an operator (e.g., an expression containing "8label" gets this error message).

**Floating Point Numbers Limited to 5 Digits**

All fixed numbers are limited to five digits not including negative signs and decimal points.

**Function Expected**

Letters that are immediately followed by parentheses are assumed to be a function. If the letters are not on the function list, this error message occurs.

**New Line Expected**

Indicates one of the following situations:

(1) An expression contains more than one equal sign.

(2) There is no operator between two sets of parentheses.

For Example:

This error message is displayed when a program contains any of these expressions:

```
zee=(label1)(label2)
ex=(5)(ARCTAN(data))
eee=(em)(see^2)
```

These are correct ways of entering the above expressions:

```
zee=(label1)*(label2)
ex=(5)*(ARCTAN(data))
eee=(em)*(see^2)
```

(3) There is no operator between a set of parentheses and a number.

For Example:

This error message is displayed when a program contains any of these expressions:

```
tee=5(2)
mu=(nu)103
bee=10.52(ef/2)
sigma=-17(RCP(alpha))
```

These are correct ways of entering the above expressions:

```
tee=5*(2)
mu=(nu)*103
bee=10.52*(ef/2)
sigma=-17*(RCP(alpha))
```

(4) A label or function is immediately after a set of parentheses without an operator.

For Example:

This error message is displayed when a program contains any of these expressions:

result=(ex^2)data
gamma=(10-omega)SIN(psi)
dee=(17)number

These are correct ways of entering the above expressions:

result=(ex^2)*data
gamma=(10-omega)*SIN(psi)
dee=(17)*number

**Equal Sign Expected**

An equal sign MUST immediately follow the label of the Input Location that stores the results (e.g., label = expression). An expression that contains no equal sign causes compiler error 202, "unrecognized text".

For Example:

"Equal sign expected" is displayed when a program contains any of these expressions:

zee/2=bee
data+number=volt1+volt2

These are correct ways of entering the above expressions:

bee=zee/2
data=volt1+volt2-number

## 8.1.2 Editing an Existing Program

To edit an existing file, load it into Edlog by choosing File | Open from the Edlog menu. Changes can be made as desired and then the file can be saved and compiled under the same (File | Save) or a new name (File | Save As). TABLE 8-2 provides a list of keystrokes that can be used in editing programs and moving around in Edlog.

| TABLE 8-2. Editor Keystrokes | |
|---|---|
| PgUp | Page Up |
| PgDn | Page Down |
| Up Arrow | Move Up One Line |
| Down Arrow | Move Down One Line |
| Right Arrow | Move One Character Right |
| Left Arrow | Move One Character Left |
| <Ctrl> Home | Move Cursor to Beginning of File |
| <Ctrl> End | Move Cursor to End of File |
| <Ctrl> PgUp | Move Cursor to Top of Screen |
| <Ctrl> PgDn | Move Cursor to Bottom of Screen |
| <Enter> | Move to Next Field or Create New Line |
| <Shift> <Ins> | Select an Instruction from a Dialog Box |
| <Ctrl> Right Arrow | Move Instruction 1 Tab Right (Cursor on Parameter) |
| <Ctrl> Left Arrow | Move Instruction 1 Tab left (Cursor on Parameter) or Move from Input Location label to Input Location number. |
| <Ctrl> n | Comment out a Line or Instruction |
| <Shift> <ctrl> n | Uncomment a Line or Instruction |
| <End> | Move to end of line, Add a comment if on an Instruction |
| <Ctrl>C | Copy selected text |
| <Ctrl> X | Cut selected text |
| <Ctrl>V | Paste clipboard |
| <Del> | Delete character to right or selected text |
| <Shift> Del | Delete the Instruction or Line Under the Cursor |
| <Esc> | Close Dialog Box |

#### 8.1.2.1 Editing Comments, Instructions, and Expressions

To edit Comments, Expressions, and Instruction parameters, move the cursor to the appropriate text and retype it. To delete an instruction when the cursor is somewhere within the instruction, select Edit | Delete Instruction or press <Shift> <Del>. An instruction or block of instructions can also be selected and deleted with the delete key. The entire instruction must be selected or an error message will be returned.

#### 8.1.2.2 Cut, Copy, Paste, and Clipboard Options

Edit | Delete , Edit | Cut, Edit | Copy, and Edit | Paste allow sections of the program to be deleted, moved, or copied to another area of the program or between programs. Edit | Show Clipboard shows the contents of the clipboard.

**NOTE** You cannot move, copy, delete or comment out protected text (Tables, Execution Intervals) or partial instructions. To move, copy or delete an Instruction, the entire instruction, including all of the parameters, must be selected.

Cutting and pasting between datalogger programs should only be between programs for the same datalogger type. Instructions and parameters may differ between dataloggers. The compiler will catch many of these errors; however, this may be at the expense of much time and confusion.

## 8.1.3 Library Files

Library files can be created to store portions of programs, which can then be inserted into a different program. Library files are useful if you want to write different programs for the same sensor set, or it you have several stations that have similar, but not identical, sensor sets.

To create a library file, select the text to be stored and then select Edit | Save To Library File. When the window appears, type in the library file name. To insert a library file in a program, move the cursor to the desired insertion point and select Edit | Insert Library File.

**NOTE**

Library files created for one type of datalogger type should not be used in programs for a different datalogger type; i.e., a library file for a CR10X-TD should not be used in a program for a CR10X or a CR510-TD. Instructions differ among dataloggers, and bringing in an invalid instruction to a datalogger could result in errors.

## 8.1.4 Documenting a DLD File

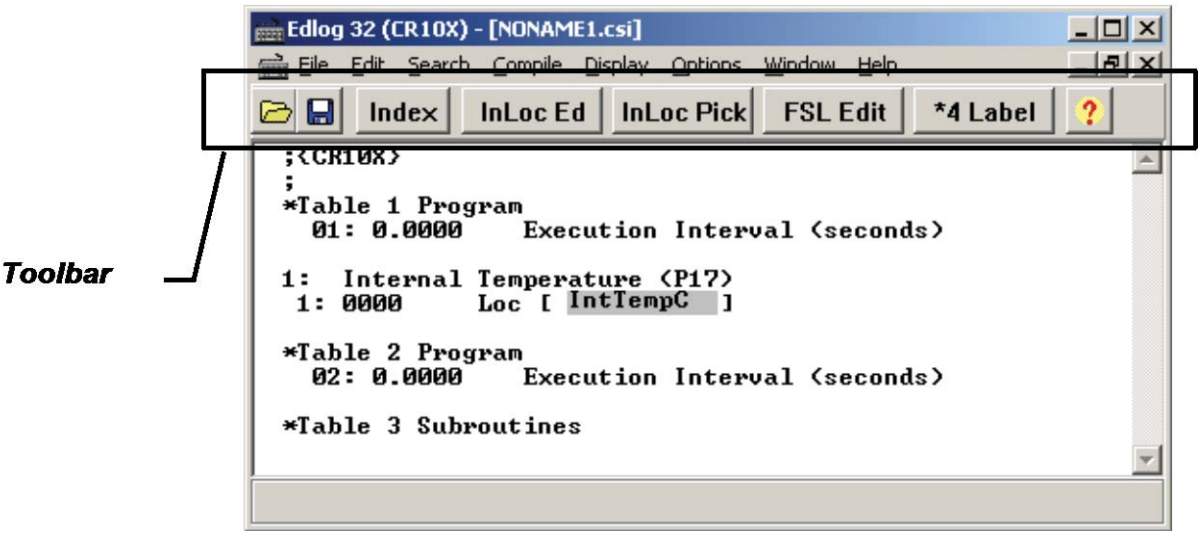
As noted in Section 8.1.1.2, *Edlog File Types* *(p. 8-4)*, the CSI file is the file created by Edlog that is used to generate the DLD code and other files. If for some reason your CSI file is missing, you can import the DLD file into Edlog to create another editable CSI file. From the Edlog menu select File | Document DLD. Select the DLD file to be imported and remember to save the file to create a new CSI file.

Programs created with the DOS versions of Edlog earlier than 6.0 were stored with the instruction description and comments in a *.DOC file instead of a *.CSI file. The DLD version of these programs can be imported into current versions of Edlog by using this Document DLD feature, though any comments will be lost.

## 8.1.5 Display Options

### 8.1.5.1 Graphical Toolbar

A graphical toolbar provides buttons for some of the more frequently used menu items in Edlog. The toolbar is made visible by choosing Options | Show Toolbar from the Edlog menu. Conversely, it is removed from the screen by choosing Options | Hide Toolbar.



Open a new file.

Save the current file to disk and optionally precompile the program.

Index — Index the parameter that is selected (for information on indexing, refer to your datalogger operator's manual).

InLoc Ed — Invoke the input location editor (for a discussion on input locations and the Editor, see Section 8.2, *Input Locations* (p. 8-16)).

InLoc Pick — Display the input location list; allows the user to select and insert an input location automatically into a parameter.

FSL Edit — Invoke the final storage label editor (for more information on editing final storage labels see Section 8.3, *Final Storage Labels* (p. 8-20)).

*4 Label — Assign a parameter a star 4 value (for information on star 4 values, refer to your datalogger operator's manual).

Open the on-line help system.

### 8.1.5.2 Renumbering the Instructions

When Automatic Renumbering is enabled, the instructions are automatically renumbered whenever instructions are inserted or deleted. By default, Automatic Renumbering is enabled. Automatic renumbering can be turned off by selecting Options | Editor if you have a very large program and auto renumbering is slowing down editing. If automatic renumbering is disabled, you can manually renumber the instructions by selecting Display | Renumber.

#### 8.1.5.3 Compress VIEW

When Display | Compress is selected, only the first line of each instruction is displayed. The compressed view makes it easier to see the program structure and to move around in the program.

Instructions cannot be edited in the compress view mode. Use Display | Uncompressed to switch back to the full view or use the <F7> function key to toggle between the compressed and full views.

#### 8.1.5.4 Indention

Indention is typically used with If Then/Else sequences and loops to provide a visual key to program flow. Indention is a visual aid; it has no meaning to the datalogger. If the programmer chooses to use indention, it can be done automatically or manually.

The settings for indention are found under Options | Editor. Turn on Automatic Indention by checking the box next to it. The distance for each indention (in spaces) is set on the same dialog box. To manually indent an instruction, place the cursor on one of the instruction's parameters and press either <Ctrl>+right arrow or <Ctrl>+left arrow; the instruction is indented the direction the arrow is pointing.

The Display | Rebuild Indention menu item resets all existing indentions and rebuilds automatic indentions. Automatic indentions may need to be rebuilt when editing instructions causes the indentions to misalign.

## 8.2 Input Locations

An input location is the space in datalogger memory where the most recent value is stored for each sensor. Each time a sensor is scanned, the input location is overwritten with a new value. Input locations are referenced in the datalogger by number.

In an Edlog program, each Input Location has an Input Location number and a label that appear whenever the Input Location is referenced in the program. Edlog automatically assigns Input Location numbers as labels are entered.

### 8.2.1 Entering Input Locations

When a parameter requires an Input Location, the cursor automatically advances to where the label is keyed in. When a new label is entered, the next available Input Location number is automatically assigned to that label. To select an existing label from a list, press the right mouse button or <F6>.

You may prefer to enter all input locations into the Edlog program before writing the program. This makes all the labels available from the input location pick list, and can help reduce programming errors because of typos. See Section 8.2.3, *Input Location Editor (p. 8-17)*.

Labels can have up to 9 characters for mixed-array dataloggers and 14 characters for table-based dataloggers. The first character must be a letter. The allowed characters are letters, numbers, and the underscore character ( _ ). The following labels are reserved for expressions and should not be entered by the user: CSI_R, CSI_2, CSI_3,... CSI_95.

To enter the Input Location number instead of the label, use the mouse or press <ctrl> left arrow.

## 8.2.2 Repetitions

Many input/output and output processing instructions have a repetitions parameter. Repetitions (REPS) allow one programming instruction to measure several identical sensors or to process data from several Input Locations. When REPS are greater than 1, the Input Locations are assigned consecutive numbers (e.g., with REPS of 2 and LOC of 5, the Input Locations are 5 and 6). Each rep label is the initial label with a "_" and the next consecutive number (i.e., with 3 REPS and a label of "data" the labels for each REP are: data_1, data_2, and data_3).

Only the first input location of an instruction is linked to the instruction. Reps of input/output instructions and output processing instructions are not linked, so use care if altering their sequence in the Input Locations Editor.

As an example, in the following section of an Edlog program, the TempC and BatteryV Input Locations are sampled with one sample (P70) instruction, with the REPS parameter of 2.

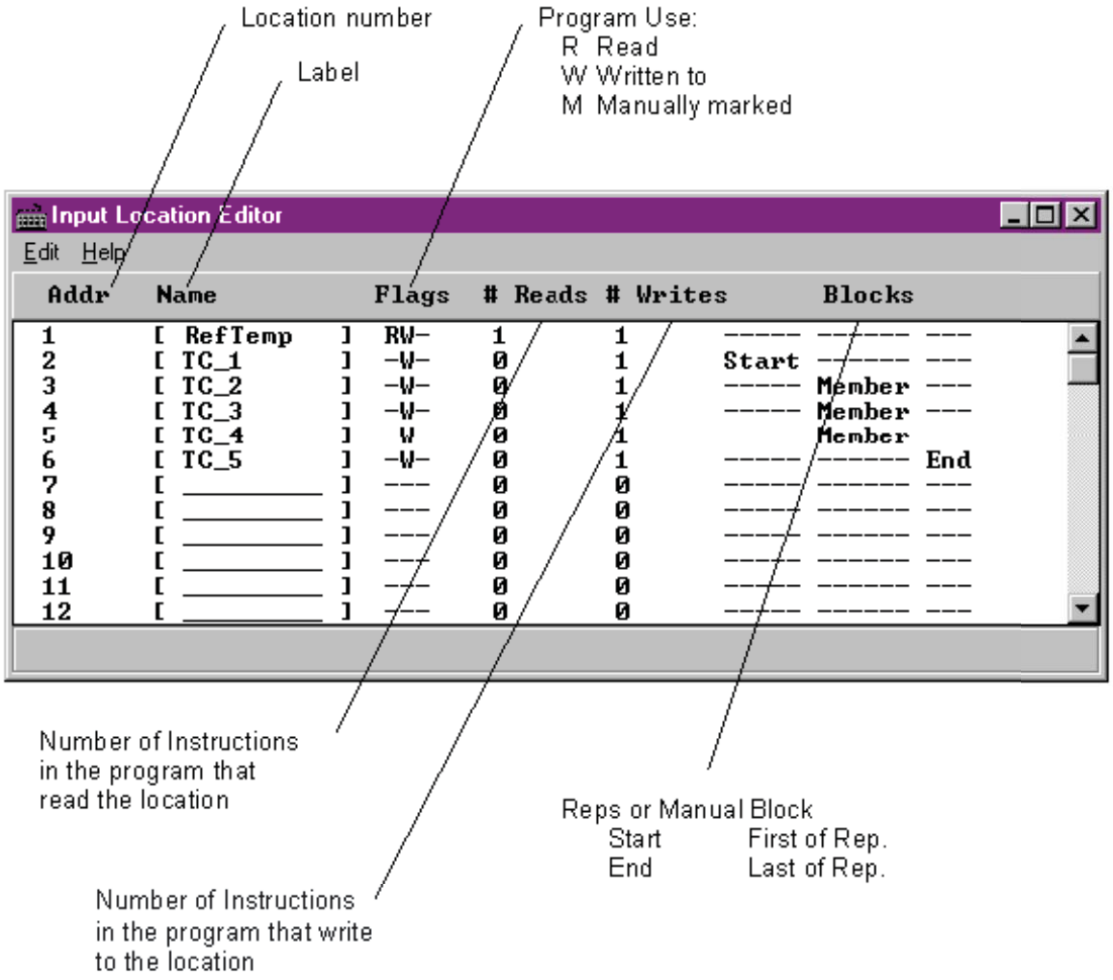
```
10:  Temperature (107) (P11)
   1:    1              REPS
   2:    2              Channel
   3:    1              Excitation Channel
   4:    1              Loc [TempC]
   5:    1              Mult
   6:    0              Offset

11:  Battery, Volt (P10)
   1:    2              Loc [BatteryV]
12:  If time is (P92)
   1:     0             minutes into interval
   2:    60             minute interval
   3:    10             Set high Flag 0(output)

13:  Sample (P70)
   1:    2              Reps
   2:    1              Loc [TempC]
```

When the program is executed, the datalogger will perform the Sample (P70) instruction twice. The first time, it will sample the value stored in the TempC location. The second time, it will sample the value stored in the BatteryV location.

**NOTE** If an Input Location is inserted between the TempC and BatteryV location, the inserted location will be sampled instead of BatteryV.

## 8.2.3 Input Location Editor

Input Location labels can be entered and edited by using the Input Location Editor. To access the Input Location Editor, select Edit | Input Labels or press <F5>.




Editing functions are available from the Input Location Editor's Edit menu and a hot key:

Insert (<F2>) – Inserts blank Input Locations. This is used to provide space for new input labels between existing labels. This automatically changes the Input Location numbers for all of the labels that are after the inserted location.

Delete (<F3>) – Deletes the Input Location label, flags, number of reads and writes, and block information for a designated location number. Wherever the datalogger program references a deleted location label, the Input Location's number automatically becomes 0.

Move (<F4>) – Moves the Input Location to a different number. This may change several Input Location numbers.

Toggle Manual (<F5>) – Allows the programmer to manually toggle a location as "in use". This is used for burst mode, indexed loops, or other situations where it's not clear to Edlog that the locations are being written to. Input Locations not marked as read, write, or manual are deleted by the Optimize command.

Optimize (<F6>) – Deletes Input Locations that aren't read, written to, or marked as Manual. Optimize tries to reduce the total number of locations used by moving existing Input Location labels to fill in unused locations. This might change several Input Location numbers. Any changes in location number made by the Optimize command are reflected in the Edlog program.

Insert Block (<F7>) – Inserts and labels a block of Input Locations and marks them as "Manual". The locations are labeled in the same manner as reps.

Esc – The escape key closes the Input Location Editor and updates the label assignments in the program.

## 8.2.4 Input Location Anomalies

In most instances, Edlog will automatically assign Input Locations for locations which are generated by the datalogger program. An example of this is Edlog's handling of Input Locations for the REPS parameter. Though only one Input Location is specified, if REPS is greater than 1, additional Input Locations are created by Edlog.

There are certain instructions that generate multiple Input Locations for which Edlog does not automatically allocate Input Locations. The user should manually allocate these locations in the Input Location Editor. These are:

- Instruction 15, Serial I/O with Control Port
- Instruction 23, Burst Measurement
- Instruction 49, Spatial Maximum
- Instruction 50, Spatial Minimum
- Instruction 54, Block Move
- Instruction 75, Histogram
- Instruction 80, Store Area
- Instruction 81, Rainflow Histogram
- Instruction 100, TDR Measurement
- Instruction 101, SDM-INT8
- Instruction 105, SDI-12 Recorder
- Instruction 106, SDI-12 Sensor
- Instruction 113, SDM-SIO4
- Instruction 118, SDM CAN
- Instruction 119, TDR100
- Instruction 120, Data Transfer to TGT
- Instruction 127, HDR Goes Status and Diagnostics
- Instruction 128, SHEF Data Transfer to TGT
- Instruction 139, Detailed Program Signatures
- Instruction 188, SDI-IO16

- Instruction 189, SDM-LI7500
- Instructions P190-199 PakBus control
- Indexed input locations in a loop

See Edlog Help for each instruction to get a detailed description of input location usage. You can also refer to the datalogger user's manual for more information on these instructions.

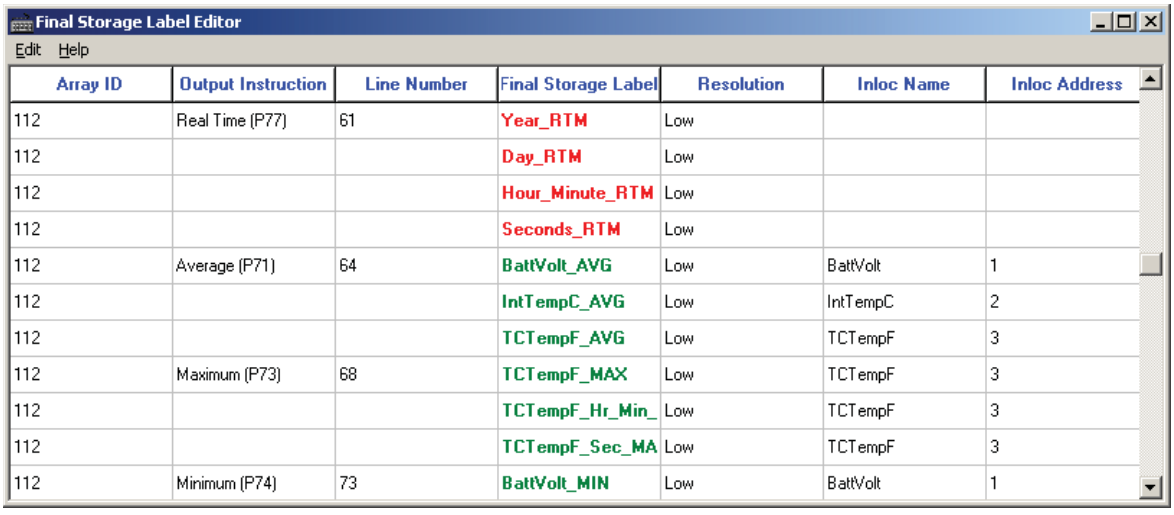
When these instructions are used in a program, the Toggle Manual feature can be used to manually mark Input Locations for use by the program.

# 8.3 Final Storage Labels

When output processing instructions are added to the datalogger program, Edlog creates final storage labels for the each of the values that will be stored. The default labels are normally the input location label with a suffix indicating the type of output processing instruction that created it. In the example below BattVolt_AVG is the average battery voltage that is stored as part of array 112.

For mixed-array dataloggers the final storage labels are stored in an *.FSL file when the program is compiled, as well as in the DLD files. For table-based dataloggers the final storage labels are included as part of the datalogger program in the *.DLD file; no FSL file is created. PC400 gets the final storage labels as part of the table definitions from the datalogger. Split, the Graphical and Numeric Displays, View Pro, and the PC400 Data applications use the final storage labels.

The user can create a custom label to reflect the meaning of the value that is being stored. Click the FSL Edit button on the toolbar or press F9 to bring up the Final Storage Label Editor as shown below.




| Array ID | Output Instruction | Line Number | Final Storage Label | Resolution | Inloc Name | Inloc Address |
|---|---|---|---|---|---|---|
| 112 | Real Time (P77) | 61 | Year_RTM | Low | | |
| 112 | | | Day_RTM | Low | | |
| 112 | | | Hour_Minute_RTM | Low | | |
| 112 | | | Seconds_RTM | Low | | |
| 112 | Average (P71) | 64 | BattVolt_AVG | Low | BattVolt | 1 |
| 112 | | | IntTempC_AVG | Low | IntTempC | 2 |
| 112 | | | TCTempF_AVG | Low | TCTempF | 3 |
| 112 | Maximum (P73) | 68 | TCTempF_MAX | Low | TCTempF | 3 |
| 112 | | | TCTempF_Hr_Min_ | Low | TCTempF | 3 |
| 112 | | | TCTempF_Sec_MA | Low | TCTempF | 3 |
| 112 | Minimum (P74) | 73 | BattVolt_MIN | Low | BattVolt | 1 |

In this example from a mixed-array datalogger, the final storage output data for Array ID 112 is shown. Each of the columns indicate the essential characteristics of the data value being stored.

- Array ID or Table Name identifies the set of output data instructions the data is associated with. For mixed-array dataloggers the array ID is at the beginning of each output record. In table-based dataloggers, the table name shows the name of the table where the data values will be stored.
- Output Instruction lists the output instruction that was used to store the data value.
- Line Number is the line number in the Edlog program for the output instruction.
- Final Storage Label is the label that is associated with this final storage value. Red labels are associated with automatically created data entries such as time stamps and record numbers. The red labels cannot be changed with the Final Storage Label Editor. The green labels are associated with user programmed sensor data. To change the label, click in the box and type in the new label.
- Resolution shows whether the data will be stored in low or high resolution. (High resolution stores data as a 4-byte floating point number, Low resolution uses a 2-byte number)
- Inloc Name is the label of the input location that the final storage data is based on.
- Inloc Address is the numeric label for the input location used for the final storage data value.

**NOTE** If changes are made to measurement or output instructions after custom final storage labels have been created, you should review the custom final storage labels to make sure the correct labels are still assigned to the desired output values. Some program changes involving an increase or decrease in input locations or output values could cause a label to no longer correspond with the value being output.

The final storage labels created by Edlog can be restored by selecting the menu item Edit | Restore Default Labels from the Final Storage Label Editor menu.

## 8.4 Datalogger Settings Stored in the DLD File

Certain settings for the datalogger, which are normally accessed through the datalogger's * modes, can be included in the DLD file. These settings include options such as program security, final storage allocation, the type of labels saved in the DLD file, power up and compilation settings, and PakBus address and router settings. When the new program is downloaded to the datalogger and compiled, the settings will take affect. These settings are accessed using Edlog's Options menu.

## 8.4.1 Program Security

Setting security in the datalogger allows you to restrict access to certain functions, which helps ensure the program or data are not altered. Security is unlocked in the datalogger when, upon attempting to connect, PC400 sends the code entered in the Setup window.

### 8.4.1.1 Setting Passwords in the DLD

In the Program Security Dialog Box, there is a field for three levels of security. Enter a non-zero number in the appropriate field to enable security at the desired security level. This number is used as a password to unlock the associated security level when needed. If you choose to set level 02, level 01 must also be set. Likewise, if you set level 03, levels 01 and 02 must be set.

**NOTE** CR7 and 21X dataloggers have only 1 level of security.

### 8.4.1.2 Disabling Passwords

Passwords of 0000 disable the program security. When you disable level 01, you also disable program security for levels 02 and 03. Similarly, disabling level 02 disables level 03. All passwords are set to 0000 upon power-up of the datalogger. When the program is run, security is enabled.

Refer to the datalogger manual or Edlog's help file for additional information on Security in the datalogger.

## 8.4.2 Final Storage Area 2

The ring memory for CR10, CR10X, CR510, and CR23X dataloggers can be divided into two final storage areas. By default, all memory is allocated to final storage area 1. However, the datalogger's memory can be partitioned into two final storage areas using this option. To allocate memory to Final Storage Area 2, enter the number of locations into the Final Storage Area 2 Locations field.

Final storage area 1 is reduced by the amount of memory allocated to final storage area 2. Data stored in final storage area 2 is protected when Input and/or Intermediate Storage is reallocated. Data stored in final storage area 1 is erased when any of the memory areas are reallocated.

## 8.4.3 DLD File Labels

This option allows you to determine the labels that are saved in the DLD file for the datalogger. While labels are useful, they can increase the size of the datalogger program considerably. If program size is a concern, you can limit or completely eliminate the labels that are saved in the DLD file.

### 8.4.3.1 Mixed-array Dataloggers

Mixed-array dataloggers can store the labels for input locations and final storage output in the DLD file. PC400 uses this information on the Monitor Data display. If you do not include these labels in the DLD file, you will see generic names for input locations, and will not be able to display final storage locations at all.

Options for mixed-array dataloggers are:

**Minimize DLD Size –** No input location labels or final storage labels are saved in the DLD file.
**Default** – Up to 255 input location labels and all final storage labels are saved in the DLD file.
**All** – All input location labels and all final storage labels are saved in the DLD file.

#### 8.4.3.2 Table-Based Dataloggers

Table-based (both TD and PB) dataloggers store all final storage labels in the DLD file and there is no option to remove or reduce them. If you do not include input location labels in the DLD file, you will not be able to display input locations on the displays in PC400.

The label options for table-based dataloggers are:

**Include All Input Location Labels –** All input location labels are saved in the DLD file.
**Include First X Input Location Labels** – Allows you to specify a certain number of input location labels to be saved in the DLD file.

