A Technical Reference Manual

for the

dataTaker 50,500,600 Series 3

Data Loggers

Warranty

DataTaker Pty Ltd warrants the instruments it manufactures against defects in materials or workmanship for a period of 3 year from the date of delivery to the original customer. This warranty is limited to the replacement or repair of such defects, without charge, when the instrument is returned to *dataTaker Pty Ltd*, or to one of its authorized dealers.

This warranty excludes all other warranties, either expressed or implied, and is limited to a value not exceeding the purchase price of the instrument. Where *dataTaker Pty Ltd* supplies equipment or items manufactured by a third party, then the warranty provided by the third party manufacturer remains.

DataTaker Pty Ltd shall not be liable for any incidental or consequential damages resulting from use of the instrument, or for damage to the instrument resulting from accident, abuse, improper implementation, lack of reasonable care, or loss of parts.

Warning

DataTaker Pty Ltd products are not authorized for use as critical components in a life support system where failure of the product is likely to effect its safety or effectiveness.

Trademarks

dataTaker is a trademark of *dataTaker Pty Ltd*
Windows is a trademark of Microsoft Corp
PC Card and PCMCIA are trademarks of the Personal Computer Memory Card Industry Association

Firmware Versions

This Technical Reference Manual applies to *Series 3 dataTaker 50, 500, 505, 515, 600, 605 and 615* model data loggers, which have firmware version 7.xx installed.

The earlier *Series 1* and *Series 2* of the *dataTaker* data loggers are very similar to the *Series 3* models, provided these loggers have the latest version firmware installed:

Series 1 Firmware Version 4.02
Series 2 Firmware Version 5.00

This manual can be used in support of earlier version data loggers, and differences can be ascertained by comparison with your original Getting Started and Concise Manuals.

Related Documents

Getting Started with *DT50, DT500 and DT600 Series dataTakers*
User's Manual *DT50, DT500 and DT600 Series dataTakers Series 3*
Advanced Communications Manual

Related Products

PMD-01 Panel Mount Display

MC-512, MC-1024, MC-2048 and MC-4096 SRAM Memory Cards

MCI-04 Memory Card Reader

SS-500 Sensor Simulation Panel

PE-500, SIE-500 and LIE-500 enclosures

DeTransfer and DePlot

DeLogger 4
DeLogger Pro 4

Visual Basic Terminal Suite

SMSX Modem

905U-D Radio Modem

Document Details

Document Number: UM-0078-A0-X0
Published: September 2003

Regulatory Compliances

The *Series 3 dataTaker 50, 500, 505, 515, 600, 605 and 615* model data loggers meet the following standards:

EUROPE CE	This product complies with the requirements of European Directives 89/336/EEC and 73/23/EEC, and conforms with EN55022 Class A (emissions) and EN50082-1 (susceptibility). Mains adaptors use to power this product must comply with EN60950, EN60742 or EN61010.
AUSTRALIA and NEW ZEALAND C-Tick	This product complies with the requirements of the Australian and New Zealand standard for EMC emissions AS/NZS 3548:1992 Class A. ACN 006 134 863
USA	This device complies with Part 15 of the FCC rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.
CANADA	This digital apparatus does not exceed the Class A limits for radio noise emissions from digital apparatus as set out in the Radio Interference Regulations of the Canadian Depertment of Communications. Le présent appareil numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la Classe A prescrites dan les réglements sur le brouillage radioélectrique édictés par le Ministére des Communications du Canada.
CAUTION – USE APPROVED ANTISTATIC PROCEDURES	The input and general circuitry of this device is extremely sensitive, and therefore susceptible to damage by static electricity. Always follow approved anti-static procedures when working with this device.
	

DataTaker Pty Ltd
7 Seismic Court,
Rowville, 3178, AUSTRALIA.

Tel: +61 3 9764 8600
Fax: +61 3 9764 8997
Web: www.datataker.com
Email: support@datataker.com.au

Contents

A Technical Reference Manual for the **dataTaker 50,500,600 Series 3** *Data Loggers* *1*

Contents *4*

Section I An Introduction to the **dataTaker 50,500,600 Series 3** *Data Loggers* *6*

Introduction to the *dataTaker* 7
Construction of the *dataTaker 50* 14
Construction of the *dataTaker 500/600* Series 19
Construction of the Channel Expansion Module 23
Getting Started 25

Section II Interfacing to the **dataTaker 50,500,600 Series 3** *Data Loggers* *40*

Interfacing to the *dataTaker* 41
Powering the *dataTaker* 46
Powering Sensors from the *dataTaker* 55
The Serial Interfaces 57
The RS232 COMMS Serial Interface 58
The NETWORK Interface 65
Analog Processing 68
Connecting Analog Input Signals 79
The Analog Input Channels 85
Measuring Low Level Voltages 95
Measuring High Level Voltages 102
Measuring Currents 114
Measuring 4–20 mA Current Loops 124
Measuring Resistance 138
Measuring Frequency and Period 149
Measuring Analog Logic State 156
Measuring Temperature 167
Measuring Temperature with Thermocouples 169
Measuring Temperature with RTDs 185
Measuring Temperature with Integrated Circuit Temperature Sensors 198
Measuring Temperature with Thermistors 220
Measuring Bridges and Strain Gauges 230
Measuring Vibrating Wire Strain Gauges 252
The Digital Input Channels 263
Monitoring Digital State 267
Low Speed Counters 270
The Phase Encoder Counter 274
High Speed Counters 277
The Digital Output Channels 280
The Channel Expansion Module 286
Installing the Panel Mount Display 293

Section III Programming the dataTaker 50,500,600 Series 3
Data Loggers 296
Programming the *dataTaker* 297
Communication Protocols and Commands 299
Entering Commands and Programs 304
Format of Returned Data 308
Specifying Channels 322
The Analog Input Channels 326
The Digital Input Channels 336
The Counter Channels 344
The Digital Output Channels 356
The Real Time Clock 364
The Internal Channels 371
Channel Options 380
Schedules 396
Immediate Schedule 399
Triggered Schedules 402
Schedule Triggered by Poll 405
Schedules Triggered by Time 408
Schedules Triggered by Digital Events 412
Schedules Triggered by Counter Events 415
Statistical Sub Schedule 418
Schedules Triggered While Condition 426
General Schedule Commands 428
Alarms 437
Scaling Data - Polynomials, Spans & Functions 467
Channel Variables and Calculations 480
Calculations 490
Histogram 497
Logging Data to Memory 499
Logging Data to the Internal Memory 509
Logging Data to Memory Cards 511
Returning Data from Memory 520
Programming from Memory Cards 529
STATUS Command 536
RESET Command 542
TEST Command 544
Switch Commands 549
Parameter Commands 556
dataTaker Networks 565
Writing *dataTaker* Programs 572
Keypad and Display Module 580
Warnings and Error Messages 591
Appendices 593
Appendix A - ASCII Character Table 594
Appendix B - ADC Sampling Speed 597
Index 603

Section I

An Introduction

to the

dataTaker 50,500,600 Series 3

Data Loggers

Introduction to the *dataTaker*

About This Manual

This Technical Reference Manual applies directly to Series 3 *dataTaker 50, 500, 505, 515, 600, 605* and *615* model data loggers, which have firmware version 7.xx installed. The firmware version is returned in the first line of the TEST command.

The earlier *Series 1* and *Series 2* of the *dataTaker* data loggers are all very similar to the *Series 3* models, provided that these loggers have the latest version firmware installed as follows:

Series 1 Firmware Version 4.02
Series 2 Firmware Version 5.00

Therefore this manual can also be used in support of these earlier version data loggers, and differences can be ascertained by comparison with your original Getting Started and Concise Manuals.

This manual is complimentary to the *Getting Started with DT50, DT500 and DT600 Series dataTaker* manual, and the *Users Manual DT50, DT500, DT600 Series dataTaker Series 3*, that are supplied with *Series 3* loggers. This manual covers all of the *dataTaker* data loggers including:

1. *dataTaker 50*
2. *dataTaker 500, dataTaker 505, dataTaker 515*
   - these data loggers will be referred to as the *dataTaker 500* series loggers in this manual
3. *dataTaker 600, dataTaker 605, dataTaker 615*
   - these data loggers will be referred to as the *dataTaker 600* series loggers in this manual
4. Channel Expansion Modules (CEM-AD)
   - for *dataTaker 500* series loggers and *dataTaker 600* series loggers
5. Panel Mount Display modules
   - for *dataTaker 50* and *dataTaker 500* series loggers

Wherever *dataTaker* is used solely in this manual without reference to a model number, then the information pertains to all of the *dataTaker* models.

Since there are many common features for the *dataTaker* models *500*, *505*, *515*, *600*, *605* & *615*, these models will be referred to throughout this manual as the *'dataTaker 500/600* series loggers', unless otherwise stated.

Introduction to the *dataTaker*

The *dataTaker* data loggers are microprocessor based intelligent data acquisition instruments able to monitor, record (log) data, and raise alarms for a wide variety of physical parameters including temperature, pressure, flow, counts, events, etc.

The key features of the *dataTaker* are accuracy, flexibility, low power consumption and optionally a large and removable data storage memory.

The *dataTaker* can be supervised from any host computer using companion software such as DeLogger or DeTransfer, or any generic terminal program such as Windows HyperTerminal or other freeware, shareware or commercial terminal programs. The *dataTaker* accepts instructions in ASCII text issued from the computer to configure the logger, to specify data acquisition, to control return and logging of data, to define testing and annunciation for alarms, to manage the digital output channels, etc. Analog and digital input signals are converted to data in ASCII text, suitable for reading directly into the host computer via the RS232 serial interface.

The *dataTaker* is intelligent such that when the logger has been programmed from the host computer, it is capable of operating off line acquiring data into memory for later recovery, all the while monitoring alarms and performing simple control functions.

The *dataTaker 50* and *dataTaker 500/600* series loggers share many common features and capabilities. The fundamental differences between these models are the number of channels supported, the ability to operate in a network, mechanical construction and the ability to support the optional Channel Expansion Modules.

The *dataTaker 50* has 5 differential (double ended) or 10 single ended analog input channels, 5 digital input/output channels, and 3 high speed counters. The *dataTaker 50* does not support the Channel Expansion Module.

In contrast the *dataTaker 500/600* series loggers have 10 differential or 30 single ended analog input channels, 4 digital input/output channels, and 3 high speed counters. The *dataTaker 500/600* series loggers support the Channel Expansion Module.

The *dataTaker 600* series loggers also have a built in display. The *dataTaker 50* and *dataTaker 500* series loggers have the option of a separate Panel Mount Display module.

The analog to digital converter of the *dataTaker* data loggers has 15 bit plus sign bit resolution. This high degree of resolution allows the *dataTaker* to monitor very low level signals such as thermocouples and strain gauges with high accuracy and resolution.

Linearization and compensation for standard sensors such as thermocouples, RTDs, thermistors and solid state temperature sensors is automatically performed whenever these sensor types are monitored. Commands are also provided to allow user defined linearization for most sensor types that may be used with the *dataTaker*.

The *dataTaker* is supplied as a free standing module. The module is suitable for use directly on any laboratory bench, and can be mounted into virtually any type of enclosure suitable to the application.

Wall mounting industrial quality steel enclosures rated to the IP55 standard are available as optional enclosures. A weatherproof carry case rated to IP67 standard is also available for those applications where portability or remote siting is required.

The principal features and operation of the *dataTaker* data loggers is introduced below, and described in detail in the following chapters.

The *dataTaker* Features

The major features of the *dataTaker* data loggers and CEM·AD are summarized in the table below:

	DT50	***DT500***	***DT505***	***DT515***	***DT600***	***DT605***	***DT615***	***CEM-AD***
Analog Inputs Diff/SE	5/10	10/30	10/30	10/30	10/30	10/30	10/30	10/30
CMOS Multiplexed	•	•			•			
Relay Multiplexed			•	•		•	•	•
Vibrating Wire Support				•			•	
Digital I/O - State	5	4	4	4	4	4	4	20/10
Low Speed Counters	5	4	4	4	4	4	4	
High Speed Counters	3	3	3	3	3	3	3	
RS232 COMMS	•	•	•	•	•	•	•	
Max. COMMS baud rate	9600	9600	9600	9600	9600	9600	9600	
RS232 electrically isolated	•	•	•	•	•	•	•	
Can be networked		•	•	•	•	•	•	
Internal 1.2 AHr battery		•	•	•	•	•	•	
Built in display					•	•	•	
Panel Mount Display	Option	Option	Option	Option				
CEM Support		•	•	•	•	•	•	

Communicating with the *dataTaker*

The *dataTaker* is operated in conjunction with a host computer, which is used to enter commands and programs to the logger, to process the data returned after input channels are scanned, to respond to alarms, and to directly manage output channels. The computer can either be running companion software such as DeLogger or DeTransfer, or any generic terminal program such as Windows HyperTerminal or other freeware, shareware or commercial terminal program.

The *dataTaker* has an RS232 serial interface for communicating with the host computer, either directly or via modems, and for outputting data to printers. This RS232 serial interface of the *dataTaker* is referred to as the RS232 COMMS interface.

All communications between the host computer and the *dataTaker* is in standard ASCII characters. There are no special communications formats or protocols used that require specific management by the host computer. However a protocol mode can be selected for noisy communication links.

The baud rate of the RS232 COMMS serial interface is set via DIP switches to 300, 1200, 2400, 4800 or 9600 baud all *dataTaker* loggers. The number of data bits and stop bits, and parity checking are permanently set to 8 data bits, 1 stop bit and no parity.

The communications link between the *dataTaker* and the host computer may be direct for distances of up to 100 metres (110 yards), or via simple line drivers for distances of up to 5000 metres (5500 yards), or via modems, radio and satellite links for longer distances.

The *dataTaker 500/600* series loggers also have an RS485 serial interface for communications between loggers in a network. Up to 32 *dataTaker 500/600* series loggers can be connected into a local network.

The RS485 serial interface of the *dataTaker 500/600* series loggers is referred to as the NETWORK interface. Network communication via the NETWORK interface is permanently set to 1200 baud, and is fully error checked.

Modes of Operation and Networking

The *dataTaker* may be operated in one of a number of different modes, depending on the availability of computer time, the data volume, and the location and number of sensors.

The most common mode of operation is where the *dataTaker* is periodically supervised by a permanently connected computer. Scanning of input channels and data collection can be progressively varied, the resultant data can be managed by the computer in real time, active alarms can be responded to directly in a reactive or corrective manner, and digital output channels can be managed directly. DeLogger and DeTransfer/DePlot are suited to this mode of operation where acquired data is to be progressively displayed, graphed and archived.

The alternative mode of operation is where the *dataTaker* is initially programmed from a temporarily connected computer, and the logger is then left to scan input channels, log the data into the internal memory or memory card, and monitor and annunciate alarms, independently of the host computer. The data can be recovered from memory when the host computer is reconnected at a later time. DeLogger and DeTransfer/DePlot are also suited to this mode of operation.

The *dataTaker 500/600* series loggers can also be used in a mode where up to 32 *dataTaker*s are connected into a local or widely deployed network. The individual *dataTaker*s within the network are linked via the RS485 NETWORK serial interfaces, and the host computer is connected to the RS232 COMMS serial interface of any logger in the network. This mode of operation generally requires a permanently connected host computer to supervise the individual loggers.

Other serial devices such as other computers, terminals, printers, or data storage devices may be connected to the RS232 COMMS serial interface of other loggers in the network, and accessed by the host via the network.

Supervising the dataTaker

Supervision of the *dataTaker* by the host computer may be periodic, partial or total, depending on the application, the volume of data acquired, and the management of alarms.

Commands which configure the *dataTaker*, specify the acquisition of data, define the testing and annunciation of alarms, manage the digital output channels, etc. are all entered directly from a host computer. Data resulting from the scanning of input channels can be returned to the computer as the channels are scanned.

However once the *dataTaker* is programmed, the computer can be disconnected and the logger left to operate independently. The scan data can be stored in the internal data memory of the *dataTaker*, or into a memory card, and returned to the host computer at a later time. The host computer can be disconnected during the period of data logging, and the data can be recovered when the host computer is later reconnected.

Host Computer Software

The software requirement for the host computer depends on the nature of the task to be performed.

The simplest form of host computer software is any terminal program that supports an RS232 port of the computer, and can be used to supervise the *dataTaker* by sending commands as they are typed, and displays and stores raw data as it is received from the logger. A terminal program does not support particular management of data, responses to alarms, real time commanding of the logger, etc.

The Windows HyperTerminal program, or any freeware, shareware or commercial terminal program can be used to supervise a *dataTaker* logger in an emergency if proprietary software is unavailable.

DeTransfer software has been developed specifically for the *dataTaker.* DeTransfer is essentially a text editor and a terminal, used for creating and sending programs to the *dataTaker*, and for receiving, displaying and storing raw data from the *dataTaker*. The DePlot software is used to plot data collected by DeTransfer either in real time or from memory. DeTransfer and DePlot are usually supplied with the *dataTaker* data loggers.

When a computer is used as the host, then a variety of software packages can be used to supervise the *dataTaker*, to manage and present the data, to respond to alarms, and to directly manage digital output channels, etc.

By contrast, the DeLogger software supplied with the *dataTaker* loggers provides for much more supervision of the logger, including graphical programming interface, displaying data as raw data, as charts, as meters, and as tables or forms, responding to alarm states, storing data, etc. In addition applications developed in DeLogger can be automated to make repetitive use simpler. Consult your DeLogger manuals for details of use of this software.

There are also some special purpose application software packages that can be used for supervising the *dataTaker*, provided appropriate drivers are available. These include National Instrument's LabView, CI Technologies' CiTech, DasyLab, Adroit, etc.

Because all data is returned from the *dataTaker* in standard ASCII character text, the data can also be imported into text editors, word processors, spreadsheets, graphics packages, etc.

dataTaker Input and Output Channels

The *dataTaker 50* has 5 differential (double ended) analog input channels. These input channels can also be used as 10 single ended analog input channels, or in any mix of differential and single ended channels.

The *dataTaker 500/600* series loggers have 10 differential analog input channels. These input channels can also be used as 30 single ended analog input channels, or in any mix of differential and single ended channels.

The analog to digital converter (ADC) of the *dataTaker* recalibrates automatically, based on drift of the internal zero reference voltage. Autocalibration corrects for errors due to circuit aging, and to temperature drifts in the measurement circuits.

The ADC of the *dataTaker* is also auto-ranging, and automatically changes range as follows

6. the *dataTaker 50, 500* and *600* all have a solid state multiplexer for the analog input channels, and autorange over three range decades of ±25.000 mV, ±250.00 mV and ±2500.0 mV.

7. the *dataTaker 505, 515, 605* and *615* all have a relay multiplexer for the analog input channels, and autorange over three low range decades of ±25.000 mV, ±250.00 mV, and ±2500.0 mV when the internal attenuators are deselected, and autorange over three high range decades of ±7.000 V, ± 70.00 V and
   ±100.0 Volts when the internal attenuators are selected.

The conversion ranges for analog input signals do not need to be specified when programming the logger. Only the channel numbers which are to be scanned, the type of input on each channel (voltage, current, frequency, resistance, thermocouple, RTD, etc.) and the interval between scans, needs to be specified in scan commands.

The *dataTaker 50* also has 5 bi-directional digital channels, which can be used as digital inputs or digital outputs. The *dataTaker 500/600* series loggers have 4 bi-directional digital channels, which can be used as digital inputs or digital outputs.

The digital input channels of the *dataTaker* can be used to monitor logic states or contact closures, for low speed counting and for event detection.

The digital output channels of the *dataTaker* are open collector TTL / CMOS compatible logic outputs, which can drive relays and logic circuits directly.

All of the *dataTaker*s also have 3 high speed counters, capable of operating at 1 KHz or 1 MHz.

Channel Expansion Modules (CEM-AD)

The channel capacity of the *dataTaker 500/600* series loggers can be expanded by adding Channel Expansion Modules (CEM-AD). The *dataTaker 500/600* series loggers can support 1 or 2 Channel Expansion Modules (CEM-AD) connected in a daisy chain with a total cable length not exceeding 2 metres (6 feet) in length.

The Channel Expansion Modules (CEM-AD) have 10 differential analog input channels, which can also be used as 30 single ended analog input channels, or in any mix. The Channel Expansion Module (CEM-AD) has the same input channel specification and sensor support as the *dataTaker* to which it is connected.

The Channel Expansion Modules (CEM-AD) also have 20 digital input channels for state monitoring, and 10 digital outputs configured as 5 relay outputs and 5 open collector TTL/CMOS compatible outputs. The Channel Expansion Module (CEM-AD) does not support counting.

Reading Input Channels

The scanning of the analog and digital input channels can be at regular intervals of time from 1 second to months, or in response to poll requests by the host computer, or in response to events occurring on the digital input channels or high speed counters.

Up to 7 scan commands for scanning input channels can operate simultaneously. Input channels can be scanned such that all data is returned, or scanned over a period of time and the average, standard deviation, minimum, maximum and integral data only is returned for the period.

The input channel data may be returned either in terms of the base electrical units of the transducers being monitored, or in terms of the actual physical quantities which the transducers are sensing. The *dataTaker* has inbuilt linearization for thermocouples, RTDs, solid state temperature sensors, thermistors and strain gauges, and has facilities for declaring calibration or linearization functions for other transducer types.

All signal input and output lines connect to the *dataTaker* and Channel Expansion Module (CEM-AD) via removable screw terminals.

Real Time Clock

The *dataTaker* incorporates a real time clock which keeps the time of day in HH:MM:SS, decimal hours and seconds formats, and keeps the date as day number in DDDDD format, or date in DD/MM/YY, MM/DD/YY and decimal day formats.

The *dataTaker* real time clock has a resolution of 1 second, and an accuracy of 2 seconds per day. The clock is battery backed, and will maintain date and time when the *dataTaker* is powered off.

The real time clock is used for many purposes including to initiate repeated time based scanning of input channels, for raising alarms as a function of time, and to timestamp and datestamp data from the input channels scanned.

There are also 4 internal system timers for use with the alarms functions.

Data Logging

The *dataTaker* has internal battery backed SRAM data memory, which can store approximately 166,500 readings (Series 1 and Series 2 loggers have smaller memory capacity). All data is logged with full precision over the range of ±1.0e-18 to ±1.0e18. Data is logged serially in the data memory, which may be defined to function either as a stop when full stack, or as a circular buffer where the newest data overwrites the oldest data after memory fills.