If you are trying to minimize the size of your DLD file but still want to be able to monitor input locations on PC400's Monitor Data tab, you can put all of the labels that you want to view at the beginning of your list of input locations, and put the labels for scratch and less important values at the end. Then, use the second option above to display only those values of interest.

### 8.4.4 Power Up Settings/Compile Settings

These two options allow you to clear or retain settings for ports, flags, storage locations, and timers when the datalogger is powered-up or when a program is compiled. Whether it is advantageous to clear or retain these settings depends on your application. For most applications, it is best to keep the default option of Do not change current datalogger Power-up settings. The affected settings are:

**Port Status** – The state of the ports (high/low) the last time the datalogger was on.
**Flag Status** – The state of the flags (high/low) the last time the datalogger was on.
**User Timer** – Allows you to continue timing events that occurred when the datalogger was on last.
**Input Storage** – Allows the values that were stored in the input locations before you turned the datalogger off to be included in the sample, average, and total when you turn the datalogger back on.
**Intermediate Storage** – Allows data processing to continue from when the datalogger was on last.

**NOTE** Not all dataloggers have a Compile Settings option. This option refers only to the CR510, CR10X, and CR23X.

## 8.4.5 Datalogger Serial Port Settings

The serial port settings are used to set the baud rate to which the datalogger's port(s) should be set when the datalogger is powered-up or when a program is compiled. If the "Do not change current CS I/O Port settings" option is selected, the baud rate option used will be that at which the datalogger is currently using.

When the "Fixed Baud Rate" check box has been selected, the datalogger is forced to communicate at the baud rate selected. When it is not selected, the datalogger will first try to use the initial baud rate, but will try the other baud rates if it cannot connect.

The CR23X has an "RS232 Power Always On" check box. This keeps the power to the RS232 port on at all times. In some instances, this may be desirable but it consumes much more power than when the datalogger turns on the port as needed.

**NOTE** Not all dataloggers have a Serial Port Settings option. This option refers only to the CR510, CR10X, and CR23X.

## 8.4.6 PakBus Settings

PakBus dataloggers have various settings that allow them to function properly in a PakBus network. In Edlog dataloggers with PB operating systems, these options can be set in the datalogger's *D mode with a keyboard/display, but they can also be set in the DLD program file

For any of the options, if the check box Do Not Change Current Settings is enabled, then those settings will not be changed when the program is downloaded to the datalogger.

### 8.4.6.1 Network

The Network option is used to set the PakBus address in the datalogger and to configure the datalogger as a router if required. This option is the same as the datalogger's *D15 mode.

**Address** – Enter the PakBus address that should be assigned to the datalogger. Each node (PakBus addressable device) in the PakBus network should have a unique PakBus address.
**Maximum number of nodes –** Enter the total number of nodes (including leaf node and router dataloggers, non-datalogger routers such as NL100s, and PCs) in the PakBus network.
**Maximum number of neighbors –** Enter the number of dataloggers in the PakBus network that the datalogger can communicate with directly (i.e., without going through another router).
**Maximum number of routers –** Enter the number in the PakBus network, including the PC.

While it is possible to calculate the exact number of nodes, neighbors, and routers in a PakBus network, it is often advisable to build in some "room to grow". For example, you might want to add 3-4 nodes, neighbors and routers. Be aware that each device you add means the datalogger must allocate memory for its routing table, so if you add too many, the datalogger won't have enough memory left to run its program.

### 8.4.6.2 Beacon Intervals

This option is used to set the interval on which the datalogger will transmit a beacon out a particular port to the PakBus network. Use the drop-down list box to select the port over which the beacon will be transmitted, and enter the desired interval in the Communications Interval field. This option is the same as the datalogger's *D18 mode.

**NOTE** In some networks, a beacon interval might interfere with regular communication in the PakBus network (such as in an RF network), since the beacon is broadcast to all devices within range. In such cases, it may be more appropriate to use the Neighbor Filter instead, which broadcasts a beacon only to those dataloggers which it has not received communication from within a specified interval.

### 8.4.6.3 Neighbor Filter

This option allows you to list expected neighbors that are available to the datalogger in the PakBus network. The datalogger will attempt to issue a "hello" command to all the dataloggers listed in the neighbors filter list, and will transmit an expected communication interval. The communication interval is the interval on which the datalogger expects to receive communication from the neighbors. If communication is not received from a neighbor within 2.25 times this interval, then the datalogger will attempt to issue another "hello" command to that datalogger only (thus, creating less network traffic than the Beacon Interval).

The expected interval is entered into the Communication Interval field in seconds. The neighbors are defined by entering their addresses into the table. A range of addresses can be entered by using the Swath field. For example, entering 1 for the address and 5 for the swath will set up dataloggers with PakBus addresses 1, 2, 3, 4, and 5 as neighbors to the current datalogger. This option is the same as the datalogger's *D19 mode.

### 8.4.6.4 Allocate General Purpose File Memory

PakBus dataloggers have the ability to store files transmitted from an NL100 in a general purpose memory area. This memory area is configured as ring memory. A value can be entered to specify the number of 64K blocks of memory that should be used for this purpose. Final storage memory will be reduced by the amount of memory specified in this option. This option is the same as the datalogger's *D16 mode.

# Section 9. Datalogger Program Creation with CRBasic Editor

*This section provides information on the CRBasic Editor used to program the Campbell Scientific CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, CR9000, and CR200 Series dataloggers. CRBasic is a full-featured programming language providing the power and flexibility to set up complex datalogger programs to support demanding measurement tasks.*

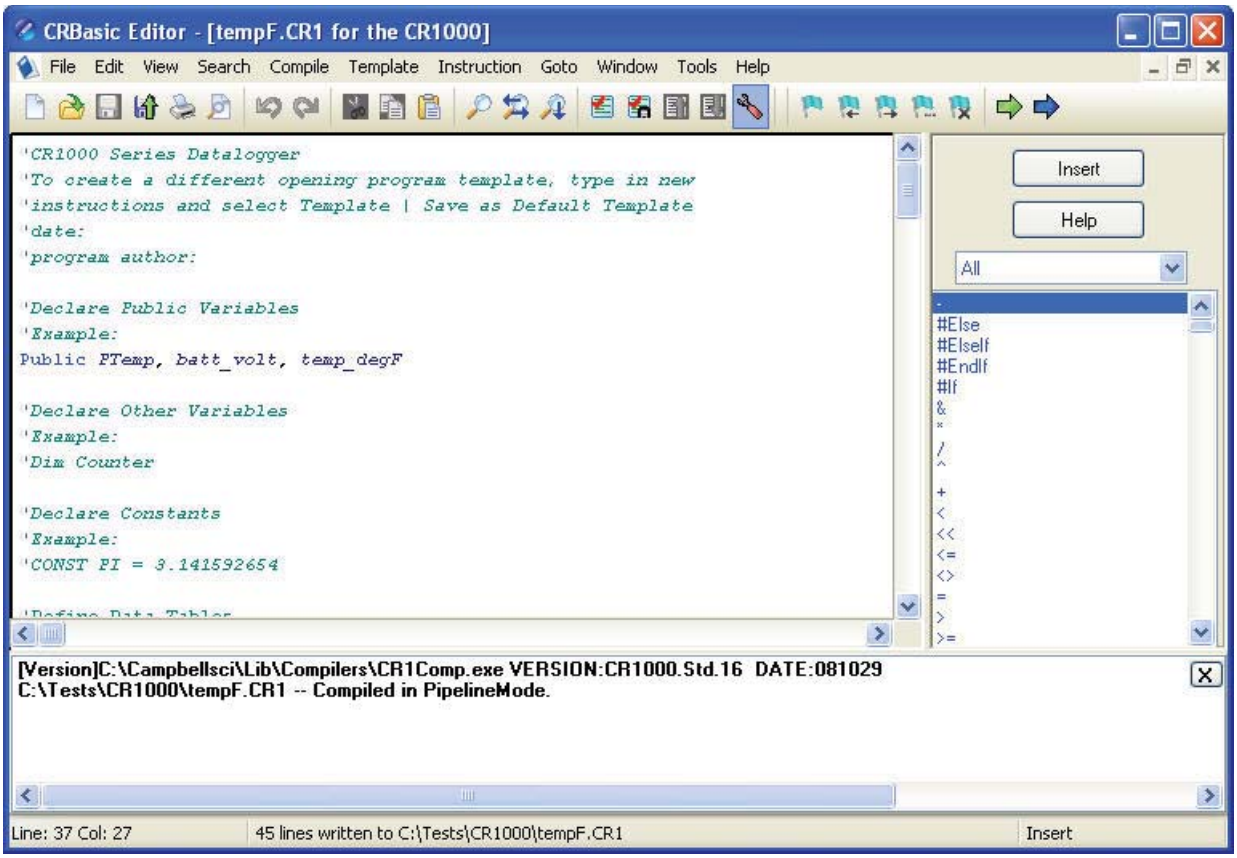
*Datalogger programs can also be created using the Short Cut program generator, see Section 7, Short Cut Program Generator (p. 7-1).*

*See Section 8, Datalogger Program Creation with Edlog (p. 8-1), for information about Edlog, the program editor for other Campbell Scientific dataloggers.*

## 9.1 Overview

The **CRBasic Editor** is a programming tool which can be used with the CR1000, CR6 Series, CR300 Series, CR3000, CR200 Series, CR800 Series, CR5000, CR9000 and CR9000X dataloggers. It is intended for use by experienced datalogger programmers who need more flexibility and control over the datalogger operation than what can be achieved using Short Cut. This programming language is similar in syntax, program flow, and logic to the Structured BASIC programming language.

As shown below, the CRBasic Editor's main window is divided into three parts: the *Program Entry Window*, the *Instruction Panel*, and the *Message* area. The Instruction Panel on the right side is a list that comprises the instructions for a particular datalogger in the CRBasic language. Instructions can be selected from this list or entered directly into the Program Entry Window on the left. The Message area at the bottom becomes visible after a program is compiled and shows results of the compile and any errors detected.




# 9.2 Inserting Instructions

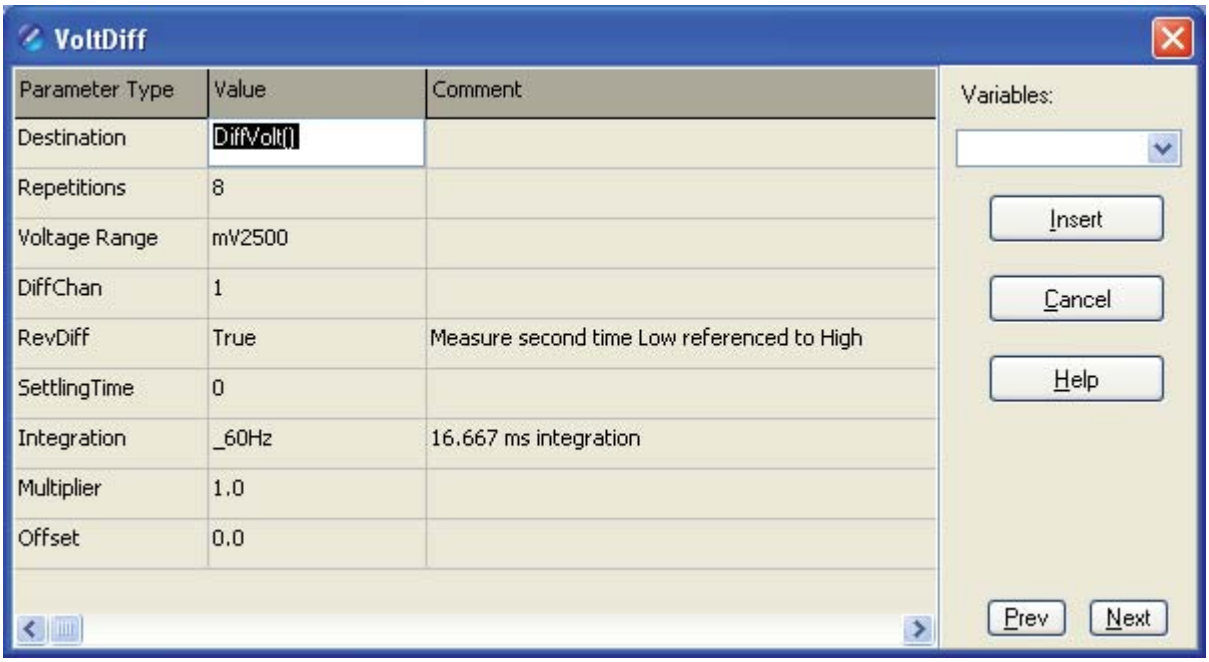
An instruction can be easily inserted into the program by highlighting it in the Instruction Panel list and pressing the **Insert** button or by double-clicking the instruction name. If an instruction has one or more parameters, an instruction dialog box will be displayed to facilitate editing the parameters. Complete the information in the parameter fields and press Insert to paste the instruction into the program. (You may disable this instruction dialog box by clearing the option in the **View | Instruction Panel Preferences | Show Instruction Dialog** check box.)

You can filter the list of instructions available in the Instruction Panel by clicking the drop-down arrow to the right of the text box above the list. This will allow you to display only instructions of a specific type such as *Measurement* or *Program Structure/Control*. This provides a smaller list to select from and makes it easier to find the instruction you want. Switch back to *All* to see all of the instructions available. You can create custom instruction filter lists as described later in this section.

## 9.2.1 Parameter Dialog Box

The **Parameter** dialog box will appear when an instruction is added that has one or more parameters or when the cursor is placed on an existing instruction and the right mouse button is pressed. This dialog box contains a field for each of the parameters in the instruction. Edit these fields as necessary and then press the **Insert** button to paste the instruction into the program.

Below is an example of the **Parameter** dialog box for the differential voltage instruction (*VoltDiff*).




The **Prev** (Previous) and **Next** buttons can be used to move to the next (or previous) instruction with the parameter entry box opened.

### *Short Cuts for Editing the Parameters*

Right-clicking or pressing *F2* on a parameter that uses a variable as an input type will display a list of variables that have been defined in the program. A sample list is shown below.

The variable list is sorted by variable type and then alphabetically by name. In the list above, the first green *A* denotes that the variable *AIRCOOL* is set up as an Alias.

Constants are listed with a blue *C*, Dimensioned variables are listed with a red *D*, and Public variables are listed with a black *P*.

At any time you can press *F10* to bring up the list of variables, regardless of the input type for the selected parameter. Also, defined variables can be selected from the **Variables** drop-down list box at the upper right of the **Parameter** dialog box.

Pressing *F9* at any time will also bring up a list of variables. However, when a variable is chosen from the list brought up by *F9*, it will simply be inserted at the cursor without overwriting anything.

Right-clicking or pressing *F2* on a parameter that has a finite number of valid entries will bring up a list of those available options.

Right-clicking or pressing *F2* on a parameter that does not fall within the two categories above will bring up help for that parameter.

Pressing *F1* with any parameter selected will bring up help for that parameter along with a list of possible options where appropriate.

#### *Changing Default Parameters Values for an Instruction*

Each instruction offers default values for each parameter. For instance, in the Parameter box above, the default for the Range is *mV5000*. If you wanted to edit this so that each time you inserted the *VoltDiff* instruction the Range value defaulted to *mV1000*, you would highlight the instruction in the Instruction Panel, select **Instruction | Edit Instruction Defaults** from the menu, and make the change in the resulting dialog box.

## 9.2.2 Right-Click Functionality

The result of a right-click action varies, depending upon your cursor location.

Right-click an instruction name to show the **Parameter** dialog box to edit the instruction parameters.

Right-click a parameter that uses a variable as an input type to bring up a list of variables that have been defined in the program as described in the previous section.

Right-click a parameter that has a finite number of valid entries to bring up a list of those available options. You can change the option by clicking the desired option.

Right-click another type of parameter to bring up help for that parameter.

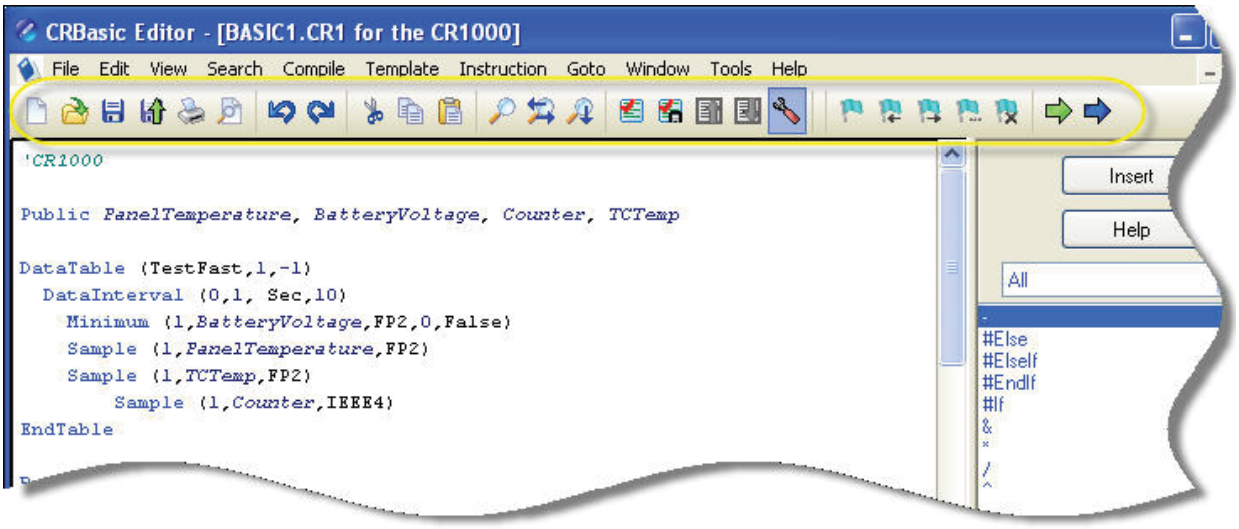
Right-click a block of text that is highlighted to bring up a short cut menu with the following options:

- **Comment/Uncomment Block**: Only one of these options will be available, depending upon the status of the highlighted text. If the text has been marked as a comment, you can choose to uncomment it. If the text is not commented, you can chose to make it into a comment. Commented text has a single quote ( ' ) at the beginning of the line. Comments are ignored by the datalogger's compiler.
- **Decrease/Increase Indent**: You can increase or decrease the indention of the selected text. The spacing is increased or decreased by one.
- **Cut/Copy/Paste/Delete**: Standard editing functions can be accessed through this menu.
- **Save as .CRB File**: Saves highlighted text to a file with a *.CRB extension. This file is referred to as a "library file". The file can then be reused by inserting it into another CRBasic program.

- **Insert File**: Inserts a library file into the current program overwriting the highlighted text.

## 9.3 Toolbar

The toolbar of the CRBasic Editor provides easy access to frequently used operations.



**New** – Creates a new program window to start writing a new program. If you have defined a default template, the new program will start with the defined template instructions.

**Open** – Brings up a File Open dialog to select a program file to open. File extension filters are provided to list only files of a certain type such as .cr5 files for CR5000 programs. Data files (*.dat) can also be opened.

**Save** – Saves any changes to the currently opened program. If this is a new program and has not been saved yet, a Save As dialog will prompt you for the file name and location to save the file. A table definition file (*.tdf) of the same name as the saved program will also be created. Refer to the online documentation for more information about using table definition files.

**Compile, Save, and Send** – Saves any changes to the currently opened program, checks it for errors with the pre-compiler, and sends the file to the datalogger via LoggerNet, PC400, or RTDAQ. LoggerNet, PC400, or RTDAQ must be running for this function to work properly.

**Print** – Prints the currently opened program.

**Print Preview** – Opens a Print Preview screen that will show what the program will look like when printed. You can check and set the margins and printer options.

**Undo** – Each time the Undo button is clicked it will step back through the last changes made to the program.

**Redo** – Cancels the undo and steps forward restoring the changes.

**Cut** – Removes the selected part of the program and puts it on the clipboard to be pasted elsewhere.

**Copy** – Places a copy of the selected part of the program on the clipboard to be pasted elsewhere.

**Paste** – Inserts a copy of the contents of the clipboard into the program at the cursor location.

**Find** – Brings up a Find dialog to specify a text string to search for in the program. Click the Find Next button or press F3 to go to successive occurrences of the text.

**Replace** – Brings up a Find and Replace dialog that allows you to specify a text string to search for and a text string to replace it with. You can replace all occurrences of the text or check them one at a time to make sure they should be replaced.

**Find Next** – Finds the next occurrence of the text string specified in the Find dialog.

**Compile** – Starts the compiler to check the current program for errors and consistency. Compile results and errors will be displayed in the message area at the bottom of the screen.

**Save and Compile** – Saves and then compiles the opened file.

**Previous Error** – Moves the cursor to the part of the program where the previous error was identified.

**Next Error** – Moves the cursor to the part of the program where the next error was identified.

**Instruction Panel** – Controls whether the Instruction Panel is displayed. Hiding the Instruction Panel allows more room in the window to view the program.

**Toggle Bookmark** – Adds a bookmark to the line where the cursor resides. If a bookmark already exists, it will remove the bookmark.

**Previous Bookmark** – Moves backward to the previous bookmark in the program.

**Next Bookmark** – Moves down to the next bookmark in the program.

**Browse Bookmarks** – Displays a list of all bookmarks in the program. When a bookmark is selected, the cursor moves to that line in the program.

**Clear Bookmarks** – Erases all bookmarks from the program.

**GoTo** – Moves the cursor to a particular section of the program. Choose the section type from the list box that appears.

**User-Defined Functions and Subroutines** – Provides a list box containing all of the user-defined functions and subroutines. Functions are identified with a purple *F*. Subroutines are marked with a black *S*. Clicking on a name in the list box moves the cursor to the start of that function or subroutine.

## 9.3.1 Compile

**Compile** is a function provided by the CRBasic Editor to help the programmer catch problems with the datalogger program. **Compile** is available from the toolbar and the Compile menu.

When the Compile function is used, the CRBasic Editor checks the program for syntax errors and other inconsistencies. The results of the check will be 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. You can double-click an error preceded by a line number and that line will be highlighted in the program editing window. To move the highlight to the next error in the program, press the **Next Error** button or choose **Next Error** from the **Compile** menu. To move the highlight to the previous error in the program, press the **Previous Error** button or choose **Previous Error** from the **Compile** menu.

It is important that the compilers used for checking programs match the OS version loaded in the datalogger, otherwise errors may be returned when the program is sent. When a CR200 program is being edited, the **Pick CR200 Compiler** menu item is available. This item opens a dialog box from which a compiler can be selected for the CR200 datalogger.

The error window can be closed by selecting the **Close Message** Window menu item from the **View** menu, or by clicking the **X** in the upper right corner of the message window.

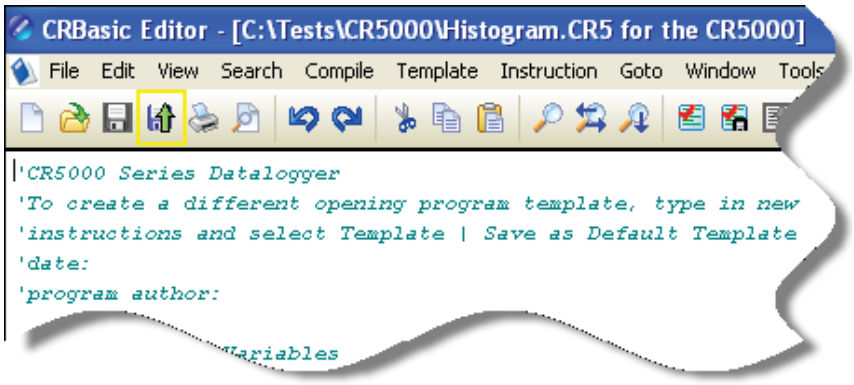
**NOTE** For the CR1000, CR6 Series, CR300 Series, CR3000, CR800 series, CR5000 and CR9000X dataloggers, the Compile function only verifies the integrity of the program. Actual compilation of the program takes place in the datalogger. When using the CR200 datalogger, however, this function creates a binary image of the program to be loaded to the datalogger at a later time. This function is not available for the CR9000 datalogger.

## 9.3.2 Compile, Save, and Send

The CRBasic Editor allows you to send a program to a datalogger that has already been defined on the network map in LoggerNet, PC400, or RTDAQ. This only works if LoggerNet, PC400, or RTDAQ is running at the time you attempt to send the program.

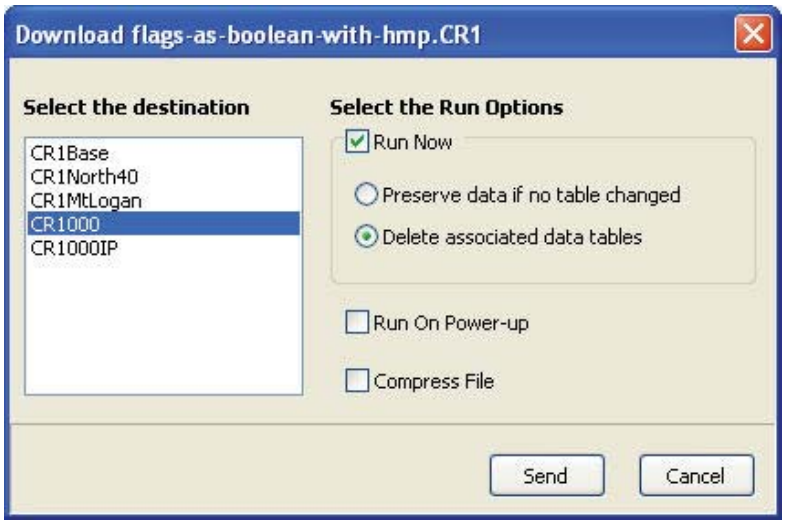
This function first checks the program for errors using the pre-compiler, then saves the program (using the current name, or by prompting the user for a name if the program is new). After the compile and save, this function sends

the program to a user-specified datalogger. To do this, use the **Compile, Save and Send** item on the **File** menu or **Compile** menu, or you can press the corresponding button on the toolbar.




**NOTE** When a file is sent to the datalogger using Compile, Save, and Send and the software is not actively connected to the datalogger, the software connects to the datalogger, sends the file, retrieves table definitions, and then disconnects. There will be little indication in the software that a connection was established.

When this function is chosen a dialog box is displayed. Below is the dialog box for a CR1000 datalogger:




The **Select the destination** list shows all dataloggers configured within LoggerNet, PC400, or RTDAQ that may receive a program matching the extension of the current CRBasic program to be sent. Assume, for example, that you have three CR1000s and some other dataloggers in your LoggerNet, PC400, or RTDAQ network map. When you send a *.CR1 program, this screen will show only the three CR1000 dataloggers. Any other dataloggers will be excluded from the list in this case, even when they are defined in the network map, because those dataloggers are not associated with *.CR1 programs. A program with the extension of .DLD will be associated with all CRBasic-programmed datalogger types.

Select the datalogger to send the file to, and then select the Run Options.

### Run Now

The Run Now run options are different for the different datalogger types.

#### *CR1000/CR6 Series/CR300 Series/CR3000/CR800 Series Datalogger Run Now Options*

When Run Now is checked, the file will be sent with the Run Now attribute set. With this attribute, the program is compiled and run in the datalogger. You may choose to preserve existing data tables on the datalogger's CPU if there has been no change to the data tables (**Preserve data if no table changed**) or to delete data tables on the CPU that have the same name as tables declared in the new program (**Delete associated data tables**).

**CAUTION**

Neither of these options affects existing data files on a card if one is being used. If a data table exists on the card that has the same name as one being output with the new program, the message will be returned "Data on Card is from a different program or corrupted". **Data will not be written to the card until the existing table is deleted.** Data tables on the card that have different names than those declared in the new program will be maintained and will not affect card data storage when the new program is running.

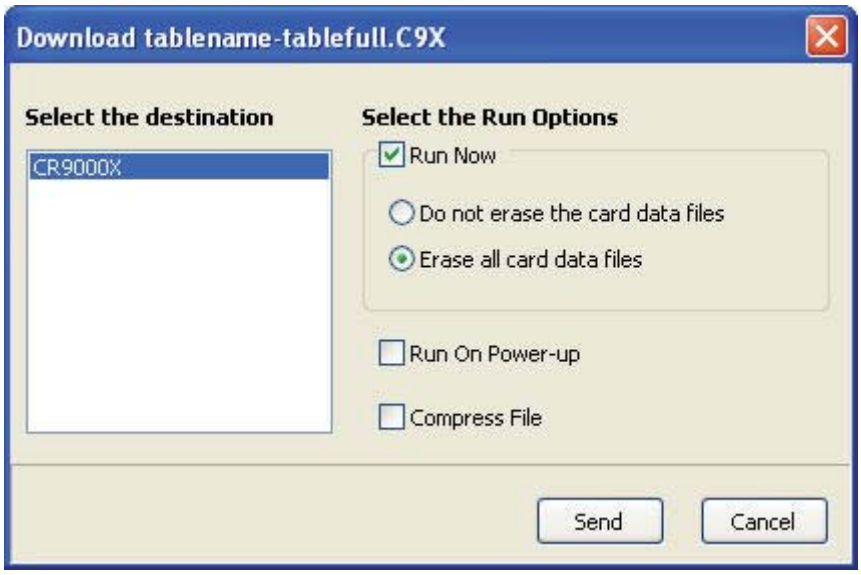
When using the **Preserve data if no table changed** option, existing data and data table structures are retained unless one of the following occurs:

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

To summarize, any change in data table structure will delete all tables on the datalogger's CPU, regardless of whether or not the Preserve Data option was chosen. If the Preserve Data option was chosen but the datalogger was unable to retain the existing data, the following message will appear in the Compile Results: Warning: Internal Data Storage Memory was re-initialized.

### *CR9000(X)/CR5000 Datalogger Run Now Options*

The Run Now options and behavior for the CR9000(X) and CR5000 dataloggers are different from the CR1000, CR6 Series, CR300 Series, CR3000, and CR800 Series dataloggers. Below is a dialog box for a CR9000X datalogger.




When Run Now is checked, the file will be sent with the Run Now attribute set. With this attribute, the program is compiled and run in the datalogger. All data tables on the CPU are erased. You have the option of whether or not to erase data files stored on a card.

#### Run On Power-up

The file will be sent with the Run On Power-up attribute set. The program will be run if the datalogger loses power and then powers back up.

#### Run Always

Run Now and Run On Power-up can both be selected. This sets the program's file attribute in the datalogger as Run Always. The program will be compiled and run immediately and it will also be the program that runs if the datalogger is powered down and powered back up.

#### CR200 Datalogger Run Options

The CR200 does not have an on-board compiler. A compiled binary (*.bin) file is sent to the datalogger. Run options are not applicable to this datalogger and are therefore disabled.

#### Compress File

If the **Compress File** check box is selected, a renamed version of the CRBasic program which has all unnecessary spaces, indentation, and comments removed in order to minimize the file size will be sent to the datalogger instead of the original program.

### Sending the Program

To send the file and perform the associated functions you have selected in the screen, press the **Send** button. If LoggerNet, PC400, or RTDAQ is not running, an error message will appear indicating that there is no communications server currently running. If LoggerNet, PC400, or RTDAQ is running and the program compiles properly on the hardware, you will receive a message indicating that the program is now running on the datalogger. If something goes wrong when sending the program, a message will appear indicating the error conditions. This may be a hardware-level compile error or another failure as reported to the software by the datalogger's program load and run process.

Press **Cancel** if you do not wish to send the program to the datalogger.

**NOTE** When sending a program with the Compile, Save, and Send feature to a CR9000X datalogger while you are connected to the datalogger, you may get a disconnect message or similar notification. This is unique to the CR9000X datalogger and does not indicate any problem with the sending of the program. You can simply reconnect to the datalogger and continue your work.

## 9.3.3 Conditional Compile and Save

The **Conditional Compile and Save** option is used to generate a new CRBasic program from code that uses conditional compile syntax (#If/Else/ElseIf statements) or constant customization. (See conditional compilation in the CRBasic Editor's online help for more information on conditional compile syntax. See Section 9.3.9.2, *Constant Customization* *(p. 9-17)*, for more information on constant customization.)

When a program is compiled that uses conditional syntax, any conditional compilation statements that do not evaluate as true are removed from the program and the program is compiled. When a program is compiled that uses constant customization, the constant values selected in the Tools | Customize Constants menu item are used when compiling the new program. In either instance, you are prompted to save the file under a user-specified name or the file will be saved under the name of the original program with _CC# appended. The # is a number that increments to create a unique filename. For instance, if the program name is myprogram.cr1, the first time it is compiled the default name will be myprogram_CC1.cr1. If myprogram_CC1.cr1 exists, the program will be named myprogram_CC2.cr1.

## 9.3.4 Templates

The use of templates can be a powerful way to quickly create a set of similar datalogger programs. All or part of a program can be saved so that it can be used when creating new programs. These files are called templates. The **Template** menu provides access to create and use templates.

**Save as Template** – Saves the comments and instructions in the active file as a template. To save part of a program as a template, copy the selected part to a new program file and then **Save as Template**.

**Save as Default Template** – Saves the comments and instructions in the active file as a template that will be used each time **File | New** is selected for that type of datalogger.

**Delete** – When selected, a list of all dataloggers is displayed. Select a datalogger to open a dialog box containing a list of saved templates. A template can then be highlighted and deleted from disk.

**(Datalogger Types)** – When a datalogger type is selected, a list of all templates is displayed.

**NOTE** Template files are associated with a specific datalogger type. For example, templates for a CR5000 cannot be used for CR9000X programming and vice versa. Each datalogger has its own set of instructions that may be different than the other.

## 9.3.5 Program Navigation using BookMarks and GoTo

Bookmarks are lines of code in the program that the user marks, which can be quickly navigated to using the **Next**, **Previous**, and **Browse Bookmark** functions. Buttons for the bookmark function are available on the toolbar or in the **GoTo | Bookmarks** menu. Selecting the **Toggle Bookmark** option will add a bookmark to a line. Selecting it a second time will remove the bookmark. When a line is bookmarked, the entire line will be highlighted with a color (the color can be changed using the **View | Editor Preferences** menu item). You can then navigate from bookmark to bookmark by selecting **Previous** or **Next**. All bookmarks can be removed from the program by selecting **Clear Bookmarks**. Bookmarks are persistent when you close a program (i.e., they are saved and will exist the next time the program is opened).

All programs have certain common instructions, such as the declaration of variables, data table definitions, the *BeginProg/EndProg* statements and *Scan/NextScan*. The **Goto** function is used to move the cursor to the next occurrence of a common instruction in the program (**GoTo | Navigation** or choose the **GoTo** button from the toolbar). In addition, you can move to a particular line number in the program by selecting **GoTo | Go To Line**.

## 9.3.6 CRBasic Editor File Menu

Many of the functions available from the CRBasic Editor Toolbar are found in this menu. They have been discussed previously. Other options include:

**Open as Read-Only** – Opens a copy of a program file in read-only view. In this mode, the file cannot be edited. However, you can copy text from a read-only file into another file. Read-only allows the same program to be opened twice – once in regular view and once in read-only view. This allows the user to examine multiple areas of a very large program at the same time.

**Save and Encrypt** – Encrypts the active file. Encrypted files can be compiled in the datalogger but cannot be read by a user. (Refer to FileEncrypt in the CRBasic Editor's online help for dataloggers that support file encryption.)

## 9.3.7 CRBasic Editor Edit Menu

This menu item allows you to edit and manipulate the text currently being displayed in the Editor. Standard text editing functions such as **Cut**, **Copy**, **Paste, Delete**, **Select All**, **Undo** and **Redo** are found in this menu.

### 9.3.7.1 Other Options

**Create Compressed File** – Creates a new file with a *_str* extension. All user comments and line spacing in the program are removed from the file. Removing comments and spaces can significantly reduce the file size in larger programs.

**Rebuild Indentation** – Reworks the indentation of loops, *If/Then/Else* statements and other logic nesting, and removes blank lines based on the Vertical Spacing rules (**Options | Editor Preferences, Vertical Spacing** tab).

**Save As CRB** – Saves highlighted text to a file with a *.CRB extension. This file is referred to as a library file. The file can then be reused by inserting it into another CRBasic program.

**Insert File** – Inserts a library file (*.CRB) into the current program at the location of the cursor.

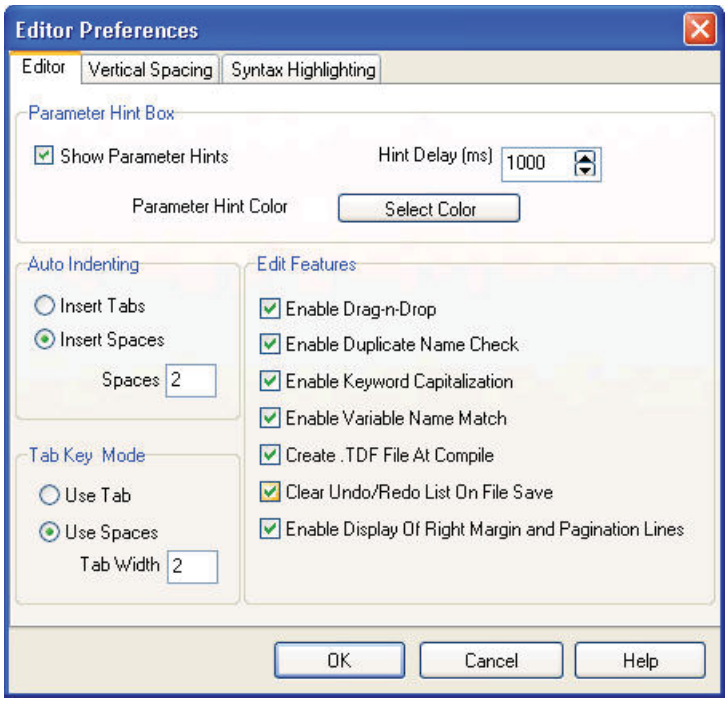
## 9.3.8 CRBasic Editor View Menu

This menu item allows you to specify the files used in the CRBasic Editor and customize its look and syntax highlighting.

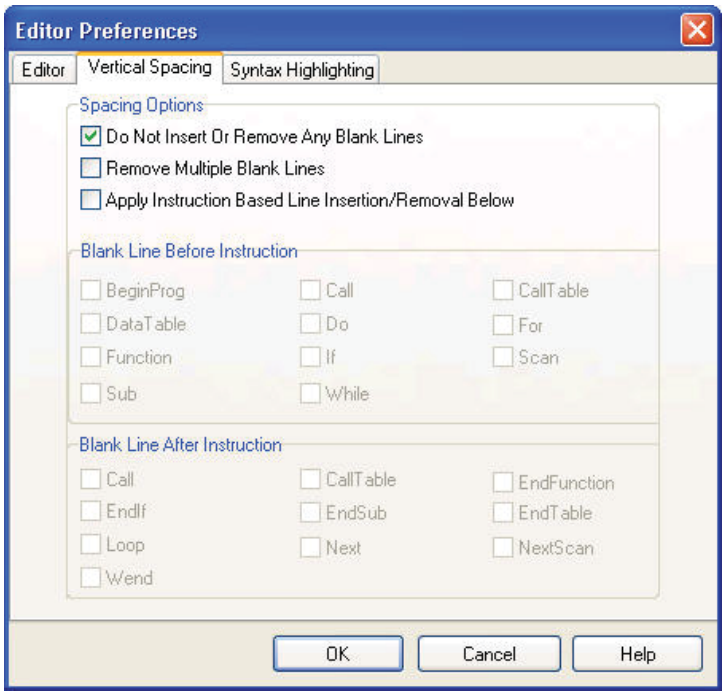
### 9.3.8.1 Editor Preferences

This option sets up the appearance options for the text instructions and the behavior of pop-up hints.

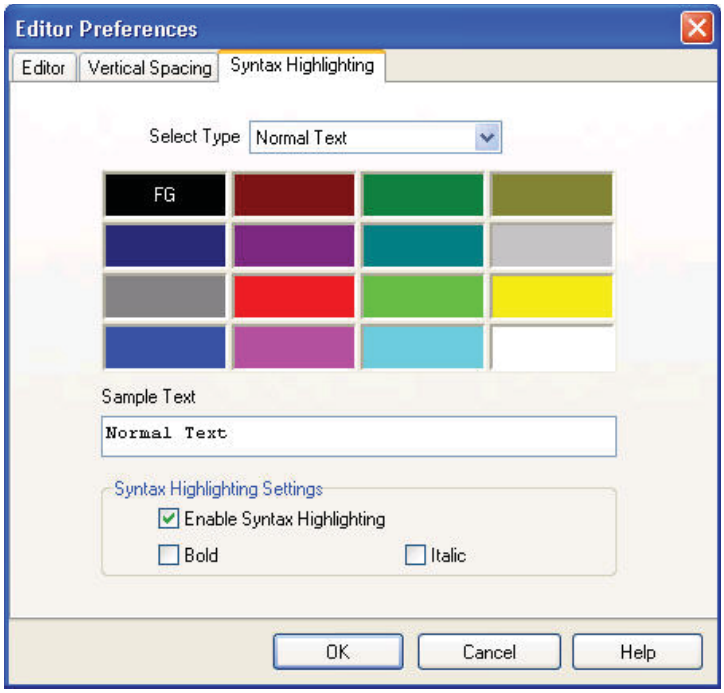
The **Editor** tab allows the user to toggle on or off the pop-up hints for parameters in instructions, set the amount of time the cursor must hover over the instruction before the pop-up hint appears, and the background color of the pop-up hint. This is also used to choose whether CRBasic automatic instruction indenting indents using tabs or spaces, and set the number of spaces if that option is chosen. Other options relating to the use of the tab key, file compression, capitalization and name checking are also available. Press the Help button for more information.




The **Vertical Spacing** tab is used to set up the rules for the CRBasic Editor's **Rebuild Indentation** function (Edit | Rebuild Indentation). You can control whether blank lines are inserted before or after certain instructions, and how the CRBasic Editor will process multiple blank lines in the program. If **Do Not Insert or Remove Any Blank Lines** is selected, all other fields on this tab will be disabled. If either of the other two line options is chosen, the remaining fields will be available for the user to customize as desired.



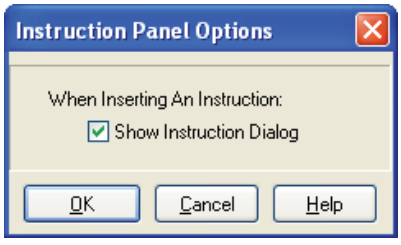
The **Syntax Highlighting** tab sets up the appearance of different text elements in the program using different font styles and colors. You can customize the appearance of the text by giving normal text, keywords, comments, operators, numbers, strings, and parentheses each a different font style and color to make the program easier to read and edit. Text colors and styles can be disabled by clearing the **Enable Syntax Highlighting** check box.




Note that if special formatting (font style, color) is assigned to **Matched Parentheses**, when your cursor is on an opening or closing parenthesis it will be highlighted with the formatting, and the "other half" of that parenthesis will also be highlighted. When your cursor moves off the parenthesis, the formatting will return to normal text.

### 9.3.8.2 Instruction Panel Preferences

This option determines whether or not the instruction dialog box will be displayed when the user inserts an instruction.




### 9.3.8.3 Other Options

**Font** – Displays a font selection dialog to select the font typeface and size for the text in the CRBasic Editor. Font style and color are set under **Editor Preferences.**

**Background Color** – Displays a color selection dialog to set the color of the CRBasic program window.

**Wrap Text When Printing** – When this option is selected, long lines that extend past the right margin will be wrapped to the next line. This option affects printing, as well as the **Print Preview** mode. A check mark will appear next to the option in the menu when it is selected.

**Display Last Window Used** – When this option is enabled, the program that was active in the CRBasic Editor when it was last closed will be visible when the Editor is reopened. If this option is disabled, no program will be loaded when the Editor is opened. A check mark will appear next to the option in the menu when it is selected.

**Close Message Window** – After you have pre-compiled your program with the **Compile | Compile** menu item, or using the toolbar, a message window opens up at the bottom of the CRBasic Editor main screen. This option will close down that message window.

**View Instruction Panel** – Select this option to View or Hide the instruction panel which displays a list of available instructions which can be used in your datalogger program based on the pre-defined instruction filter selected with the drop-down selection box.

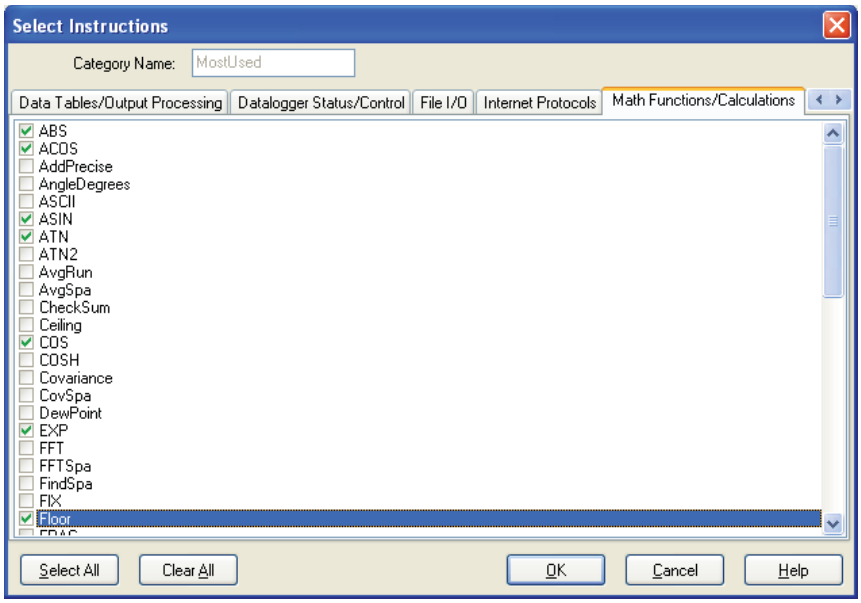
## 9.3.9 CRBasic Editor Tools Menu

This menu item allows you to use special tools associated with the operation of the editor.

### 9.3.9.1 Edit Instruction Categories

**Edit Instruction Categories** allows the user to create one or more custom list of instructions. If a category of instructions is selected from the Instructions Panel, the entire list of instructions in the Editor will be filtered to show only those instructions in the selected category. (Note: The default categories cannot be edited or deleted.)

To create a new list, first select the **Add New Category** button and provide a name for the user-created category. Next, ensure the category name is selected and click the **Edit Category** button to bring up the **Select Instructions** dialog (shown below). Instructions that should be included in the new list are indicated by a check in the box to the left of the instruction name. This feature allows the user to display a filtered instruction list containing only those instructions most often used. Press **OK** to save the list.




### 9.3.9.2 Constant Customization

The Constant Customization feature allows you to define values for one or more constants in a program prior to performing a conditional compile (**Compile | Conditional Compile and Save** menu item). The constants can be set up with an edit box, a spin box field for selecting/entering a value, or with a list box. A step increase/decrease can be defined for the spin box, as well as maximum and minimum values.

To set up Constant Customization, place the cursor on a blank line within the CRBasic Editor and choose **Tools | Set Up Constants Customization Section**. This will insert two comments into the program:

```
'Start of Constants Customization Section

'End of Constants Customization Section
```

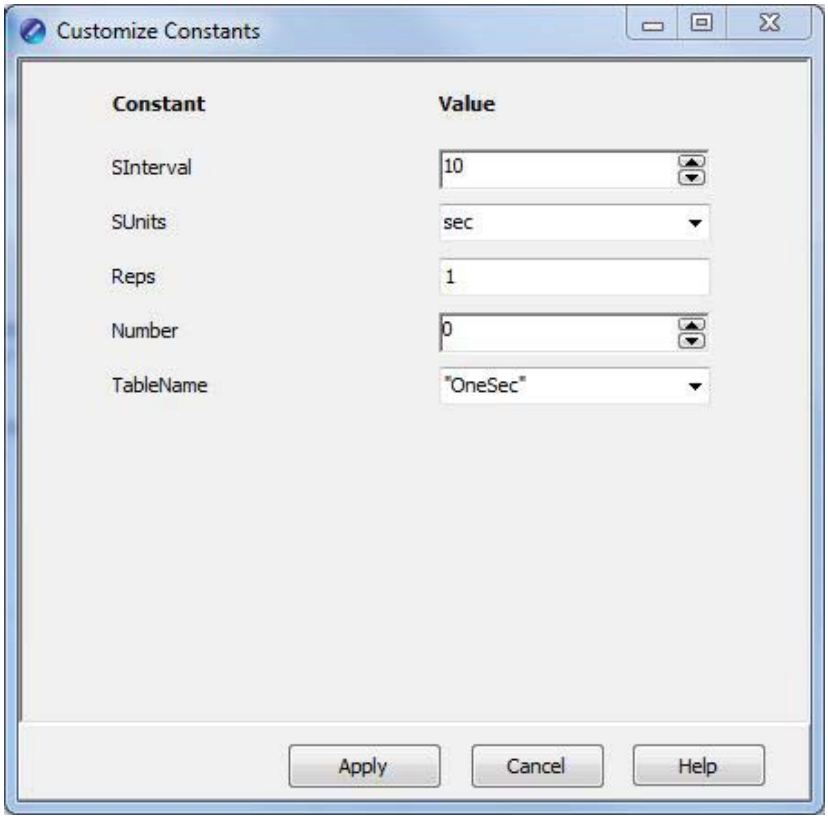
Within these two comments, define the constants. Following each constant, use the keywords noted below formatted as a comment, to set up edit boxes, spin boxes, or list boxes for the constant values. The fields are edit boxes by default. If a maximum/minimum are defined for a constant, the field will be a spin box. If a discrete list is defined for the constant, the field will be a list box.

| | |
|---|---|
| Constant *Name*=(value) | Sets a default value for a constant |
| Min=(value) | Sets a minimum value for a spin box control |
| Max=(value) | Sets a maximum value for a spin box control |
| Inc=(value) | The number of steps for a value each time the up or down control on the spin box is selected |
| Value=(value) | Defines a pick list value |

The Constant Customization syntax may be best understood by looking at an example. Consider the following program code:

```
'Start of Constants Customization Section
Const SInterval=10
‘Min=5
‘Max=60
‘Inc=5

Const SUnits = sec
'value=sec
'value=min

Const Reps=1

Const Number=0
'Min=-100
'Max=100

Const TableName="OneSec"
'value="OneMin"
'value="OneHour"
'value="OneDay"

'End of Constants Customization Section
```

This code will create the following constant customization dialog box:



The constant SInterval is defined with a default value of 10, a maximum of 60 and a minimum of 5, with a step of 5 each time the up or down control is selected.

The constant SUnits has a list box with sec and min; sec is the default.

The constant Reps is defined with a default value of 1. It is an edit box, into which any value can be entered.

The constant Number is defined with a default value of 0, a minimum of -100 and a maximum of 100. The value will increase by 1 each time the up or down control is selected.

The constant TableName is defined with a list box of "OneSec", "OneMin", "OneHour", and "OneDay"; the default value is "OneSec".

Before compiling the program, open the Customize Constants dialog box, select the constant values you want to compile into the program, and then perform the Conditional Compile and Save.

#### 9.3.9.3 Other Options

**Associate Files** – This option is used to set up file associations within the Windows operating system, so that if a program file is double-clicked while in Windows Explorer, that file with be opened in the CRBasic Editor.

Check one or more boxes for file extension(s) you want to associate and press the **Associate Files** button.

**Show Keyboard Shortcuts** – This option displays a list of the functions of the CRBasic Editor which are accessible via the keyboard. The list can be copied to the clipboard for printing or other uses.

**Show Tables** – This option displays details about the output tables and the items they store as they are defined in the current CRBasic program. The list can be copied to the clipboard for printing or other uses.

**Insert Symbol** – Opens a dialog box that lets you insert Unicode symbols into your CRBasic program for use in strings and units declarations.

**Set DLD Extension** – This option selects which datalogger's pre-compiler will be used when performing a pre-compile check on a DLD program which uses conditional compile statements. A CRBasic program must be named with the DLD extension for this item to be active.

**Open Display Settings File** – Opens a previously saved display setting file.

**Save Display Settings File** – The look and feel of the CRBasic Editor can be changed from the default. The Font and Background can be changed, as well as the syntax highlighting. These changes can be saved to a file (with an ini extension) using the Save Display Settings File menu item. The file can be reloaded on the same or different computer running CRBasic using the Open Display Settings File.

### 9.3.10 Available Help Information

Pressing the **Help** button of the Parameter dialog box will bring up a detailed help topic for the instruction being edited. Pressing *F1* when your cursor is within a parameter field will bring up help only on that parameter. Some fields also have text in the **Comments** column, which provides a short description of the option that has been selected for the parameter.

## 9.4 CRBasic Programming

CRBasic is a programming language that has some similarities to a structured BASIC. There are special instructions for making measurements and for creating tables of output data. The results of all measurements are assigned variables (given names). Mathematical operations are written out much as they would be algebraically. This section provides a summary of a program, its syntax, structure, and sequence. Refer to the datalogger user's manual or the on-line help for detailed information on program instructions.

### 9.4.1 Programming Sequence

The structure of a datalogger program requires that variables, data tables, and subroutines be declared before they can be used. The best way to do this is to put all the variable declarations and output table definitions at the beginning, followed by the subroutines, and then the program. Below is the typical layout of a program. Note that the online help has example code for each instruction to demonstrate the use of the instruction in a program.

| | |
|---|---|
| **Declarations** | *Make a list of what to measure and calculate.* |
| **Declare constants** | *Within this list, include the fixed constants used,* |
| **Declare Public variables** | *Indicate the values that the user is able to view while the program is running,* |
| **Dimension variables** | *the number of each measurement that will be made,* |
| **Define Aliases** | *and specific names for any of the measurements.* |
| **Define data tables** | *Describe, in detail, tables of data that will be saved from the experiment.* |
| **Process/store trigger** | *Set when the data should be stored. Are they stored when some condition is met? Are data stored on a fixed interval? Are they stored on a fixed interval only while some condition is met?* |
| **Table size** | *Set the size of the table in RAM.* |
| **Other on-line storage devices** | *Should the data also be sent to the external storage?* |
| **Processing of Data** | *What data are to be output (current value, average, maximum, minimum, etc.).* |

| | |
|---|---|
| **Define Subroutines** | *If there is a process or series of calculations that needs to be repeated several times in the program, it can be packaged in a subroutine and called when needed rather than repeating all the code each time.* |
| **Program** | *The program section defines the action of datalogging.* |
| **Set scan interval** | *The scan sets the interval for a series of measurements.* |
| **Measurements** | *Enter the measurements to make.* |
| **Processing** | *Enter any additional processing with the measurements.* |
| **Call Data Table(s)** | *The Data Table must be called to process output data.* |
| **Initiate controls** | *Check measurements and Initiate controls if necessary.* |
| **NextScan** | *Loop back (and wait if necessary) for the next scan.* |
| **End Program** | |

## 9.4.2 Program Declarations

Variables must be declared before they can be used in the program. Variables declared as *Public* can be viewed in display software. Variables declared using *Dim* cannot be viewed. Variables assigned to a fixed value are used as constants.

For example, in a CRBasic program there may be multiple temperature (or other) sensors that are wired to sequential channels. Rather than insert multiple instructions and several variables, a *variable array* with one name and many elements may be used. A thermocouple temperature might be called TCTemp. With an array of 20 elements the names of the individual temperatures are TCTemp(1), TCTemp(2), TCTemp(3), ... TCTemp(20). The array notation allows compact code to perform operations on all the variables. For example, to convert ten temperatures in a variable array from C to F:

```
For I=1 to 10
        TCTemp(I)=TCTemp(I)*1.8+32
Next I
```

Aliases can also be created that will allow an element of an array or another data result to be referred to by a different name. To continue the example above, TCTemp(3) could be renamed using the following syntax:

```
Alias TCTemp(3) = AirTemp
```

In the display software, the more descriptive alias, AirTemp, would be used for the cell name.