The data logging capacity of the *dataTaker* can be increased using plug in PC-Card or PCMCIA standard SRAM memory cards. These memory cards can be removed from the *dataTaker* at any time, and the data logged onto the card can be unloaded elsewhere either using another *dataTaker*, or a *dataTaker* MCI-04 Memory Card Reader, or by inserting the memory card into the PC-Card slot of some Notebook computers (this will depend on support provided by the computer manufacturer, and the particular version of Windows software used). While the memory card is removed, current data logging continues into the internal data memory.

The memory cards are available with the following memory capacities

8. 512 Kbyte, stores approx 173,750 readings
9. 1 Mbyte, stores approx 347,500 readings
10. 2 Mbyte, stores approx 695,000 readings
11. 4 Mbyte, stores approx 1,390,000 readings

Stored data is unloaded from the internal data memory and memory cards in the order of oldest data first.

Programming the *dataTaker*

The *dataTaker* is programmed using simple English-like commands. The individual commands are combined together to form command lines to program the data logger for particular tasks.

The different commands provide for

12. scanning of up to five different groups of selected analog and digital input channels
13. scanning a group of input channels in response to poll requests by the host computer
14. scanning up to four groups of input channels, each at a different time interval, or in response to digital events or counter events
15. conditionally scanning groups of input channels while nominated digital input channels are high or true
16. determining the average, the standard deviation, the minimum, the maximum and the integral readings for selected groups of input channels
17. inspection of input channels at any time without disrupting other scan commands in progress
18. performing calculations on the raw data from input channels
19. returning acquired data to the host computer in real time
20. storing acquired data in the internal data memory or memory cards, in stack or circular buffer mode
21. returning data from the data storage memory or memory cards
22. defining the format of returned data

23. setting the state of digital output channels

24. defining alarm conditions and the method for annunciating or reporting those alarms.

25. setting system variables, channel variables and general system configuration.

26. local viewing of acquired data and some system information, and executing predefined tasks, from the Display Keypad module (where fitted).

The various commands for programming the *dataTaker* are briefly discussed in context in Section II of this manual, and are discussed in detail in Section III of this manual.

Construction of the *dataTaker 50*

The *dataTaker 50* is constructed on two printed circuit boards, which are mounted into a modular chassis.

The Processor printed circuit board is located in the bottom of the *dataTaker 50* chassis. The Processor printed circuit board comprises the

- microcomputer, system logic and system timing circuitry
- system software in ROM memory
- system and data storage RAM memory, and interface for memory cards
- analog to digital converter circuitry

The Signal I/O printed circuit board is located under the top cover of the chassis, and provides all of the interfacing to the logger on its surface. The Signal I/O printed circuit board comprises the

- the RS232 COMMS serial interface
- the analog input channels, and multiplexer
- digital input/output channels
- high speed counter channels
- power supply, and battery charger for a lead acid gel cell battery
- 12 pin connector for a connecting a Panel Mount Display module to the logger
- a regulated 5 Volt sensor power supply
- ground and external reference terminals
- the Wake terminal for externally waking the *dataTaker 50* from low power or sleep mode

An 8 way DIP switch is also located on the Signal I/O printed circuit board, beside the memory card socket. The DIP switch provides for

- setting the baud rate for the RS232 COMMS serial interface
- setting an address for the logger
- setting the local mains frequency

The top cover of the *dataTaker 50* must be removed to access this DIP switch.

The function block diagram for the *dataTaker 50* is illustrated in Figure 1, which also indicates on which of the printed circuit boards the various sub systems reside.

All signal input and output connections to the *dataTaker 50* are made via removable screw terminals (Series 3) or fixed screw terminals (Series 1 and 2) that are arranged on the upper surface of the data logger. The input and output channel numbers are labelled adjacent to the screw terminals.

The RS232 COMMS serial interface is implemented as a female 9 pin D type connector, located along the upper edge of the data logger.

Screw terminals for the external power supply connection and the battery connection are also located along the upper edge of the logger.

Figure 2 illustrates the layout of all terminals and connectors for the *dataTaker 50*.

Multiplexer

The *dataTaker 50* has a solid state multiplexer for selecting analog input channels. The multiplexer uses CMOS 4052 quad analog gate devices.

The 4052 devices have a maximum input voltage of approximately 15 Volts, and will withstand a common mode voltage of up to ±3 Volts.

The multiplexer devices also provide input protection for the *dataTaker 50* against excessive input voltages, because the 4052 devices will generally fuse to open circuit when damaged. These devices are socketed for easy replacement.

The *dataTaker 50* data logger has three decade voltage input ranges of ±25.000 mV, ±250.00mV and ±2500.0 mV.

Enclosures

The *dataTaker 50* is normally supplied in the basic format as a free standing module, and can be used in this format in work rooms, laboratories and other clean areas.

Alternatively the *dataTaker 50* can be mounted into enclosures to suit the particular application, eg. heavy duty industrial enclosures, weatherproof enclosures, portable carry cases, etc.

A variety of enclosures are available from your *dataTaker* supplier, for industrial, weatherproof and portable applications.

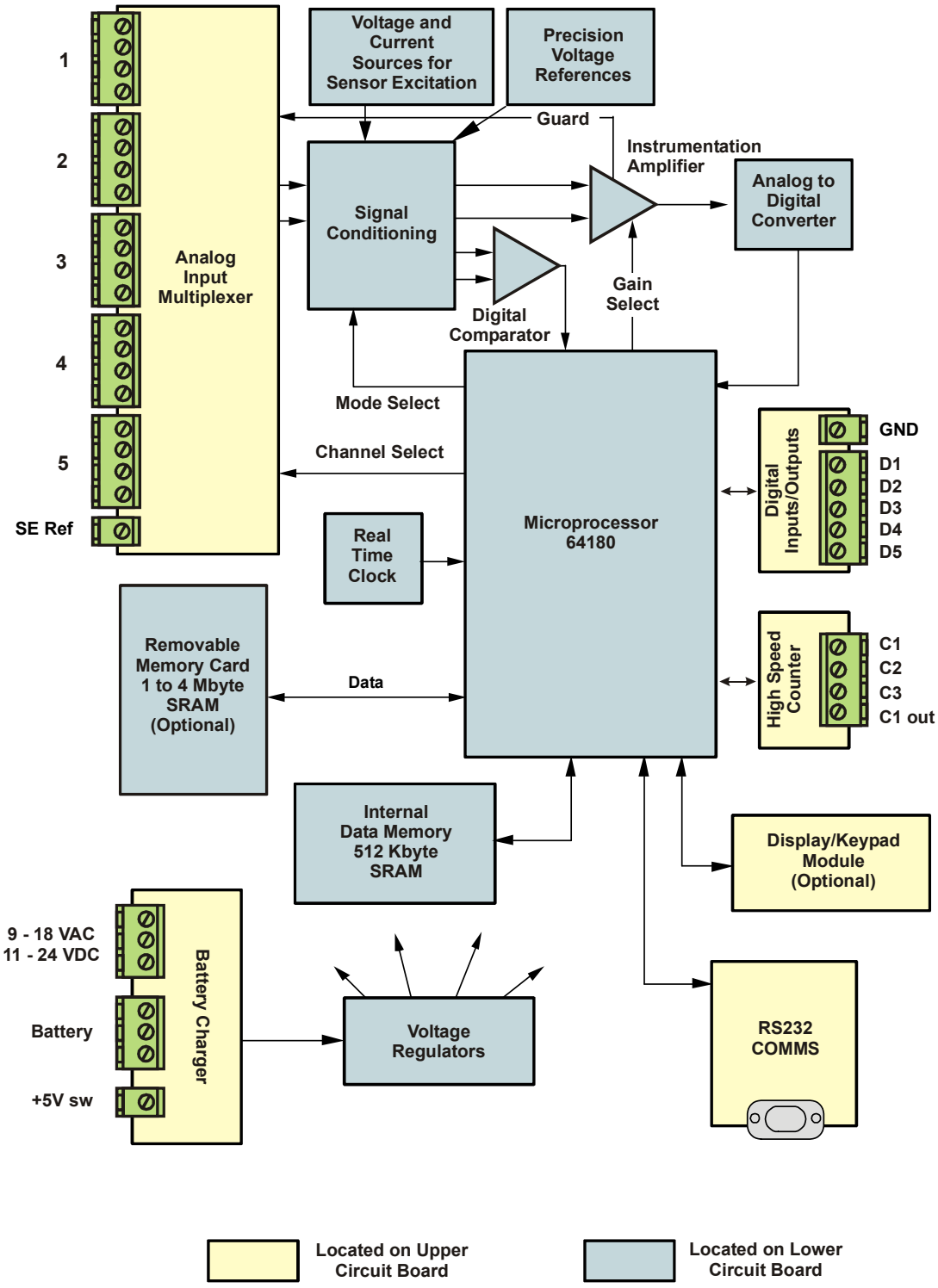
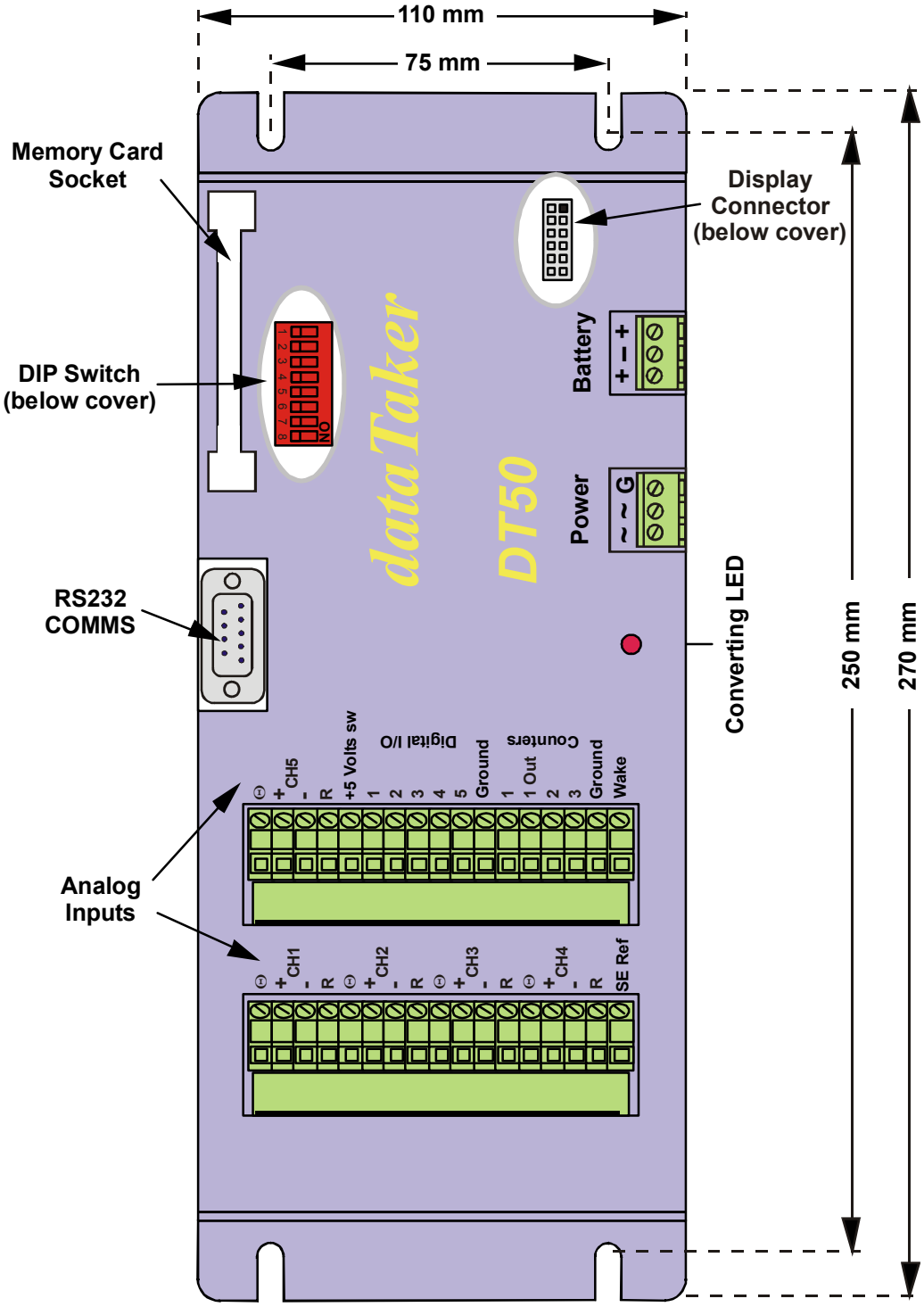


Figure 1 – *dataTaker 50* Function Block Diagram



Height - 50mm without memory card
105mm with memory card

Figure 2 – *dataTaker 50* Screw Terminal and Connector Layout.

Construction of the *dataTaker 500/600* Series

The *dataTaker 500/600* series loggers are constructed on two printed circuit boards, which are mounted into a modular chassis which also houses the internal battery.

The *dataTaker 600* series also have a Display/Keypad printed circuit board fitted above the battery tray.

The Processor printed circuit board is located in the bottom of the *dataTaker 500/600* chassis. The Processor printed circuit board comprises the

microcomputer, system logic and system timing circuitry

system software in ROM memory

system and data storage RAM memory, and interface for memory cards

analog to digital converter circuitry

The Signal I/O printed circuit board is located immediately above the lower chassis of the data logger, and provides all interfacing to the data logger along its edges. The Signal I/O printed circuit board comprises the

the RS232 COMMS serial interface

the RS485 NETWORK serial interface

the analog input channels and multiplexer
- *dataTaker 500 & 600* have solid state (CMOS) multiplexers
- *dataTaker 505, 515, 605 & 615* have relay multiplexers

digital input/output channels

high speed counter channels

power supply and battery charger for the internal lead acid gel cell battery, and any external batteries.

12 pin connector for a connecting a Panel Mount Display module to *dataTaker 500* series loggers

a regulated 5 Volt sensor power supply

ground and external reference terminals

the Wake terminal for externally waking the *dataTaker 500/600* logger from the low power or sleep mode

a 25 pin plug for connection of Channel Expansion Modules.

An 8 way DIP switch is also located on the Signal I/O printed circuit board, beside the memory card socket. The DIP switch provides for

setting the baud rate for the RS232 COMMS serial interface

setting the network address of the logger

setting the local mains frequency

The top cover of the *dataTaker 500/600* must be removed to access this DIP switch.

The function block diagram for the *dataTaker 500/600* series loggers is illustrated in Figure 3, and indicates on which of the printed circuit boards the various sub systems reside.

All signal input and output connections to the *dataTaker 500/600* loggers are made via removable screw terminals (Series 3) or fixed screw terminals (Series 1 and 2) that are arranged along the edges of the data logger. The input and output channel numbers are labelled adjacent to the screw terminals

Screw terminals for the power supply input and RS485 NETWORK serial interface are located along the rear edge of the Signal I/O printed circuit board.

The RS232 COMMS serial interface is implemented as a female 9 pin D type connector, located along the rear edge of the Signal I/O printed circuit board.

Figure 4 illustrates the layout of all terminals and connectors for the *dataTaker 500/600* series loggers.

Multiplexer Types

The *dataTaker 500/600* series loggers have 2 different types of analog input channel multiplexer, solid state (CMOS) multiplexer or relay multiplexer.

The major difference between these multiplexer types is the common mode voltage and the voltage ranges that the *dataTaker*s with each multiplexer type can measure directly.

Solid State Multiplexer

The *dataTaker 500* and *dataTaker 600* data loggers have a solid state multiplexer for selecting analog input channels. The multiplexer uses CMOS 4052 quad analog gate devices.

The 4052 devices have a maximum input voltage of approximately 15 Volts, and will withstand a common mode voltage of up to ±3 Volts.

The multiplexer devices provide input protection for the *dataTaker 500* and *dataTaker 600* against excessive input voltages, because the 4052 devices will generally fuse to open circuit when damaged. These devices are socketed for easy replacement.

The *dataTaker*s that have the solid state multiplexer have three decades of voltage input ranges of ±25.000 mV, ±250.00mV and ±2500.0 mV.

Relay Multiplexer

The *dataTaker 505, 515, 605* and *615* data loggers all have a relay multiplexer, which uses high precision electromechanical relays.

The *dataTaker*s which have the relay multiplexer also have internal input attenuation which provides two categories of voltage input range as follows

when the internal attenuators are not selected, there are three decades of low voltage input ranges of ±25.000 mV, ±250.00mV and ±2500.0 mV

when the internal attenuators are selected, there are three decades of high voltage input ranges of ±7.000 V, ±70.00 V and ±100.0 V

The input attenuation, and hence voltage measuring ranges, are selected under program control.

The relay multiplexer has high withstanding input voltages when open contact as follows

1.5 KVolts for 10 µS

500 Volts for 50 mS

100 Volts indefinitely

Enclosures

The *dataTaker 500/600* series loggers are normally supplied in the basic format as a free standing module, and can be used in this format in work rooms, laboratories and other clean areas.

Alternatively the *dataTaker 500/600* series loggers can be mounted into enclosures to suit the particular application, eg. heavy duty industrial enclosures, weatherproof enclosures, portable carry cases, etc.

A variety of enclosures are available from your *dataTaker* supplier, for industrial, weatherproof and portable applications.

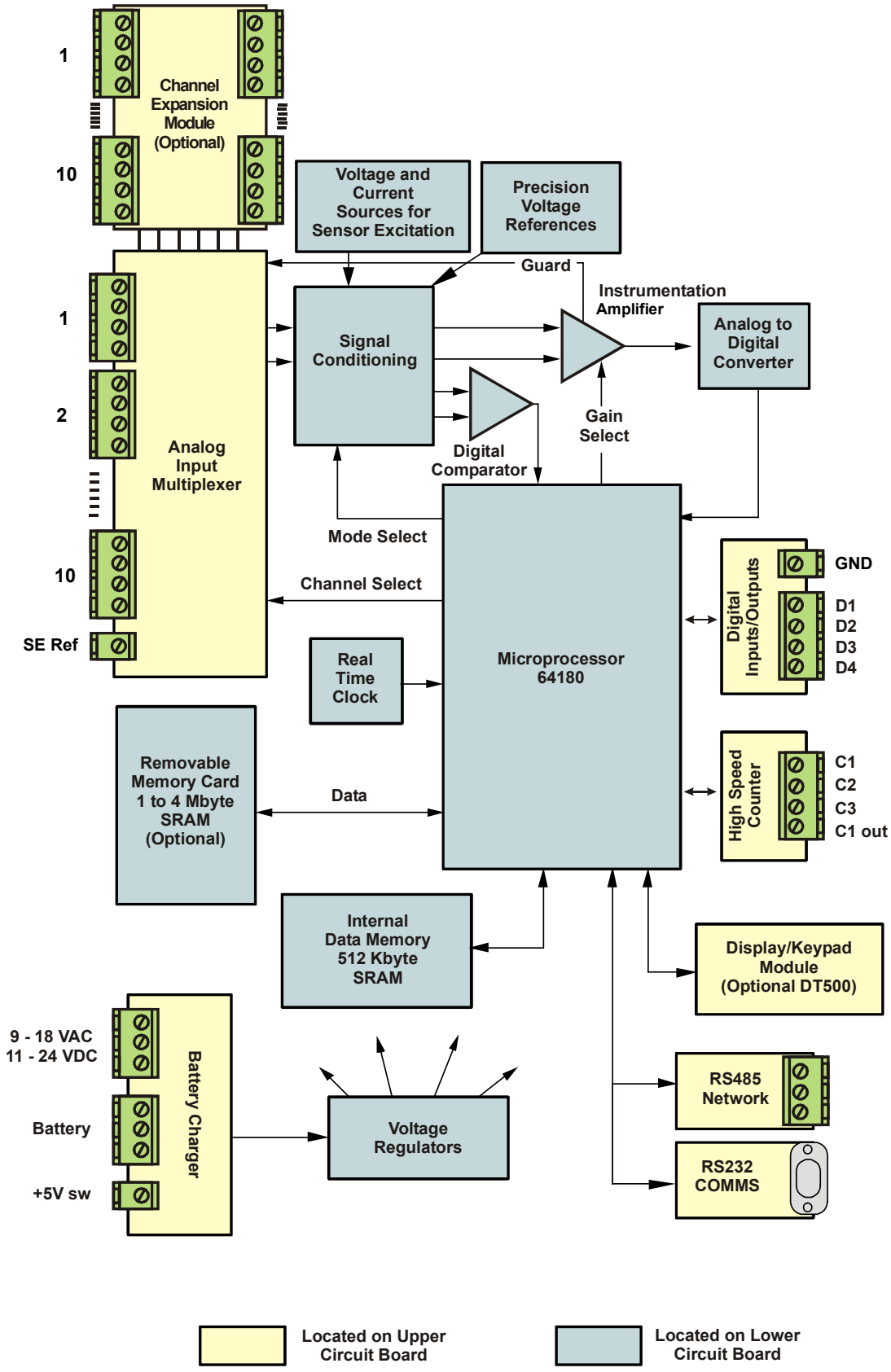
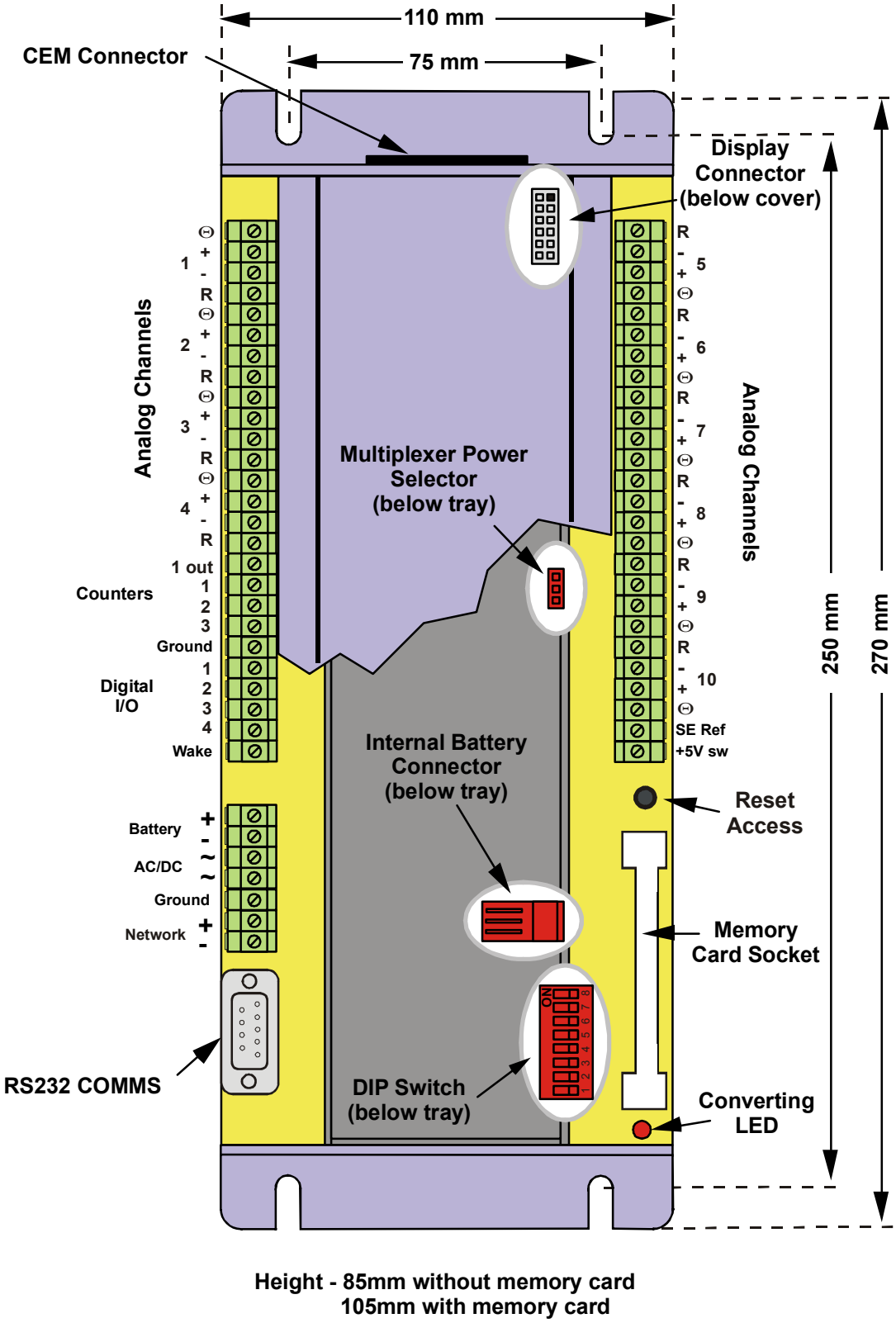


Figure 3 – *dataTaker 500/600* Series Loggers Function Block Diagram



Height - 85mm without memory card
105mm with memory card

Figure 4 – *dataTaker 500/600* Series Loggers Screw Terminal and Connector Layout.