## 9.4.3 Mathematical Expressions

Mathematical expressions can be entered algebraically into program code to perform processing on measurements, to be used for logical evaluation, or to be used in place of some parameters.

As an example of **Measurement Processing,** to convert a thermocouple measurement from degrees Celsius to degrees Fahrenheit, you could use the following expression:

TCTempF=TCTemp(1)*1.8+32

Logical Evaluation expressions could be used to determine the flow of a program:

```
If TCTemp(1) > 100 Then
Call Subroutine1
Else
'enter code for main program
End If
```

Many parameters will allow the entry of expressions. In the following example, the DataTable will be triggered, and therefore data stored, if TCTemp(1)>100.

DataTable(TempTable, TCTemp(1)>100, 5000)

## 9.4.4 Measurement and Output Processing Instructions

Measurement instructions are procedures that set up the measurement hardware to make a measurement and place the results in a variable or a variable array. Output processing instructions are procedures that store the results of measurements or calculated values. Output processing includes averaging, saving maximum or minimum, standard deviation, FFT, etc.

The instructions for making measurements and outputting data are not found in a standard basic language. The instructions Campbell Scientific has created for these operations are in the form of procedures. The procedure has a keyword name and a series of parameters that contain the information needed to complete the procedure. For example, the instruction for measuring the temperature of the CR5000 input panel is:

PanelTemp *(Dest, Integ)*

PanelTemp is the keyword name of the instruction. The two parameters associated with PanelTemp are: *Dest*ination, the name of the variable in which to put the temperature; and *Integ*ration, the length of time to integrate the measurement. To place the panel temperature in the variable RefTemp (using a 250 microsecond measurement integration time) the code is:

PanelTemp(RefTemp, 250)

## 9.4.5 Line Continuation

Line continuation allows an instruction or logical line to span one or more physical lines. This allows you to break up long lines of code into more readable “chunks”. Line continuation is indicated by one white space character that immediately precedes a single underscore character as the last character of a line of text. Following is an example of line continuation:

```
Public Temp, RH, WindSp, WindDir, _
BatteryV, IntRH, IntTemp, RainTot, _
RainInt, Solar
```

## 9.4.6 Inserting Comments Into Program

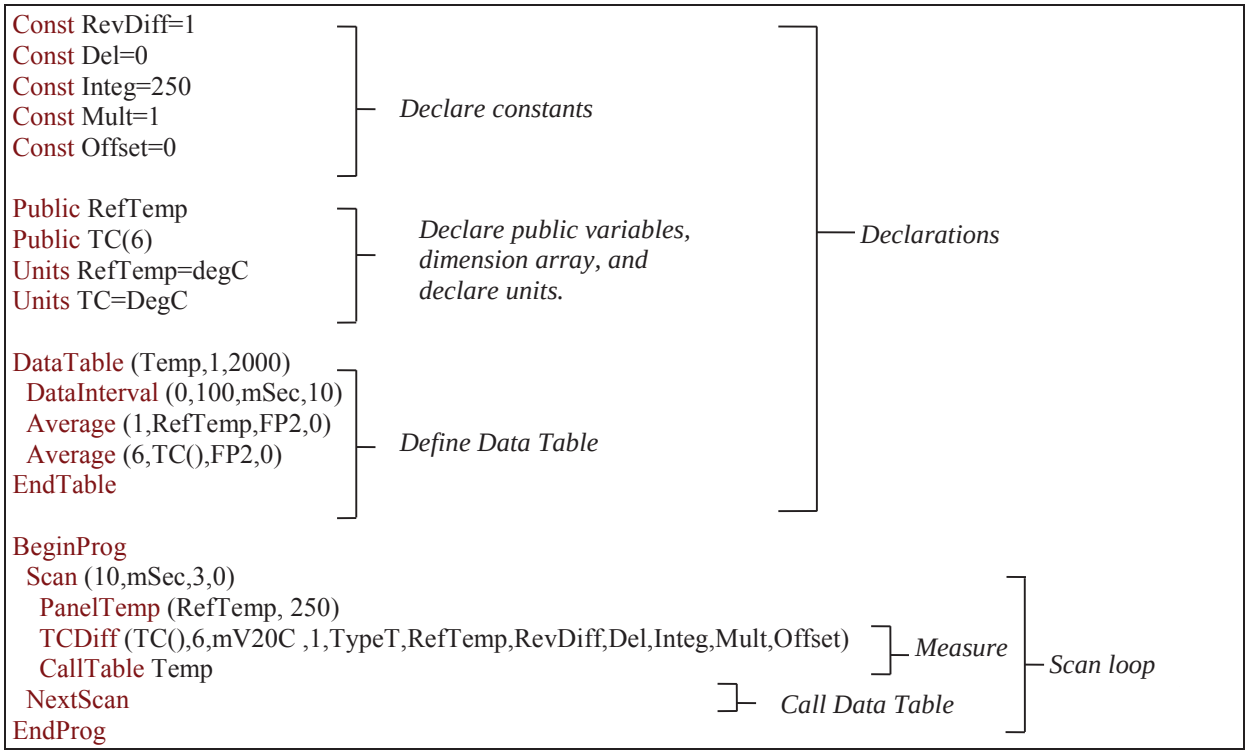
It is often useful to provide comments in your datalogger program so that when you review the program at a later date, you will know what each section of code does. Comments can be inserted into the program by preceding the text with a single quote. When the program is compiled, the datalogger compiler will ignore any text that is preceded by a single quote. A comment can be placed at the beginning of a line or it can be placed after program code. If Syntax Highlighting is enabled (Options | Editor Preferences | Syntax Highlighting), commented text will appear formatted differently than other lines of code.

```
'CR5000

‘The following program is used to measure
'4 thermocouples

'VARIABLE DECLARATION
Dim TCTemp(4)                          'Dimension TC measurement variable
Alias TCTemp(1)=EngineCoolantT         'Rename variables
Alias TCTemp(2)=BrakeFluidT
Alias TCTemp(3)=ManifoldT
Alias TCTemp(4)=CabinT
```

In the sample code above, the datalogger compiler will ignore the commented text.

## 9.4.7 Example Program

The following program will serve as a programming example in this section to illustrate the concepts and program structure. This is a program for a CR5000 datalogger. Note that other dataloggers may have slightly different parameters for some instructions.




## 9.4.8 Data Tables

Data storage follows a fixed structure in the datalogger in order to optimize the time and space required. Data are stored in tables such as:

| TOA5<br>TIMESTAMP<br>TS | StnName<br>RECORD<br>RN | Temp<br>RefTemp_Avg<br>degC<br>Avg | TC_Avg(1)<br>DegC<br>Avg | TC_Avg(2)<br>degC<br>Avg | TC_Avg(3)<br>degC<br>Avg | TC_Avg(4)<br>degC<br>Avg | TC_Avg(5)<br>degC<br>Avg | TC_Avg(6)<br>degC<br>Avg |
|---|---|---|---|---|---|---|---|---|
| 1995-02-16 15:15:04.61 | 278822 | 31.08 | 24.23 | 25.12 | 26.8 | 24.14 | 24.47 | 23.76 |
| 1995-02-16 15:15:04.62 | 278823 | 31.07 | 24.23 | 25.13 | 26.82 | 24.15 | 24.45 | 23.8 |
| 1995-02-16 15:15:04.63 | 278824 | 31.07 | 24.2 | 25.09 | 26.8 | 24.11 | 24.45 | 23.75 |
| 1995-02-16 15:15:04.64 | 278825 | 31.07 | 24.21 | 25.1 | 26.77 | 24.13 | 24.39 | 23.76 |

The user's program determines the values that are output and their sequence. The datalogger automatically assigns names to each field in the data table. In the above table, TIMESTAMP, RECORD, RefTemp_Avg, and TC_Avg(1) are fieldnames. The fieldnames are a combination of the variable name (or alias if one exists) and a three letter mnemonic for the processing instruction that outputs the data. Alternatively, the FieldNames instruction can be used to override the default names.

The data table header may also have a row that lists units for the output values. The units must be declared for the datalogger to fill this row out (e.g., Units

RefTemp = degC). The units are strictly for the user's documentation; the datalogger makes no checks on their accuracy.

The above table is the result of the data table description in the example program:

```
DataTable (Temp,1,2000)
        DataInterval(0,10,msec,10)
        Average(1,RefTemp,fp2,0)
        Average(6,TC(1),fp2,0)
EndTable
```

All data table descriptions begin with **DataTable** and end with **EndTable**. Within the description are instructions that tell what to output and the conditions under which output occurs.

DataTable(*Name, Trigger, Size*)
DataTable (Temp,1,2000)

The DataTable instruction has three parameters: a user specified name for the table, a trigger condition, and the size to make the table in RAM. The trigger condition may be a variable, expression, or constant. The trigger is true if it is not equal to 0. Data are output if the trigger is true and there are no other conditions to be met. No output occurs if the trigger is false (=0). The size is the number of records to store in the table. You can specify a fixed number, or enter –1 to have the datalogger auto allocate the number of records. The example creates a table name Temp, outputs any time other conditions are met, and retains 2000 records in RAM.

DataInterval(*TintoInt, Interval, Units, Lapses*)
DataInterval(0,10,msec,10)

DataInterval is an instruction that modifies the conditions under which data are stored. The four parameters are the time into the interval, the interval on which data are stored, the units for time, and the number of lapses or gaps in the interval to track. The example outputs at 0 time into (on) the interval relative to real time, the interval is 10 milliseconds, and the table will keep track of 10 lapses. The DataInterval instruction reduces the memory required for the data table because the time of each record can be calculated from the interval and the time of the most recent record stored. The DataInterval instruction for the CR200 does not have lapses.

**NOTE**

Event driven tables should have a fixed size rather than allowing them to be allocated automatically. Event driven tables that are automatically allocated are assumed to have one record stored per second in calculating the length. Since the datalogger tries to make the tables fill up at the same time, these event driven tables will take up most of the memory leaving very little for the other, longer interval, automatically allocated data tables.

The output processing instructions included in a data table declaration determine the values output in the table. The table must be called by the program using the CallTable (*Tablename*) instruction in order for the output processing to take. That is, each time a new measurement is made, the data table is called. When the table is called, the output processing instructions within the table process the current inputs. If the trigger conditions for the data table are true, the processed values are output to the data table. In the example below, several averages are output.

Average(*Reps, Source, DataType, DisableVar*)
Average(1,RefTemp,fp2,0)
Average(6,TC(1),fp2,0)

Average is an output processing instruction that will output the average of a variable over the output interval. The parameters are repetitions (the number of elements in an array to calculate averages for), the Source variable or array to average, the data format to store the result in (TABLE 9-1), and a disable variable that allows excluding readings from the average if conditions are not met. A reading will not be included in the average if the disable variable is not equal to 0; the example has 0 entered for the disable variable so all readings are included in the average.

**TABLE 9-1. Formats for Output Data**

| Code | Data Format | Size | Range | Resolution |
|---|---|---|---|---|
| FP2 | Campbell Scientific floating point | 2 bytes | ±7999 | 13 bits (about 4 digits) |
| IEEE4 | IEEE four byte floating point | 4 bytes | 1.8 E -38 to 1.7 E 38 | 24 bits (about 7 digits) |
| LONG | 4 byte Signed Integer | 4 bytes | -2,147,483,648 to +2,147,483,647 | 1 bit (1) |

## 9.4.9 The Scan — Measurement Timing and Processing

Once the measurements and calculations have been listed and the output tables defined, the program itself may be relatively short. The executable program begins with BeginProg and ends with EndProg. The measurements, processing, and calls to output tables bracketed by the Scan and NextScan instructions determine the sequence and timing of the datalogging.

```
BeginProg
Scan(1,MSEC,3,0)
        PanelTemp(RefTemp, 250)
        TCDiff(TC(),6,mV50,4,1,TypeT,RefTemp,RevDiff,Del,Integ,Mult,Offset)
        CallTable Temp
NextScan
EndProg
```

The Scan instruction determines how frequently the measurements within the scan are made:

Scan*(Interval, Units, BufferOption, Count)*
Scan(1,MSEC,3,0)

The Scan instruction has four parameters (the CR200 datalogger's Scan instruction has only two). The Interval is the time between scans. Units are the time units for the interval. The BufferSize is the size (in the number of scans) of a buffer in RAM that holds the raw results of measurements. Using a buffer allows the processing in the scan to at times lag behind the measurements without affecting the measurement timing (see the scan instruction in the CR5000 help for more details). Count is the number of scans to make before proceeding to the instruction following NextScan. A count of 0 means to continue looping forever (or until ExitScan). In the example the scan is 1 millisecond, three scans are buffered, and the measurements and output continue indefinitely.

## 9.4.10 Numerical Entries

In addition to entering regular base 10 numbers there are 3 additional ways to represent numbers in a program: scientific notation, binary, and hexadecimal (TABLE 9-2).

**TABLE 9-2. Formats for Entering Numbers in CRBasic**

| Format | Example | Value |
|---|---|---|
| Standard | 6.832 | 6.832 |
| Scientific notation | 5.67E-8 | $5.67X10^{-8}$ |
| Binary: | &B1101 | 13 |
| Hexadecimal | &HFF | 255 |

The binary format makes it easy to visualize operations where the ones and zeros translate into specific commands. For example, a block of ports can be set with a number, the binary form of which represents the status of the ports (1= high, 0=low). To set ports 1, 3, 4, and 6 high and 2, 5, 7, and 8 low; the number is &B00101101. The least significant bit is on the right and represents port 1. This is much easier to visualize than entering 72, the decimal equivalent.

## 9.4.11 Logical Expression Evaluation

### 9.4.11.1 What is True?

Several different words are used to describe a condition or the result of a test. The expression, X>5, is either **true** or **false**. However, when describing the state of a port or flag, **on** or **off** or **high** or **low** is more intuitive. In CRBasic there are a number of conditional tests or instruction parameters, the result of which may be described with one of the words in TABLE 9-3. The datalogger evaluates the test or parameter as a number; 0 is false, not equal to 0 is true.

| TABLE 9-3. Synonyms for True and False | | |
|---|---|---|
| Predefined Constant | True (-1) | False (0) |
| Synonym | High | Low |
| Synonym | On | Off |
| Synonym | Yes | No |
| Synonym | Trigger | Do Not Trigger |
| Number | ≠0 | 0 |
| Digital port | 5 Volts | 0 Volts |

### 9.4.11.2 Expression Evaluation

Conditional tests require the datalogger to evaluate an expression and take one path if the expression is true and another if the expression is false. For example:
**If X>=5 then Y=0**
will set the variable Y to 0 if X is greater than or equal to 5.

The datalogger will also evaluate multiple expressions linked with **and** or **or**. For example:
**If X>=5 and Z=2 then Y=0**
will set Y=0 only if both X>=5 and Z=2 are true.
**If X>=5 or Z=2 then Y=0**
will set Y=0 if either X>=5 or Z=2 is true (see And and Or in the help). A condition can include multiple **and** and **or** links.

### 9.4.11.3 Numeric Results of Expression Evaluation

The datalogger's expression evaluator evaluates an expression and returns a number. A conditional statement uses the number to decide which way to branch. The conditional statement is false if the number is 0 and true if the number is not 0. For example:
**If 6 then Y=0,**
is always true, Y will be set to 0 any time the conditional statement is executed.
**If 0 then Y=0**
is always false, Y will never be set to 0 by this conditional statement.

The expression evaluator evaluates the expression, X>=5, and returns -1, if the expression is true, and 0, if the expression is false.
**W=(X>Y)**
will set W equal to -1 if X>Y or will set W equal to 0 if X<=Y.

The datalogger 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.

## 9.4.12 Flags

Any variable can be used as a flag as far as logical tests in CRBasic are concerned. If the value of the variable is non-zero the flag is high. If the value of the variable is 0 the flag is low. LoggerNet, PC400, or RTDAQ looks for the variable array with the name **Flag** when the option to display flag status is selected from the Connect Screen. If a Flag array is found, as many elements of that array which can fit will be displayed in the Port and Flags dialog box.

## 9.4.13 Parameter Types

Instruction parameters allow different types of inputs. These types are listed below and specifically identified in the description of the parameter in the following sections or in CRBasic help.

Constant
Variable
Variable or Array
Constant, Variable, or Expression
Constant, Variable, Array, or Expression
Name
Name or list of Names
Variable, or Expression
Variable, Array, or Expression

TABLE 9-4 lists the maximum length and allowed characters for the names for Variables, Arrays, Constants, etc.

**TABLE 9-4. Rules for Names**

| Name for | Maximum Length (number of characters) | Allowed characters |
|---|---|---|
| Variable or Array | 39 (17 ) | Letters A-Z, upper or lower case, underscore " _ ", and numbers 0-12. The name must start with a letter. CRBasic is not case sensitive. |
| Constant | 39 (16) | |
| Alias | 39 (17 ) | |
| Data Table Name | 20 (8) | |
| Field name | 39 (16) | |

Values in parentheses refer to the CR5000, CR9000 and CR9000X dataloggers.

### 9.4.13.1 Expressions in Parameters

Many parameters allow the entry of expressions. If an expression is a comparison, it will return -1 if the comparison is true and 0 if it is false. An example of the use of this is in the DataTable instruction where the trigger condition can be entered as an expression. Suppose the variable TC(1) is a thermocouple temperature:

*DataTable(Name, TrigVar, Size)*
DataTable(Temp, TC(1)>100, 5000)

Entering the trigger as the expression, TC(1)>100, will cause the trigger to be true and data to be stored whenever the temperature TC(1) is greater than 100.

### 9.4.13.2 Arrays of Multipliers and Offsets for Sensor Calibration

If variable arrays are used for the multiplier and offset parameters in measurements that use repetitions, the instruction will automatically step through the multiplier and offset arrays as it steps through the channels. This allows a single measurement instruction to measure a series of individually calibrated sensors, applying the correct calibration to each sensor. If the multiplier and offset are not arrays, the same multiplier and offset are used for each repetition.

*VoltSE(Dest,Reps,Range,SEChan,Delay, Integ,Mult,Offset)*

```
'Calibration factors:
Mult(1)=0.123 : Offset(1)= 0.23
Mult(2)=0.115 : Offset(2)= 0.234
Mult(3)=0.114 : Offset(3)= 0.224
VoltSE(Pressure(),3,mV1000,6,1,1,100,Mult(),Offset()
```

Note that one exception to this is when the Multiplier or Offset points to an index into the array, then the instruction will not advance to the next Multiplier or Offset but use the same for each repetition. For instance in the above example, if Mult(2) and Offset(2) were used, the instruction would use 0.115 and 0.234 for the Multiplier and Offset, respectively, for each repetition. To force the instruction to advance through the Multiplier and Offset arrays while still specifying an index into the array, use the syntax **Mult(2)()** and **Offset(2)()**.

## 9.4.14 Program Access to Data Tables

Data stored in a table can be accessed from within the program. The format used is:

*Tablename.Fieldname(fieldname index,records back)*

Where *Tablename* is the name of the table in which the desired value is stored. *Fieldname* is the name of the field in the table. The fieldname is always an array even if it consists of only one variable; the *fieldname index* must always be specified. *Records back* is the number of records back in the data table from the current time (1 is the most recent record stored, 2 is the record stored prior to the most recent). For example, the expression:

Tdiff=Temp.TC_Avg(1,1)–Temp.TC_Avg(1,101)

could be used in the example program to calculate the change in the 10 ms average temperature of the first thermocouple between the most recent average and the one that occurred a second (100 x 10 ms) earlier.

In addition to accessing the data actually output in a table, there are some pseudo fields related to the data table that can be retrieved:

*Tablename*.record(1,n) = the record number of the record output n records ago.

*Tablename*.output(1,1) = 1 if data were output to the table the last time the table was called, = 0 if data were not output.

*Tablename*.timestamp(m,n) = element m of the timestamp output n records ago where:

timestamp(1,n) = microseconds since 1990
timestamp(2,n) = microseconds into the current year
timestamp(3,n) = microseconds into the current month
timestamp(4,n) = microseconds into the current day
timestamp(5,n) = microseconds into the current hour
timestamp(6,n) = microseconds into the current minute
timestamp(7,n) = microseconds into the current second

*Tablename*.eventend(1,1) is only valid for a data table using the DataEvent instruction, *Tablename*.eventend(1,1) = 1 if the last record of an event occurred the last time the table was called, = 0 if the data table did not store a record or if it is in the middle of an event.

*TableName*.EventCount = the number of data storage events that have occurred in an event driven DataTable.

*TableName*.Tablefull = 1 to indicate a fill and stop table is full or a ring-mode table has begun overwriting its oldest data, = 0 if the data table is not full/begun overwriting oldest data.

*TableName*.TableSize = the size allocation, in number of records, of the selected DataTable.

**NOTE** The values of Tablename.output(1,1) and Tablename.eventend (1,1) are only updated when the tables are called.

# Section 10. Utilities

*CardConvert is a utility that is used to quickly read and convert binary datalogger data that is retrieved from a compact flash, microSD, or PCMCIA card. The binary data can also be a file already saved to the user's PC. The binary data is converted to a TOA5 format and saved on the user's PC.*

*Transformer is an application that converts a datalogger program created in Edlog to a CRBasic program. With CSI's introduction of the CR1000, this program was specifically designed to help ease the transition from programming in Edlog to programming in CRBasic. Currently, conversion is supported for CR10X or CR510 to CR800 or CR1000 or CR23X to CR3000.*

*The Device Configuration Utility, or DevConfig, is a utility for setting up datalogger and communication peripherals for use in a datalogger network.*

*File Format Convert is an application that is used to convert data files form one format to another.*

*The utilities in PC400 are opened by going to the Window's Start menu and selecting Programs | Campbell Scientific | PC400 | Utilities | (desired utility). CardConvert can also be opened from PC400's Tools menu. DevConfig can be opened from a button on PC400's toolbar.*

## 10.1 CardConvert

CardConvert is a utility that is used to quickly read and convert binary datalogger data that is retrieved from a compact flash, microSD, or PCMCIA card. The converted data is saved on the user's PC.

### 10.1.1 Input/Output File Settings

The file settings are used to specify the directory where the binary data is stored, and the directory in which the converted file(s) should be saved.

Press the Select Card Drive button to bring up dialog box that helps you browse for the drive assigned to the card reader. Note that you can also select a directory on your hard drive in which binary data files have been copied. When a card drive or directory is selected, any files found with a *.dat extension will be displayed in the Source Filename column in CardConvert.

By default, the converted data files will be saved to the same drive or directory as the source files. To change the destination, press the Change Output Dir button. Once again you will be provided with a dialog box that helps you to browse for the desired drive or directory. When the drive or directory is selected, the path and the filename that will be used for the converted files will show up in the Destination Filename column.

The default filename for a converted file is comprised of the table name in the datalogger program, along with a prefix that reflects the file format and a *.dat extension. For instance, the default name for a table called MyData stored in TOA5 format would be TOA5_MyData.dat.

The destination directory or filename for a converted file can be changed on an individual file basis. Click on the row for the file that you wish to change. It will be highlighted. Select Options | Change Output File from the CardConvert menu, and browse for or type in a new path and/or filename. You can apply a directory path change to all files by selecting Options | Apply Directory to All.

You do not have to convert all files that are found in the selected directory. Select one or more files for conversion by selecting or clearing the checkbox beside the individual file name. If a box is checked the file will be converted; if a box is cleared the file will not be converted. To quickly select or clear all check boxes, choose Options | Check All or Clear Check All from the CardConvert menu.

The list of files displayed for a particular drive or directory can be updated by selecting Options | Rebuild File Lists from the menu. Any new files that have been stored since you last selected the drive (or since the last rebuild), will be added to the list.

Tip: Right-click within the file list to display a shortcut menu containing the items on the Options menu.

## 10.1.2 Destination File Options

The Destination File Options determine whether the data will be stored on the PC in ASCII or binary format, how filemarks will be processed, and what should happen when existing files with the same name are found.

### 10.1.2.1 File Format

The File Format is used to specify the format in which the data file should be saved. Select the desired option from the list box:

ASCII Table Data (TOA5) – Data is stored in an ASCII comma separated format. Header information for each of the data values is included, along with field names and units of measure if they are available.

Binary Table Data (TOB1) – Data is stored in a binary format. Though this format saves disk storage space, it must be converted before it is usable in other programs.

Array Compatible CSV – Data is stored in a user-defined comma separated format. This option can be used to produce output files from table data dataloggers that are similar to those created by mixed array dataloggers. When this option is chosen, the Array CSV Options button becomes available, so that you can customize the data string for the CSV file.

If an array ID is desired, select the **Include Array ID** check box and enter a value into the field. The value can range from 1 to 1023. The array ID will be the first value in the array of data.

Select the appropriate timestamp options for the type of timestamp to write to the file. Each time element will be output as a separate data value in the array and the data values will be separated by a comma. Selecting **Year** will output the year represented by four digits, YYYY (e.g., 2006). The **Day** will be represented as a Julian Day. The **Hour/Minutes** will be

represented by four digits (hhmm). When **Midnight is 2400** is selected, the timestamp will reflect midnight as the current date with 2400 for the Hour/Minutes. Otherwise, the timestamp will reflect midnight as the next day's date, with the Hours/Minutes as 0000.

The **Max and Min Timestamp Options** is used to determine the type of timestamp that will be used for Maximum and Minimum outputs that include a timestamp along with the value. You can choose to output No Timestamp, a timestamp that includes Hours/Minutes/Seconds (produces two values, hhmm and seconds), a timestamp that includes Hours/Minutes only, or a timestamp that includes Seconds only.

CSIXML – Data is stored in XML format with Campbell Scientific defined elements and attributes.

The file format is reflected in the default filename by the prefix of either TOA5, TOB1, CSV, or CSIXML added to the table name.

### 10.1.2.2 File Processing

**Use Filemarks –** CRBasic dataloggers have a FileMark instruction that allows you to store a filemark along with the data. These filemarks are ignored by the LoggerNet or PC400 data collection process. However, in CardConvert you can convert the file with the **Use Filemarks** option selected, and the file will be stored as multiple files, based upon the filemarks. Each file created will be given a numeric suffix prior to the *.dat extension. The first file is stored with a _1 at the end of the root file name (e.g., TOA5_Mytable_1.dat). The number is incremented by one with each new file saved. If a file with the same name is found, the number will be incremented to the next available number.

**Use Removemarks –** When a compact flash card is removed from a CR1000 or CR3000 datalogger, a special mark is inserted in the last record. The Removemark is similar in nature to the Filemark. In CardConvert, you can split a file into multiple files, separated at the Removemarks, by converting the file with the **Use Removemarks** option selected. As with the **Use Filemarks** option, the first file stored uses a _1 at the end of the root file name and the number is incremented by one with each new file saved.

**Use Filemarks** and **Use Removemarks** can be selected at the same time, to create a new file from the data table any time either of the marks is encountered.

**Use Time** – This option is used to store the converted data into files based on the timestamp of the data. When the **Use Time** check box is selected, the **Time Settings** button becomes available. This button opens a dialog box that is used to set a **Start Date** and **Time**, along with an **Interval**, which are used to determine the time frame for the data that goes into each file. Note that the Start Date and Time are not used to specify the actual time and date to begin processing the file; rather, they are used as a reference for the file interval. Processing always starts at the beginning of the file.

When **Use Filemarks, Use Removemarks, or Use Time** is selected, the **Create New Filenames** option is disabled. New file names will always be created.

**Convert Only New Data** – When this option is selected, only data that has been collected since CardConvert's last conversion of the specified file(s) will

be converted. The first time CardConvert is used on a file, all data will be converted. On subsequent conversions, only new data will be converted. However, if CardConvert cannot tell what data is new (i.e. if data on the card has wrapped since the last conversion), all data will be converted. This option can be used with Append to Last File to create a continuous file with no repetition of data.

#### 10.1.2.3 File Naming

**Time/Date Filenames** – When this option is selected, the date and time of the last record of data in the file will be appended to the end of the base file name. The suffix includes a four digit year, a two digit month, a two digit day of month, and a four digit hour/minute. When this option is selected, **Use Day of Year** becomes available. If this option is selected, the Julian day (day of year) will be used for the suffix instead of the year/month/day/hour/minute suffix.