Construction of the Channel Expansion Module

The Channel Expansion Module (CEM-AD) is constructed on one printed circuit board, which is mounted into a module chassis. This chassis has the same footprint size as the *dataTaker* data logger modules.

Note: The *dataTaker 50* does not support the Channel Expansion Modules.

The Channel Expansion Module (CEM-AD) printed circuit board comprises

1. the analog input channels, and relay multiplexers
2. digital input/output channels
3. a regulated 5 volt sensor power supply
4. ground and external reference terminals
5. A male 25 pin plug for connection to the *dataTaker* 500/600 series loggers
6. A female 25 pin socket for connection to other Channel Expansion Modules

All signal input and output connections of the Channel Expansion Module (CEM-AD) are made by screw terminals arranged along the edges. The input and output channel numbers are labelled adjacent to the screw terminals.

Figure 5 illustrates the layout of all terminals and connectors of the Channel Expansion Module (CEM-AD).

Enclosures

The Channel Expansion Module (CEM-AD) is normally supplied in the basic format as a free standing module, and can be used in this format in work rooms, laboratories and other clean areas.

Alternatively the Channel Expansion Module can be mounted with the *dataTaker* data logger into enclosures to suit the particular application, eg. heavy duty industrial enclosures, weatherproof enclosures, portable carry cases, etc.

A variety of enclosures are available from your *dataTaker* supplier, for industrial, weatherproof and portable applications.

The Large Industrial Enclosure (LIE) will house 1 *dataTaker* and 1 Channel Expansion Module, and the Small Industrial Cabinet (SIC) will house 1 *dataTaker* and 2 Channel Expansion Modules.

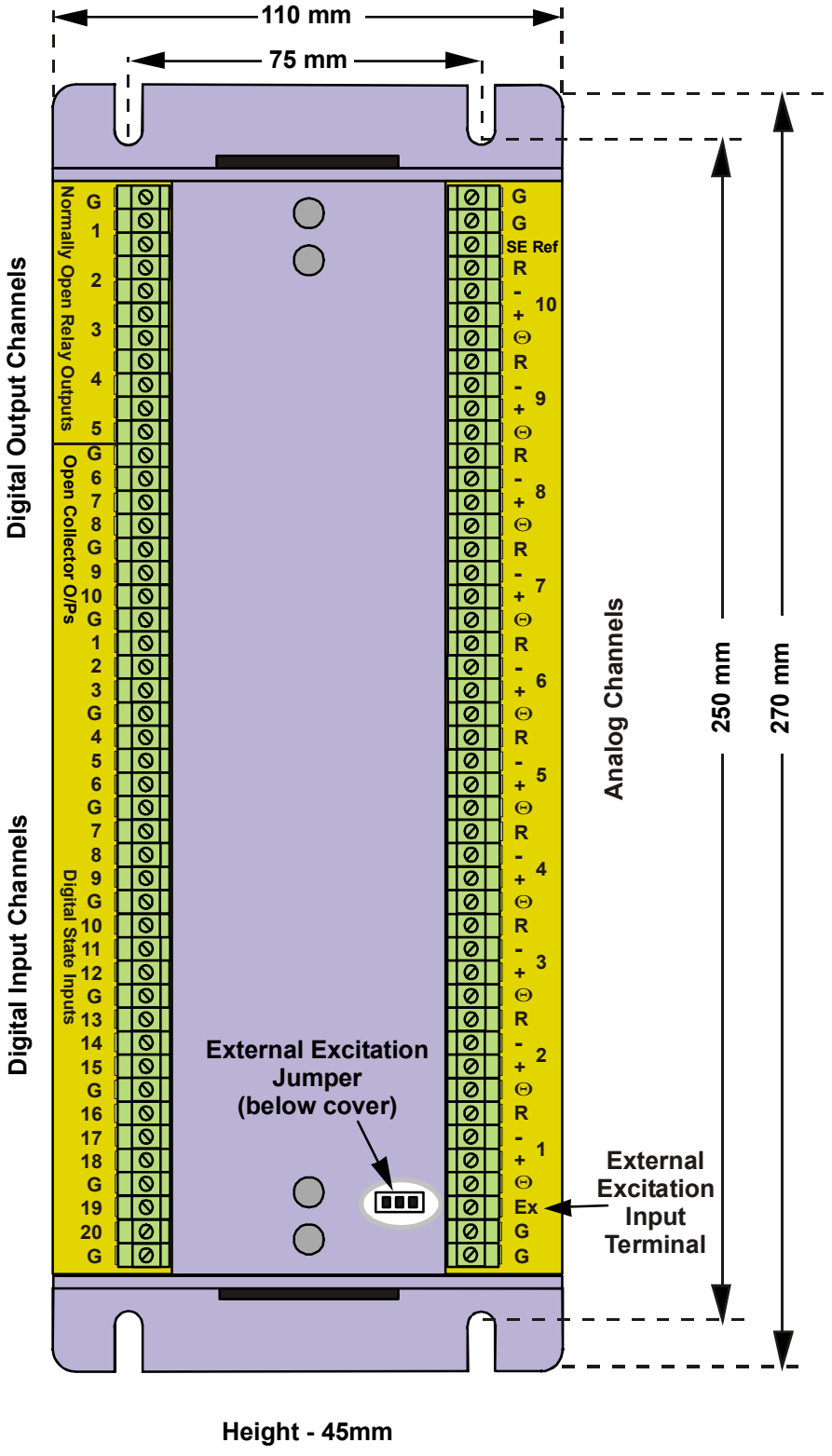


Figure 5 - Channel Expansion Module (CEM-AD) Screw Terminal and Connector Layout.

Getting Started

This chapter describes the simple step by step procedure for initially preparing the *dataTaker* data loggers for use with your host computer. The chapter also provides some introductory programming details to perform some simple initial tasks.

The *dataTaker* is supplied with a number of standard accessories. These should be checked when the logger is originally unpacked.

dataTaker 50

12 volt mains/line power adaptor

RS232 host communications cable

Screw driver for signal I/O terminals

Getting Started with *DT50*, *DT500* and *DT600* Series *dataTaker* Manual

User's Manual *DT50*, *DT500* and *DT600* Series *dataTaker Series 3*

dataTaker Software Suite CD

There are a number of optional accessories which are also available for your *dataTaker* data logger including

Wall mounting industrial enclosures rated to IP55 or NEMA 4

Waterproof portable enclosure rated to IP67

Sensor Simulator Panel

PC-Card SRAM Memory Cards – 512Kbyte, 1Mbyte, 2Mbyte, 4Mbyte capacity

1.2 AHr and 4 AHr gel cell batteries

17 AHr alkaline battery

Display/Keypad Module for *dataTaker 50* and *dataTaker 500* series

Channel Expansion Module for *dataTaker 500/600* series loggers

This Technical Reference Manual

Standard accessories and the optional accessories can be obtained either from your *dataTaker* supplier.

Installing the *dataTaker*

The *dataTaker* is normally supplied as a free standing module. The logger can optionally be supplied in an industrial quality wall mounting steel enclosure, or in a weatherproof portable enclosure.

The industrial enclosures are rated to IP55 or NEMA 4, which is described as both shower proof and dust proof. This enclosure is constructed of pressed steel, and can be mounted directly onto walls and panels. The enclosure is suitable for most industrial applications, particularly where permanent installation of the *dataTaker* is required.

The weatherproof portable enclosure is waterproof (IP67), and is rated to 10 metres (33 feet) in water. This enclosure has a clamshell design, and is constructed of moulded ABS plastic. The portable enclosure is suitable for outdoor and portable applications.

The *dataTaker* must be installed in a location where ambient temperature does not exceed –20 to 70 Deg C, and the humidity does not exceed 80% and is non condensing. If the *dataTaker* is likely to be subjected to low temperatures, then a desiccant should be included in the enclosure to reduce internal humidity.

Where signal cable runs are close to mains/line power or high voltage cables, or other sources of electrical noise, then the signal cables should preferably be shielded and the shield grounded. This is particularly important for low level signals.

Open Installation of the dataTaker

The free standing modular *dataTaker* can be located onto any bench surface, or mounted into user supplied enclosures.

The location for the data logger must be dry, relatively free of dusts and corrosive vapours, and not subject to temperature extremes.

The *dataTaker* can be mounted in any orientation, to suit the signal I/O cabling.

Protected Installation of the dataTaker

The optional IP55 or NEMA 4 wall mounting steel enclosures allow the *dataTaker* to be installed into dusty locations, and locations subject to intermittent water showers or sprays.

There are 3 industrial quality steel enclosures available. The outside dimensions of these enclosures are as follows

Small Industrial Enclosure (SIE)

- length 400 mm (15.75 inches)
- width 200 mm (7.85 inches)
- depth 120 mm (4.72 inches)
- weight 4.5 Kg (9.9 lbs)

Large Industrial Enclosure (LIE)

- length 300 mm (11.81 inches)
- width 380 mm (14.96 inches)
- depth 155 mm (6.10 inches)
- weight 7.0 Kg (15.4 lbs)

Small Industrial Cabinet (SIC)

- length 600 mm (23.62 inches)
- width 380 mm (14.96 inches)
- depth 210 mm (8.26 inches)
- weight 15.0 Kg (33.0 lbs)

The industrial enclosures are constructed of 1.25mm sheet steel, which is corrosion protected and coated with a baked enamel finish.

The doors have a polyurethane door seal, and the enclosure bodies have a protective gutter around the door opening to prevent ingress of dirt and water when the door is opened.

The doors have 110° hinges for easy access into the enclosure and have cam locks with removable handles. Key locks are optionally available.

The enclosures have a custom gear plate for mounting the *dataTaker* data loggers, Channel Expansion Modules, batteries, modems and other accessories.

The optional enclosures for the *dataTaker* can be installed in any orientation, and may be fixed to any surface. Where the data logger is to be permanently installed, location should be such as to facilitate the routing and entry of signal input and output cables.

Facilities for cable entry into the enclosure are not provided during manufacture, because of considerable variation in the requirements for different applications. Cable entry must be effected by the user via holes drilled into the gland plates or the walls of the enclosure, which can be fitted with cable glands.

Portable Installation of the dataTaker

The optional portable enclosure allows the *dataTaker* to be used in roving applications, in locations which may be dusty and subject to intermittent water showers or sprays.

The outside dimensions of the portable enclosure is as follows

- length 355 mm (14.0 inches)
- width 260 mm (10.23 inches)
- depth 155 mm (6.10 inches)
- weight 3.5 Kg (7.7 lbs)

The enclosure is a clamshell design, constructed of moulded black ABS plastic with a full length stainless steel hinge. The lid is sealed by a neoprene gasket, and held closed by two sturdy latches which can be fitted with padlocks. The enclosure will withstand 800Kg stacking loads, and will not dent, warp, or become misaligned. A contoured and padded carry handle makes the case comfortable and easy to carry.

The portable enclosure houses one *dataTaker* data logger, and one 4Ah gel cell or 17Ah alkaline battery.

The *dataTaker* and battery are mounted on a supporting frame which sits in the base of the enclosure, and is held in place by the enclosure lid when closed. The frame can be lifted out of the enclosure to provide easy access to the screw terminals and connectors of the *dataTaker*.

The frame has lugs for wrapping long cables, and a tray for storage of sensors and shorter cables. The tray can be inverted to provide a surface for custom mounting of connectors, switches, etc.

Powering the *dataTaker*

The *dataTaker* may be powered from a variety of power sources, depending on the needs of the application and the availability of power sources.

A mains/line power adaptor with 12 volt output is supplied with the *dataTaker*, for powering the logger from the local mains/line power supply.

The *dataTaker* can be powered from any 11–24 VDC source or 9–18 VAC source. This source must be able to provide at least 600 mA to allow normal operation of the *dataTaker* and proper recharging of the lead acid rechargeable battery.

Suitable alternative power sources include solar panels and lead acid auto batteries.

The external power supply is connected to the *dataTaker* via the terminal pair labelled AC/DC. The AC/DC power input is bridge rectified before being applied to the battery charger and regulator circuits.

Therefore for a floating external AC or DC power supply, the polarity of connection is not important.

However if an external DC power supply is to be grounded to the *dataTaker* ground, then a negative ground is mandatory. A positive ground may damage the rectifier.

The *dataTaker 50* can be fitted with an external 6 Volt sealed lead acid gel cell battery (not supplied with the logger) for powering the logger in the absence of an external power supply. This battery is automatically recharged whenever an external power supply is connected to the logger.

The *dataTaker 500/600* series loggers have an internal 1.2 AHr 6 Volt sealed lead acid gel cell battery, for powering the logger in the absence of an external power supply. This battery is automatically recharged whenever an external power supply is connected to the logger.

The rechargeable lead acid battery can be replaced with a non-rechargeable alkaline battery. This battery offers a higher charge density, and therefore a longer period of operation in the absence of other external power supplies.

The Communications Cable

The *dataTaker 50* has one serial interface - an RS232 serial interface nominated the RS232 COMMS interface.

The *dataTaker 500/600* series loggers have two serial interfaces - an RS232 serial interface nominated the RS232 COMMS interface, and an RS485 serial interface nominated the NETWORK interface.

The RS232 serial interface of the *dataTaker* Series 2 and Series 3 data loggers is electrically isolated, providing protection to approximately 500Volts.

The *dataTaker* communicates with a host computer via the RS232 COMMS serial interface.

The communications cable supplied with the *dataTaker* is 2 metres (6 feet) in length, with a 9 pin male D connector for the *dataTaker* installed at one end and a 9 pin female D connector for connection to a Windows computer. A communications cable suitable for use with Apple computers is available from your *dataTaker* supplier.

The wiring schematics for the dataTaker communications cables is detailed in Section II - The RS232 COMMS Serial Interface.

Where a number of *dataTaker 500/600* series loggers are linked into a network, the individual loggers communicate with each other via the RS485 NETWORK interface. The host computer supervises all loggers in the network, and communicates with the network via the RS232 COMMS interface of any one of the loggers in the network.

Establishing Communications

This host computer used to supervise the *dataTaker* can be any of the following

- a computer with a Microsoft Windows operating system, running DeTransfer or DeLogger data acquisition software package
- any computer running a terminal emulation or communications program

Establishing communications between a host computer and the *dataTaker* for the first time can be simplified by following the procedure outlined below:

Connect the COM1 serial port of the computer to the RS232 COMMS serial interface of the dataTaker, using the communications cable supplied with the logger.

Note: If you are using a notebook computer and do not have a serial COM port on your computer, then you will require either a docking module for your notebook computer which has a COM port, or a generic USB to serial COM port converter. Contact to your dataTaker supplier for assistance.

Connect the mains/line power adaptor to a mains power outlet, and connect the output of the adaptor to the two power supply terminals labelled AC/DC POWER. The polarity of this connection does not matter. Power up the *dataTaker*.

Install the DeTransfer and/or DeLogger software onto your computer. These are supplied on the dataTaker Software Suite CD supplied with your *dataTaker*

Note: Throughout this manual it is assumed that DeTransfer or DeLogger will be used as the host software for communicating with your *dataTaker*. This manual deals with the language of the *dataTaker*, which is interacted with using direct terminal-like communications with the logger. You could also use Windows HyperTerminal, or any other generic communications software.

If you are using DeLogger as the host software, then consult your DeLogger Manual for details of first communications. The DeLogger Text View can be used as a direct interface to the *dataTaker*, in much the same way as DeTransfer.

The *dataTaker* is shipped with the RS232 COMMS serial interface set to 9600 baud, 8 data bits, 1 stop bit and no parity (See Section II - The RS232 COMMS Serial Interface). The *dataTaker* baud rate is altered by setting relevant bits of the 8 way DIP switch (See Section II - The RS232 COMMS Serial Interface).

DeTransfer or DeLogger will initially attempt to connect to a *dataTaker* on COM 1 serial interface of the computer, and automatically perform a baud rate match to establish a communications link.

For further details of connecting to a *dataTaker* with the DeTransfer program, refer to Help within DeTransfer. If connection is unsuccessful, follow the suggestions listed in the help windows.

For further details of connecting to a *dataTaker* with the DeLogger program, refer to the DeLogger

Users Manual and the Help within DeLogger. If connection is unsuccessful, follow the suggestions listed in the manual and help windows. Also refer to Identifying Causes of Communications Failure below for further assistance.

Resetting the *dataTaker*

When you have physically and logically connected to the *dataTaker*, you are ready to send commands to the logger.

If you are using DeTransfer, type the *dataTaker* reset command RESET (this must be in upper case) in the lower window of DeTransfer, then press the Enter key

```
RESET [Enter]
```

Pressing the Enter key will only send commands to the *dataTaker* if the main menu Send|To dataTaker Mode option is checked.

Commands can also be sent to the *dataTaker* by placing the cursor on the command, and

holding down the Alt key and pressing the L key (send line)

by pulling down the Send menu and clicking on the Line option

by clicking the Send Line toolbar button

When the *dataTaker* executes the reset command, the following initialisation message from the *dataTaker* will be displayed in the upper window of DeTransfer

```
dataTaker nn Version 7.xx
Initializing...Done
```

where *nn* is the address of the particular *dataTaker*, and 7.*xx* is the Version of the firmware installed.

Note: This example of a *dataTaker* command is enclosed in a box. This format for illustrating *dataTaker* commands will be used throughout the manual.

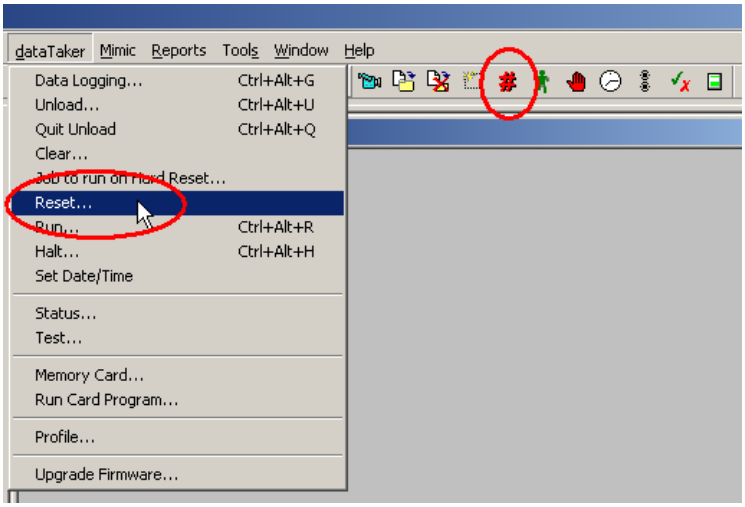
During the initialization sequence, the Converting LED which is located

beside the external AC/DC POWER connector of the *dataTaker 50*

beside the memory card connector of the *dataTaker 500/600* series loggers

will flash intermittently, indicating that the analog to digital converter is automatically recalibrating.

If you are using DeLogger, then a RESET command can be sent to the *dataTaker* by pulling down the dataTaker option on the main menu bar and selecting Reset..., or clicking the Reset button on the toolbar



DeLogger will respond with a warning dialog that you are about to reset the *dataTaker*, which will remove antsy stored data. Click the Yes button to proceed with the reset command.

If you have successfully connected to the *dataTaker*, and executed a reset command, then proceed to the section Entering Commands in Low Power Mode below.

Identifying Causes of Communication Failure

Failure of the communication between the host computer and the *dataTaker* is indicated either by no response being displayed by DeTransfer, or by 'garbage' characters being displayed. The following checks should be performed to identify and correct the problem.

No Message

If no initialization message is displayed, then

- check that the *dataTaker* is connected to the same COM port of the computer that is specified in the DeTransfer connection
- check that autobauding is selected in the DeTransfer connection definition
- check the baud rate of logger matches the DeTransfer setting if autobauding is not selected
- check that the transmit and receive wires from the *dataTaker* are connected to the receive and transmit pins respectively of the serial interface of the computer
- check that the signal ground wire from the *dataTaker* is connected to the signal ground pin of the serial interface of the computer
- check if any handshake lines of the serial interface of the computer should be linked together to defeat handshaking
- check that the RS232 serial interface of the computer functions correctly

Garbage Characters

If 'garbage' characters are displayed on the screen of the host, then

- check that autobauding is selected in the DeTransfer connection definition
- check the baud rate of logger matches the DeTransfer setting if autobauding is not selected
- check that the communications cable is wired correctly for the host computer used, and that there are no loose connections
- check that the RS232 serial interface of the computer functions correctly

Entering Commands in the Low Power Mode

The *dataTaker* generally operates in a low power mode whenever external power is not connected and the logger is being powered by the internal battery. The logger waits for the next scan to become due, or for commands to be received into the RS232 COMMS serial interface, at which time it wakes to normal power mode to process the event.

When the *dataTaker* is in the low power mode, various sections of the logger including the microprocessor and serial interface circuitry are powered off.

When a scan becomes due, or a command is are received at the RS232 COMMS serial interface, the *dataTaker* wakes up to the normal power mode and performs the function required.

If commands are sent to the *dataTaker* while the logger is in the low power mode, then a number of leading characters of the command will be missed as the logger wakes up. The *dataTaker* takes approximately 250 mS to wake, and any characters received during this period will be missed. The number of characters missed depends on the communications baud rate, such that at the higher baud rates the entire command may be missed.

These missed characters will result in command syntax errors due to incomplete commands being received. The *dataTaker* will return command error messages for what would appear to be syntactically correct commands.

This problem can be avoided by sending the logger a [CR] or [LF] character before sending the command. The [CR] or [LF] character will wake the logger ready to receive the command. DeTransfer and DeLogger automatically manage this scenario.

Whenever the *dataTaker* wakes from the low power mode by characters received into the RS232 COMMS serial interface, the logger remains in the normal power mode for 30 seconds after the last character is received, before dropping to the low power mode again (See Section II - Low Power Delay).

If the *dataTaker* is externally powered (this is detected by a positive internal battery current such that the battery is being charged), then the low power mode is not automatically selected unless the low power mode is forced by the Parameter15=1 command (See Section II - Powering the *dataTaker*).

For applications where low power mode operation is not required, or is inconvenient, then automatic selection of low power mode can be disabled by the Parameter15=2 command (See Section II - Powering the *dataTaker*).

Whenever the *dataTaker* is externally powered, or is operating in the normal power mode, the Converting LED will flash every second as the battery current is tested (See Section II - Powering the *dataTaker*).

The Real Time Clock

The real time clock of the *dataTaker* is hardware clock, which is backed up by the system lithium battery.

The clock maintains time and date even when all logger power supplies have been removed.

Once communications have been established with the *dataTaker*, then the first simple data acquisition command can be executed – to return the current time.

If you are using DeTransfer, the following command will return the current time for the *dataTaker* real time clock.

```
T  [Enter]
```

The command is entered to the *dataTaker* by typing an upper case T, followed by the Enter key.

The *dataTaker* will return the current time in HH:MM:SS format as follows

```
Time 00:10:30
```

If the time is displayed, then full two way communications has been established.

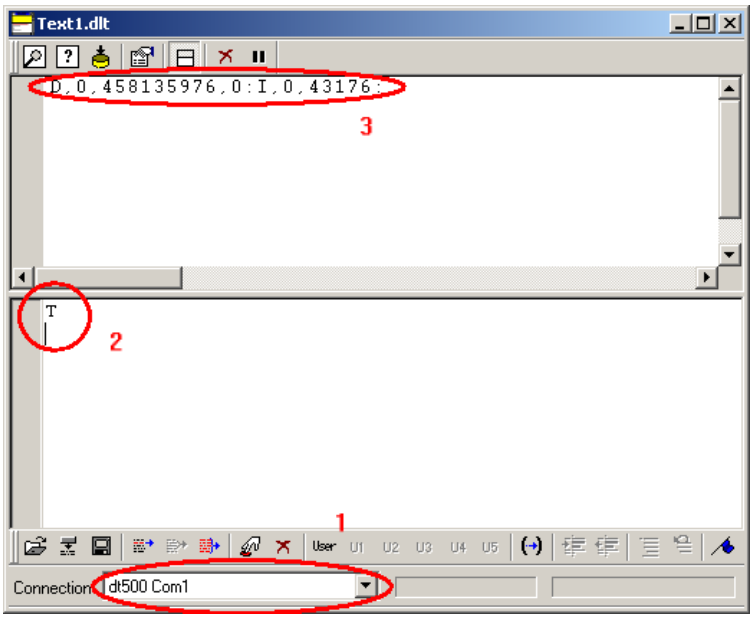
If you are using DeLogger, the *dataTaker* time can be read from the Text View window. Pull down the Window option on the main menu bar, and select the Text1.dlt option. This will bring the Text View window to the front.

The *dataTaker* time can be read using this window as follows (see also illustration overleaf):

Drop down the Connections list box at the bottom of the window, and select the connection name for the current connection to your *dataTaker*

Type an uppercase character T in the lower (Entry) screen, then hit the Enter key. This will send the 'Read Time' command to the logger

The *dataTaker* will return its current time, which is displayed by DeLogger in the upper (Display) screen.



The *dataTaker* time is displayed in the Fixed Format mode, in which the *dataTaker* time is displayed as the number of seconds elapsed today. In this example, the time is 43176 seconds today (ie. since midnight), which is 11:59:36. The format for *dataTaker* data is discussed in detail in Section III.

Setting the Real Time Clock

When you read the *dataTaker's* time for the first time, it is unlikely to be correct for your time zone. The real time clock of the *dataTaker* can now be set for the current date and time.

The format for the date is dependent on the setting of the mains or line frequency bit (bit 1) of the DIP switch (See Section III - The Real Time Clock).

If the line frequency switch is set to 50 Hz (DIP switch bit 1 OFF) then the date format is Day/Month/Year. The date is therefore set in the format

```
D=dd/mm/yyyy  [Enter]
```

where dd, mm and yyyy are day, month and year respectively. The year can be entered either as yyyy (eg. 2003) or yy (eg. 03).

Alternatively, if the line frequency switch is set to 60 Hz (DIP switch bit 1 ON) then the date format is Month/Day/Year, and the date is therefore set in the format

```
D=mm/dd/yyyy  [Enter]
```

where mm, dd and yyyy are month, day and year respectively.

The time is set for the current time of day in the format

```
T=hh:mm:ss  [Enter]
```

where hh:mm:ss is the current time in standard format.

If you are using DeTransfer, the date and time can be set directly by the commands for example

```
D=28/6/2003  [Enter]

T=14:46:30  [Enter]
```

However, DeTransfer also allows for the time and date of your computer's system clock to be used to set the *dataTaker*'s clock. This command is performed as follows:

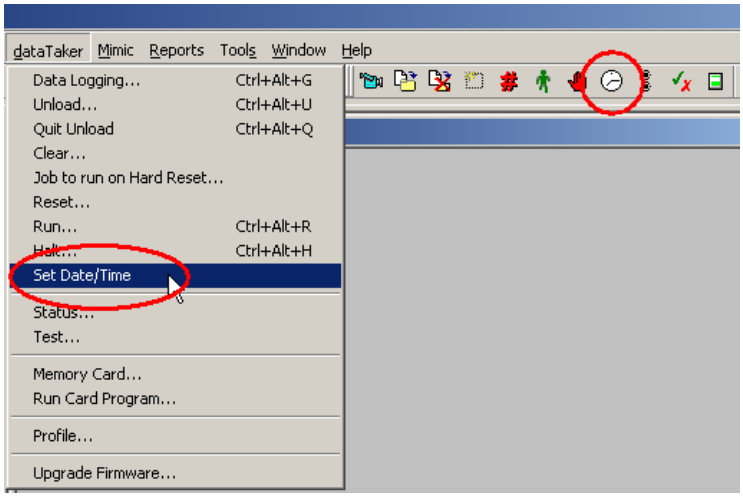
```
D=\D   [Enter]

T=\T   [Enter]
```

where DeTransfer substitutes the \D with the current computer system date, and substitutes the \T with the current computer system time.

Note : Ensure that your computer's system clock is set correctly, and that the country or regional setting for your computer matches the date format selected by the *dataTaker* DIP switch. The computer date format is defined in the Regional Options applet in the Windows Control Panel (click Start:Settings:Control Panel:Regional Options).

If you are using DeLogger, then the date and time of the *dataTaker* can be set either by pulling down the dataTaker option on the main menu bar and selecting Set Date/Time…, or by clicking on the Set Date and Time tool on the toolbar



Note : Again, ensure that your computer's system clock is set correctly, and that the country or regional setting for your computer matches the date format selected by the *dataTaker* DIP switch. The computer date format is defined in the Regional Options applet in the Windows Control Panel (click Start:Settings:Control Panel:Regional Options).

Reading the Real Time Clock

The date and the time of day for the *dataTaker* real time clock are treated as a data source, in the same way as any other channel of the logger. In fact the date and time 'channels' can be included with input channels in data collection commands, to produce time and date stamps with the collected data (see Section III).

Using DeTransfer, the date and time of the *dataTaker* can be read by the commands

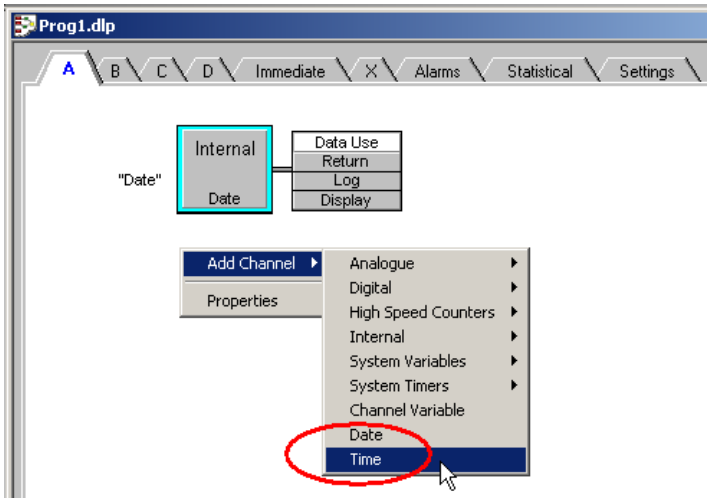
```
D  T   [Enter]
```

which returns the date and time in the format

```
Date   28/4/2003
Time   14:46:35
```

Note : Where 60 Hz line frequency operation is selected, then the date is set and returned in the Month/Day/Year format.

Using DeLogger, the date and time of the *dataTaker* can be read by the following icons in the Program Builder



The current date and time returned by the *dataTaker* can be seen in the upper Display Screen of the Text

Reading the Temperature of the *dataTaker* and the CEM

The *dataTaker* and the Channel Expansion Module (CEM-AD) both have an internal temperature sensor which is used to monitor internal temperature. This sensor is an LM35 solid state device, and is permanently connected to an internal analog input channel identified by 1%.

Using DeTransfer, the internal or case temperature of the *dataTaker* is read by the command

```
1%LM35  [Enter]
```

which specifies the temperature sensor as a solid state LM35 type, powered directly from the analog input channel of the logger (See Section II - Measuring Temperature with Monolithics).

The internal temperature is returned in the format

```
1%LM35  21.2 Deg C
```

The internal or case temperature of a Channel Expansion Module is read by the command

```
1:1%LM35  [Enter]
```

The internal temperature sensor is read once in response to the command.

The internal temperature is returned for the Channel Expansion Module in the format

```
1:1%LM35  21.2 Deg C
```

Note: The Channel Expansion Module is identified by a module number which is automatically allocated by the *dataTaker* during RESET, and refers to the position of the Channel Expansion Module on the expansion bus (ie: CEM number 1 is closest to the *dataTaker* and CEM number 2 is the last module on the bus).

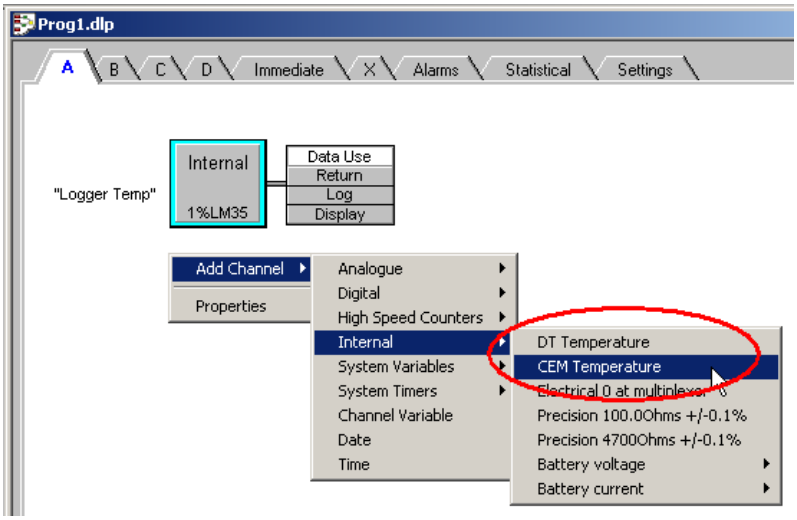
The module number of the Channel Expansion Module is prefixed to the command for the module channel to be read. In the above example a single Channel Expansion Module is assumed, which has the module number 1:

The *dataTaker* has the module number 0: on the expansion bus, and so the internal temperature of the *dataTaker* could also have been read by the command

```
0:1%LM35
```

Whenever any analog input channel is scanned or 'converted', the Converting LED will flash, indicating that the analog to digital conversion is active.

Using DeLogger, the internal or case temperature of the *dataTaker* and CEM can be read by the Program Builder icons



A First Program - Reading the Time and Temperature of the *dataTaker*

The time and internal temperature of the *dataTaker* or the Channel Expansion Module can be read at regular intervals, and returned to0 the host computer.

Using DeTransfer, the time and internal temperature can be returned every 5 seconds by the command

```
RA5S  T  1%LM35  [Enter]
```

The *dataTaker* interprets the command and begins scanning the internal temperature channel every 5 seconds. This command combines a repetitive scanning command (RA5S), and timestamp command (T), and a channel from which data is required (1%LM35). This single line is in fact a small *dataTaker* program.

The *dataTaker* can run up to 4 different scanning commands in the same program, each scanning at different intervals, and each scanning different groups of channels. Each of these is known as a Schedule, and the 4 Schedules are identified a Schedule A, B, C and D.

The single line program above could also have been entered as a multi-line program, as would be necessary for larger programs, as follows

```
BEGIN     [Enter]
 RA5S     [Enter]
  T       [Enter]
  1%LM35  [Enter]
END       [Enter]
```

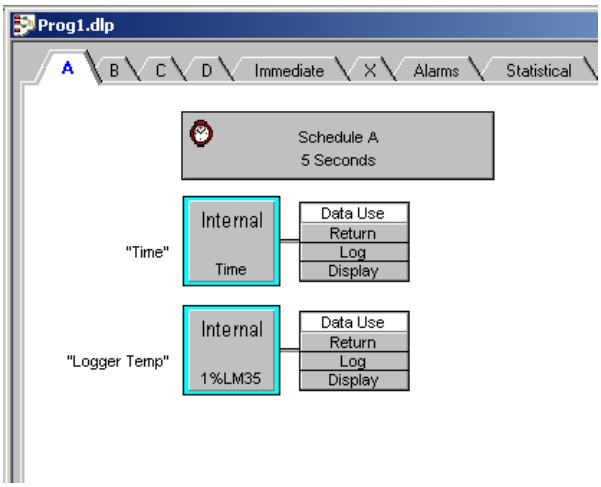
This multi-line format for entering programs is discussed fully in Section III, however we will use this format through the Sections I and II of this manual.

The temperature data and a timestamp is returned every second in the general format

```
Time  12:22:45
1%LM35  21.2 Deg C

Time  12:22:46
1%LM35  21.1 Deg C
```

Using DeLogger, this small program can be entered into the *dataTaker* by creating the program graphically in the Program Builder as follows



where the Schedule A has a Time channel and the internal temperature channel added to it. Refer to your DeLogger manual for further information on use of the Program Builder in DeLogger.

There is no need to understand the structure of multi-line programs if you are using DeLogger, because DeLogger automatically takes care of program structure. You only need to be aware of the separate tabs in the Program Builder for the 4 Schedules A, B, C and D.

Halting Scanning

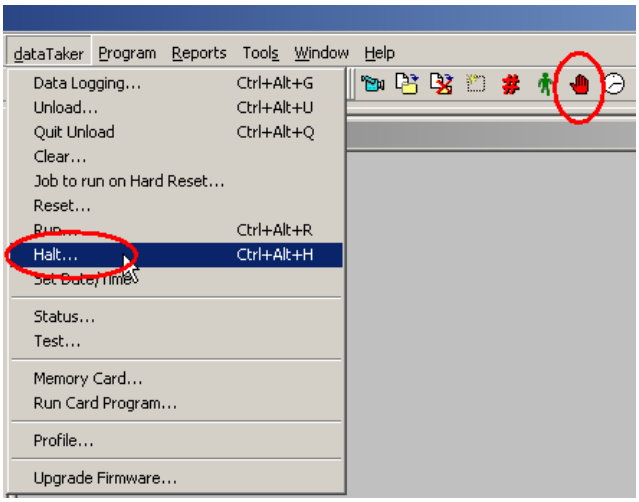
The *dataTaker* will continue to read the internal temperature channel and return data until a Halt command is received.

Using DeTransfer, a Halt command is entered as follows

```
H [Enter]
```

Note: If DeTransfer does not display data as fast as it is returned from the *dataTaker*, then entering a Halt command will not immediately halt data scrolling in the Receive window of DeTransfer. Although scanning by the *dataTaker* is halted, data that is held in the internal buffers of DeTransfer continues to be displayed. However these buffers will eventually empty, and the screen will stop scrolling data.

Using DeLogger, scanning of data can be halted either by pulling down the dataTaker option on the main menu bar and selecting Halt…, or by clicking on the Halt tool on the toolbar



Resuming Scanning

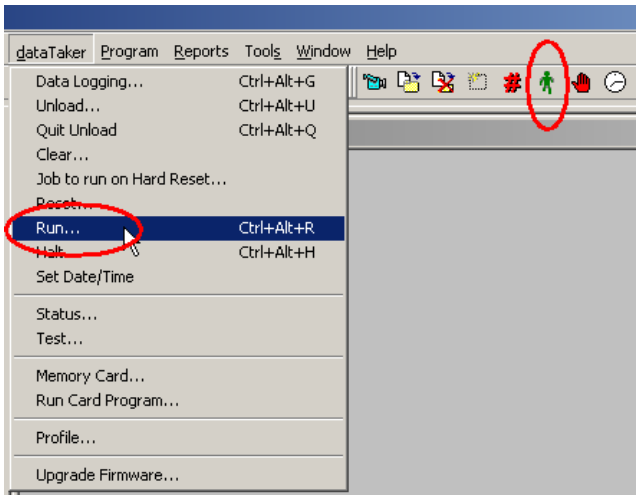
Halting a Schedule does not delete the schedule from the logger. Schedules that have been halted can be resumed scanning by entering a Go command to the logger.

Using DeTransfer, a Go command is entered as follows

```
G [Enter]
```

and scanning of the temperature channel will resume.

Using DeLogger, the scanning of data can be resumed either by pulling down the dataTaker option on the main menu bar and selecting Run…, or by clicking on the Run tool on the toolbar



Command Styles

The *dataTaker* commands are composed of printable upper case ASCII characters and numerals. Any lower case characters are ignored by the *dataTaker*, and can be used to self document commands.

In the case of the DeTransfer program (we will no longer show the [Enter] key in our programs) entered earlier,

```
BEGIN
 RA5S
  T
  1%LM35
END
```

this could have been entered in DeTransfer as follows

```
BEGIN
 Repeat_the_A_schedule_every_5_Seconds
  Time
  internal_channel_1%_with_a_LM35_sensor
END
```

This is of course extreme, however only the digits and the upper case characters are recognised by the *dataTaker* command processor.

A limitation of self documenting commands and programs in this way is that the *dataTaker* must receive the RA1S in this example as a continuous sequence of characters without intervening spaces. The underscore character can be used to space the components of the command, to improve the readability.

The exception to use of lower case characters to self-document programs are Switch commands (See Section III – Switch Commands). The Switch commands are used to configure the *dataTaker*, and an upper case character specifies that a particular feature is enabled while a lower case character specifies that the feature is disabled.

DataTaker programs may be simple like those above, or may be more extensive where more involved data logging tasks are to be performed. For example the following command programs the *dataTaker* for a more extensive task

```
BEGIN
 Repeat_B_schedule_every_10_Secs
  Time
  channel_4_Volts
  channel_5_Tcouple_type_K
END
```

or more concisely

```
BEGIN
 RB10S
  T
  4V
  5TK
END
```

which returns data every 10 seconds as follows

- the current time (timestamp)
- the voltage measured for analog input channel 4
- the temperature measured by a type K thermocouple connected to analog input channel 5

Command lines from DeTransfer can contain several separate commands, but are limited to a maximum of 250 characters.

If you are using DeLogger, then program structure is managed entirely by DeLogger and you do not need to consider these programming options.

The Internal Power Supply Backup Battery

The *dataTaker* has a sealed lead acid gel cell battery which can be used to provide primary power, or to provide backup power in the event of failure of the primary external power supply.

In the *dataTaker 500/600* series loggers, the battery is contained within the data logger chassis and is often referred to as the internal battery.

By contrast, the battery is physically external to the *dataTaker 50* chassis, and is mounted by the user in the enclosure containing the data logger. Nonetheless the battery still performs the role of a backup battery.

A 4 AHr backup battery is able to power the logger for up to 65 days. If the battery is allowed to fully discharge, then data logging function will cease, however data will still be retained in the internal data storage memory because this memory is backed up by the lithium system battery.

The *dataTaker 50* is shipped with the gel cell lead acid battery (if purchased) disconnected and separately packed.

The *dataTaker 500/600* series loggers are shipped with the gel cell lead acid battery installed but disconnected. The battery must be connected to the internal battery connector before first use (see the tag advising of this requirement).

The internal backup battery is recharged whenever external power is supplied to the *dataTaker*. A full recharge takes approximately 8 hours.

Whenever the *dataTaker* is not in use the battery should either be disconnected, or left on charge. The battery charger circuit of the *dataTaker* is designed for long term battery maintenance.

When the Internal Battery Goes Flat . . .