**Create New Filenames –** When the **Create New Filenames** option is selected, CardConvert will add a _01 to the filename, if a file of the same name is found (e.g., TOA5_Mydata_01.dat). If a *_01.dat file is found, the file will be named with a _02 suffix. If the **Create New Filenames** check box is cleared and a file with the same name is found, you will be offered the option to Overwrite the existing file or Cancel the conversion.

The **Create New Filenames** option is disabled when the **Use Filemarks, Use Removemarks**, or **Use Time** option is enabled.

**Append to Last File** – When this option is selected, converted data will be appended to the end of the destination file. If the destination file does not exist when a conversion is done, a new file will be created. On subsequent conversions, converted data will be appended to the end of that file. If the header of the new data does not match that of the data in the destination file, an error will be generated. This option is most useful with the **Convert Only New Data** option to create a continuous file with no repetition of data.

#### 10.1.2.4 TOA5/TOB1 Format

These two options are available when the ASCII Table Data (TOA5) or the Binary Table Data (TOB1) output option is selected.

**Store Record Number –** By default, the record number for each row of data is stored in the data file. This record number can be omitted from the converted file by clearing the **Store Record Number** check box.

**Store Time Stamp** – The time stamp can be omitted from the file by clearing the **Store Time Stamp** check box.

### 10.1.3 Converting the File

Once the File and Conversion settings are selected, press the Start Conversion button. CardConvert will begin processing the file. When the file is being processed, the estimated number of records and a percentage of the conversion completed will be displayed at the bottom edge of the window. Note that the values reflect an *estimate* of the amount of data in a table. If the table is set to a fixed size, CardConvert returns a fairly close estimate. However, if the table is set to auto-allocate, CardConvert essentially returns an estimate that reflects the maximum number of records that can be stored based on card size (even if

the table is not completely full). Because of this, you may see the progress reported as something less than 100% when the conversion is complete.

If a conversion is in progress and you wish to stop it, press the Cancel Conversion button.

After file conversion is complete, summary information is provided in the field below the file list. The summary provides a listing of the new files that were created, and the total number of records converted for each table (if filemarks are being processed for a table, the number of records returned is the cumulative number of records for all files).

#### 10.1.3.1 Repairing/Converting Corrupted Files

If you attempt to convert a file and receive a message that the input file contained no data, you may want to consider using the Repair File option. You may also want to consider using the Repair File option if you think there is additional data on the card that is not being converted and included in the output file. With either case, it is possible that data on the card has become corrupted. The Repair File Option will attempt to scan the card for good frames of data and output that data to a new binary file.

In some instances, data on a card can become corrupted. Corruption can occur if the card is subjected to electrostatic discharge or if it is removed when data is being written to the card (e.g., the card is removed from the CFM100 without pressing the Card Control button to stop data storage to the card). This corruption can be at the beginning of the data file or anywhere within the stored data. Using the standard conversion option, CardConvert will stop if it encounters a corrupted frame of data because it assumes it has come to the end of the data file. If corrupted frames of data are found at the beginning of the file, CardConvert will display a message indicating that no data could be found on the card. If corrupted frames of data are found within the data file, you may get some, but not all, of the data that you expect in the converted file.

CardConvert offers a repair option, which will attempt to scan the card for good frames of data and output that data to a new binary file (the original file is unchanged). To start the repair of a file, highlight the suspected corrupt file in the list of Source Filenames and right-click to display a floating menu. Select the Repair File option from the list. The repair process will create a new TOB3 file (the default name is Repair_existingfilename), which can then be converted to an ASCII file using the standard CardConvert process.

When CardConvert comes to what it believes is the end of the data file during the repair process (the end of valid frames), it will stop and display a message. The message prompts the user either to continue searching the file for more good data frames or to stop the repair process. CardConvert displays the last time stamp for data in the repaired file. If you think there should be additional data on the card, you can continue to run the repair process. If it appears that all the data has been stored to the new file, you can stop. The option to continue processing the file allows you to recover all good data on a card with more than one corrupted frame.

Note that CardConvert can repair only TOB2 or TOB3 files. TOB1 files cannot be repaired.

**NOTE** The Repair File option should be used only if a standard conversion cannot be done.

### 10.1.4 Viewing a Converted File

Converted data files can be reviewed using the View Pro file viewing application. View Pro can be launched by pressing the View Files button. If a file is highlighted in the list of files, that file will be displayed when View Pro is opened. Otherwise you can select the file to view from View Pro's File | Open menu.

### 10.1.5 Running CardConvert from a Command Line

In order to run CardConvert from a command line without user interaction, you will first need to create a CCF file that contains the CardConvert settings to be used when running from a command line. To create the CCF file, open CardConvert and select the desired source directory (Select Card Drive), destination directory (Change Output Dir), and Destination File Options. When CardConvert is closed, it will produce a file named "lastrun.ccf" that contains the designated settings. The file will be written to the C:\Campbellsci\CardConvert directory. You should rename this file as it will be overwritten the next time that CardConvert is closed.

When running CardConvert from a command line, you can designate the CCF file using the command line option runfile. For example,

"C:\Program Files\Campbellsci\CardConvert\CardConvert.exe" runfile="C:\Campbellsci\CardConvert\myfile.ccf"

The above command line will run CardConvert using the settings contained in myfile.ccf.

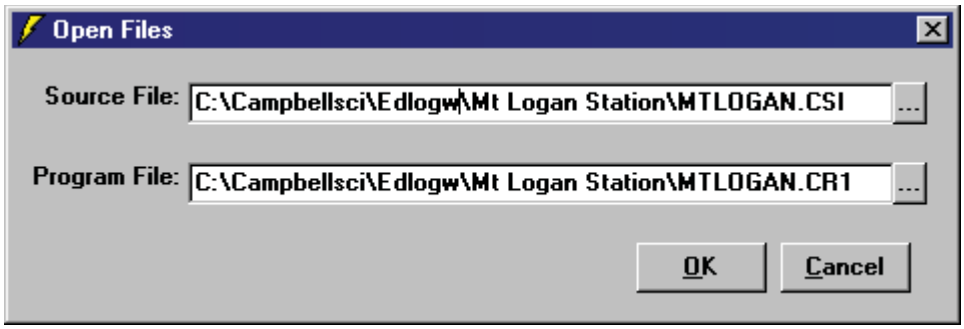
If there are no problems or questions encountered, CardConvert will start, convert the file(s), and then exit with no user interaction. However, if any problems or questions are encountered, CardConvert will display a dialog box as usual and then wait for a user response. Therefore, this command line option allows some automation of CardConvert but does not allow for completely unattended automation. To minimize the user interaction required, the Destination File Option "Create New Filenames" should be used as this prevents CardConvert from asking whether a file should be overwritten.

## 10.2 Transformer Utility

The Transformer application converts a datalogger program created in Edlog to a CRBasic program. The original file and the transformed file are displayed side-by-side so they can easily be compared. The transformer can be an excellent tool for those familiar with programming in Edlog to learn CRBasic programming.

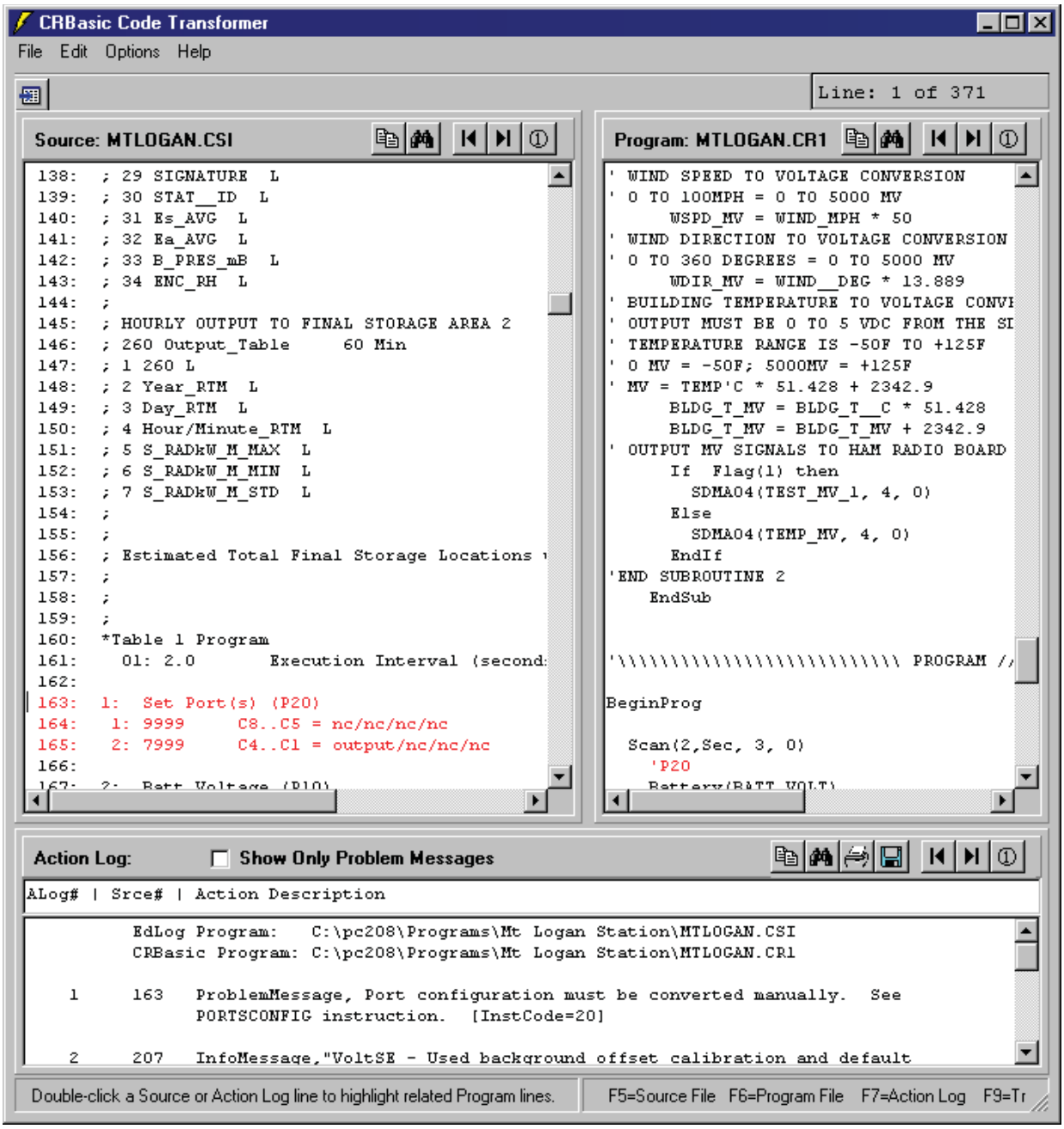
### 10.2.1 Transforming a File

When the Transformer is first opened, a dialog box is displayed in which to enter the Source File and the Program File.




The Source File is the CSI or DLD file to be converted. The Program File is the new CR* file that will be created. By default, the resulting file name for the CR800, CR1000, or CR3000 program that will be created is the name of the original program with a CR* extension. This can be changed if desired by typing in a new path and/or file name directly, or by pressing the Browse button to the right of the Program File field.

When OK is chosen, the Edlog program file — or Source file — is opened up in the left window of the Transformer. The CR800, CR1000, or CR3000 program file is created in the right window.




Comments about the conversion are shown in the Action Log (bottom portion of the window). The Action Log should be reviewed carefully; it provides useful comments and alerts you to any problems that may exist in the converted file. To view only the messages related to problems in the field, enable the Show Only Problem Messages check box.

If a comment in the Action Log is double-clicked, the associated instructions in both the Edlog program and the CRBasic program will be highlighted. If an instruction is double-clicked in the Edlog file, the associated instruction in the CRBasic program will be highlighted. In the window above, the first comment was selected, resulting in the Edlog instruction and a comment in the CRBasic program both being highlighted in red.

The transformed file cannot be edited in the Transformer. Once transformed, it can be opened in the CRBasic Editor or saved under a new file name. To open the program in the CRBasic Editor, press the CRBasic program icon at the top left of the window. To save the file under a different name, choose File | Program File | Save As.

If an Edlog file previously has been opened in the Transformer, when the file is opened a second time you will receive a message "This file, <*filename*>, already exists. If you overwrite it, the information it contains will be lost. Do you want to overwrite it?" If you choose Yes, the existing CR1 file will be overwritten. If you choose No, you will be given the opportunity to provide a new name for the file. This message can be suppressed by selecting Options | Suppress "Overwrite File" Warning from the Transformer menu. However, note that you should strongly consider keeping this message intact to avoid the possibility of overwriting a file that you transformed and then subsequently edited in the CRBasic Editor.

### 10.2.2 Controls

The following buttons are used within the Transformer to move to a different location in the file, or save or print the file.

Copies the highlighted text to the Windows clipboard. The information can then be pasted into another application.

Searches for specific text in the file.

Moves the mouse cursor to the beginning of the file.

Moves the mouse cursor to the end of the file.

Jumps to a specific line number in the file.

Prints the contents of the window (Action Log only).

Saves the contents of the window (Action Log only).

## 10.3 Device Configuration Utility

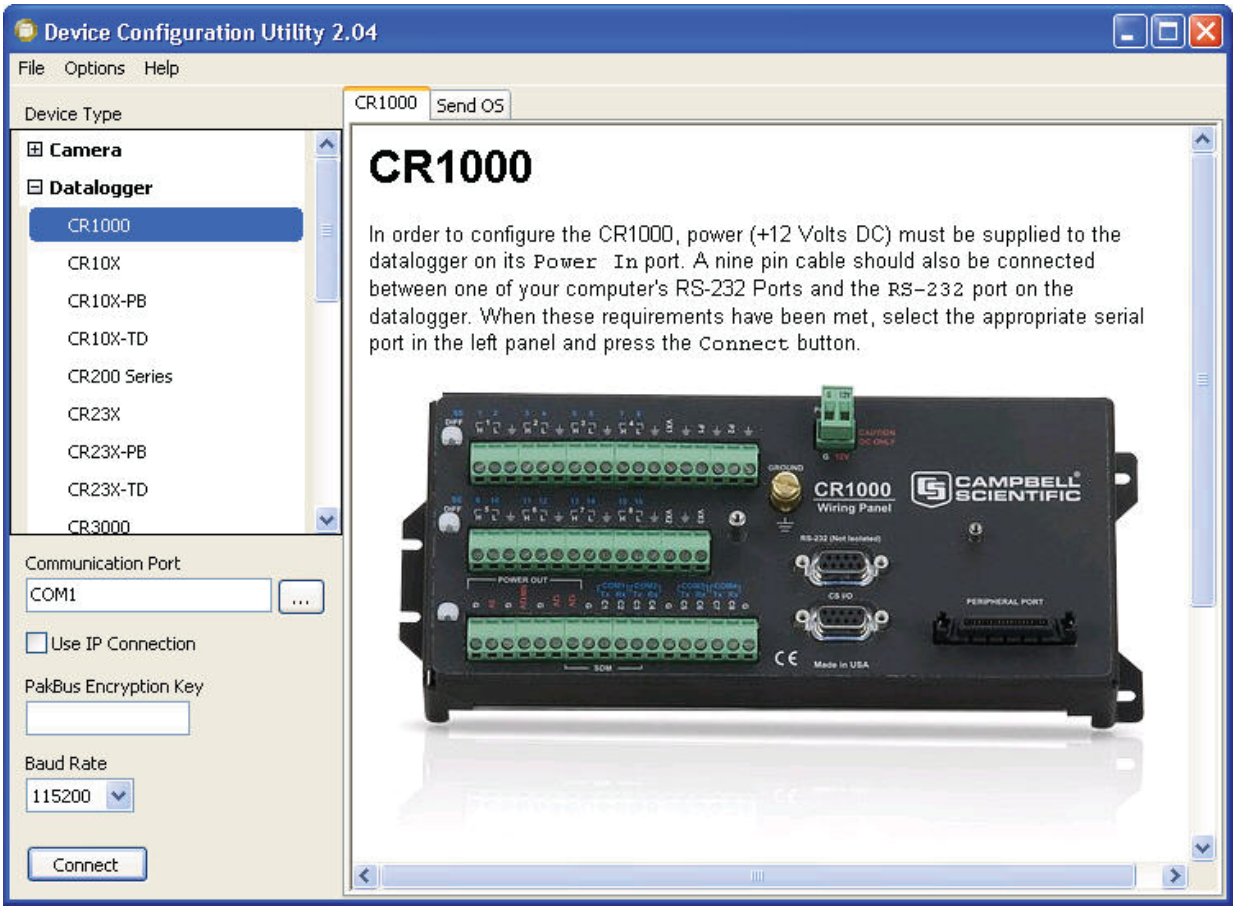
### 10.3.1 Overview

The Device Configuration Utility (DevConfig) is used to set up dataloggers and intelligent peripherals before those devices are deployed in the field and before the devices are added to networks in Campbell Scientific datalogger support software such as LoggerNet or PC400. Some key features of DevConfig include:

- To keep the process as simple as possible, DevConfig supports only serial and IP connections between the PC and devices.

- DevConfig cannot only send operating systems to supported device types, but can also set datalogger clocks and send program files to dataloggers.
- DevConfig allows you to determine operating system types and versions, which can be very useful in classic dataloggers, such as the CR10X, where the operating system version in the datalogger is not known.
- DevConfig provides a reporting facility where a summary of the current configuration of a device can be shown on the screen and printed. This configuration can also be saved to a file and used to restore the settings in the same or a replacement device.
- Some devices may not support the configuration protocol in DevConfig, but do allow configurations to be edited through the terminal emulation screen.
- Help for DevConfig is shown as prompts and explanations on its main screen. Help for the appropriate settings for a particular device can also be found in the user's manual for that device.
- Updates to DevConfig are available from Campbell Scientific's website. These may be installed over the top of older versions.

### 10.3.2 Main DevConfig Screen



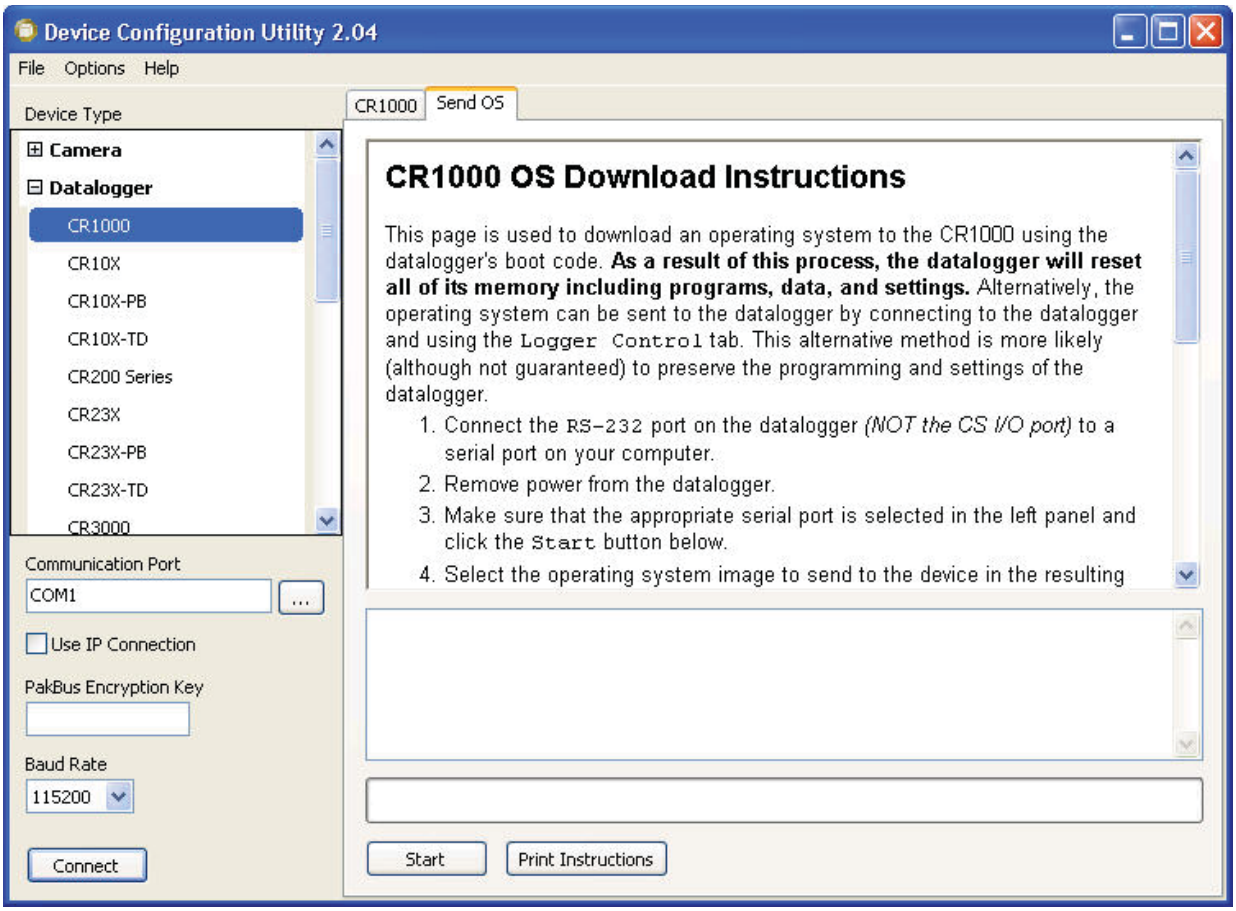
The DevConfig window is divided into two main sections: the device selection panel on the left side and tabs on the right side. After choosing a device on the left, you will then have a list of the serial ports (COM1, COM2, etc.) installed on your PC. If the device supports IP communication, the Use IP Connection check box will be enabled. In order to communicate via IP, click on the Use IP Connection check box and enter the IP address or domain name for the device in the Communication Port field. For some devices, you may be able to click on the browse button to the right of the Communication Port control to bring up a dialog that searches your local area network for any available devices. If the device has a TCP Password, you will need to enter it in the TCP Password field. You'll be offered a choice of baud rates only if the device supports more than one baud rate in its configuration protocol. The page for each device presents instructions about how to set up the device to communicate with DevConfig. Different device types will offer one or more tabs on the right.

When the user presses the Connect button, the device type, serial port, and baud rate selector controls become disabled and, if DevConfig is able to connect to the device, the button will change from "Connect" to "Disconnect". The tabs on the right side of the window will be replaced with tabs that represent the various operations that are available for that device in a connected state. These operations can vary from device to device.

Two common operations are sending a new operating system and communicating with a device in terminal emulation mode. These two operations are covered below. Refer to DevConfig's embedded help or the user manual for a device for more device-specific operations.

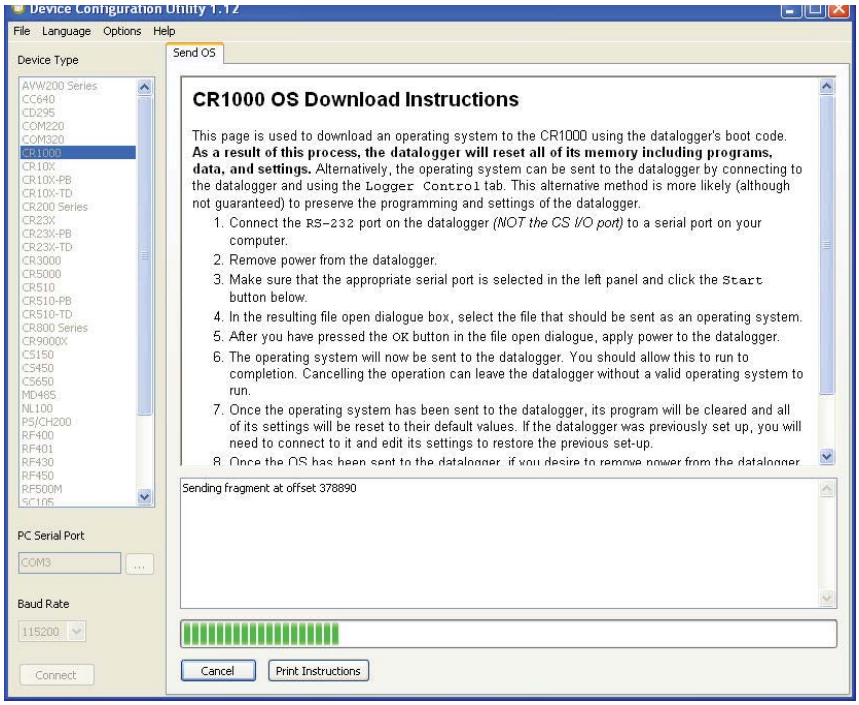
### 10.3.3 Downloading an Operating System

DevConfig can send operating systems from the Send OS tab to all Campbell Scientific devices with flash replaceable operating systems. An example for the CR1000 is shown below:

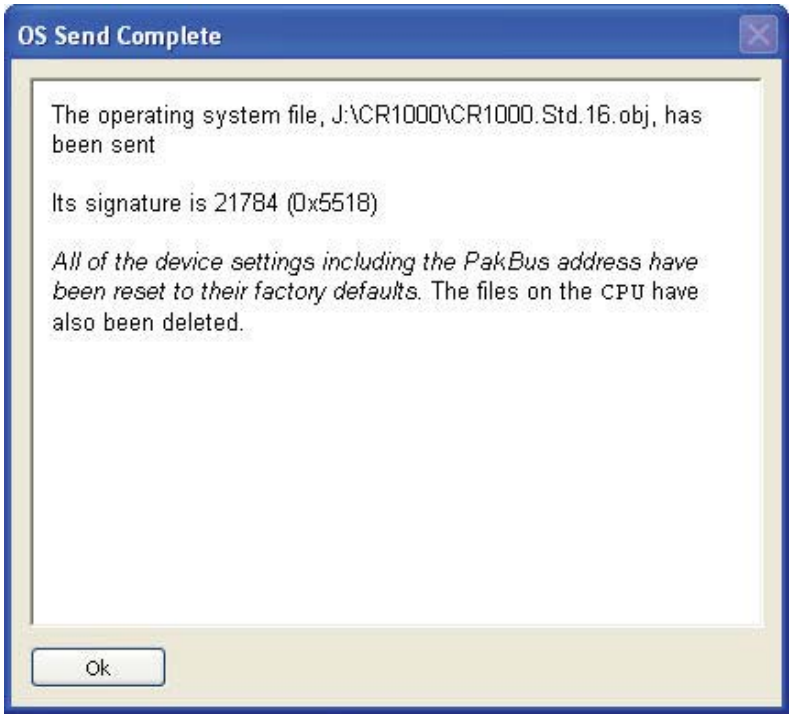



The text at right describes any interface devices or cabling required to connect the PC to the device. Screens for other devices vary only in the text on the right side. This screen differs from other screens that are available in DevConfig in that it can be accessed from either a connected or disconnected state.