If the logger is being powered from the internal battery, and this battery discharges, then the logger will respond as follows

Series 1 and Series 2 loggers will discharge the battery to the point where operation of the logger is no longer possible and the microprocessor will eventually stop. The program will be lost, and any data stored in the internal data memory will be lost. Data stored in a memory card however will be OK.

Series 3 loggers will discharge the battery to a ‘grey out’ threshold, at which point the logger will power down to an inactive mode and will not respond to any communications, scan triggers or other activity. The program and stored data are fully preserved in this state. When external power is applied, the logger will re-activate, and resume the program it was running before becoming inactive.

Note: A Series 3 logger will not enter the inactive state if power supply or internal battery is suddenly disconnected – in this case there is insufficient time to preserve the status of the logger, and so program and data loss will usually result.

The Lithium System Battery

The *dataTaker* has a lithium battery which maintains the real time clock and internal data memory when the logger is not powered.

The ½AA sized battery must be replaced every 2 years. The battery is located on the bottom (processor) circuit board, and the logger must be disassembled to replace the battery.

Section II

Interfacing

to the

dataTaker 50,500,600 Series 3

Data Loggers

Interfacing to the *dataTaker*

This section of the manual deals with all aspects of interfacing with the *dataTaker* including

connecting power supplies

interfacing to the host computer

configuring local networks of *dataTaker 500/600* series loggers

connecting analog signals to the analog input channels

connecting digital signals to the digital input channels

connecting digital outputs to digital devices

connecting a Panel Mount Display module to the *dataTaker 50* and *dataTaker 500* series loggers

connecting Channel Expansion Modules to the *dataTaker 500/600* series loggers

A Note on Commands

Throughout Section II of the manual, numerous examples of commands are given in the context of reading the different input channel types, setting the digital output channels, defining the system configuration, etc. as each is individually discussed.

In each of these situations the commands are only briefly discussed. A complete description of all commands can be found in Section III of this manual.

A Note on Examples

Throughout Section II of the manual, numerous examples of commands are given in the context of reading the different input channel types, setting the digital output channels, defining the system configuration, etc as each is individually discussed.

These examples often contain analog input channel numbers which relate to the *dataTaker 500/600* series loggers, and are often greater than the maximum analog input channel for the *dataTaker 50*.

However the same concepts for input channel identification and specification used in these examples also apply to the *dataTaker 50*.

A Note on Entering Commands into the *dataTaker* Data Loggers

There are two different host computer applications supplied with the *dataTakers* for programming and supervising the loggers – DeTransfer and DeLogger 4.

These programs run on any of the Microsoft Windows operating systems, Windows 98 Second Edition, Windows NT4 Service Pack 5, Windows 2000 and Windows XP.

DeTransfer

DeTransfer is a simple terminal like application in which you send commands to the *dataTaker* from the bottom screen, and receive data and information back from the *dataTaker* into the top screen. In order to use DeTransfer with your *dataTaker*, you will need a good knowledge of the *dataTaker* programming language that is presented in Section II and Section III of this manual, and in the User's Manual supplied with your logger.

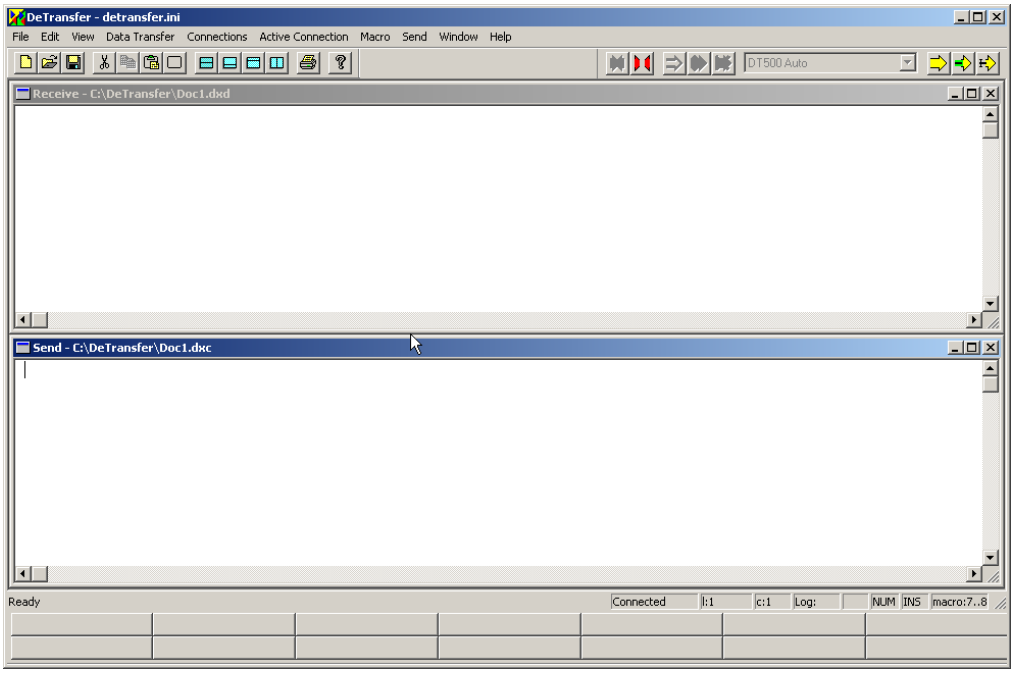
When you first run DeTransfer, you first need to 'connect' to your *dataTaker* before you can send commands to the logger or receive data and information back from the logger. To connect DeTransfer to the *dataTaker* you must

electrically connect the RS232 COMMS port of the logger to the COM1 or COM2 port of the computer using the *dataTaker* communications cable supplied.

logically connect DeTransfer to the *dataTaker* attached to the COM port of the computer

See the DeTransfer Help menu, or the Getting Started Manual supplied with Series 1 and Series 2 *dataTakers* for details of running DeTransfer.

The DeTransfer windows appears similar to



Commands are sent to the *dataTaker* from the lower 'Send' window, and the various information and data returned by the logger is displayed in the upper 'Receive' window.

The examples throughout this manual are shown in the following formats for single line commands,

```
T=12:30:00
```

single line programs,

```
RA5S 1TK 2TK 3TK 5V
```

and multi line commands or programs

```
D=10/04/03
T=12:30:00

BEGIN
 RA5S
  1TK
  2TK
  3TK
  5V
END
```

In all cases, the examples given can be entered into the *dataTaker* by typing the command(s) into the lower 'Send' window of DeTransfer.

DeLogger 4 and DeLogger Pro 4

DeLogger 4 and DeLogger Pro 4 are higher level applications for programming and supervising the *dataTaker* data loggers. DeLogger Pro 4 has some additional capability to process and display the data, however both versions have the same interface to programming and supervising the *dataTaker*.

It is possible to use DeLogger to program and supervise the *dataTaker* to collect and return data without having any knowledge of the *dataTaker* language. The DeLogger user interface is mostly graphical, both for programming the logger to collect required data, and for supervising the logger while it is collecting that data.

When you first run DeLogger, you first need to 'connect' to your *dataTaker* before you can send commands to the logger or receive data and information back from the logger. To connect DeLogger to the *dataTaker* you must

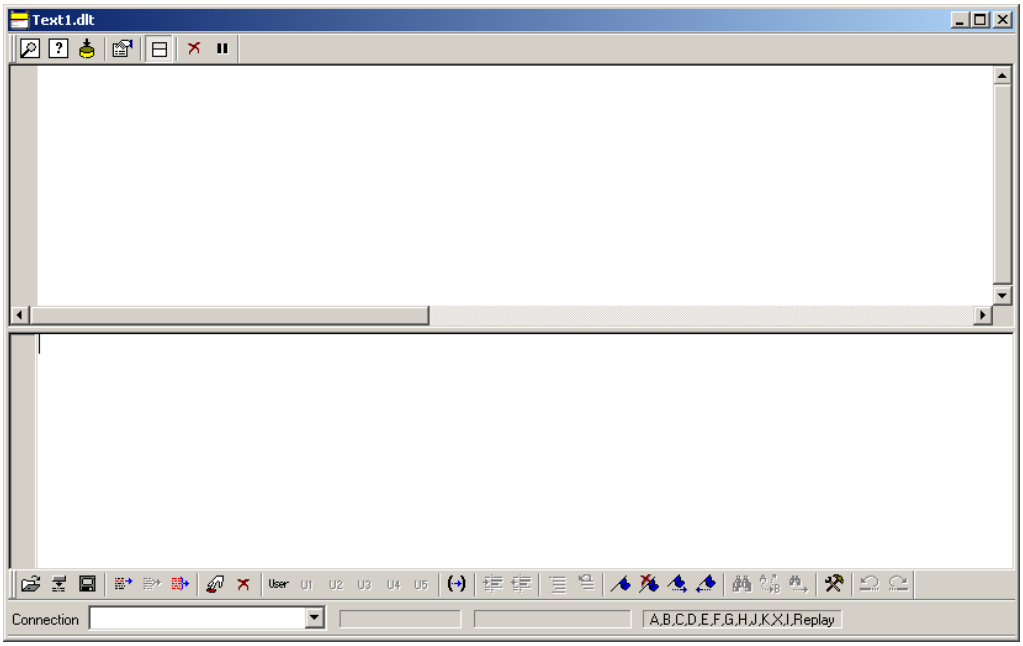
electrically connect the RS232 COMMS port of the logger to the COM1 or COM2 port of the computer using the *dataTaker* communications cable supplied.

logically connect DeLogger to the *dataTaker* attached to the COM port of the computer

See the DeLogger Manual supplied with your *dataTaker* for complete details of running DeLogger.

Sending Commands Directly from DeLogger

DeLogger has a number of windows or views, including the Text View which is similar to the DeTransfer user interface.



Commands are sent to the *dataTaker* from the lower 'Entry' screen or window, and the various information and data returned by the logger is displayed in the upper 'Display' screen or window.

The examples throughout this manual are shown in the following formats for single line commands,

```
T=12:30:00
```

single line programs,

```
RA5S 1TK 2TK 3TK 5V
```

and multi line commands or programs

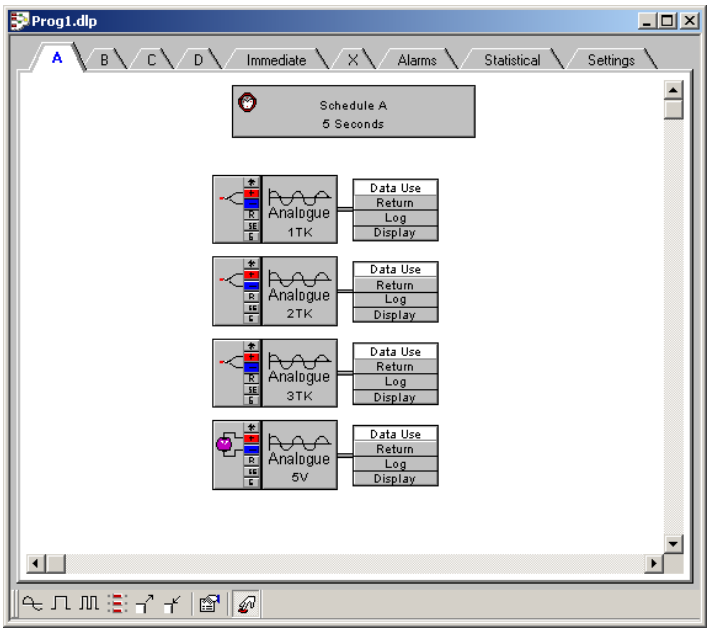
```
D=10/04/03
T=12:30:00

BEGIN
 RA5S
  1TK
  2TK
  3TK
  5V
END
```

In all cases, the examples given can be entered into the *dataTaker* by typing the command(s) into the lower 'Entry' screen or window of DeLogger.

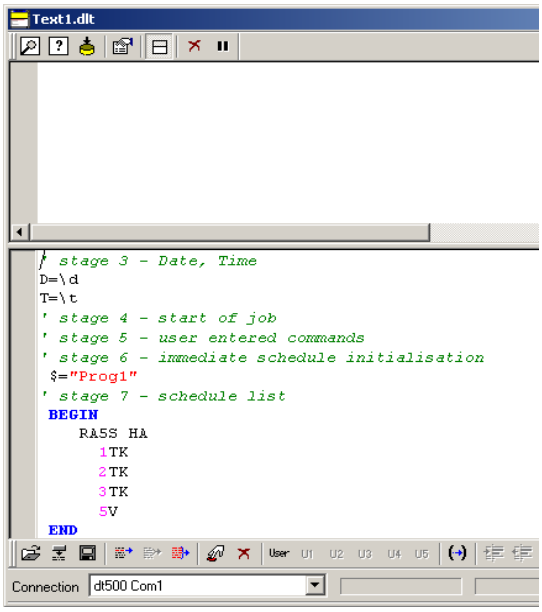
However, complete data acquisition programs as is shown in the last example are usually created in the Program Builder of DeLogger, and the *dataTaker* commands are never actually seen by the user.

In the DeLogger Program Builder the program example above would be similar to



The DeLogger Program Builder takes the program that you build up graphically, and converts it into *dataTaker* language script before sending it to the logger.

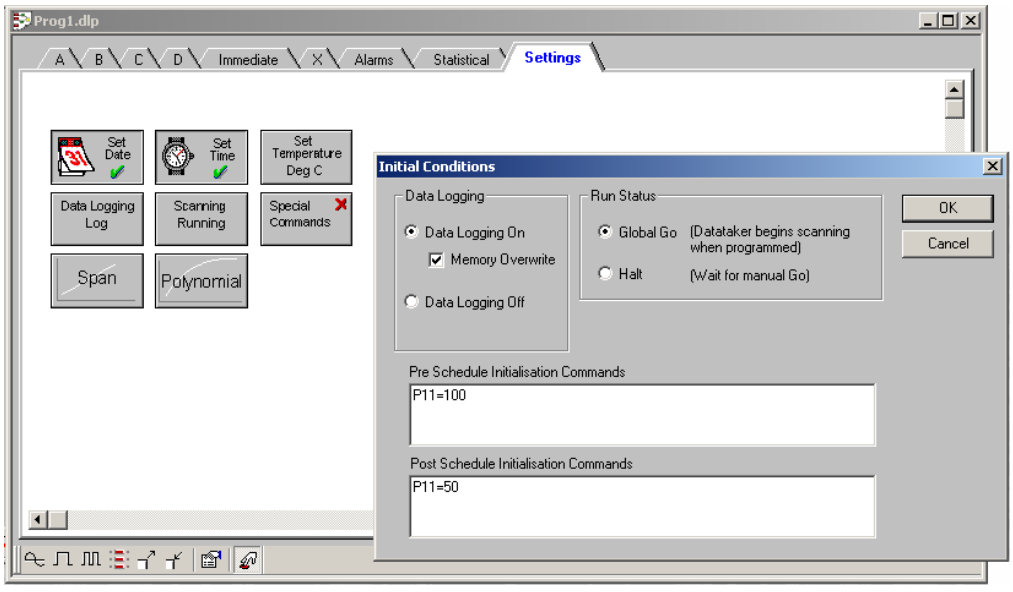
This script can be seen in the lower Entry screen of the Text View when you click on the Load Program Text button (second from left) on the bottom toolbar.



DeLogger does not support all aspects of the *dataTaker* command set, and occasionally you may have to enter these unsupported commands manually. This can be done directly from the Entry screen or window of the Text View, in much the same way as would be done from DeTransfer.

Sending Commands with a Program from DeLogger

If unsupported commands need to be part of the program developed in the Program Builder, then the commands are entered as Pre Schedule or Post Schedule Initialization Commands under the Special Commands button of the Settings tab in the Program Builder.



Any commands entered here are integrated into the script produced by the Program Builder.

DeLogger 3 and DeLogger Pro 3 that was supplied with Series 1 and Series 2 *dataTaker* data loggers is very similar to DeLogger 4 outlined above. Consult your DeLogger 3 manuals for any differences.

Powering the *dataTaker*

The *dataTaker* can be powered from a variety of power sources. These sources are categorised as external power supplies and internal or backup power supplies.

External power supplies include

- the AC mains/line voltage via the 12 volt power adaptor supplied with the logger
- various 12 to 18 Volt external batteries that can be connected to the *dataTaker* as external power supplies
- solar panels

Internal power supplies include

- a rechargeable 6 Volt lead acid battery which is connected as the 'internal' or backup battery. This battery is mounted externally for the *dataTaker 50*, and can be mounted internally or externally for the *dataTaker 500/600* series loggers. The battery is automatically re-charged whenever external power is supplied.
- a 9 Volt alkaline D cell battery pack

The basic power requirements of the *dataTaker* depends on the operating conditions, but is approximately 120 mA for normal operation, and approximately 600 mA when recharging the battery, at 12 Volts AC or DC power supply.

If power is drawn from the excite terminals of the analog input channels, or if power is drawn from the 5 Volt sensor power supply terminal, then the total current drawn will be higher during periods of scanning activity.

The *dataTaker* has two power modes - normal power mode and low power mode. These modes are automatically selected depending on activity.

There is no general power supply or battery ON/OFF switch on the *dataTaker*.

Low Power Mode

The *dataTaker* is designed to operate as a low power data logger, conserving power during periods of no activity by entering a low power mode. Only the real time clock, memory, wake detect circuit and high speed counters remain powered in the low power mode.

The low power mode allows the internal batteries to be used as the sole power supply for long periods, depending on operating conditions and scanning activity.

The *dataTaker* draws approximately 120 mA when operating in the normal power mode, while scanning input channels, monitoring alarms, and communicating with the host.

However whenever there is no scanning required, and there is no communications, then the *dataTaker* automatically drops to the low power or 'sleep' mode when it draws less than 400 µA.

The *dataTaker* 'wakes' up to the normal power mode in response to a number of conditions as follows

- a scan of the input channels becomes due, including for alarms and statistics
- communications is received by the serial interface
- a memory card is inserted or removed
- a change of state (edge) is received on the Wake terminal
- a key is depressed on the Display/Keypad (where fitted)
- the *dataTaker* also wakes every 18 minutes for time keeping

Note : Pulses or changes of state on the digital input channels or the high speed counter channels do not wake the *dataTaker*.

Selecting the Power Mode

The automatic low power mode of operation is enabled or disabled by the Parameter15 command (See Section III – Parameter Commands) as follows

Parameter15=0	Automatic low power selection (Default)
Parameter15=1	Force low power mode
Parameter15=2	Disable low power mode

The command for example

```
P15=1
```

will force the *dataTaker* into the low power mode immediately. The current setting for the power mode can be determined by the command

```
P15
```

Selecting Automatic Low Power Mode

The *dataTaker* defaults to automatic low power selection (Parameter15=0) whenever it is initially powered up or executes a RESET command.

Automatic low power selection only operates when the *dataTaker* is powered from the internal or backup battery. If the *dataTaker* is powered from an external power supply, then the logger remains in the normal power mode.

When awake, the *dataTaker* measures the internal battery current every second to determine the method of powering. If automatic low power mode is selected, then

- if the battery current is negative (battery is providing power) then the low power mode is selected, and the logger sleeps during periods of inactivity
- if the battery current is positive (battery is recharging from external power) then the low power mode is not selected, and the logger does not sleep during periods of inactivity

Force Low Power Mode

When the *dataTaker* is externally powered, the logger does not normally drop to the low power mode during periods of inactivity (see above).

However the low power mode can be forced when the *dataTaker* is externally powered, by the Parameter15 command as follows

```
P15=1
```

This allows power consumption from an external power supply to be reduced whenever the logger is inactive. This is particularly relevant if the power supply is a 12 Volt wet cell battery connected to the AC/DC power supply input.

Disable Low Power Mode

Whenever the *dataTaker* is powered from the internal or backup battery, it automatically selects the low power mode during periods of inactivity (see above).

Automatic low power mode selection can be disabled if necessary when the *dataTaker* is powered from the internal battery, by the Parameter15 command as follows

```
P15=2
```

which will force the logger to remain in the normal power mode.

Low Power Delay

When automatic low power selection is enabled, and the *dataTaker* has powered down to the low power mode, then the logger powers up or 'wakes' to the normal power mode in response to a number of conditions as follows

when a new scan of the input channels becomes due. The scan is performed and the logger then drops to the low power mode again.

when any communications are received from a host computer. The logger remains powered up for a period after the last character is received, and then drops to the low power mode again.

when a memory card is inserted or removed. The logger remains powered up for a period, and then drops to the low power mode again.

when the Wake terminal is connected to a GND terminal. The logger remains powered up for a period, and then drops to the low power mode again.

The delay before automatic power down following communication, memory card insertion or removal, or wake terminal activity is 30 seconds by default. This delay period can be defined from 1 to 255 seconds by the Parameter17 command (See Section III – Parameter Commands).

The command for example

```
P17=60
```

defines the delay before automatic power down to be 1 minute. The current setting for the delay before automatic power down can be determined by the command

```
P17
```

The *dataTaker* will not power down in the automatic low power mode unless it can sleep for at least one full second.

Controlling Waking of the *dataTaker*

The *dataTaker* will normally wake when time triggered Schedules are triggered for scanning. Schedules can be disabled from waking the logger by the Parameter20 command, which defines a bit mask of Schedules not to wake the logger as follows

Bit	***Schedule***	
0	Alarms Schedule (RZ)	– decimal value =1
1	Polled Schedule (RX)	– decimal value =2
2	Statistical Schedule (RS)	– decimal value =4
3	Triggered Schedule A (RA)	– decimal value =8
4	Triggered Schedule B (RB)	– decimal value =16
5	Triggered Schedule C (RC)	– decimal value =32
6	Triggered Schedule D (RD)	– decimal value =64
7	Immediate Schedule	– decimal value =128

Figure 6 - Bit Map for Schedules to Wake *dataTaker*

The default is Parameter20=0, which means that all Schedules can wake the logger when they become due.

Parameter20 is set to the decimal value of the bit mask defining which Schedules are not to wake the logger. The decimal value for each Schedule in the mask is indicated in the diagram above.

The command for example

```
P20=65
```

will disable the Alarm Schedule (1) and Triggered Schedule D (64) from waking the logger.

The Parameter20 command does not disable scanning of Schedules if the *dataTaker* is already awake.

The *dataTaker* will not wake for event based scanning, and so steps must be taken to ensure that external pulses wake the logger via the Wake terminal (see below).

The Wake Terminal

Waking of the *dataTaker* from low power mode can also be controlled by the Wake terminal. A low state of less than 1.0 Volts on the Wake terminal will wake the logger in less than 300 mS.

The wake signal can be generated by relay closures, an open collector NPN transistor switching to ground, etc. The wake terminal has an internal 1800 Ohm pull up resistor to 5 Volts. A permanent low state on the wake terminal will not prevent short periods (100 mS) of sleep if there is no Schedule activity. Only setting Parameter15=2 will keep the logger permanently awake.

The External Power Supply

The *dataTaker* can be powered from any external power supply which has the following specifications

dataTaker 50

9 to 15Volts AC, at 600 mA

10 to 17 Volt DC, at 600 mA

dataTaker 500/600 series loggers

9 to 18 Volts AC, at 600 mA

11 to 28 Volt DC, at 600 mA

The *dataTaker* is supplied with a 12 Volt power adaptor, which can be used to power the logger from the local mains/line power supply.

However other external power sources can be used to power the *dataTaker*, including external 12 Volt dry cell or wet cell batteries, solar panels, vehicle power supplies, etc.

The *dataTaker* draws approximately 120 mA in the normal power mode to power the logger. Whenever the backup lead acid battery is being recharged, then an additional charging current of up to 300 mA is drawn if the battery is fully discharged. Therefore the external power supply must be able to supply up to 450mA if a rechargeable lead acid battery is fitted, or 150 mA if a battery is not fitted.

The external power supply is connected to the *dataTaker 50* via the removable screw terminals labelled Power AC/DC (See Figure 7), located near to the Converting LED.

The external power supply is connected to the *dataTaker 500/600* series loggers via the screw terminals labelled AC/DC (See Figure 7), located near the 9 pin D connector of the RS232 COMMS serial interface.

The external AC/DC power input is bridge rectified before being applied to the battery charger and regulator circuits. Therefore if a floating external power supply is used, the polarity of the external power supply connection is not important.

Note: If an external DC power supply is to be grounded to dataTaker ground, then negative ground is mandatory. A positive ground will damage the bridge rectifier.

In this case the positive side of the external power supply must be connected to either of the AD/DC screw terminals, and the negative side connected to a GND terminal.

The Internal or Backup Battery

The *dataTaker* internal or backup battery provides primary power in the absence of an external power supply, and acts as a backup when the external power supply is changed or fails. The battery can power the *dataTaker* for long periods, depending on the scanning activity and the frequency of communications.

The battery for the *dataTaker 50* is always mounted outside of the body of the logger. The battery for the *dataTaker 500/600* series data loggers is either mounted within the body of the logger, or outside of the body of the logger. The internal battery can be either a 6 Volt sealed lead acid gel cell, or a 9 Volt alkaline D cell battery pack.

A lead acid internal battery is recharged whenever the logger is powered from external power. Full recharge takes 8 hours for the 1.2 Ah lead acid battery, and longer for the larger capacity batteries. The battery charger circuit of the *dataTaker* is designed for long term battery maintenance.

The *dataTaker 500/600* series loggers are shipped with the internal lead acid battery installed but not connected. The battery must be connected at first use of the logger, in accordance with the tag attached to the battery cable.

When the *dataTaker* is not in use, then the lead acid battery should be disconnected or the logger be left on charge. Failure to do this will allow the logger to is fully discharge the battery, which will eventually shorten the effective battery life.

Lead Acid Backup Battery dataTaker 50

The backup battery for the *dataTaker 50* is not installed in the chassis of the logger. The battery is always located outside of the logger, and connects to the battery screw terminals. A backup battery is not supplied with the *dataTaker 50*.

The backup battery can be any 6 Volt lead acid gel cell battery, for which a number of capacities are available including 1.2 AHr, 4 AHr and 10 AHr. The battery is connected to the screw terminals labelled Lead Acid + and Lead Acid –.

Lead Acid Backup Battery dataTaker 500/600 Series

The *dataTaker 500/600* series loggers is supplied with a 6 Volt 1.2 Ah lead acid gel cell battery. This battery is located under the top cover of the logger.

A larger capacity 6 Volt lead acid battery can be externally connected to the *dataTaker 500/600* series loggers, to provide greater backup capacity and hence longer periods of operation under battery power. This larger battery is located outside the chassis of the *dataTaker 500/600* series loggers, but generally within the enclosure which houses the logger. This external battery is connected to the two screw terminals labelled Batt + and Batt – . These terminals are illustrated in Figure 7 below.

An external battery is connected to the Batt + and Batt – terminals is recharged whenever the logger is externally powered.

Alkaline Backup Battery

The backup battery for the *dataTaker* data loggers can also be a 9 Volt alkaline D cell battery pack. These typically have 17-20 Ahr capacity, and so can provide longer term operation of the logger when it is powered solely from the battery.

Standard alkaline batteries are not rechargeable, and so there are separate battery terminals for lead acid and alkaline batteries. There is no charging current output on the alkaline battery terminals whenever the logger is powered from external sources

The 9 Volt Alkaline battery is connected to the *dataTaker 50* via the battery screw terminals on the top of the logger, labelled Alkaline + and Alkaline – (see Figure 2).

The 9 Volt Alkaline battery is connected to the *dataTaker 500/600* series loggers via a three pin internal battery connector on the upper printed circuit board of the logger (See Figure 4). The internal battery connector is accessed by removing the battery cover. The connector is polarized. and care must be taken when installing the alkaline battery that the battery connects to the Alkaline + and Negative pins.

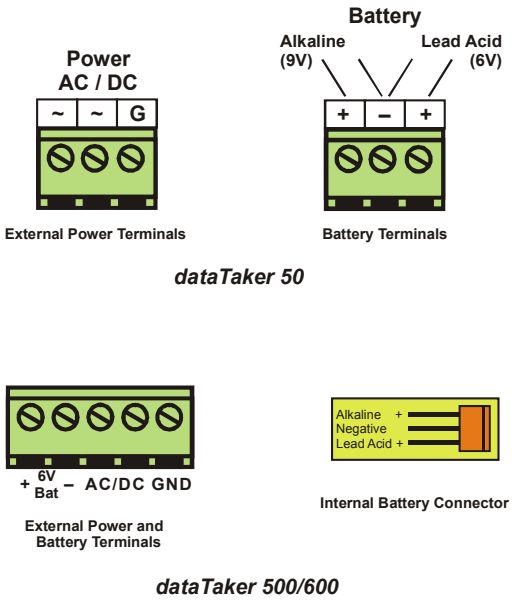


Figure 7 – *dataTaker* Power Supply Connections

Caution : The 9 Volt Alkaline battery must not be connected to *dataTaker 500/600* series loggers via the external Bat + and Bat – screw terminals if an external power supply is also used. A charging current is output from these terminals whenever these loggers are externally powered, that will destroy the alkaline battery.

The alkaline batteries are available from your *dataTaker* supplier.

Battery Life

The *dataTaker* battery will power the logger for varying periods, depending on the battery type and operating conditions.

A general guide to battery life for the 1.2 AHr and 4 AHr lead acid gel cell and the alkaline battery pack is presented in the following table for different sampling intervals. The table assume that automatic selection of low power mode is enabled, and that the *dataTaker* is allowed to drop to the low power or sleep mode between scans.

Battery life is substantially reduced if the *dataTaker* is used to continuously monitor alarm inputs, continuously scan for statistical data, power sensors or excite sensors.

Sampling 10 Channels	1.2Ah Lead Acid Battery	4Ah Lead Acid Battery	17 Ah Alkaline Battery
Continuously	10 hours	30 hours	6 days
1 minute	15 days	45 days	200 days
15 minutes	60 days	180 days	800 days
1 hour	90 days	300 days	900 days

Using the Battery Life Chart

The Battery Life Chart illustrated in Figure 8 below can be used to estimate battery life.

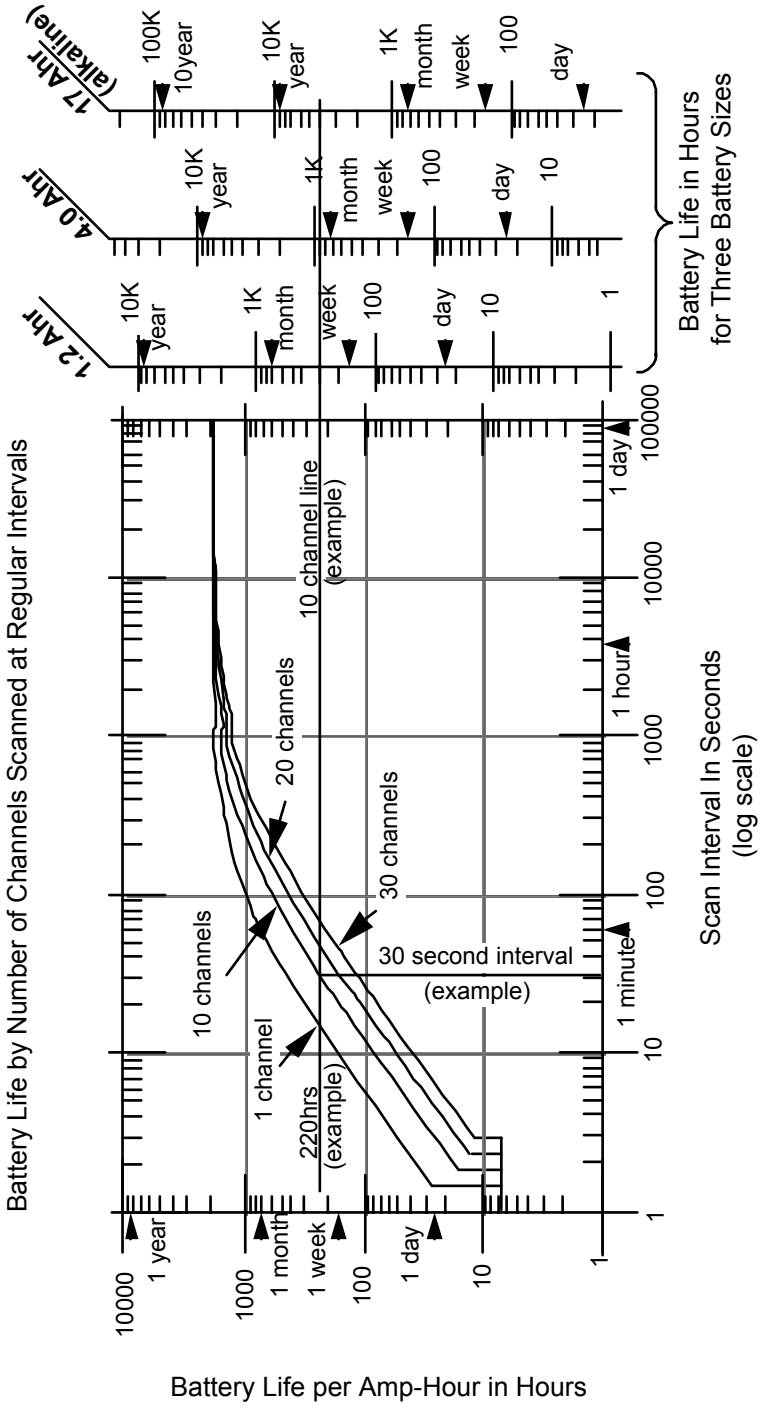


Figure 8 - *dataTaker* Battery Life Chart

Battery life is estimated for different operating conditions as follows

Establish the scan interval, and find it on the Scan Interval axis. The chart shows an example of 30 seconds.

Project a vertical line to the curves. These curves correspond to the different numbers of channels. Interpolate if necessary. Channels are assumed to be analog, however for this estimation a digital channel is equivalent to one quarter of an analog channel. The chart shows an example of 10 channels.

From the curve intersect project a horizontal line to the right across three scales, which correspond to standard *dataTaker* battery sizes, where the battery life can be read. The example in the chart shows that the battery life would be 300, 800 and 4000 hours for the 1.2, 4 and 17 Ahr batteries respectively.

If a non-standard battery is employed, then the chart axis on the left is used and the battery life is equal to the scale value multiplied by the battery's amp-hour rating (220 hours for the example).

A Few Details

The *dataTaker* takes approximately 300mS to wake up, 40mS to sample each analog channel, 10mS to sample each digital channel and 10 to 50mS for each calculation.

Also the *dataTaker* must also wake at least every 18 minutes for time keeping, and will not sleep unless it can sleep for at least one full second. Current draw is 120mA while awake and less than 0.4mA while asleep.

Converting LED

The *dataTaker* has a LED, which flashes when the *dataTaker* carries out an analog to digital conversion. The Converting LED flashes for a period of time equal to the number of channels being scanned during data acquisition or an autocalibration.

The *dataTaker 50* this LED is located near the external power supply screw terminals, while for the *dataTaker 500/600* series loggers the LED is located beside the memory card connector.

The Converting LED also flashes every second when the logger is either externally powered, or powered from the internal battery in normal power mode. This flashing coincides with the internal battery current being sampled to determine if the logger should select the low power mode.

Multiplexer Powering

The *dataTaker* has an option for switching off power to the analog input multiplexer while the logger is asleep. This can reduce current draw from the battery, thereby extending battery life.

However removing power reduces the input impedance of the solid state CMOS multiplexers of the *dataTaker 50, 500* and *600,* from millions of ohms to only a few hundreds of ohms. This can result in excessive current flow from sensors, which may not be acceptable.

The *dataTaker 505, 605, 515* and *615* loggers do not have this option.

Switching the Multiplexer Power for the dataTaker 50

The multiplexer powering for the *dataTaker 50* is set by switch 4 of the DIP switch located under the top cover. This switch is labelled Power.

When this switch is in the ON position (default), then power is continually applied to the positive side of the multiplexer. The negative side switches from –4 Volt to ground in the sleep mode. This option will only safeguard sensors with a positive output relative to ground.

When this switch is in the OFF position, both the positive and the negative multiplexer power supplies are removed while the logger is in the sleep mode.

Switching the Multiplexer Power for the dataTaker 500 dataTaker 600

The *dataTaker 500* and *600* multiplexer powering option is selected by a jumper located midway along the memory card side of the top circuit board. The jumper is labelled Mux Power, and has two positions which are labelled as SW (switched) and USW (unswitched).

When the jumper is in the USW position (default), then power is continually applied to the multiplexer.

When the jumper is in the SW position, the multiplexer power is switched off while the logger is in the sleep mode.

Setting the Mains/Line Frequency

The local mains or line frequency is defined to the *dataTaker* via bit 1 of the DIP switch (See Figure 9). The mains frequency options are 50 Hz for Australia, Europe, Asia, New Zealand, etc, and 60 Hz for Canada and USA.

For 50Hz operation, bit 1 of the DIP switch should be set OFF, while for 60Hz operation bit 1 should be set ON.

The mains frequency setting is used by the analog to digital converter of the *dataTaker* to synchronize the analog conversion period to the local mains frequency. This maximizes the rejection of noise due to mains or line induced hum.

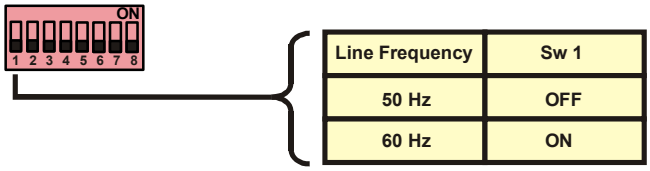


Figure 9 – The Line Frequency DIP Switch

Powering Sensors from the *dataTaker*

The *dataTaker* is able to power sensors, relays, etc. from a regulated 5 Volt DC power supply terminal.

In the *dataTaker 50* the 5 Volt sensor power supply terminal is located beside analog input channel 5, and is labelled +5V sw.

In the *dataTaker 500/600* series loggers the 5 Volt sensor power supply terminal is located near the memory card socket, and is labelled +5V sw.

This power supply can source a current of up to 100 mA.

Caution : If more than 100 mA current is drawn from the 5 Volt sensor power supply terminal, then the performance of the *dataTaker* will be affected.

The 5 Volt sensor power supply is automatically switched OFF when the *dataTaker* drops to the low power or sleep mode.

The 5 Volt sensor power supply is automatically switched ON when the *dataTaker* is in the normal power or wake mode.

If the 5 Volt sensor power supply is used to power sensors when the *dataTaker* is powered by the internal battery, then battery life will be reduced.

The circuit schematic for the *dataTaker* 5 Volt switched power supply is illustrated below

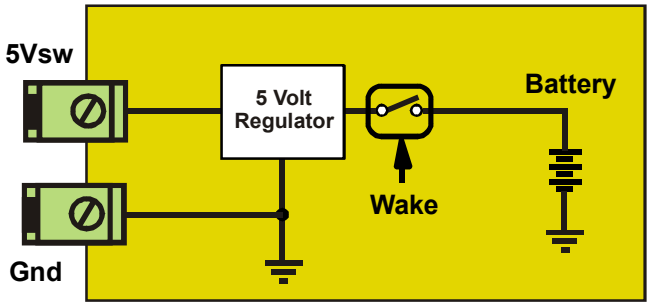


Figure 10 – *dataTaker* 5 Volt Switched Sensor Power Supply

Powering Sensors

The sensor(s) to be powered from the 5 Volt sensor power supply are connected between the +5V sw screw terminal and any GND screw terminal.

The sensors are powered down whenever the *dataTaker* drops to the low power or sleep mode, and are powered up when the *dataTaker* wakes and powers up to the normal power mode.

If sensors powered from the 5 Volt sensor power supply have long warm up times, then a warm up delay can be introduced between the switching of the 5 Volt sensor power supply and the commencement of scanning of the sensors.

The *dataTaker* normally pauses briefly between waking, at which time the 5 Volt sensor power supply is switched ON, and commencing to scan the input channels. This wake delay can also be used to provide time for sensors to warm up.

This wake delay is defined to the *dataTaker* by the Parameter16 command in the range of 1 to 255 (See Section III – Parameter Commands). Parameter16 is specified in increments of 100's of mS.

Using DeTransfer, the command for example

```
P16=30
```

sets the wake up delay to 3000 mS or 3 seconds.

The wake up delay can be set in the range of 100 to 25,500 mS or 0.1 to 25.5 seconds. The default wake up delay is 100 mS.

The currently defined wake up delay can be determined by the command

```
P16
```

Powering Relays

Small 5 Volt relays can be powered directly by the *dataTaker*, using the 5 Volt sensor power supply.

The relay coil is connected between the +5V sw screw terminal and the digital output channel being used to control the relay.

The Serial Interfaces

All *dataTakers* have an RS232 serial interface which is used for communication with the host computer. This serial interface is labelled RS232 COMMS on the body of the *dataTaker*.

The *dataTaker 500/600* series loggers also have an RS485 standard serial interface for communication between up to 32 *dataTaker*s connected into a local area network. This interface is labelled the NETWORK interface on the body of the logger.

The RS232 COMMS serial interface of any *dataTaker 500/600* series loggers in a local network is used by a host computer to communicate with all loggers in the network, via the respective NETWORK interfaces. The relationship between the serial interfaces of the *dataTaker 500/600* series loggers is illustrated in Figure 11.

The dataTaker *50* does not have an RS485 NETWORK interface.

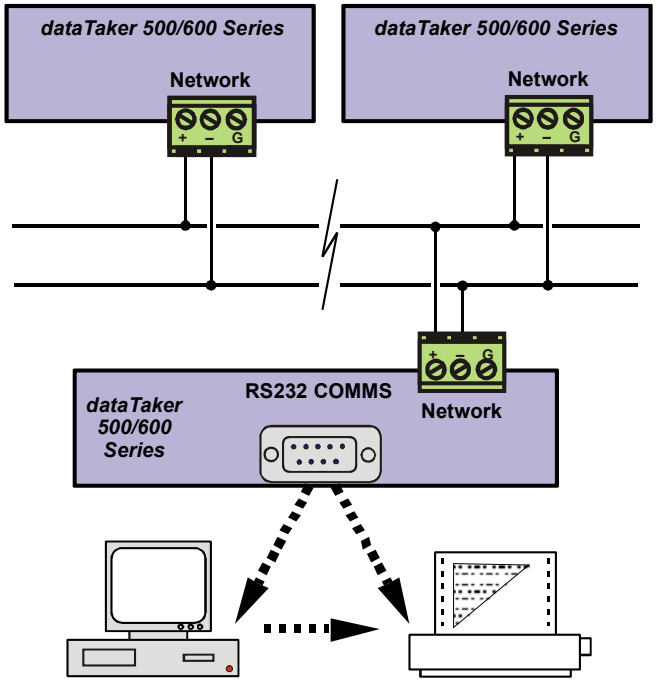


Figure 11 – Relationship Between the RS232 COMMS and NETWORK Serial Interfaces of *dataTaker 500/600* Series Loggers

The RS232 COMMS Serial Interface

The *dataTaker* data loggers communicate with a host computer via the RS232 COMMS serial interface. The interface is configured to the RS232 serial interface standard, and is implemented as a 9 pin female D connector for all *dataTaker* models.

The RS232 COMMS serial interface of all *dataTaker Series2* and *Series 3* models is electrically isolated to 500 Volts, which provides for

protection from damaging electrical currents passing between the host computer and the logger

reduces interference from electrical noise

eliminates the possibility of host computer to logger ground loops being established which will cause erroneous analog measurements

Note: The RS232 COMMS serial interface of *dataTaker Series1* models is not normally electrically isolated, however can be isolated by modifying the communications cable (see below).

The RS232 COMMS serial interface of the *dataTaker*s supports 300, 1200, 2400, 4800 and 9600 baud rates. The communications baud rate is set by the Baud Rate switches of the DIP switch (see below).

Communication parameters for the RS232 COMMS serial interface are permanently set to 8 data bits, 1 stop bit, and no parity.

RS232 COMMS Serial Interface Connector

The RS232 COMMS serial interface of the *dataTaker* has a female 9 pin D connector. The connector is labelled RS232 COMMS. Pin assignment for the RS232 COMMS serial interface connector of the *dataTaker* is illustrated in Figure 12 below

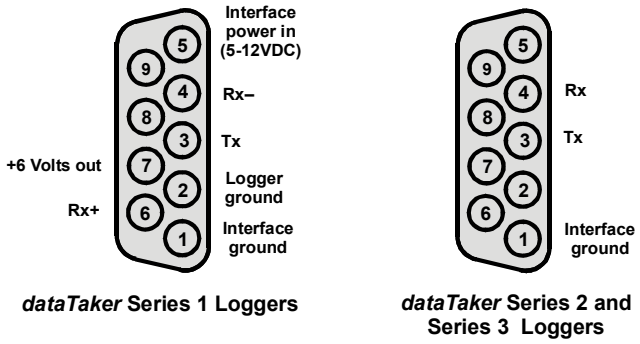


Figure 12 – RS232 COMMS Serial Interface Connector Pin Assignments

(Viewed looking at face of connector)

RS232 COMMS Serial Interface - *dataTaker* Series 1 Only

The RS232 COMMS interface of Series 1 *dataTaker* models is not electrically isolated. For most applications the serial interface does not need to be electrically isolated.