When you click the **Start** button, DevConfig offers a file open dialog box to prompt you for the operating system file (usually a *.obj file). You may be required to cycle power the device or press a special “program” button. When the device issues the appropriate prompts, DevConfig starts to send the operating system:



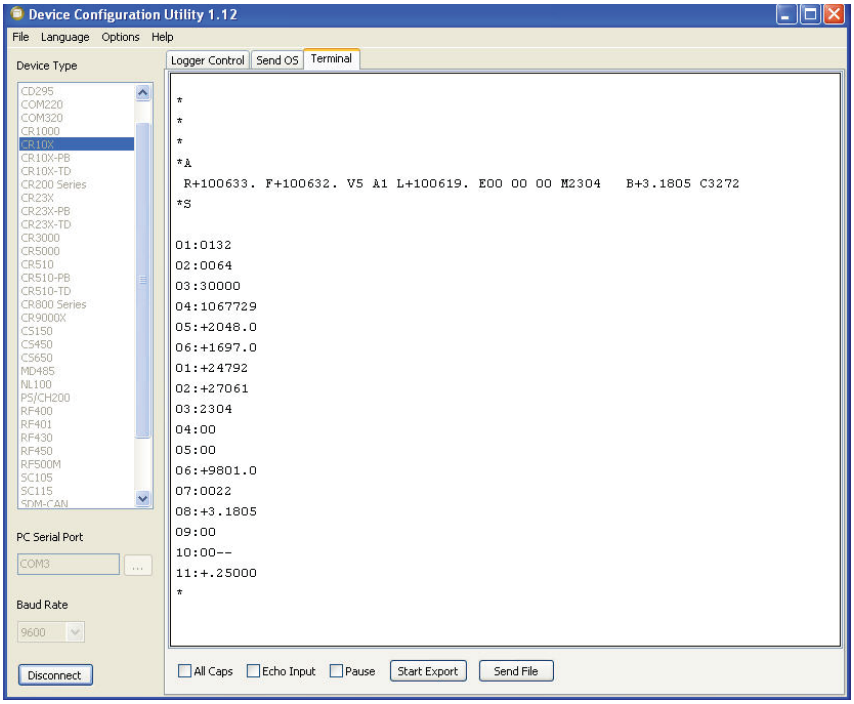

When the operating system has been sent to the device, a message dialog will appear similar to the one shown below:




The information in the dialog helps to corroborate the signature of the operating system sent. For devices such as the CR10X (especially those with extended memory) that can take a long time to reset following an OS download, text warns you against interrupting the memory test.

## 10.3.4 Terminal Tab

The **Terminal** tab will be available when the application is connected to any device type that can be communicated with in a remote terminal mode. The Terminal tab offers a terminal emulator that can be useful in accessing settings or status information that are not exposed in other windows. For example, classic dataloggers with PakBus operating systems that are configured as routers contain routing tables that list the other PakBus nodes that are known to that datalogger. This routing table is only available through the *D17 mode (see *D descriptions in the datalogger's operators' manuals) using the keyboard/display or a terminal emulator. Another example is that the status table in mixed-array dataloggers (*B) can also be accessed via an "S" command in terminal mode. This status information can provide important data for troubleshooting purposes.

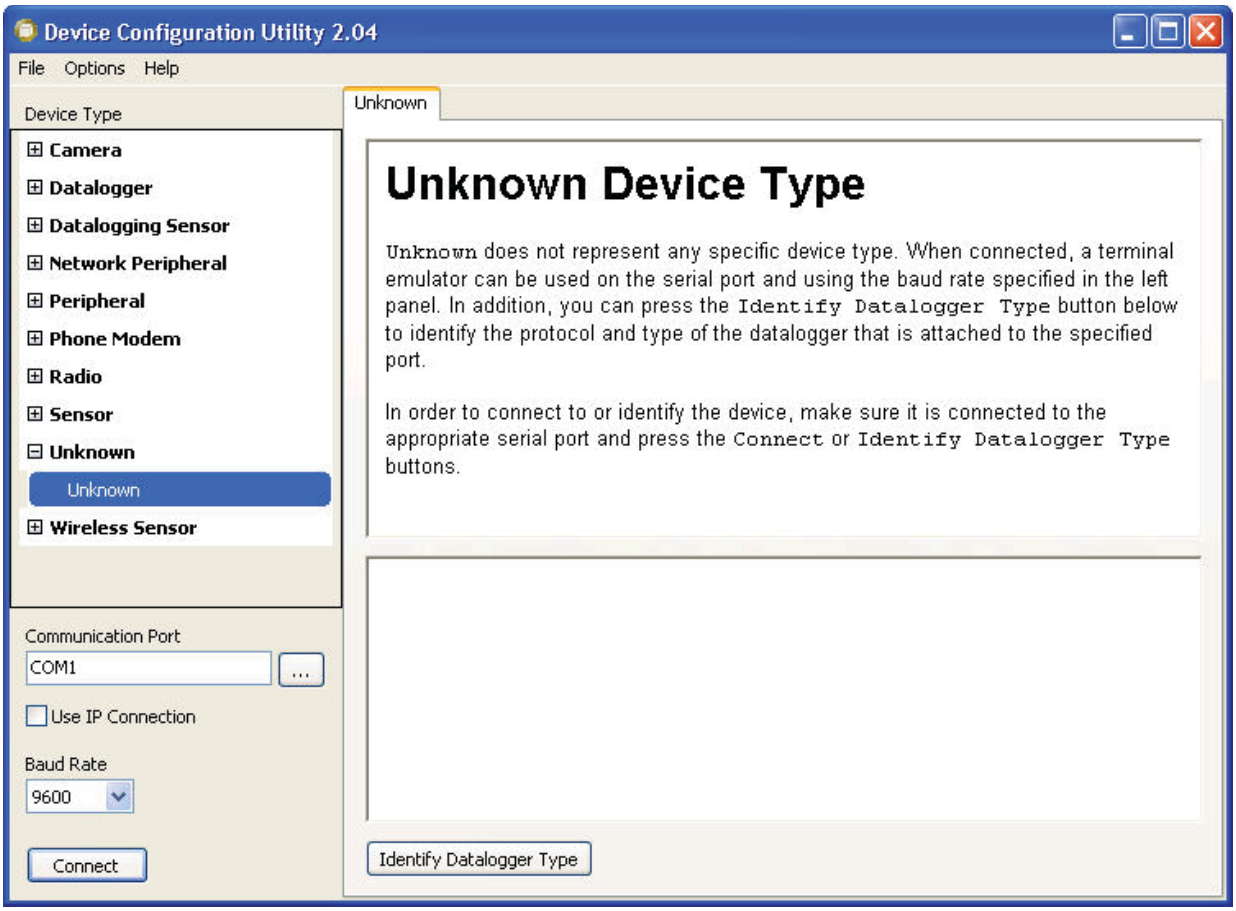



The default for the Terminal tab is to only show characters that are returned from the device. However, if the **Echo Input** check box is enabled, the screen will also display the characters actually typed by the user.

The **All Caps** check box controls whether the keyboard input will be forced to upper case before the characters are sent to the device. It will be disabled for some device types that require upper case input.

## 10.3.5 The Unknown Device Type

When the Unknown device type is selected, a panel will be shown in the tab control similar to that shown below:




Clicking **Connect** puts DevConfig into Terminal emulation mode on the Serial Port and at the Baud Rate selected.

When you click on **Identify Datalogger Type**, DevConfig will attempt to identify the type of device that is connected on the specified serial port. It will attempt to communicate using each of the datalogger protocols (mixed-array, table-data, and PakBus) in turn. If it fails to get any answer to any of these attempts, the baud rate will be automatically changed and the various protocols will be attempted again. When DevConfig recognizes the response from the device and the device type is one of the supported types, that device type will automatically be selected.

### 10.3.6 Off-line Mode

Many devices in DevConfig have an off-line mode available that allows you to browse the device's settings without actually being connected to a device. You can select a device, and then select Off-line Mode from DevConfig's File menu. You will be able to see the device's settings and associated help. You can also make changes to the settings and then press Apply to bring up the option to Save or Print the configuration. Saving the configuration will allow you to load it into a device at a later time.

# 10.4 File Format Convert

File Format Convert is not available from the PC400 toolbar. It can be opened from the Window's Start menu under Programs | Campbell Scientific | PC400 | Utilities

## 10.4.1 Overview

File Format Convert is used to convert data files from one format to another. It can also perform the following functions:

- Break large files into smaller files (also known as baling).
- Check for missing records by checking the record number and or timestamp.
- Bale based on time
- Bale based on File Marks and Remove Marks (TOB2,TOB3 files)
- Bale files when missing records are discovered.
- Fill in missing records with 'Null' or empty records.

More than one of the above functions can be performed in one pass.

In general files can be converted from:

- TOA5
- TOACI1
- TOB2
- TOB3
- TOB1
- CSIXML

To:

- TOA5
- TOACI1
- TOB1
- CSIXML
- CSV

**NOTES**

File Format Convert cannot produce TOB2 or TOB3 files, and it cannot read CSV files.

Some file headers have less information than other formats. If you convert from a file with more information in the header to one with less, information will be lost. If you convert from a format with less information, some fields will be left blank.

Some formats (e.g. TOB1) store string in fixed length fields and have headers that specify how big that field is. Other formats use variable length strings. If you convert from a format that uses variable lengths to a fixed length, the length is assigned to 64. If the string is longer than this, it is truncated.

#### *Converting a File*

Press the **Open** button to browse to a file to be converted. After a file is selected, press the **Options** button and set up the options for the conversion. Then press the **Convert/Check** button to convert the file.

If a conversion is in progress and you wish to stop it, press the **Abort** button.

#### *Log File*

If the **Write Log File** checkbox is checked, a "Log.Txt" file will be created in the same directory as the source data file. The log.txt file will be overwritten if it exists.

## 10.4.2 Options

#### *Check*

**Record Numbers** – Checks for missing record numbers.

**Timestamps** – Checks for missing timestamps based on entered interval.

Both can be checked in the same pass. If a file is written, other options are available.

Files can be baled if missing records are found. See the ***Bale based*** on information below. When checking timestamps, "null" records can be written to "fill" missing records. See ***Missing Records*** information below.

#### *File*

Check **Write File** to cause an output file to be created. The file will be created in the same directory as the source file. The base name will be the same as the source name with the new format prepended. For example, test.dat becomes TOA5_test.dat. Use the drop-down list to select the format of the new file.

For all output options except TOACI1, the browse button to the right of the field becomes available and can be pressed to set additional file output options.

### *File Naming*

**Date Time Filename** – When this option is selected, the date and time of the first record of data in the file will be appended to the end of the base file name. The suffix includes a four digit year, a two digit month, a two digit day of month, and a four digit hour/minute. When this option is selected, **Use Day of Year** becomes available. If this option is selected, the Julian day (day of year) will be used in the suffix instead of the month and day of the month.

**Create New Filenames –** When the **Create New Filenames** option is selected, File Format Convert will add a _1 to the filename, if a file of the same name is found (e.g., TOA5_Mydata_1.dat). If a *_1.dat file is found, the file will be named with a _2 suffix. If the **Create New Filenames** check box is cleared and a file with the same name is found, you will be offered the option to Overwrite the existing file or Cancel the conversion. (Note that if any of the baling options are selected, new filenames will automatically be created as described below.)

### *Missing Records*

If Timestamps are checked (see ***Check*** section above), then missing records can be filled. These will be Null records. A timestamp and record number will be added. Values will be "NAN". Strings will be empty. etc.

**Just Log –** Missing records are not filled.

**Fill Null Records –** All missing records are filled.

**Prompt –** Shows what records are missing and lets you choose to fill or not. If you have big gaps (e.g. bad timestamp), filling can be quite slow.

### *Bale based on*

This allows a file to be broken into smaller files. A new file is started based on:

**Time –** A new file is created based on interval.

**Remove Marks –** A new file is created when a remove mark is found in the data file (TOB3 only).

**File Marks –** A new file is created when a file mark is found in the data files (TOB3 and TOB2 only).

**Discontinuity** – A new file is created when missing records are encountered (see ***Check*** section above). The **How Many?** can be used so that small gaps do not start a new file. The file will be baled only if the entered number of records (or more) are missing.

### *Bale Info*

Use to specify the **Start Time** and **Interval** and start time for baling based on time.

# Appendix A. Glossary of Terms

## A

**Advise –** See Data Advise

**ASCII File** – A computer file containing letters, numbers, and other characters using the ASCII character encoding.

**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 that designate the beginning and ending points of the information (see Synchronous). The transfer of information is not otherwise coordinated between the sender and receiver.

**Analog Channel** – A terminal on the datalogger's wiring panel where leads for analog signals are connected. The analog channels are designated single-ended (SE) or differential (DIFF) on the wiring panel. Many sensors, such as thermistor temperature probes and wind vanes, output analog signals.

**Array-based Datalogger** – See Mixed-array Datalogger.

## B

**Batch Files** – An ACSII text file that contains one or more DOS commands or executable file commands. When the batch file is run, the commands in the file are executed sequentially.

**Battery** – This entry in the status table returns the datalogger battery voltage.

**Baud** – The rate at which a communication signal travels between two devices.

**Binary File** – A file based on software defined formatting. A binary file can only be interpreted by the software programmed to decode the formatting. This format is used for more efficient data storage than is provided by ASCII.

**BMP (Block Mode Protocol)** – The communications protocol used by the server to communicate with table-based dataloggers and RF modems.

**Broadcast –** Part of the radio (RF) technique of polling remote radio modem datalogger sites. A single modem sends a message (broadcast) that all affected remotes hear and respond to.

## C

**Call-back** – When a datalogger is programmed for Call-back, it will automatically call the host computer when a specified condition is met. The computer must be set up to look for such an incoming call.

**Call-back ID Number** – A three-digit number that is used to identify what datalogger has called the host computer. (Not available for Table-based dataloggers.)

**Cancel** – Choosing Cancel from a dialog box will typically ignore any changes made and close the box.

**Carrier** – An electrical signal used to convey data or other information. For example, radio and phone modems use carrier signals. Phone modems attempt to detect carrier when the call is placed. The red LED on the RF95T lights when the modem detects a carrier.

**Child Node –** See Node. A node that is accessed through another device (parent node). For example a remote radio frequency (RF) site is accessed through and a child of the base RF232T. All nodes are child nodes of the PC.

**Client –** a software application designed to connect to a server. Usually provides some type of user interface or data acquisition. Email programs running on individual PCs are typically client applications that connect to an email server program running on a computer at an Internet Service Provider to receive and send email messages.

**Coaxial cable –** Special type of cable with two conductors (center conductor and outer shield conductor). Classified by size, impedance, and loss characteristics. Used to connect MD9 modems and to connect radios to antennas.

**Collection –** (see Data Collection)

**COM Port** – A computer's serial communications port. Cables and other interface devices are connected between the computer's COM port and the datalogger.

**Communication Server** – The software (typically packaged as a DLL) that provides the communications functions within other software such as PC200W, PC400, or LoggerNet.

**Control Port** – Dataloggers have digital output ports that can be used to switch power to sensors such as the HMP35C relative humidity circuit or to control relays. These digital outputs are called Control Ports and are labeled C1, C2, etc., on the wiring panel. Control ports on some dataloggers can also be used as inputs to sense the digital (high or low) state of a signal, monitor pulse signals, control Synchronous Devices for Measurement (SDM), or used as data input/output connections for SDI-12 sensors.

**CoraScript** – A command line interpreter client to the LoggerNet server that allows the user access to many of the capabilities of the LoggerNet server using direct commands or programmed script files.

**CR10X-TD Family of Dataloggers** – Any of the Edlog dataloggers with table-data operating systems become "TD" dataloggers, including the CR10T, CR510-TD, CR10X-TD, and CR23X-TD.

**CRBasic –** The programming language used for CR200, CR800 Series, CR1000, CR6 Series, CR300 Series, CR3000, CR5000 or CR9000 dataloggers. Short Cut or the CRBasic Editor are used to create program files for these dataloggers.

**CRBasic Datalogger** – A CR200/205, CR800 Series, CR1000, CR6 Series, CR300 Series, CR3000, CR5000 or CR9000 datalogger. Sometimes referred to as "CRx000 dataloggers."

**CRx000 Datalogger** – Generally, a CR200/205, CR800 Series, CR1000, CR6 Series, CR300 Series, CR3000, CR5000 or CR9000 datalogger. More correctly referred to as "CRBasic dataloggers."

# D

**Data Advise (Datalogger)** – A mutual agreement between the communication server and the datalogger about which tables are to be collected every time the datalogger is contacted. Based on the dataloggers table definitions.

**Data Advise (Server)** – an agreement between a client application and the communication server to provide specified data as it is collected by the server.

**Data Advise Notification** – The packet of data sent by the datalogger based on the Data Advise agreement.

**Data Cache –** The storage for data collected from the datalogger by the communication server. This data is stored in binary files on the hard disk of the computer where the server is running.

**Data Collection** – Getting a copy of the data stored in the datalogger and saving it in the communication server's data cache (compare to Data Retrieval).

**Data Point** – A data value that is sent to Final Storage as the result of an Output Instruction. A group of data points output at the same time makes up a record in a data table.

**Data Retrieval** – Sending a copy of the data from the communication server's data cache to a file, network, or data display (compare to Data Collection).

**Data Storage Table, Data Table** – A portion of the datalogger's Final Storage allocated for a particular output. Each time output for a given data table occurs, a new record is written to the table. The size of the table (in number of records) and when records are written to the data table are determined by the datalogger's Data Table Instruction (P84). The fields (columns) of the table are determined by the Output Processing Instructions that follow the Data Table Instruction.

**Data Table Instruction** – Instruction 84. Used to create a Data Table and to cause records to be written to the Data Table.

**DaysFull** – A field in the status table that shows the number of days before any of the tables using automatic record allocation are filled.

**DevConfig** – Short for "Device Configuration Utility", a software application that provides a graphical user interface to configure settings in dataloggers and communications peripherals. Available in PC400, LoggerNet, and as a stand-alone application from the Campbell Scientific website. (Supplants CSOS.EXE, PakCom, and stand-alone terminal emulators.)

**Differential Analog Input** – Some sensors have two signal wires and the measurement is reflected in the voltage difference between them. This type of sensor requires two analog connections. The channels marked DIFF on the datalogger wiring panel are used to connect differential sensors.

**DLD File** – An ASCII file that can be sent to program an Edlog datalogger. Dataloggers must be programmed to perform measurements, convert data to final units, and to save data for retrieval. Edlog is used to create these files that are saved to disk with a DLD file name extension. A program must be sent to the datalogger before the datalogger will begin to collect data.

## E

**Edlog** – Campbell Scientific's software application used to create new or edit existing datalogger programs. Edlog supports all of the programming capabilities in the dataloggers it supports. (Program generators such as Short Cut are necessarily more limited in the features they can support.)

**Edlog Datalogger –** Any of the dataloggers, 21X, CR7, CR10, CR500, CR10X, CR510, or CR23X. The default operating system for these dataloggers is a mixed-array configuration. Some of these, specifically the last three, can have alternative operating systems installed by users. These include mixed-array, table-data (TD), or PakBus (PB) operating systems.

**EEPROM** – Electrically erasable programmable read only memory; the memory CR10X-TD, CR510-TD, and CR23X-TD dataloggers use to store their operating system. A new operating system can be transferred to the datalogger using a special software package (see PROM and DevConfig).

**Execution Interval** – The periodic interval on which the datalogger program is run. The execution interval is sometimes referred to as the Scan Interval. For example, when an execution interval of 60 seconds is set, the datalogger will execute its program table every 60 seconds. Between executions the datalogger enters a sleep (quiescent) mode. This conserves battery power and creates predictable measurement intervals. The execution interval is synchronized with the datalogger's real-time clock.

**Execution Time** – The time required to execute an instruction or group of instructions. If the total execution time of a Program Table exceeds the table's Execution Interval, the Program Table will be executed less frequently than programmed. Each time this occurs, a Table Overrun occurs. Table Overruns are considered to be "errors" and are reported in the datalogger status information table.

**Excitation Channel** – Sensors utilizing electrical bridge circuits require a precise electrical voltage to be applied. The excitation channels, marked as E1, E2, etc., on the datalogger wiring panel, provide this required precision voltage.

## F

**Fault –** Message relating to network activity where repeated problems or errors have occurred. Repeated faults usually indicate a failure of some kind.

**F1** – In most instances, pressing the F1 key will provide context sensitive help for the highlighted object on the screen.

**Final Storage** – Final Storage is an area in the datalogger's memory where data is stored for collection to a PC. When you collect data from the datalogger you are collecting data from a Final Storage area or table.

**Flag** – Memory locations where the program can store a logical high or low value. These locations, called User Flags, are typically used to signal a state to another part of the program.

# G

**Ground Connection** – Most sensors require one or more ground connections in addition to excitation or signal inputs. Ground connections may serve any of several purposes:

- a reference for a single-ended (SE) analog voltage (use analog ground if available)
- a power return path (do NOT use analog ground for power return)
- a connection for cable shield wire to help reduce electrical noise (do not use analog ground for shield wires, also known as drain wires)

# H

**Highlight** – Text or objects can be highlighted, by positioning the cursor where you want the highlight to begin, holding the left mouse button, and dragging it across the words or group of objects to be highlighted. A single object can be highlighted, by clicking it once with the left mouse button. Highlighted items can then be edited or activated.

**Holes** – When using Data Advise, the communications server always gets the most recent data records, so if there are more records to be returned than can fit in one packet there can be sequences of older data available from the datalogger that have not yet been collected to the data cache. The server tracks and collects these holes only if that option is enabled. This entry in the status table shows the number of data points in missed records for the data storage tables in that station.

**Hole Collection** – The process used by the server to collect data records missing from the data cache but possibly still in the datalogger. If Hole Collection is delayed or disabled, the memory in the datalogger can ring around and overwrite the missing data records resulting in an Uncollectable Hole.

**Host Computer** – The machine where the communication server software is running.

# I

**INI Files** – Configuration files that are used to preserve the last known setups or states of a program or device.

**Initialization String** – A string of alphanumeric characters that are sent to a device, such as a modem, to prepare that device for communications.

**InLocs** – Abbreviation for “Input Locations”. This entry in the status table shows the number of input locations allocated for the program.

**Input Location Storage** – Each time a measurement or calculation is performed the resultant value is stored in an Input (memory) Location, sometimes abbreviated as “InLoc.”

**Input/Output Instructions** – Datalogger program instructions used to make measurements or send data automatically to other devices.

**Intermediate Storage** – Datalogger memory used to temporarily store values (such as a running total and number of samples for an average calculation), typically to be used for output calculations. The datalogger uses Intermediate Storage to accumulate sensor readings until output.

## L

**Link** – Communications route between two devices, for example the phone link between two phone modems.

**LDEP** – Logger Data Export Protocol, a protocol and client application that provides for data distribution from the communications server to a third party application through a standard TCP/IP socket. Installed with LoggerNet Admin; see the associated PDF file for more information. Requires record-specific acknowledgements for record flow control. See LDMP.

**LDMP** – Logger Data Monitoring Protocol, a protocol and client application that provides for data distribution from the communications server to a third party application through a standard TCP/IP socket. Installed with LoggerNet Admin; see the associated PDF file for more information. Requires very simple acknowledgements for record flow control. See LDEP.

**Log Files** – Text files that are stored on the computer's hard drive that record activity. They contain information about communications between the communications server and other devices in the datalogger network. Log files are typically used for troubleshooting purposes. LoggerNet has four types of log files: Transaction, Communications Status, Object State, and Low Level I/O. Refer to Appendix C, *Log Files and the LogTool Application (p. C-1)*, or the help within the LogTool (in PC400 click the Tools | LogTool menu item) application for information on these log files.

## M

**MD9** – An MD9, or multi-drop modem, is a communications device that uses twisted pair cable for connection. Typically, the system consists of one MD9 base modem that is attached to the user's computer, with one or more remote modems at the datalogger field site. One remote modem is needed for each datalogger at the field site.

**Measurements** – Values stored by the datalogger in an Input Location after reading an electronic signal from a sensor and converting the raw signal into meaningful units.

**Mixed-array** – Dataloggers with mixed-array operating systems save output in a common area of the datalogger's final storage memory. When data is directed to final storage, a unique array ID number is stored, followed by other values as determined by the datalogger program. These are called "elements". "Mixed-array dataloggers" typically save all information that is directed to output storage to the same area of datalogger memory (as opposed to table-based dataloggers that always store different output processing intervals to separate tables in datalogger memory). Data retrieved by the PC must be processed by PC software to separate the data based on the array IDs.

**Modem** – From "modulator-demodulator"; a device used to transmit and receive digital data over normally analog communications lines, such as an audio signal on telephone circuits. A modem attached to a computer performs

a digital-to-analog conversion of data and transmits them to another modem that performs an analog-to-digital conversion which permits its attached computer to use the data.

## N

**Net Description** – Description of dataloggers and communications devices that form the datalogger network. Created using the EZWizard in PC400 or Setup screen in LoggerNet to communicate with the various dataloggers.

**Node** – Part of the description of a datalogger network. Each node represents a device that the communications server will dial through or communicate with individually. 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 node.

## O

**ObjSrlNo** – This entry in the status table provides the revision number of the datalogger PROM.

**Output Interval** – The output interval is the interval at which the datalogger writes data to Final Storage. The output interval is defined by Instruction 84 in Edlog (for table-based dataloggers) or the instructions that set the output flag high in mixed-array dataloggers.

**Output Processing** – Writing to final storage memory a sample or summary statistic of data measurements. Output processing options include sending a sample, average, maximum, minimum, total, or wind vector of data to Final Storage. Each Output Processing data value is kept in a separate location within the datalogger. This allows multiple output processing for each measurement. For example, you can average air temperature over a 60-second interval, a one-hour interval, and a 24-hour interval. See the operator's manual or programming software for output processing options available for each datalogger model.

**Overrun Errors** – Overrun errors occur when the actual program execution time exceeds the execution interval. This causes program executions to be skipped. When an overrun error occurs, the Table Overrun parameter in the datalogger's status table is incremented by 1.

**Overruns** – This entry in the status table provides the number of table overruns that have occurred. A table overrun occurs when the datalogger has insufficient time between execution intervals to complete one pass through the program. This counter is incremented with each table overrun.

## P

**Packet** – a unit of information sent between two BMP or PakBus devices that are communicating. Each packet can contain data, messages, programming, etc. Usually contains addressing and routing information.

**PakBus** – A packet-based and packet-switched networking protocol used by newer dataloggers. PakBus allows for robust transmission of commands and data, dynamic routing between PakBus devices, and peer-to-peer communications (such as when one datalogger needs to control another datalogger without involving the PC).

**Parameter** – Number or code which helps to specify exactly what a given datalogger instruction is to do.

**Path** – The modems, or other devices that make up a link to communicate with a remote site datalogger.

**Polling** – Process where a datalogger or other communications device is periodically checked for any packets it needs to send. The server polls dataloggers for most communications links. Some communications devices, such as RF232T radio bases or repeaters can also poll datalogger sites.

**Polling Interval** – The user-specified interval that determines when to poll a given device.

**PrgmFree** – An entry in the status table that shows the amount of remaining program memory, in bytes.

**PrgmSig** – An entry in the status table that shows the signature of the datalogger program. The signature is a unique number derived from the size and format of the datalogger program.

**PromID** – An entry in the status table that shows the version number of the datalogger PROM or OS.

**PromSig** – An entry in the status table that shows the signature of the datalogger PROM or OS. As with the PrgmSig, if this signature changes, the datalogger instruction set has somehow been changed.

**Processing Instructions** – Datalogger instructions that further process input location data values and typically return the result to Input Storage where it can be accessed for output processing. Arithmetic and transcendental functions are included in these instructions.

**Program Control Instructions** – Datalogger instructions that modify the sequence of execution of other instructions in the datalogger program; also used to set or clear user flags.

**Program Signature** – A program signature is a unique value calculated by the datalogger based on program structure. Record this signature in a daily output to document when the datalogger program is changed.

**Program Table** – The area where a datalogger program is stored. Programming in Edlog dataloggers can be separated into two tables, each having its own execution interval. A third table is available for programming subroutines that may be called by instructions in Tables 1 or 2. Programming in CRBasic dataloggers can be separated into different “scans”. The length of the program tables or scans is constrained only by the total memory available for programming.

**PROM** – Programmable Read-Only Memory – integrated circuit chips that are used to store the Operating System (OS) in the CR10T datalogger and some other communications peripherals. The PROM can be replaced to install a new operating system (also see EEPROM).

**Pulse Channel** – Some sensors output voltage pulse signals. Such sensors can be connected to Pulse Channels for measurement (labeled as P1, P2, etc., on the datalogger’s wiring panel).

# Q

**Quiescent Mode** – Often referred to as "sleep mode" – a low power state between program execution intervals.

# R

**Real-Time Clock** – All dataloggers have an internal clock. The date and time information from this clock are used in the time stamp for stored data. The datalogger's execution interval and timer are synchronized with the clock. Some Edlog dataloggers (CR10X, CR510, and CR23X) and all CRBasic dataloggers have battery backups that maintain the clock even when 12V power is not available.