The serial interface is normally powered from the internal power supply of the *dataTaker* by installing links on the back of the *dataTaker* connector of the communications cable as follows

a link must be installed between pin 2 (*dataTaker* ground) and pin 1 (serial interface ground) of the 9 pin connector

a link must be installed between pin 7 (*dataTaker* internal 6 Volt power supply) and pin 5 (serial interface power input) of the 9 pin connector

The communications cable supplied with the *dataTaker* has these links installed.

The links on the communications cable connector for powering the serial interface from the *dataTaker* power supply are shown below in Figures 14 and 15.

Isolating the RS232 COMMS Serial Interface - *dataTaker* Series 1 Only

The RS232 COMMS serial interface of Series 1 *dataTakers* can be electrically isolated from the remainder of the *dataTaker* circuitry. This may be done for a number of reasons including

to protect the host computer from any damaging voltages accidentally applied to the logger channels

to reduce effects of electrical noise on the communications link

to eliminate ground loops between the host computer ground and the data logger ground will cause erroneous analog measurements

The receiver output and the transmitter input to the serial interface circuits are both optically coupled to the remainder of the *dataTaker* circuitry.

In order to maintain electrical isolation of the serial interface from the remainder of the *dataTaker*, the serial interface must be powered from an external power supply. The *dataTaker* and the serial interface must not share the same ground.

Isolation is achieved by removing the powering links installed on the back of the *dataTaker* connector of the communications cable (see above), and providing power to the interface from an external source.

The schematic for isolating and externally powering the serial interface of the *dataTaker* is illustrated in Figure 13 below.

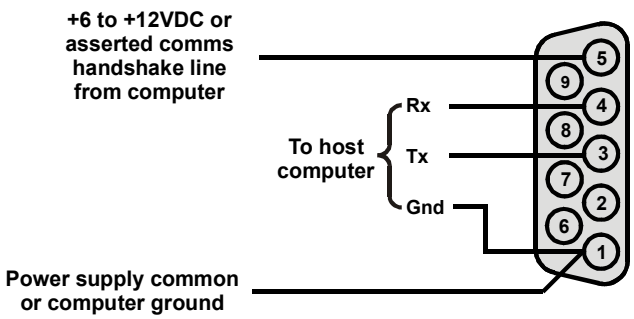


Figure 13 – Externally Powering the RS232 COMMS Serial Interface of *dataTaker* Series 1 Loggers

The *dataTaker* RS232 COMMS serial interface requires a power supply of 5 to 12 Volts DC at 20 mA. An external power supply for the serial interface is connected as follows

the +ve line from the external power supply is connected to pin 5 (serial interface power input) of the 9 pin connector

the –ve line from the external power supply is connected to pin 1 (serial interface ground) of the 9 pin connector

Reverse polarity protection for externally powering the serial interface is provided by a series diode on pin 5. The negative rail required for the RS232 signal voltage swings is generated by a DC–DC converter within the isolated section of the serial interface.

External power for the serial interface is usually supplied via an extra wire pair in the communication cable. The power requirement of the serial interface is dependent on the transmission speed and

distance. However +6 Volts at 20 mA is sufficient for most communications. An asserted Request To Send (RTS) line for many computers is able to supply this power requirement.

Communications Cable

The communications cable supplied with the *dataTaker* is used to connect a computer to the logger. The cable is wired to suit Windows computers which have a 9 pin serial interface connector.

Some notebook computers now are supplied without any RS232 serial interfaces. In this case an RS232 serial interface can be provided via a USB to RS232 adaptor, or via a docking module.

Older style Windows computers may have a 25 pin connector on the COM port. There is a cable schematic below for these computers.

Cable Type

The use of shielded communications cables is recommended where the cables are long, or where environmental electrical noise is likely to be high. If a shielded cable is used, the shield should be connected to the Frame Ground or Ground pin of the computer serial interface connector.

Short communications cables do not need to be shielded, and can be of any conveniently available cable type.

Long Cables

The RS232 communications standard specifies a maximum cable length of 15 metres (17 yards). However longer cables can be used, and lengths up to 150 metres (170 yards) are possible.

The limitation of communications cable length is determined by cable resistance and capacitance, the level of local electrical noise, the communications baud rate, and the sensitivity of the host serial interface receiver. Longer cables usually require a lower baud rate.

Communications Cable for Windows Computers

Communication cable schematics are illustrated in the following Figures for Windows computers with 25 pin and 9 pin COM port connectors.

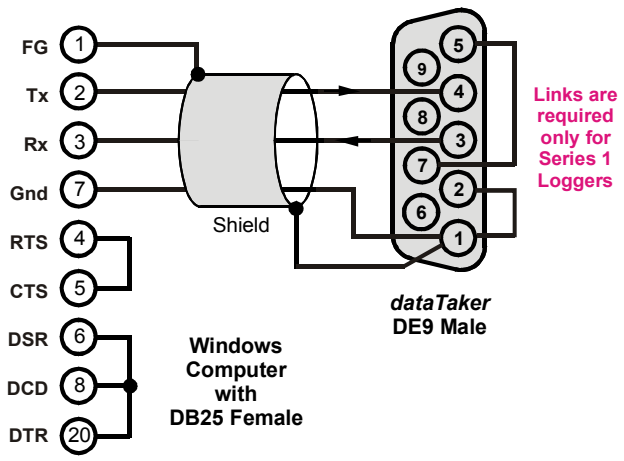


Figure 14a – RS232 Communications Cable for Windows Computers with 25 pin COM Port Connectors

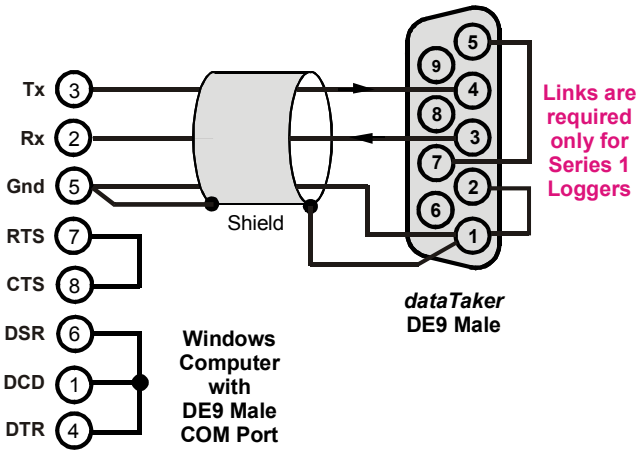


Figure 14b – RS232 Communications Cable for Windows Computers with 9 pin COM Port Connectors

Communication Cable for Apple Macintosh Computers

Apple Macintosh computers have an RS422 serial interface with a female 8 pin Mini DIN connector, mounted in the rear panel of the computer. This interface can also be used as an RS232 serial interface.

A communications cable wiring schematic for connecting an Apple Macintosh series computer with the *dataTaker* is shown in Figure 15 below.

Figure 15 also details the 8 pin Mini DIN interface connector of the Apple Macintosh series computers. A communication cable for the Macintosh computers is available from your *dataTaker* supplier.

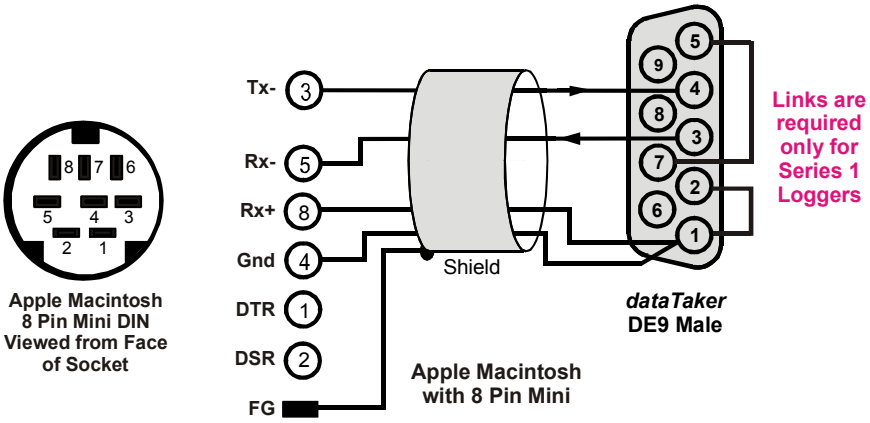


Figure 15 – RS232 Communications Cable for Apple Macintosh Computers

RS232 Control and Handshake Lines

The RS232 COMMS serial interface of the *dataTaker* does not support hardware handshake lines (RTS, CTS, DSR, DCD, DTR, etc.).

Setting the RS232 Communications Parameters

All *dataTaker* data loggers are shipped with the communications parameters for the RS232 COMMS serial interface pre-set to the following settings

– 9600 baud
– no parity checking
– 8 data bits
– 1 stop bit

The communications baud rate of the *dataTaker* is set by the Baud Rate switches of the DIP switch to 300, 1200, 2400, 4800 or 9600 baud.

The communication parameters of the RS232 COMMS serial interface are permanently set to 8 data bits, 1 stop bit, and no parity checking. These settings suit all types of host computers, and cannot be changed.

The Baud Rate

Whenever the host computer or terminal and the *dataTaker* are communicating, they must transmit and receive information at compatible rates. The receiver of each device must be able to receive information at the rate at which the other device is transmitting the information.

The rate of information transfer between two communicating devices is referred to as the baud rate, and is defined in terms of the number of pulses or bits of information transferred per second. The baud rates must be the same for each direction of data transfer.

The *dataTaker* models support baud rates of 300, 1200, 2400 and 9600 baud

The choice of baud rate for serial communication with the host computer depends on a number of parameters

the length of communications cable

the level of local electrical noise

the sensitivity of the computer interface receiver

At lower baud rates the maximum scanning rate of the analog input channels by the *dataTaker* will be reduced if the data is returned as the channels are scanned. This is because of the priority to transmit acquired data to the host rather than to commence a new scan (See Section III – Priorities for Returning and Logging Data).

In general practice a baud rate of 9600 baud should be used.

The baud rate of the *dataTaker* is set by the Baud Rate switches of the DIP switch. The baud rate settings available are illustrated in Figures 16 and 17 below.

The host computer must be set to the same baud rate as the *dataTaker*. DeTransfer and DeLogger have automatic baud rate matching, and will set the baud rate of the computer to that of the *dataTaker* when connection is being made or requested.

The Number of Data Bits

The *dataTaker* transfers all information as ASCII characters. Each ASCII character is represented by a unique series of data bits, or a bit pattern, comprised of 0's and 1's (See Appendix A - ASCII Character Table).

The ASCII character codes may be made up of 7 or 8 data bits. For example the ASCII representation of the character Z is 1011010 as 7 bit code, and 01011010 as 8 bit code.

Whenever the host computer and the *dataTaker* are communicating, both devices must transmit and receive the ASCII characters in compatible formats. In order to achieve this compatibility, both devices must be set to represent ASCII characters either as 7 or 8 bit code. This is referred to as the number of data bits.

The RS232 COMMS serial interface of the *dataTaker* is permanently set to 8 data bits. DeTransfer and DeLogger automatically set the computer to 8 data bits.

Parity Checking

Parity checking provides a means for error checking each ASCII character as it is received by a communicating device. If parity checking is enabled, a bit of 0 or 1 is added after the last bit of the ASCII character. The parity bit is set according to the number of 1's in the ASCII character, and can be defined to return 0 for an odd number of 1's (odd parity) or 0 for an even number of 1's (even parity).

When two devices are communicating, they must both be set for the same parity checking. This enables the receiving device to interpret the parity bit sent by the transmitting device.

The RS232 COMMS serial interface of the *dataTaker* has the parity checking permanently disabled. DeTransfer and DeLogger automatically set the computer to no parity checking.

The Number of Stop Bits

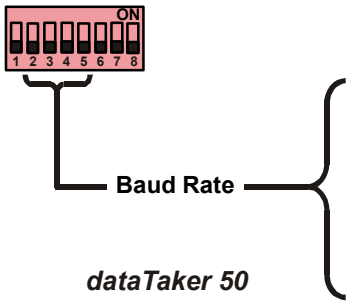
The stop bits are added to each character by the transmitting device, following the parity bit if parity checking is enabled, or following the last data bit if parity checking is disabled. The stop bits provide a means of character synchronization between the transmitting device and the receiving device. The number of stop bits can usually be set to 1 or 2.

The RS232 COMMS serial interface of the *dataTaker* is permanently set for 1 stop bit. DeTransfer and DeLogger automatically set the computer to 1 stop bit.

The Baud Rate DIP Switches

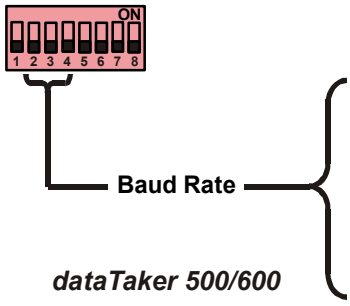
The baud rate for the *dataTaker* RS232 COMMS serial interface is set by the Baud Rate switches of the DIP switch. The DIP switch of the *dataTaker* is accessed by removing the top cover of the logger.

The range of baud rates available for the RS232 COMMS serial interface, and their selection, is illustrated in Figure 16 & 17 below



Baud Rate	Sw 2	Sw 3	Sw 5
1200	off	off	x
9600	off	on	off
300	off	on	on
2400	on	off	x
4800	on	on	x

Figure 16 – The RS232 COMMS Baud Rate DIP Switch - *dataTaker 50*



Baud Rate	Sw 2	Sw 3	Sw 4
1200	off	off	x
9600	off	on	off
300	off	on	on
2400	on	off	x
4800	on	on	x

Figure 17 – The RS232 COMMS Baud Rate DIP Switch - *dataTaker 500/600* Series

The factory setting for the Baud Rate is 9600 baud.

The Baud Rate switches are read periodically, and so if the baud rate is altered while the *dataTaker* is operating, the change has an immediate effect.

Caution : The DIP switches should be set using a pointed implement such as a small screwdriver, a ball-point pen tip or similar. Pencils should not be used because of the risk of graphite particles entering the switch mechanism, and producing short circuits.

Preparing The Host Computer

The computer used for supervising the *dataTaker* must have a serial interface which supports the RS232 communication standard.

The baud rate of the computer must be set to be the same as that for the *dataTaker*, and the start bits, stop bits and parity must be set to 8, 1 and None respectively.

If you have a notebook computer which has no RS232 serial interface(s), then you will need a docking module for your particular computer, or a USB to RS232 adaptor, which have an RS232 interface.

The DeTransfer software package supplied with the *dataTaker* provides for easy communications with the logger. DeTransfer is used to send commands to the *dataTaker*, and to receive data back from the *dataTaker*.

A large number of general purpose communications programs can also be used to communicate with the *dataTaker*.

DeLogger also provides an easy method for supervising the *dataTaker*, and also has facilities for generating data files, graphs and reports.

Password Protection

The *dataTaker* has an optional password protection scheme. Whenever a password is defined, no communications are possible until the defined password is entered.

Password protection is most useful when the *dataTaker* is connected to a modem. The use of a password eliminates the possibility of line noise being interpreted as commands during the establishment of a call.

The Password protection also protects against unauthorized access to the *dataTaker*.

Using DeTransfer, the password is set by the following general command for example

```
PASSWORD="password "
```

Where the *password* may be any string of up to 10 characters, and is case sensitive.

To establish communications when a password is defined, enter the password followed by a carriage return. This will sign the user on, and communications can proceed until the SIGNOFF command is issued, or if there is no communications activity for 300 seconds (5minutes).

The period before automatic signoff is defined by the Parameter14 command, defines the signoff delay period in seconds (see Section III – Parameter Commands). For example the command

```
P14=15
```

sets the automatic signoff delay to 15 seconds.

To remove a defined password, firstly establish communications by entering the current password and then enter the command

```
PASSWORD=""
```

The NETWORK Interface

Up to 32 *dataTaker 500/600* series loggers can be connected into a network, where individual loggers are linked together via their NETWORK interfaces.

Note : The *dataTaker 50* does not provide networking capability.

The network for *dataTaker 500/600* series loggers can be up to 1000 meters (1100 yards) in length. The maximum length of the network cabling is limited only by the attenuation of the communications signals. Therefore data quality cable should be used where long distance networks are to be established, while simple telephone cable is sufficient for short distance networks.

The NETWORK interface of the *dataTaker 500/600* series loggers is implemented as a RS485 serial interface, and communicates at a fixed 1200 baud. The interface is not electrically isolated, and is powered from the logger.

There are no master or slave units in a *dataTaker* network. The host connects to the network via the RS232 COMMS of any *dataTaker* in the network. All *dataTaker*s in a network have equal status, and each is identified by its unique network address.

A proprietary clash detection and error correcting protocol operates to ensure error free transmission on the network. However this can slow network communications if a number of loggers are trying to communicate simultaneously. The network should be managed such that only one logger is communicating at any one time.

Wiring a *dataTaker* Network

The NETWORK interface has screw terminals located beside the 9 pin D connector for the RS232 COMMS interface. The screw terminals are labelled NET + and NET–.

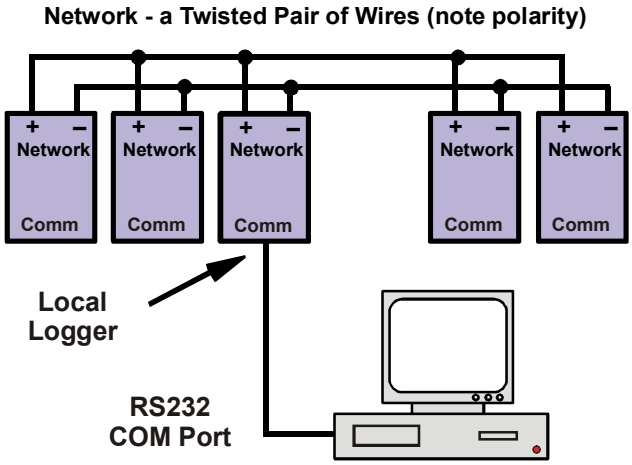


Figure 18 – Network Cabling Schematic for *dataTaker 500/600* Series Loggers



Network Terminals

Figure 19 – Network Terminals for *dataTaker 500/600* Series Loggers

A *dataTaker* network is established by connecting the positive terminals of each logger in the network together, and connecting the negative terminals of each logger together.

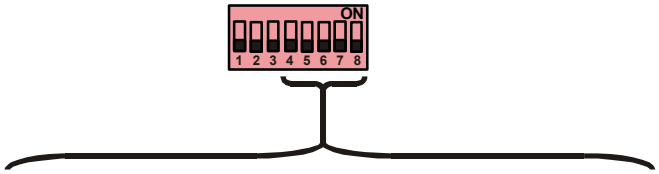
It is recommended that shielded network cables be used. The shield continuity between successive *dataTaker*s is maintained via the GND terminals of the loggers. The network cable shields should be grounded at one point only, and care should be taken that the shield is not grounded at any other point.

Network Address

Each *dataTaker* in a network must have a unique network address. The network address is set via the Network Address switches of the DIP switch. The DIP switch is located behind the memory card connector, under the battery cover.

The Network Address switch settings of the DIP switch for the 32 available addresses are illustrated in Figure 20 below.

The use of the network address when sending commands to a group of *dataTaker*s that are connected in a network is discussed in the Section III – *dataTaker* Networks.



Address	Sw4	Sw5	Sw6	Sw7	Sw8	Address	Sw4	Sw5	Sw6	Sw7	Sw8
0	off	off	off	off	off	16	on	off	off	off	off
1	off	off	off	off	on	17	on	off	off	off	on
2	off	off	off	on	off	18	on	off	off	on	off
3	off	off	off	on	on	19	on	off	off	on	on
4	off	off	on	off	off	20	on	off	on	off	off
5	off	off	on	off	on	21	on	off	on	off	on
6	off	off	on	on	off	22	on	off	on	on	off
7	off	off	on	on	on	23	on	off	on	on	on
8	off	on	off	off	off	24	on	on	off	off	off
9	off	on	off	off	on	25	on	on	off	off	on
10	off	on	off	on	off	26	on	on	off	on	off
11	off	on	off	on	on	27	on	on	off	on	on
12	off	on	on	off	off	28	on	on	on	off	off
13	off	on	on	off	on	29	on	on	on	off	on
14	off	on	on	on	off	30	on	on	on	on	off
15	off	on	on	on	on	31	on	on	on	on	on

Figure 20 – Setting the Network Address of the *dataTaker 500/600* Series Loggers

The Network and Modems

The *dataTaker* network can be extended by modems or radio modems that are able to operate at 1200 baud and can automatically 'turn around' (change direction of data transfer). This is an important issue for radio modems, where changing from transmit mode to receive mode can take as long as 500 mS.

The *dataTaker* network turn around time is defined by the Parameter7 command in increments of 14 seconds. For example the command

```
P7=22
```

sets the turnaround time to 300 seconds.

The *dataTaker* does not issue an electrical signal or a message preamble to indicate turn around. The modem must detect when the *dataTaker* is sending data, and rapidly switch to the transmit mode.

Networking and Low Power Mode

When the *dataTaker* is asleep, the logger will not wake when network activity begins. To ensure proper operation of loggers in a network, the following should be done

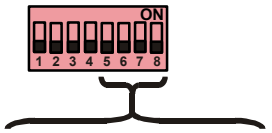
the loggers can be kept awake by setting Parameter15=2

the Wake terminals of all networked loggers can be linked together such that when one logger wakes they all wake

diode couple the negative network wire between loggers to the Wake terminal of each logger. Any activity on the negative network line will then wake all loggers. The diode is necessary to prevent the 10 Hz signal which is output from the Wake terminal while the logger is awake, from entering the network.

dataTaker 50 Address

The *dataTaker 50* can be assigned an address, which provides a form of password security to the logger.



Address	Sw5	Sw6	Sw7	Sw8
0	off	off	off	off
1	off	off	off	on
2	off	off	on	off
3	off	off	on	on
4	off	on	off	off
5	off	on	off	on
6	off	on	on	off
7	off	on	on	on
8	on	off	off	off
9	on	off	off	on
10	on	off	on	off
11	on	off	on	on
12	on	on	off	off
13	on	on	off	on
14	on	on	on	off
15	on	on	on	on

Figure 21 – Setting the Address of the *dataTaker 50*

The address can be set in the range of 0 to 15 by the four Address switches of the DIP switch under the top cover of the logger.