**Record** – A group of data values output at the same time to the same data table. Records are written in response to the Data Table Instruction (84) in TD dataloggers or the DataTable declaration in CRBasic dataloggers. The individual fields within each record are determined by the Output Processing instructions following the instruction that created the data table.

**RecNbr** – An entry in a table that shows the sequential record number in the table.

**Remote Site** – Typically where a datalogger is located at the other end of a communications link. Also can refer to the site where a radio (RF) repeater is located.

**Repeater** – a radio (RF) site that relays packets of information to a remote site. Used to extend the range of radio transmissions. Most remote datalogger sites with radios can act as repeaters.

**Retries** – When a transaction or communication between two devices or programs fails, the transaction or communication can often be triggered to repeat until it succeeds.

**Retrieval –** (see Data Retrieval).

**RF** – Radio Frequency.

**RTMC** – Real Time Monitoring and Control software. A client application to the communications server that displays data from the server's data cache (only) and updates as new data is collected. RTMC is relatively easy to set up, and ships with LoggerNet.

# S

**Scan Interval** – See Execution Interval.

**SDI-12** – SDI-12 stands for Serial Digital Interface at 1200 baud. It is an electrical interface standard and communications protocol that was originally developed by Campbell Scientific and other manufacturers for the U.S. Geological Survey for hydrologic and environmental sensors. SDI-12 was designed to be a simple interface (ground, 12 volts, and signal) that improves compatibility between dataloggers and "smart" microprocessor-based sensors.

Other goals of the SDI-12 standard are:

- low power consumption for battery powered operation via the datalogger
- low system cost
- use of multiple sensors on one cable connected to one datalogger
- allow up to 200 feet of cable between a sensor and a datalogger

**Security Code –** A code entered into the datalogger either directly with a keypad or via the datalogger's program to prevent unauthorized access to datalogger settings, programs, and data.

**Server** – Also "communication server", a software application that accepts connections from client applications and provides data or other information as requested. The LoggerNet server manages all the communications and data collection for a network of dataloggers. The collected data is made available for client applications. PC200W and PC400 also use the communication server but in a more limited configuration.

**Short Cut** – A program generator application that ships with PC400, LoggerNet, and is available as a stand-alone product from the Campbell Scientific website. Short Cut does not require knowledge of individual program instructions. Users need only know what kind of datalogger and sensors they're using and decide what output they require. Short Cut generates the program for them. (Contrast a "program generator" with the full-featured "program editors", Edlog and CRBasic Editor.)

**Signature** – Number calculated to verify both sequence and validity of bytes within a packet or block of memory.

**Single-ended Analog Input** – Some analog sensors have only one signal wire. (They will also have another wire that can be grounded and that is used as the reference for the signal wire.) With this type of sensor, only one analog connection is required. Hence, it needs a "single-ended" or SE analog input. The single ended channels are marked as SE on the datalogger wiring panel.

**Socket Data Export** – a software application that connects to the LoggerNet server and provides a TCP/IP socket for a user created application to receive data records from the server data cache.

**Station** – A datalogger site is often referred to as a station.

**Station Number** – The LoggerNet server assigns and uses station numbers for routing packets to the dataloggers. These numbers can be modified using CoraScript. Not to be confused with datalogger serial numbers, PakBus addresses, or addresses set in communications peripherals such as RF or MD9 modems.

**Storage** – An entry in the status table that shows the number of final storage locations available.

**Synchronous** – The transmission of data between devices occurs as groups 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 Asynchronous).

# T

**Tab Windows** – Some screens depict a series of related windows in a multi-tabbed notebook format. When you click the file folder tab, the information on the tab you chose will be displayed.

**Tables** – An entry in the status table that shows the number of user-created data tables. (See also Data Table.)

**Table-based Dataloggers** – Table-based dataloggers store each record of data that follows an output instruction in a table. Each separate occurrence of an output instruction directs the datalogger to store the data in a separate table. "Table-based" includes both "TD" table-data and "PB" PakBus versions of the Edlog dataloggers as well as the CRBasic dataloggers.

**Table Definitions** – List of data available from a table-based datalogger. The datalogger supplies this list on request. The tables are determined by the datalogger program. The LoggerNet server must have a current version of the table definitions to collect data from the datalogger.

**Time Stamp** – The date and time when data are stored in the datalogger.

**TMStamp** – An entry in the status table that shows the date and time the status information was recorded.

**Transaction** – The exchange of data or information between two devices or programs. For example, setting the clock in a datalogger requires a transaction between the server and the datalogger.

# U

**Uncollectable Hole** – Occurs when a hole in the data cache cannot be collected from the datalogger before the data table wraps around and the records are overwritten.

# V

**Variable Name** – Edlog uses variable names in expressions. Variables are another name for input location labels. For instance, in the equation TempF = (TempC*1.8) + 32, TempC is an input location label and TempF is a new location calculated from TempC. CRBasic dataloggers use variables for all measurements, processing values, including variables to be used in Boolean form as "high" or "low".

# W

**Wiring Panel** – The set of terminals and underlying circuits that enable connections of sensors, control and power supply wiring to the datalogger itself. Some dataloggers such as the CR23X have built-in wiring panels. Others, such as the CR10X, have removable wiring panels.

**Watchdog** – An entry in the status table that shows the number of watchdog errors that have occurred. The watchdog checks the processor state and resets it if necessary. If an error occurs, the watchdog error counter is incremented.

# Appendix B. Table-Based Dataloggers

*This section describes some of the characteristics and features of the Edlog TD and PB dataloggers and CRBasic dataloggers. These include the CR510-TD, CR510-PB, CR10T, CR10X-TD, CR10X-PB, CR23X-TD, CR23X-PB, CR200, CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, and CR9000. See the operator's manual for the specific datalogger for detailed information about its operation.*

## B.1 Memory Allocation for Final Storage

The datalogger memory includes four important areas: the datalogger program storage, input storage, intermediate storage, and final storage. When a program is downloaded to the datalogger and compiled, datalogger memory is allocated for each of these areas.

The Edlog mixed-array and table-based dataloggers are identical in hardware and differ only in the operating system. The primary distinction between mixed-array and table-based dataloggers is how final storage is allocated and filled. CRBasic dataloggers use CRBasic programs and have a different memory allocation structure.

### B.1.1 Edlog TD Family Dataloggers

CR510-TD, CR10T, CR10X-TD, and CR23X-TD table-based dataloggers store data from different intervals in different final storage tables. Final storage 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 output processing instructions in the datalogger program that follow the Data Table output instruction (P84 Output Table; refer to your datalogger user's manual for more information). The total number of fields for each table will be the number of output processing instructions multiplied by the number of values stored by each of the output instructions.

The number of records to be kept in a table before the oldest data is overwritten can be fixed by the user, or left for the datalogger to determine automatically. With automatic allocation the datalogger tries to set the sizes of automatically allocated tables such that all of the tables will fill up at about the same time. Once the sizes of the tables are determined, the datalogger allocates available final storage to these tables.

Note that the tables are allocated by size with the smallest tables first. For dataloggers with extended flash memory, any tables that will not fit in SRAM memory are allocated to flash memory. Flash memory is allocated in 64 K blocks; therefore, even a very small table will take 64 K of flash memory. If the table sizes specified by the user exceed the amount of memory available for that purpose in the datalogger, an error will occur when the program is compiled by the datalogger.

**NOTE** Event driven tables should have a fixed size rather than allowing them to be allocated automatically. Event driven tables in Edlog TD dataloggers that are automatically allocated are assumed to have one record stored per execution interval in calculating the length. Since the datalogger tries to make the tables fill up at the same time, with programs using short execution intervals these event driven tables may take up most of the memory leaving very little for the other, longer interval, automatically allocated data tables.

## B.1.2 CR800/CR1000/CR6 Series/CR300 Series/CR3000/CR5000/ CR9000 Memory for Programs and Data Storage

The datalogger memory for the CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, and CR9000 is divided between Random Access Memory (RAM) and Electrically Erasable Programmable Read Only Memory (EEPROM). The EEPROM, or flash memory, is used to store the operating system and the user programs that have been saved in the datalogger. When the datalogger powers up, the program marked as "Run on Power-up" is transferred to RAM and executes from there. Additional storage is available using PCMCIA, microSD, or Compact Flash cards.

When a datalogger program is sent to the datalogger, it is divided into two tasks that run simultaneously. All of the program instructions that deal with measuring sensors, controlling outputs, or are time sensitive, are placed in the **measurement task**. The instructions that deal with data processing, including calculations, data storage, averaging, minimum and maximum tracking, and data I/O operations, are placed in the **processing task**.

The measurement task is executed at the precise specified scan rate and stores the raw data into a memory buffer. As soon as the measurement task has completed filling the buffer for the current scan, the processing task starts the data processing on the buffered data. There are at least two memory buffers, allowing the measurement task to fill one buffer while the processing task is working with the data in the other.

The data processing task stores data as records in final storage data tables. PC400 can collect the records from these data tables. The datalogger program can also make some or all of the variables used for measurement storage or calculations available to PC400. These variables are found in the Public table, which is similar to the Input Location table in Edlog dataloggers.

Final storage 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 Data Table definition.

The number of records to be kept in a table before the oldest data is overwritten can be limited by the user, or left for the datalogger to determine automatically. The datalogger tries to set the sizes of automatically allocated tables such that all of the tables will fill up at about the same time. Once the sizes of the tables are determined, the datalogger allocates the available memory to these tables.

If the amount of memory requested for the data tables exceeds the available memory, the program will not run.

**NOTE** Event driven tables should have a fixed size rather than allowing them to be allocated automatically. Event driven tables in CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, and CR9000 dataloggers that are automatically allocated are assumed to have one record stored per execution interval in calculating the length. Since the datalogger tries to make the tables fill up at the same time, with programs using short execution intervals these event driven tables may take up most of the memory leaving very little for other, longer interval, automatically allocated data tables.

### B.1.3 CR200 Series Dataloggers

CR200 Series dataloggers are similar to the other CRBasic dataloggers regarding the format of final storage. Data is stored in final storage tables that are made up of records and fields. As with the other table-based dataloggers, the user can specify the number of records for each table, or table-size can be determined by the datalogger. And, as with the other dataloggers, the size of event driven tables should always be entered by the user, or else the datalogger will calculate the amount of memory for the table based on the execution interval.

The CR200 Series dataloggers have a Public table in which the current scan's measurements are held. However, the CR200 Series does not have the ability to store multiple programs and program processing takes place in a linear, or sequential, fashion similar to the Edlog family of dataloggers (e.g., each programming instruction is performed sequentially; one instruction must finish before the datalogger proceeds with the next).

The CR200 Series dataloggers do not have an on-board compiler. Programs must be precompiled into a binary format by the PC software before they are sent to the datalogger.

## B.2 Converting a Mixed-array Program to a TD or PB Table-Based Program using Edlog

The following information is provided for those users familiar with writing programs for mixed-array dataloggers or users who have existing mixed-array datalogger programs that need to be changed to table-based programs.

**NOTE** PakBus versions of the Edlog dataloggers generally use the same programming instructions as TD dataloggers.

### B.2.1 Steps for Program Conversion

If you are converting a program from mixed-array to table-data format for the same datalogger (e.g., converting a CR10X program to a CR10X-TD program) you can edit the existing program in Edlog. If you are converting a program from one datalogger series to another (e.g., CR10X to CR23X-TD), you may be better off need to start the program from scratch because different dataloggers may have very different measurements, numbers of input terminals, and parameters for similar instructions.

As an example, to convert a mixed-array program for a CR10X to the table-data version for the CR10X-TD (use the same "CR10X-TD" name for converting to CR10X-PB):

1. Open the CSI file in Edlog.

2. The first line of the file will read

   ;{CR10X}

   Change this line to

   ;{CR10X-TD}

3. **Review all of the instructions provided in the section below**. If any of these are included in your program, format them as a comment or delete them from the program.

4. Save the file to a new file name, but **do not compile the file when prompted**.

5. Open the newly created file in Edlog. It will be opened using the CR10X-TD datalogger template instead of the CR10X. Make any changes necessary to replace the commented or deleted instructions.

6. Save and compile the program, correcting any errors that may be found by the compiler.

## B.2.2 Program Instruction Changes

Several programming instructions have changed or are not used in table-based datalogger programs. Make sure you "comment out" any of these instructions before you try to convert the mixed-array program. These are listed below:

- Check any instructions that may set the Output Flag (Flag 0) high or low by using the Command Code Options. The output flag is not used in table-based programming. Instructions that may include reference to the output flag are: P83, If Case; P86, Do; P88, If (X< = > Y); P89, If (X< = > F); P91, If Port/Flag; and P92, If Time.

  If any of these instructions set the output flag high, the instruction can be replaced with Instruction 84, Data Table. Instruction 84 is used to define a table of final storage data. New records of data are stored in the table based on time (interval data) or when a user flag is set (event data). Time based output intervals are specified in seconds.

- **Instruction 18, Time** – Instruction 18 is used to store the current time into an input location. Parameter 1 designates what format will be used when storing the time. There are differences in this instruction's Parameter 1 for the two datalogger types.

- **Instructions 73 and 74, Maximum and Minimum** – These instructions are used to store the maximum or minimum for a value over a period of time. Parameter 2 in these instructions is used to designate a time option. There are differences in the instructions' Parameter 2.

- **Instruction 77, Real Time** – Instruction 77 is used to store the current time in final storage for mixed-array dataloggers. **This instruction does not exist at all in TD dataloggers should be deleted instead of commented out in the original program before reloading it as a TD program.** Time is assigned to records automatically in TD dataloggers when data is retrieved.

- **Instruction 80, Set Active Storage Area** – Instruction 80 is used to direct output processing to final storage area 1, final storage area 2, or an input location. **This instruction does not exist at all in TD dataloggers should be deleted instead of commented out in the original program before reloading it as a TD program.** Output processing can be redirected to input locations in a table-based datalogger using Instruction P84, Table Data (see Edlog's help).

- **Instruction 92, If Time** – Instruction 92 is used to perform one or more actions based on time. The interval for table-based dataloggers is in seconds only; mixed-array dataloggers offer the options of seconds or minutes. The instruction for mixed-array dataloggers defaults to minutes, so if you are using this instruction it may need to be changed.

  Also, check any Instruction 92s for Command Codes that may affect the output flag (see discussion above on output flag instructions).

- **Instruction 96, Serial Output –** Instruction 96 is used to send data in the active Final Storage area to a storage module, computer, printer, or alternate final storage area. **This instruction does not exist at all in TD dataloggers should be deleted instead of commented out in the original program before reloading it as a TD program.**

- **Instruction 98, Send Printer Character –** Instruction 98 is used to send characters to either an addressed or pin-enabled printer. **This instruction does not exist at all in TD dataloggers should be deleted instead of commented out in the original program before reloading it as a TD program.**

- **Conditional Data Output** – check to make sure that the output data is not being output conditionally. Table-based dataloggers require that the size of the output record is constant. Any instructions that dynamically change the number of data values in a record or the size of the record need to be removed. (e.g., don't change data resolution from low to high based on a conditional.)

# B.3 Table Data Overview

In the datalogger all data is organized into tables with fixed data records. Each of these tables has a definite number of records that is either fixed by the datalogger program or allocated when the program is compiled by the datalogger. Once the maximum number of records for a table has been stored, the next record stored will overwrite the oldest record in the table. The record number will continue to increment, and the oldest record will "drop off" the top.

Tables that are automatically allocated in the datalogger program are allocated a number of records based on the time interval for the records. The datalogger

attempts to allocate these tables so that all of the automatically allocated tables fill up at the same time. For example two tables with records stored every 30 minutes and 60 minutes would have twice as many records allocated for the 30-minute table.

**NOTE**

Event driven tables should have a fixed size rather than allowing them to be allocated automatically. If automatically allocated, event driven tables in the later versions of the operating systems of CR10X-TD type dataloggers are assumed to have one record stored per execution interval in calculating the length.

In CR800, CR1000, CR6 Series, CR300 Series, CR3000, CR5000, and CR9000 dataloggers event tables are assumed to have one record stored per execution interval.

Since the datalogger normally tries to allocate the table sizes so they fill up at the same time, if you let the datalogger automatically allocate table sizes these event driven tables may take up most of the memory leaving very little for the other, longer interval, automatically allocated data tables.

Within a data table, data is organized in records and fields. Each row in a table represents a record and each column represents a field. To understand the concept of records it may be helpful to consider an example.

**Example:**
A CR10X-TD is to be used to monitor three thermocouples. Each hour a temperature for each of the three thermocouples is to be stored. The table has five fields: DATE_TIME, RECORD #, TEMP1, TEMP2, TEMP3.

The program is written so that each hour an Instruction 84, Table Data, generates a new "record" in the data table. This hourly table would then be organized as follows:

| DATE_TIME | RECORD # | TEMP1 | TEMP2 | TEMP3 |
|---|---|---|---|---|
| 2002-01-27 10:00:00 | 14 | 23.5 | 24.6 | 28.2 |
| 2002-01-27 11:00:00 | 15 | 24.2 | 22.4 | 23.4 |

Only the hourly data triggered by the Instruction 84 above would be written to this table. If other table data instructions existed, the output for these tables would be written to their own tables.

Data tables can also be event driven rather than interval driven. That is, a new record is stored when a specified event occurs rather than based on time.

Each table is completely independent of any other tables and all records in a given table have the same number of fields.

# B.4 Default Tables

Each table-based datalogger has a set of default tables plus the tables created by the datalogger program. The four default tables in the CR10X-TD family of dataloggers, are Timeset, Errorlog, Inlocs, and Status. The default tables in CR1000, CR6 Series, CR300 Series, CR3000, and CR800 Series dataloggers

are Status, Public, and DataTableInfo. The default tables in CR5000, CR9000, and CR200 dataloggers are Status and Public.

- **Timeset Table** – The Timeset table contains a history of clock sets for the datalogger. It includes three fields: TimeStamp, RecordNumber, and OldTime. TimeStamp is the time and date the clock was set. RecordNumber is incremented each time the clock is set. When the datalogger is reset or a new program is loaded, RecordNumber is reset to 1. OldTime is the datalogger's clock value before the time was set (CR10X-TD family dataloggers only).

- **Errorlog Table** – The Errorlog table contains any errors that occur in the datalogger. It includes three fields: TimeStamp, RecordNumber, and ErrorCode. TimeStamp is the time and date the error occurred. RecordNumber is incremented each time an error occurs. When the datalogger is reset or a new program is loaded, RecordNumber is reset to 1. ErrorCode is the code returned by the datalogger when an error occurs. Refer to the datalogger user's manual for a list of all error codes (CR10X-TD family dataloggers only).

- **Inlocs Table** (CR10X-TD family dataloggers) or **Public Table** (CR-x000 dataloggers) – When a datalogger measures a sensor, the sensor reading is stored in a temporary register called an input location or variable. With each new measurement, the old value is overwritten by the new value. The Inlocs or Public table contains a time stamp, record number, flag status, port status, and the reading from each sensor scanned or user created input locations.

- **DataTableInfo Table** – The DataTableInfo table contains information about each data table in the datalogger including the table name, the number of skipped records for the table, the number of records in the table, the output interval of the table, and the time in days to fill the table.

- **Status Table** – The Status table contains information on the datalogger. Data is written to the table with each datalogger program execution. Note that the actual fields contained in the table are datalogger-specific. TABLE B-1 below describes typical fields that are given in the Status table. Not all fields will be present or applicable for all dataloggers. See the datalogger operator's manual for specifics.

| TABLE B-1. Example of Status Table Entries | |
|---|---|
| **TMStamp** | Date and time the status information was recorded. |
| **RecNBR** | The record number in the table. |
| **Battery** | Datalogger battery voltage. |
| **Watchdog** | The watchdog checks the processor state, software timers, and program related counters. If an error occurs, the watchdog counter is incremented. |
| **Overruns** | A table overrun occurs when the datalogger has insufficient time between execution intervals to complete one pass through the program. This counter is incremented with each table overrun. |
| **InLocs** | Number of input locations allocated for the program. |
| **PrgmFree** | Amount of remaining program memory, in bytes. |
| **Storage** | Number of final storage locations available. |
| **Tables** | Number of user-created data tables. |
| **DaysFull** | Estimated number of days of data the tables using automatic record allocation can hold. (NOTE: this number is only based on tables stored at intervals. Automatically allocating an event based table will often result in very small interval tables.) |
| **Holes** | Number of missed records in all data storage tables. |
| **PrgmSig** | Signature of the datalogger program. The signature is a unique number derived from the size and format of the datalogger program. If this signature changes, the program has been altered. |
| **PromSig** | Signature of the datalogger PROM. As with the PrgmSig, if this signature changes, the datalogger instruction set has somehow been changed. |
| **PromID** | Version number of the datalogger PROM. |
| **ObjSrlNo** | Revision number of the datalogger PROM. |
| **ROMVersion** | Version of the ROM code. This value is stored in the ROM and read by the OS at compile time. |
| **OSVersion** | Current version of the operating system. |
| **OSItem** | The CSI item number for the operating system. |
| **OSDate** | Date that the Operating System was compiled. |
| **StationName** | String stored as the Station Name of the CR5000. |
| **ProgName** | The Name of the currently running program. |
| **StartTime** | Time that the program began running. |
| **Battery** | Current value of the battery voltage. This measurement is made in the background calibration. |
| **PanelTemp** | Current Panel temperature measurement. |
| **LithiumBattery** | A Boolean variable signaling “True” (-1) if the lithium battery is OK and “False” (0) if not. The lithium battery is loaded and a comparator checked every 4 seconds to verify that the battery is charged. |

| TABLE B-1. Example of Status Table Entries | |
|---|---|
| **CPUSignature** | The Operating System signature. The value should match the value obtained by running the CSI sig program on the *name*.obj operating system file. |
| **DLDSignature** | Signature of the current running program file. |
| **ProgSignature** | Signature of the compiled binary data structure for the current program. This value is independent of comments added or nonfunctional changes to the program file. |
| **PC-CardBytesFree** | Gives the number of bytes free on the PC-Card. |
| **MemoryFree** | Amount (in bytes) of unallocated memory on the CPU (SRAM). The user may not be able to allocate all of free memory for data tables as final storage must be contiguous. As memory is allocated and freed there may be holes that are unusable for final storage, but that will show up as free bytes. |
| **DLDBytesFree** | Amount of free space in the CPU RAM disk that is used to store program files. |
| **ProcessTime** | Time in microseconds that it took to run through processing on 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. |
| **MaxProcTime** | The maximum time required to run through processing for the current scan. This value is reset when the scan exits. |
| **MeasureTime** | The time required by the hardware to make the measurements in this scan. The sum of all integration times and settling times. Processing will occur concurrent with this time so the sum of measure time and process time is not the time required in the scan instruction. |
| **SkippedScan** | Number of skipped scans that have occurred while running the current program. |
| **SlowProcTime** | Time required to process the current slow scan. If the user has slow scans then this variable becomes an array with a value for the system slow scan and each of the user's scans. |
| **MaxSlowProcTime** | The maximum Time required to process the current slow scan. If the user has slow scans then this variable becomes an array with a value for the system slow scan and each of the user's scans. |
| **LastSlowScan** | The last time that this slow scan executed. If the user has slow scans then this variable becomes an array with a value for the system slow scan and each of the user's scans. |
| **SkippedSlowScan** | The number of scans that have been skipped in this slow sequence. If the user has slow scans then this variable becomes an array with a value for the system slow scan and each of the user's scans. |
| **MeasureOps** | This is the number of task sequencer opcodes required to do all measurements in the system. This value includes the Calibration opcodes (compile time) and the system slow sequence opcodes. |

| TABLE B-1. Example of Status Table Entries | |
|---|---|
| **WatchdogErrors** | The number of Watchdog errors that have occurred while running this program. This value can be reset from the keyboard by going to status and scrolling down to the variable and pressing the DEL key. It is also reset upon compiling a new program. |
| **Low12VCount** | Keeps a running count of the number of occurrences of the 12VLow signal being asserted. When this condition is detected the logger ceases making measurements and goes into a low power mode until the system voltage is up to a safe level. |
| **StartUpCode** | A code variable that allows the user to know how the system woke up from poweroff. |
| **CommActive** | A variable signaling whether or not communications is currently active (increments each time the autobaud detect code is executed). |
| **ProgErrors** | The number of compile (or runtime) errors for the current program. |
| **ErrorCalib** | A counter that is incremented each time a bad calibration value is measured. The value is discarded (not included in the filter update) and this variable is incremented. |
| **VarOutOfBound** | Flags whether a variable array was accessed out of bounds. |
| **SkippedRecord** | Variable that tells how many records have been skipped for a given table. Each table has its own entry in this array. |
| **SecsPerRecord** | Output interval for a given table. Each table has its own entry in this array. |
| **SrlNbr** | Machine specific serial number. Stored in FLASH memory. |
| **Rev** | Hardware revision number. Stored in FLASH memory. |
| **CalVolts** | Factory calibration numbers. This array contains twenty values corresponding to the 20 integration / range combinations. These numbers are loaded by the Factory Calibration and are stored in FLASH. |
| **CalGain** | Calibration table Gain values. Each integration / range combination has a gain associated with it. These numbers are updated by the background slow sequence if the running program uses the integration / range. |
| **CalSeOffset** | Calibration table single ended offset values. Each integration / range combination has a single ended offset associated with it. These numbers are updated by the background slow sequence if the running program uses the integration / range. |
| **CalDiffOffset** | Calibration table differential offset values. Each integration / range combination has a differential offset associated with it. These numbers are updated by the background slow sequence if the running program uses the integration / range. |
| **CardStatus** | Contains a string with the most recent card status information. |
| **CompileResults** | Contains any error messages that were generated by compilation or during run time. |

# Appendix C. Log Files and the LogTool Application

## C.1 Event Logging

As PC400 performs its work, it will create records of various kinds of events and store them in ASCII log files. These logs can be very useful for troubleshooting problems and monitoring the operation of the datalogger network. You can monitor these logs using a built-in tool, called LogTool, accessible from the Tools | LogTool menu item, or open these log files in a simple text editor.

Most users will not need to understand these logs, but if you request technical assistance, a Campbell Scientific application engineer may ask you to send them one or more of the logs.

### C.1.1 Log Categories

The PC400 server logs events in four different kinds of logs as follows:

**Transaction Status (TranX.log)** — This log file documents the state of the various transactions that occur between the PC400 server and devices in the datalogger network. This is the most readable of the logs and contains event messages that are meaningful to most users. Examples of these events are:

- Datalogger clock check/set
- Datalogger program downloads
- Data collection

The format and type of records in this log are strictly defined to make it possible for a software program to parse the log records.

**Communications Status (CommsX.log)** — This log file documents the quality of communications in the datalogger network.

**Object State (StateX.log)** — This log file documents the state of an object. This is primarily for troubleshooting by software developers and the messages are relatively free in form.

**Low Level I/O (IOXSerial Port_1.log)** — A low level log file is associated with each root device in the datalogger network to record incoming and outgoing communications. While the entire network can be monitored from a single messaging session of the transaction, communications status, or object state logs, monitoring of the low-level log is performed on a session with the root device for that log.

You can monitor the logs with a special application called LogTool, accessed from the Tools | LogTool menu item in PC400. By default, PC400 stores five historical files of each type of log, each file 1.4 MB in size. After five files have been created, the oldest log file is deleted as a new one begun. PC400

stores the most recent log records in a file that has a $ character in the place of the version number or "X" in the above file names. When this file grows to the point that it will exceed the threshold set by the File Size setting for that log (default 1.4MB in PC400), the server renames the log file by replacing the dollar sign with a new version number. At the same time that the server rolls over to a new log file, the File Count parameter for that log will also be evaluated. If there are more saved files for that log than are allowed by the File Count parameter (default is five files in PC400), the server will delete the oldest of these files until the count is less than or equal to the File Count.

## C.1.2 Log File Message Formats

### C.1.2.1 General File Format Information

The communications status, transaction, and object state logs all share the same basic file format. Each record in a log file ends with a carriage return and line feed. A single record will consist of two or more fields where each field is surrounded by quotation marks and separated by commas.

The two fields that will be present in all records are:

**Timestamp** – The server time when the record was generated. It will have the following format:

YYYY-MM-DD HH:MM:SS.mmm

where "YYYY" is the 4-digit year, "MM" is the month number, "DD" is the day of the month, "HH" is the hour in the day (24 hour format), "MM" is the minutes into the hour, "SS" is the seconds into the minute, and "mmm" is the milliseconds into the second.

**Device Name** – The name of the device associated with the message. If the message is associated with the PC400 server, this will be an empty string.

### C.1.2.2 Transaction Log Format

Each record in the transaction log includes at least two fields in addition to the timestamp and device name:

**Message Type Code** – Identifies the type of event that has occurred. This is a number that corresponds to the description immediately following. If this log is being read by a software program, a number is very easy to use for comparison when looking for specific message types.

**Message Type Description** – Text that describes the message type code.

The following table is a list of the different messages that can appear in the transaction log, some of the optional parameters and what the message means. Where appropriate, a suggested response to the message is provided.

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 1 | Network device added | Device Name | A new device was added to the network map. | |
| 2 | Network branch deleted | Device Name | A branch of the network map was deleted (this may consist of a single device) | |
| 3 | Network branch moved | Device Name | A branch of the network map was moved from one parent device to another | |
| 5 | Network logon succeeded | Logon Name | A client application successfully attached to the server | |
| 6 | Network logon failed | Logon Name | A client application failed to attach to the server | If unsuccessful logon messages occur frequently, use a network monitor to determine who is trying to connect. If security is enabled this message will appear for someone trying to connect with the wrong user name or password. |
| 10 | Modem custom database read failed | | When the server started up it could not read the user customized modem settings file wmodem.cust. | If the user has not set up custom modem configurations, this file will not exist. |
| 11 | Clock check started | | A clock check has been initiated. This clock check is not sent out to the station until the transaction is sent. | |
| 12 | Clock set | Device time before set; Server time; | The device clock has been set. | |
| 13 | Clock checked | Datalogger time | The datalogger clock has been checked. | |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 14 | Clock check failed | Reason code:<br>3. Communication failure<br>4. Invalid datalogger security clearance<br>5. Invalid transaction number specified (already in use)<br>6. Communications are disabled for this device<br>7. The transaction was aborted by client request<br>8. The device is busy with another transaction | The clock check/set failed for the reason specified in the reason code. | Check the connections of the communication path to the datalogger, make sure the datalogger is connected and has power, check the security setting in the datalogger and in Setup, check that communications are enabled in Setup for all the devices in the path. |
| 26 | Data polling started | | Data collection by polling started. | |
| 27 | Data polling complete | | Data collection by polling completed | |
| 28 | Data polling failed | | Data collection by polling failed due to communication failure or a timeout. | Check communications with the datalogger by trying to check the clock. If that fails, follow the steps for message 14. |
| 29 | Directed data query start | | A user initiated query has been started. | |
| 30 | Directed data query continue | | The requested data in the directed query could not fit in one block and the next part is being requested. | |
| 31 | Directed data query complete | | The user requested data has been received by the server. | |
| 32 | Directed data query failed | | The directed query request failed. | |
| 33 | Getting logger table definitions | | The server is getting the table definitions from the datalogger. | Getting the datalogger table definitions will erase any data in the data cache. |
| 34 | Received logger table definitions | | The server has received the datalogger table definitions. | |
| 35 | Failed to get logger table definitions | | The request to get table definitions has failed. | |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 36 | Logger table definitions have changed | | The server has detected a change in the table definitions in the datalogger. | A change in table definitions indicates that the datalogger program may have changed. Before updating table definitions make sure the needed data in the data cache has been saved to a file if desired. |
| 40 | Datalogger message | Severity (S for Status, W for Warning, F for Fault); Message text. | This is a message that has been generated by the datalogger (or in some cases the RF base on behalf of the datalogger). | Datalogger warning and fault messages should be investigated using the datalogger operator's manual or contacting an applications engineer at Campbell Scientific. |
| 41 | Records received | Table name; Beginning record number; Ending record number | Datalogger records have been received and stored in the data cache. | |
| 42 | A datalogger transaction has timed out | Time out period in milliseconds | The server has waited longer than the allotted time for the expected response to a transaction. | Determine the reason for the timeout. This is usually due to a problem with the communications path between the PC and the datalogger. |
| 43 | Terminal emulation transaction started | | Terminal emulation message has been sent to the datalogger. | |
| 44 | Terminal emulation transaction complete | | Terminal emulation response message has been received from the datalogger. | |
| 45 | Terminal emulation transaction failed | | The expected terminal emulation response from the datalogger was not received. | |
| 46 | Set variable started | | The message to set an input location, flag or port has been sent to the datalogger. | |
| 47 | Set variable complete | | The datalogger has acknowledged the set of an input location, flag or port. | |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 48 | Set variable failed | | The datalogger failed to acknowledge the set variable message. | |
| 50 | Program file send start | | The server is sending a program to the datalogger. The actual program segments will appear as BMP1 message type 4. | |
| 51 | Program file send status | | The datalogger has received the program segment. | |
| 52 | Program file send complete | | The datalogger has compiled the program. | |
| 53 | Program file send failed | | The datalogger did not acknowledge the receipt of the program, the program did not compile, or communications failed with the datalogger. | If the program did not compile check the error messages. Otherwise, check communications with the datalogger by trying to check the clock. If that fails, follow the steps for message 14. |
| 54 | Program file receive start | | The server is requesting the datalogger program. The actual program segments will appears as BMP1 message type 5. | |
| 55 | Program file receive status | | A program segment has been received. | |
| 56 | Program file receive complete | | The datalogger program has been received from the datalogger. | |
| 57 | Program file receive failed | | The datalogger failed to send the program or communications with the datalogger failed. | Check communications with the datalogger by trying to check the clock. If that fails, follow the steps for message 14. |
| 64 | Device restore from file succeeded | | On server startup a device previously entered in the network map has been restored. | |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 65 | Device restore from file failed | | On server startup a device in the network map could not be restored. | This is an indication that the configuration file has been corrupted. Check the network map and the computer file system. |
| 66 | Device save to file succeeded | | The update to the device configuration file was successful. | |
| 67 | Device save to file failed | | The update to the device configuration file failed. | This may be due to a problem with directory permissions or a corrupted directory. |
| 68 | Packet delivery failed | Fault code:<br>1. Incompatible BMP1 device or malformed packet<br>2. Routing failure {unrecognized station number}<br>3. Temporarily out of resources<br>4. Link failure | This is a message from the RF base indicating that a BMP1 message didn't make it to the data logger. | Codes 1 and 3 are rare. If ever seen contact an application engineer at Campbell Scientific.<br>Code 2 indicates that the RF base has lost the network map and doesn't know how to route the message. The server automatically resends the network map.<br>Code 4 is an indication that the RF base was not able to communicate with the RF modem attached to the datalogger. These will happen occasionally as part of normal operations. Frequent occurrences indicate that the radio, antenna, connectors and RF link be reviewed. |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 69 | Unexpected change in datalogger table definitions | | As part of data collection the server has detected a change in the datalogger's table definitions. | A change in table definitions indicates that the datalogger program may have changed. This will suspend data collection and warnings will be shown in the Status Monitor. Data Collection can only be restored by updating table definitions. Before updating table definitions make sure the needed data in the data cache has been saved to a file if desired. |
| 70 | A device setting value has changed | Setting Identifier; Client's logon name; New value of the setting | A client has changed one of the device configuration settings. | |
| 71 | A LgrNet setting value has changed | Setting Identifier; Client's logon name; | A client has changed one of the server configuration settings. | |
| 72 | Client defined message | Client defined message | These messages are placed in the transaction log by client applications. The message should indicate which client entered the message. | |
| 73 | Socket listen failed | | Indicates an error in the computer system that prevents the server from listening for client connections on a socket. | This is a rare error and results in a problem with the computer operating system. If rebooting the computer does not clear the error, contact an application engineer. |
| 74 | Device renamed | | The name of a device in the network was changed. | |
| 75 | Logger locked | | This message indicates the start of a transaction such as terminal emulation that will tie up the datalogger preventing other operations. | |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 76 | Logger unlocked | | The transaction blocking datalogger access has completed. | |
| 77 | Null program sent | | The server has sent a null program to get an older datalogger (CR7X or 21X) out of keyboard emulation mode. | |
| 78 | Server started | The server version | The server has been started. | |
| 79 | Server shut down | | The server is being shut down | If a new "server started" message is seen without the shutdown message before it, this is an indication that the server or the PC crashed without exiting properly. |
| 80 | Collect area initialized | Collect area name | A data cache collect area has been created. | |
| 82 | Collect area removed | | A data cache collect area has been removed | |
| 83 | LgrNet restore failed | | On server startup the network description file, csilgrnet.dnd, could not be read. | The network setup and configuration will have to be restored from a backup or re-entered. Try to determine what corrupted or removed the network description file. |
| 85 | Data restore failed | | On server startup the data broker data storage area could not be created. | This is a computer problem. The files are either not present or are corrupted. See notes for message 83. |
| 86 | Manual poll transaction started | Client logon name | The listed client is starting a manual poll operation according to the scheduled collection settings. A manual poll is initiated from the "Collect Now" button on the Connect screen. | |
| 87 | Manual poll transaction complete | | The manual poll operation has received the data from the datalogger. | |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 88 | Manual poll aborted | | The manual poll operation was stopped or failed to complete due to communications failure or a timeout. | Check communications with the datalogger by trying to check the clock. If that fails, follow the steps for message 14. |
| 89 | Selective manual poll begun | Collect area name | A user specified poll has been started for one of the datalogger collect areas. | |
| 90 | Selective manual poll complete | Collect area name | The user specified manual poll has completed. | |
| 91 | Selective manual poll aborted | Collect area name | The user specified manual poll failed. | Check communications with the datalogger by trying to check the clock. If that fails, follow the steps for message 14. |
| 92 | Polling started on collect area | Collect area name | Data has been requested for the specified collect area. This message is always associated with another message indicating whether this is scheduled, manual or selective manual polling. | Collect areas can be table for table mode dataloggers, final storage areas, ports and flags, or input locations. |
| 93 | Collect area poll data | Collect area name | Data has been received from an array based datalogger for the specified collect area. | |
| 94 | Collect area polling complete | Collect area name | Data collection for the specified collect area has successfully completed. | |
| 95 | Collect area polling failed | Collect area name | Data collection for the specified collect area failed. | Check communications with the datalogger by trying to check the clock. If that fails, follow the steps for message 14. |
| 99 | Collect area first poll | | This message is posted either the first time data is collected for a collect area, or holes were lost for the datalogger. | If this is not the first poll for the collect area, this message indicates that data that had been stored in the datalogger was lost before it could be collected. |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 100 | Table mount failed | Table name; Operating system information regarding the failure | The server was not able to create a data collection area from the stored table configuration file or new table definitions. This could be the result of trying to create table files that are too large for the computer system. | Check the computer operating system integrity. Verify that the PC400 system configuration files exist and the directory has not been corrupted. |
| 101 | Add record failed | Table name; Beginning record number; End record number; A reason for the failure | The server was not able to write data records to the data storage area. | This indicates a problem writing to files on the computer hard disk. Verify write permissions are set and that there is sufficient space left on the disk. |
| 102 | Collect area skipped warning | Collect area name | The specified collect area was skipped because the associated table has not been initialized by the server yet. | During system startup this is a normal message. If it occurs at other times contact an application engineer. |
| 103 | Collect area skipped error | Collect area name | The specified collect area was skipped because the server could not initialize the associated table. | See message 100 |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 104 | BMP1 packet sent | The packet message type code:<br>0 Packet Delivery Fault Notification<br>1 Status/Warning/Fault Notification<br>2 Network Description Transaction<br>3 Clock Check/Set Transaction<br>4 Program Down-load Transaction<br>5 Program Up-load Transaction<br>7 Data Advise Command Transaction<br>8 Data Advise Notification Packet<br>9 Hole Collection Command Transaction<br>10 Control Command (Set Variable) Transaction<br>11 User I/O Transaction (Terminal Mode)<br>12 Memory Image Down-load Transaction<br>13 Memory Image Up-load Transaction<br>14 Get Table Definitions Transaction<br>15 RF Test Transaction<br>16 Communication Status Notification | The specified BMP1 packet was sent to the serial communication interface. The number specifies the type of message that was sent. | |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 105 | BMP1 packet received | The packet message type code:<br>0 Packet Delivery Fault Notification<br>1 Status/Warning/Fault Notification<br>2 Network Description Transaction<br>3 Clock Check/Set Transaction<br>4 Program Down-load Transaction<br>5 Program Up-load Transaction<br>7 Data Advise Command Transaction<br>8 Data Advise Notification Packet<br>9 Hole Collection Command Transaction<br>10 Control Command (Set Variable) Transaction<br>11 User I/O Transaction (Terminal Mode)<br>12 Memory Image Down-load Transaction<br>13 Memory Image Up-load Transaction<br>14 Get Table Definitions Transaction<br>15 RF Test Transaction<br>16 Communication Status Notification | The specified BMP1 packet was received over the serial communications link. The number indicates the type of message received. | |
| 106 | Data file output failed | | Data collected from a datalogger could not be written to the data output file. | Check that there is space available on the hard disk and that write permissions allow the server to write the data output files. |
| 107 | Max time on-line exceeded | The amount of time the device was connected, in milliseconds | A client kept the communication link on-line longer than the specified max time on-line. | |
| 108 | Table reset | The name of the table that was reset; The account name of the logged in client | The name of a table was changed at the request of a client. On CR1000, CR5000, and CR9000 loggers this is a reset for the table in the datalogger and on the PC. | |

**TABLE C-1. Transaction Log Messages**

| Code | Message Text | Message Parameters | Message Meaning | User Response to Message |
|---|---|---|---|---|
| 110 | Collect area setting changed | The name of the collection area; The setting identifier for the setting that was changed; The new value of the setting; The account name of the logged in client. | One of the settings for the specified collect area was changed. The identifiers for the setting can be found in CoraScript help. | |
| 111 | PakBus route added | | A new PakBus route has been added to the routing table. | |
| 112 | PakBus route lost | | A PakBus route has been lost and will be removed from the routing table. | |
| 113 | PakBus station added | | A new PakBus station was added to the network. | |
| 117 | Client logged off | The login name of the client; The reason the session was closed. | A client application has closed or lost the connection to the server. | |
| 118 | Table size reduced during creation | The name of the table that was resized; The original specified size of the table; The new size of the table. | The size of the table in the data cache was reduced because there was not enough computer disk space to create it, or the file would have exceeded the 2 Gbyte size limit. | Reduce the size of the tables in the datalogger program or get more hard disk storage space for the computer. |

**Transaction Log Example**

```
"2009-04-15 16:41:05.367","CR1000","11","Clock check started"
"2009-04-15 16:41:05.429","CR1000","13","Clock checked","2009-04-15 16:41:33.44","2009-04-15 16:41:05.421","-28"
"2009-04-15 16:41:06.367","CR1000","86","Manual poll started","admin"
"2009-04-15 16:41:06.367","CR1000","92","Collect area poll started","TestFast"
"2009-04-15 16:41:06.382","CR1000","41","Records received","TestFast","21007","21007","polling"
"2009-04-15 16:41:06.382","CR1000","20","Hole detected","TestFast","20769","21006"
"2009-04-15 16:41:06.429","CR1000","11","Clock check started"
"2009-04-15 16:41:06.492","CR1000","41","Records received","TestFast","20769","20799","polling"
"2009-04-15 16:41:06.507","CR1000","144","data file opened","TestFast","C:\Campbellsci\LoggerNet\CR1000_TestFast.dat"
"2009-04-15 16:41:06.507","CR1000","21","Hole collected","TestFast","20769","20799"
"2009-04-15 16:41:06.507","CR1000","41","Records received","TestFast","20800","20864","polling"
"2009-04-15 16:41:06.507","CR1000","21","Hole collected","TestFast","20800","20864"
"2009-04-15 16:41:06.523","CR1000","13","Clock checked","2009-04-15 16:41:34.55","2009-04-15 16:41:06.516","-28"
"2009-04-15 16:41:06.601","CR1000","41","Records received","TestFast","20865","20899","polling"
"2009-04-15 16:41:06.601","CR1000","21","Hole collected","TestFast","20865","20899"
"2009-04-15 16:41:06.601","CR1000","41","Records received","TestFast","20900","20960","polling"
"2009-04-15 16:41:06.601","CR1000","21","Hole collected","TestFast","20900","20960"
"2009-04-15 16:41:06.648","CR1000","41","Records received","TestFast","20961","20999","polling"
"2009-04-15 16:41:06.648","CR1000","21","Hole collected","TestFast","20961","20999"
"2009-04-15 16:41:06.648","CR1000","41","Records received","TestFast","21000","21006","polling"
"2009-04-15 16:41:06.648","CR1000","21","Hole collected","TestFast","21000","21006"
"2009-04-15 16:41:06.679","CR1000","145","data file closed","TestFast","C:\Campbellsci\LoggerNet\CR1000_TestFast.dat"
"2009-04-15 16:41:06.679","CR1000","94","Collect area poll complete","TestFast","956","956"
"2009-04-15 16:41:06.695","CR1000","87","Manual poll complete"
"2009-04-15 16:41:07.429","CR1000","11","Clock check started"
"2009-04-15 16:41:07.445","CR1000","13","Clock checked","2009-04-15 16:41:35.46","2009-04-15 16:41:07.438","-28"
"2009-04-15 16:41:08.429","CR1000","11","Clock check started"
```

### C.1.2.3 Communications Status Log Format

Each record in the communications status log includes two fields in addition to the timestamp and device name:

**Severity** – A single character code that indicates the type of message. The following values are legal:

- "S" (Status) Indicates that the identified operation has successfully completed.
- "W" (Warning) Indicates that the server has attempted to retry the operation with the identified device.
- "F" (Fault) Indicates that the identified operation has failed and that the server has stopped retrying.

**Description** – text providing more details about the event.

**TABLE C-2. Communications Status Log Messages**

| Message Text | Message Meaning | User Response to Message |
|---|---|---|
| Serial packet X exchanged | The low level serial BMP1 communication framing packet was sent and the response received from the device. (CR10X-TD table based type devices) | |
| Classic;;Cmd | The listed command was sent to an array based datalogger. | For a list of the commands and their meanings see the datalogger operator's manual. |
| BMP1 packet received | A BMP1 packet was received from the device. (CR10X-TD type devices only) | |
| RPC packet exchanged | A BMP3 packet was exchanged. (CR5000, CR9000 dataloggers only) | |
| Datalogger did not respond to end command | The computer tried to terminate the connection but the datalogger did not acknowledge the shutdown. | This is an indication that there is a communications problem between the computer and the datalogger. Check the cables and connectors and make sure the datalogger has power. |
| PakBus framing error | PC400 received data from the link that cannot be verified to be part of a PakBus packet. | Some possible causes: the datalogger program or its settings are configured to write data on the port on which you are attempting to connect, there is a mismatch in baud rates between your computer serial port and the datalogger, the link is dropping sequences of characters from transmission because the CPU on the computer is heavily loaded or because of faulty USB drivers.<br><br>Possible solutions: change the port you are using to communicate, try using a slower baud rate to communicate with the datalogger, use Windows Task Manager to determine whether there are processes that are loading down your CPU, check to see if there is an updated driver for your USB/RS232 adapter. |
| Invalid low level signature | The packet received from the device got corrupted and the packet signature doesn't match the packet contents. | Check to find out where in the communications link noise or signal corruption is causing the data to be disrupted. |

**TABLE C-2. Communications Status Log Messages**

| Message Text | Message Meaning | User Response to Message |
|---|---|---|
| Provider opened | The serial communications port has been initialized. | |
| Device dialed | The communications link has been initialized to transfer data packets. | |
| Provider closed | The serial communications port has been closed. | |
| Unable to Locate Serial synch byte | The low level communications synchronization byte was not received after the computer sent out a serial packet. | This indicates that the device is either not responding or responding with an invalid communications protocol. This message would appear if trying to talk to an array based datalogger that is set up as a table based datalogger in the network map. |

**Communications Status Log Example**

```
"2009-04-15 16:41:05.367","IPPort","S","Device dialed"
"2009-04-15 16:41:05.382","PakBusPort_ip","S","sending message","src: 4094","dest: 2","proto: PakCtrl","type: 0x09","tran: 214"
"2009-04-15 16:41:05.398","PakBusPort_ip","S","received message","src: 2","dest: 4094","proto: PakCtrl","type: 0x89","tran:
214"
"2009-04-15 16:41:05.398","CR1000","S","PakCtrl message received","89"
"2009-04-15 16:41:05.413","PakBusPort_ip","S","sending message","src: 4094","dest: 2","proto: BMP5","type: 0x17","tran: 213"
"2009-04-15 16:41:05.429","PakBusPort_ip","S","received message","src: 2","dest: 4094","proto: BMP5","type: 0x97","tran: 213"
"2009-04-15 16:41:05.429","CR1000","S","BMP5 message received","type: 0x97","check/set clock"
"2009-04-15 16:41:06.367","PakBusPort_ip","S","sending message","src: 4094","dest: 2","proto: BMP5","type: 0x09","tran: 217"
"2009-04-15 16:41:06.382","PakBusPort_ip","S","received message","src: 2","dest: 4094","proto: BMP5","type: 0x89","tran: 217"
"2009-04-15 16:41:06.382","CR1000","S","BMP5 message received","type: 0x89","table poll","CR1000.TestFast"
"2009-04-15 16:41:06.382","PakBusPort_ip","S","sending message","src: 4094","dest: 2","proto: BMP5","type: 0x09","tran: 218"
"2009-04-15 16:41:06.492","PakBusPort_ip","S","received message","src: 2","dest: 4094","proto: BMP5","type: 0x89","tran: 218"
"2009-04-15 16:41:06.492","CR1000","S","BMP5 message received","type: 0x89","table poll","CR1000.TestFast"
"2009-04-15 16:41:06.523","PakBusPort_ip","S","sending message","src: 4094","dest: 2","proto: BMP5","type: 0x17","tran: 219"
"2009-04-15 16:41:06.523","PakBusPort_ip","S","received message","src: 2","dest: 4094","proto: BMP5","type: 0x97","tran: 219"
```

### C.1.2.4 Object State Log Format

The object state log includes two fields in addition to the timestamp and device name:

**Object Name** – The name of the object from which the message is being generated. Typically this will be the name of an object method.

**Description** – Any extra information associated with the event.

**Object State Log Example**

"2009-04-15 16:41:05.351","CR1000","starting BMP5 operation","manage comm resource"
"2009-04-15 16:41:05.367","CR1000","starting BMP5 operation","check/set clock"
"2009-04-15 16:41:05.367","PakBusPort_ip","Request Transaction Focus","check/set clock","213"
"2009-04-15 16:41:05.367","PakBusPort_ip","Transaction focus start","PakCtrl::Hello","2","214"
"2009-04-15 16:41:05.367","PakBusPort_ip","Dev::sesBegin","01100C90"
"2009-04-15 16:41:05.367","PakBusPort_ip","Dev::cmdAdd","MyPort::serial_framing_command","3"
"2009-04-15 16:41:05.367","IPPort","Dev::reqDevice","Requesting device: PakBusPort_ip"
"2009-04-15 16:41:05.367","IPPort","Dev::cmdFinished","Callback Command"
"2009-04-15 16:41:05.367","IPPort","Dev::sesEnd","016E83B0"
"2009-04-15 16:41:05.367","IPPort","DevHelpers::HangupDelaySession","Hangup delay: 10"
"2009-04-15 16:41:05.367","PakBusPort_ip","Dev::reqDevResp","IPPort","PakBusPort_ip","success"
"2009-04-15 16:41:05.367","PakBusPort_ip","Dev::sesBegin","016E83B0"
"2009-04-15 16:41:05.367","PakBusPort_ip","Dev","Going on-line"
"2009-04-15 16:41:05.367","PakBusPort_ip","Dev::onNextCommand","Executing command","MyPort::serial_framing_command","3"
"2009-04-15 16:41:05.367","PakBusPort_ip","Csi::PakBus::SerialPortBase::link_type","watch dog timeout set at 40000"
"2009-04-15 16:41:05.367","PakBusPort_ip","send_ring","remote: 2","retries: 0"
"2009-04-15 16:41:05.382","IPPort","DevHelpers::HangupDelaySession","post completion"
"2009-04-15 16:41:05.382","IPPort","Dev::sesEnd","0166DCA8"
"2009-04-15 16:41:05.382","IPPort","Dev","Hangup delay complete received, no sessions left"
"2009-04-15 16:41:05.382","PakBusPort_ip","arm transaction watchdog","PakCtrl::Hello","2","7250","37350"
"2009-04-15 16:41:05.382","CR1000","Bmp5::Datalogger","delay_hangup created"
"2009-04-15 16:41:05.382","PakBusPort_ip","Csi::PakBus::SerialPortBase::link_type","watch dog timeout set at 40000"
"2009-04-15 16:41:05.382","CR1000","starting BMP5 operation","delay hangup"
"2009-04-15 16:41:05.382","CR1000","Bmp5::OpDelayHangup","transaction started","216"
"2009-04-15 16:41:05.413","PakBusPort_ip","PakBusTran closing","PakCtrl::Hello","2","214"
"2009-04-15 16:41:05.413","PakBusPort_ip","Csi::PakBus::Router","entering close_transaction"
"2009-04-15 16:41:05.413","PakBusPort_ip","Release Transaction Focus","PakCtrl::Hello","2","214"
"2009-04-15 16:41:05.413","PakBusPort_ip","Transaction focus start","check/set clock","213"
"2009-04-15 16:41:05.413","PakBusPort_ip","arm transaction watchdog","check/set clock","11250","37355"
"2009-04-15 16:41:05.413","PakBusPort_ip","Csi::PakBus::SerialPortBase::link_type","watch dog timeout set at 40000"
"2009-04-15 16:41:05.413","PakBusPort_ip","Csi::PakBus::Router","leaving close_transaction"
"2009-04-15 16:41:05.445","PakBusPort_ip","PakBusTran release focus","check/set clock","37355"
"2009-04-15 16:41:05.445","PakBusPort_ip","Release Transaction Focus","check/set clock","213"
"2009-04-15 16:41:05.460","PakBusPort_ip","PakBusTran closing","check/set clock","213"
"2009-04-15 16:41:05.460","PakBusPort_ip","Csi::PakBus::Router","entering close_transaction"
"2009-04-15 16:41:05.460","PakBusPort_ip","Csi::PakBus::Router","leaving close_transaction"
"2009-04-15 16:41:06.367","CR1000","starting BMP5 operation","table poll","CR1000.TestFast"
"2009-04-15 16:41:06.367","PakBusPort_ip","Request Transaction Focus","table poll","CR1000.TestFast","217"
"2009-04-15 16:41:06.367","PakBusPort_ip","Transaction focus start","table poll","CR1000.TestFast","217"
"2009-04-15 16:41:06.367","PakBusPort_ip","arm transaction watchdog","table poll","CR1000.TestFast","7250","37361"
"2009-04-15 16:41:06.367","PakBusPort_ip","Csi::PakBus::SerialPortBase::link_type","watch dog timeout set at 40000"
"2009-04-15 16:41:06.382","CR1000","Bmp5::OpTablePoll::on_bmp5_message - check newest","table poll","CR1000.TestFast"
"2009-04-15 16:41:06.382","PakBusPort_ip","Release Transaction Focus","table poll","CR1000.TestFast","218"
"2009-04-15 16:41:06.382","CR1000","Bmp5::OpTablePoll::on_check_complete","table poll","CR1000.TestFast"
"2009-04-15 16:41:06.382","PakBusPort_ip","Request Transaction Focus","table poll","CR1000.TestFast","218"
"2009-04-15 16:41:06.382","PakBusPort_ip","Transaction focus start","table poll","CR1000.TestFast","218"
"2009-04-15 16:41:06.382","PakBusPort_ip","arm transaction watchdog","table poll","CR1000.TestFast","7250","37365"

## 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 Ltd.**
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.*