The use of the *dataTaker 50* address when sending commands is the same as when commanding a network of *dataTaker 500/600* series loggers. This is discussed in the Section III – *dataTaker* Networks.

Analog Processing

The *dataTaker* range of data loggers have 2 different types of analog input channel multiplexers, solid state (CMOS) multiplexer or relay multiplexer.

The models of the *dataTaker* data loggers which have each of these multiplexer types are as follows

Solid State (CMOS) Multiplexer
- *dataTaker 50, 500* and *600*

Relay Multiplexer
- *dataTaker 505, 605, 515* and *615*
- Channel Expansion Module (CEM-AD)

Throughout this section reference will be made to solid state or relay multiplexing, rather than to individual *dataTaker* models. Confirm from the above list which *dataTaker* model you are using.

The analog processing circuits of the *dataTaker* consist of four principal sections

a multiplexer for selecting the analog input channels for sampling

a multiplexer for selecting the measuring circuits and input configuration appropriate to each input signal type

a precision instrumentation amplifier with programmable gain

an analog to digital converter (ADC)

This chapter describes the operation of each of these sections, and the degree of influence that the user has over how each of these sections function.

Analog Input Multiplexer

The analog input channel multiplexer for the *dataTaker* is an array of either solid state (CMOS) analog gate devices or small electromechanical relays, that provide the analog input channel switching.

Each of the analog input channels is in effect a five pole switch, for which each pole is normally open.

These switches select the analog input signal to be measured, configure the channel to measure the input signal, and direct any required support to the sensor.

The switches for each of the analog input channels has the following functions

Select the +ve input

Select the -ve input

Direct the excitation source to the channel

Select the 'active' end of the 100.0 Ohm current shunt.

Select the 'grounded' end of the 100.0 Ohm current shunt.

A simplified schematic for the analog multiplexer switching of a single analog input channel of the *dataTaker* is illustrated in Figure 22.

There are additional multiplexer or Selector devices in the analog input circuits, which select the various support circuits during measurement of an input signal, and select the various internal calibration channels during an autocalibration.

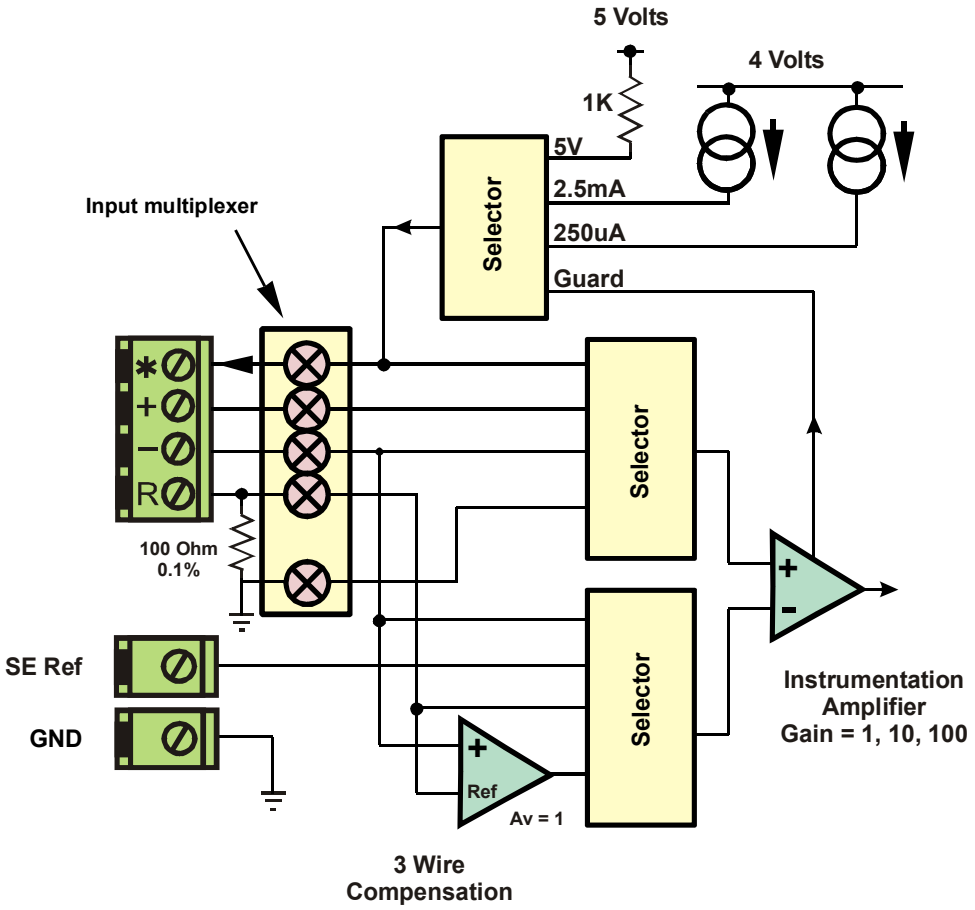


Figure 22 - *dataTaker* Simplified Analog Input Channel

Each analog input channel also has input protection resistors, which are not shown in Figure 22 for simplicity.

The major difference between the two multiplexer types are the common mode voltage and the voltage ranges that the *dataTaker*s with each multiplexer type can measure directly.

Solid State Multiplexer

The *dataTaker 50*, *dataTaker 500* and *dataTaker 600* data loggers have a solid state multiplexer, which uses CMOS 4052 quad analog gates.

The *dataTakers* which have the solid state multiplexer have three decades of voltage input ranges of ±25.000 mV, ±250.00mV and ±2500.0 mV.

The 4052 devices have a maximum input voltage of approximately 20 Volts, and will withstand common mode voltages of up to ±3.5 Volts.

The multiplexer also provides protection for the *dataTaker* against excessive input voltages, because the 4052 devices will usually fail safe when damaged.

The multiplexer devices are socketed for easy replacement – see Appendix D for details of replacing the multiplexer devices in the field.

Relay Multiplexer

The *dataTaker 505*, *dataTaker 515*, *dataTaker 605* and *dataTaker 615* data loggers all have a relay multiplexer, which uses high precision electromechanical relays.

The *dataTaker*s which have the relay multiplexer also have internal input attenuation which provides two categories of voltage input range as follows

when the internal attenuators are not selected, there are three decades of low voltage input ranges of ±25.000 mV, ±250.00mV and ±2500.0 mV

when the internal attenuators are selected, there are three decades of high voltage input ranges of ±7.000 V, ±70.00 V and ±100.0 V

The relay multiplexer has high withstanding input voltages when the contacts are open as follows

1.5 KVolts for 10 µS

500 Volts for 50 mS

100 Volts indefinitely

Instrumentation Amplifier

The precision instrumentation amplifier of the *dataTaker* is a high performance device, with three programmable gain stages that provide the three decade input ranges of the analog input channels. The instrumentation amplifier has gains of 1, 10, and 100 that correspond to voltage input ranges.

The *dataTaker*s with the solid state multiplexer have three decades of voltage input ranges of ±25.000 mV, ±250.00mV and ±2500.0 mV.

The *dataTaker*s with the relay multiplexer also have internal input attenuation which provides two categories of voltage input range as follows

when the internal attenuators are not selected, there are three decades of low voltage input ranges of ±25.000 mV, ±250.00mV and ±2500.0 mV

when the internal attenuators are selected, there are three decades of high voltage input ranges of ±7.000 V, ±70.00 V and ±100.0 V

Autoranging by Instrumentation Amplifier

The programmable gains of the instrumentation amplifier are automatically selected by the *dataTaker*, according to the level of each input signal to be measured. It is not necessary for the user to define ranges for measuring analog input signals.

During operation the *dataTaker* manages the instrumentation amplifier gain as follows

when the *dataTaker* is initially programmed to scan a group of analog channels, each channel is set to an initial gain which is optimal for the particular input signal type being measured.

as each input channel is sampled, the analog to digital converter reads the channel at the previously defined gain. If the first sample is not within this range, then the gain is changed accordingly and the channel is sampled again. This process continues until an in-range or over-range sample is obtained.

when an in-range sample is obtained, the gain setting is saved for that channel, and becomes the initial gain setting when the channel is next sampled.

when an over-range sample is obtained, the gain is set to one and becomes the initial gain setting when the channel is next sampled. An over-range error is generated.

Autoranging occurs at thresholds of approximately 15% and 120% of the full scale for the selected gain, for a gain increase and gain decrease respectively.

The tendency for 'gain hunting' is reduced by a hysteresis of approximately 10% around the threshold.

When the input signal is greater than the full scale threshold of the highest gain range, then an ‘E11–input(s) out of range’ error message is generated. A value of -99999.9 units or +99999.9 units is returned, depending on the polarity of the input signal.

Locking the Gain

The instrumentation amplifier gain can be locked for individual analog input channels by the Gain Lock channel option, which is declared as a part of the individual channel specifications in the schedule lists of Schedules.

When the gain is locked, the instrumentation amplifier is preset to the defined gain setting and does not autorange. All measurements for that channel are made at the set gain. Locking the gain for a channel does not affect autoranging for other channels.

The gain is locked by a channel option command

channel(GLn)

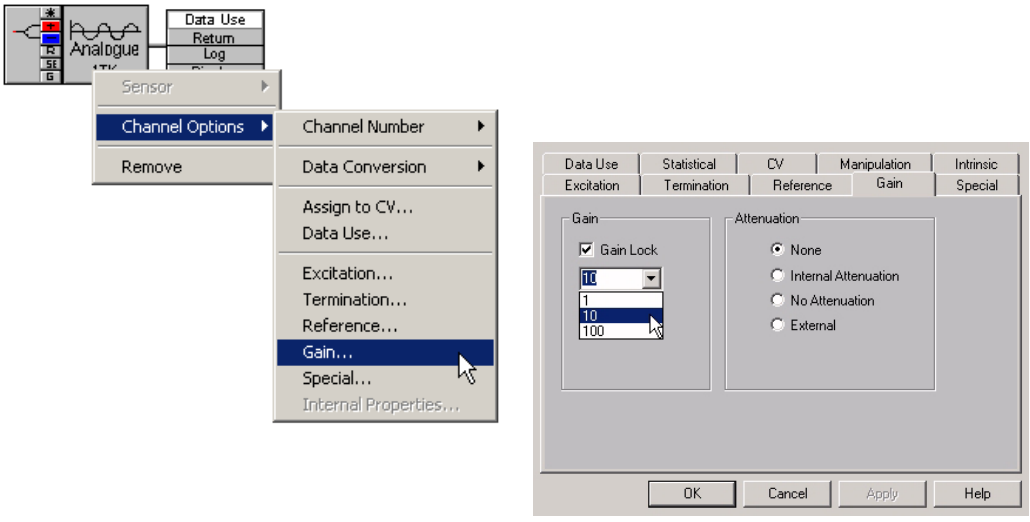
where GL specifies Gain Lock, and n is the gain required.

The gain can be locked by direct command from DeTransfer as part of the channel specification in a schedule list. For example

```
R10M 1V(GL10)
```

will lock the instrumentation amplifier gain to 10 (±250.00 mV).

In DeLogger, the gain lock setting for each analog channel is specified in the Program Builder via Channel Options : Gain...



The Input Bias Current

The instrumentation amplifier of the *dataTaker* requires only a small input bias current, typically less than 10 nanoamps, which must be supplied by the sensor providing the input signal.

By default, the *dataTaker* terminates most voltage type inputs to ground via internal 1 MOhm resistors. These input termination resistors provide a path for the bias current. However the input termination resistors may be switched out of circuit if required (See Section II – The Analog Input Channels).

If a current path is not provided, then the common mode input range will be exceeded and erroneous readings will result.

Some sensors such as pH and specific ion glass electrodes have a high output impedance, and are unable to supply the required input bias current. Such sensors should therefore be connected to the *dataTaker* via a high impedance buffer amplifier.

However if the input signal source is either connected directly to the ground of the *dataTaker*, or is related to the ground of the *dataTaker*, and the common mode range is not exceeded, then there is generally no difficulty in providing the bias current.

The Analog to Digital Converter

The analog to digital converter of the *dataTaker* data loggers is a voltage controlled oscillator (VCO) type. This conversion technique has several advantages over other techniques as follows

the input is integrated over a programmable sampling period, providing high rejection of input noise

trade-offs are possible between speed and resolution

the frequency counter also allows measurement of frequency and period signals

The main limitation of VCO type analog to digital converters is the conversion speed.

However the VCO technique removes the need for cumbersome input signal filtering for mains or line hum. Mains or line induced hum is a prominent source of noise in most measurement systems, often producing serious measurement errors. The analog to digital converter of the *dataTaker* is designed to minimize errors produced by mains or line induced noise, by sampling the input channels over a period of one mains or line cycle.

The analog to digital converter of the *dataTaker* is optimized to the extent that while an input signal for a channel is being converted, the raw data for the previous channel is being calculated through to appropriate engineering units. Calculation time for each analog channel is approximately 5-10 mS depending on whether any linearization is required for thermocouples, RTDs, etc. or if polynomials have to be applied to the data.

Timing diagrams for the sampling of the analog input channels, the calculation of the primary data, and the linearization of supported sensor types, are presented in detail in the Appendix B at the back of this manual.

ADC Sample Period and Line Frequency

The analog to digital converter of the *dataTaker* data loggers is designed to sample an input channel for one mains or line cycle period. One mains or line cycle is a period of 16.67 mS for the 60 Hz line frequency, and 20 mS for the 50 Hz line frequency.

Sampling inputs over one line or mains frequency period maximizes rejection of line or mains hum that may have been induced into the sensor cables, etc. As a consequence of this technique, the conversion period or sampling rate of the analog to digital converter is a function of the local line frequency.

The local line frequency must be defined to the *dataTaker*, and is set via the Line Frequency switch of the DIP switch. The line frequency should be set to 50 Hz for Australia, Europe, Asia, New Zealand, etc, and to 60 Hz for Canada and USA.

For 50Hz operation the Line Frequency switch must be set OFF, while for 60Hz operation the Line Frequency switch must be set ON.

The DIP switch of the *dataTaker* is accessed by removing the top cover of the logger. The settings for the Line Frequency switch is illustrated below in Figure 23.

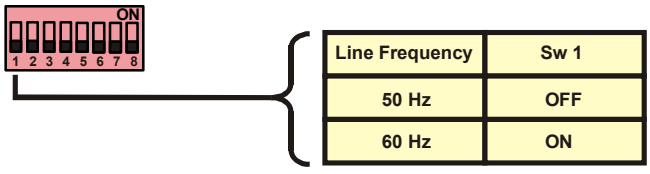


Figure 23 - The Line Frequency Switch of the DIP Switch

The *dataTaker* is factory set to a line frequency of 60 Hz (Line Frequency switch ON) for North America, and 50 Hz (Line Frequency switch OFF) for all other countries.

Increasing the ADC Sampling Rate

Increasing the setting of the line frequency for the *dataTaker* has the effect of decreasing the ADC sampling period, and as a consequence increases the ADC sampling rate.

It is not mandatory to set the mains or line frequency to the local line frequency if noise induced from line hum is not a problem. Therefore higher line frequencies may be selected in a quest for increased sampling speed.

However as the setting for the line frequency is increased and so the sampling period decreased, the resolution and accuracy of the ADC decreases.

The ADC resolution and the defined line frequency are related by the relationship

$$R = 3100000 / F_{line}$$

where the resolution R is the number of quantizing levels in the measuring range, and F_{line} is the line frequency.

This ADC resolution can also be expressed in terms of the input voltage resolution by the formula

$$\text{Input Resolution} = (V_{fs} \times F_{line}) / 3{,}100{,}000$$

where V_{fs} is the full scale voltage and F_{line} is the line frequency

For the 25 mV range and 50 Hz line frequency

$$\text{Input Resolution} = (0.025 \times 50) / 3100000$$
$$= 0.4\ \mu V$$

For the 2.5 Volt range and 60 Hz mains/line frequency

$$\text{Input Resolution} = (2.50 \times 60) / 3100000$$
$$= 48.3\ \mu V$$

The mains or line frequency can be set to frequencies other than 50 Hz or 60 Hz by the Parameter11 command (See Section III – Parameter Commands).

Using DeTransfer, the mains or line frequency can be set by the command for example

```
P11=100
```

which sets the line frequency to 100 Hz, and therefore sets the period for analog to digital conversion to 10 mS.

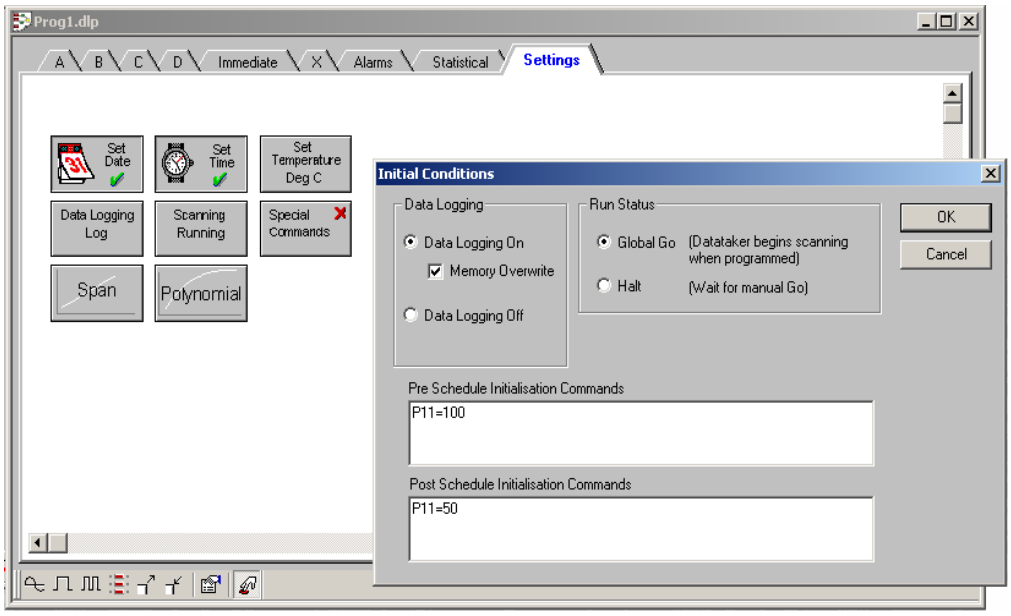
The current setting for the mains or line frequency can be read in DeTransfer by the command

```
P11
```

Using DeLogger, the mains or line frequency can be set and read using these same commands either

directly, in the Entry Screen of the Text View

as part of a program, in the Pre Schedule Initialization Commands under the Special Commands button of the Settings tab of the Program Builder



The line frequency can be set within the range of 48 Hz and 1000 Hz.

Increasing the line frequency beyond 200 Hz will not increase the conversion speed appreciably, because the *dataTaker* spends at least 5 mS per channel to scale the data into engineering units.

Timing diagrams for input channel sampling, the calculation of primary data and the linearization of supported sensor types are presented in the Appendix B at the back of this manual.

The ADC Settling Period

Whenever an input channel is selected for sampling, a settling period or delay is introduced to allow the input to stabilize before the sample is taken.

There are two reasons for varying the ADC settling period

- sensors with an output impedance of greater than 50 KOhm will require a longer settling time, because of the small internal capacitances of the *dataTaker* input channels and ADC
- the ADC settling period provides a means for modifying the sampling period or sampling rate. This is mainly useful when sampling one or two channels faster than once per second.

The ADC settling period is 10 mS by default, and may be set to any period in the range of 0 mS to 30000 mS in increments of 1 mS by the Parameter10 command (See Section III – Parameter Commands).

Using DeTransfer, the ADC settling period is set by the command for example

```
P10=20
```

sets the ADC settling period to 20 mS.

The current setting for the ADC settling period is read in DeTransfer by the command

```
P10
```

Using DeLogger, the ADC settling period is set and read using these same commands either

- casually in the Entry Screen of the Text View
- as part of a program, in the Optional Initialization Commands under the Special Commands button of the Settings tab of the Program Builder

Altering the ADC settling period alters the ADC sampling rate. This effect is separate from the effect on ADC sampling rate of changing the line frequency.

Autocalibration of the ADC

The analog to digital converter of the *dataTaker* is automatically recalibrated to compensate for any temperature or ageing drifts in the ADC circuits. Such drifts will result in measurement errors, particularly when low level signals are being measured.

The input zero voltage, the gain, and the current source for resistance measurement, are corrected by autocalibration. An autocalibration is carried out in response to changes in the measured internal zero voltage.

An autocalibration can also be forced at any time by enabling the Autocalibration Switch command /K (see below).

The Number of Calibration Samples

During each autocalibration cycle, each internal calibration channel is sampled a number of times, and the samples are averaged.

Averaging of the samples from calibration channels provides for an accurate calibration, and minimizes the effects of local noise during the calibration cycle.

The number of calibration samples taken during a calibration cycle is 3 samples by default, but may be set within the range of 1 to 10 samples by the Parameter23 command (See Section III – Parameter Commands).

Using DeTransfer, the number of calibration samples is set by the command for example

```
P23=2
```

which sets the number of calibration samples to 2.

The current setting for the number of calibration samples is read in DeTransfer by the command

```
P23
```

Using DeLogger, number of calibration samples is set and read using these same commands either

casually in the Entry Screen of the Text View

as part of a program, in the Optional Initialization Commands under the Special Commands button of the Settings tab of the Program Builder

There is little benefit in increasing the number of calibration samples beyond the default. However decreasing the number of calibration samples will reduce the calibration time, with only a minor reduction in accuracy.

The time taken for a calibration cycle when Parameter23=3 (default) is 600 mS, which can cause unpredictable discontinuity's in data collection during rapid scanning.

Autocalibration of the ADC can be enabled and disabled by the /K or /k Calibration Switch command.

Using DeTransfer, autocalibration is enabled by the command

```
/K
```

and is disabled by the command

```
/k
```

Using DeLogger, autocalibration can be enabled or disabled using the same commands either

directly, in the Entry Screen of the Text View

as part of a program, in the Optional Initialization Commands under the Special Commands button of the Settings tab of the Program Builder

When autocalibration is enabled by the /K command, a new calibration is immediately performed. Therefore ADC recalibration can be forced at any time by the /K command.

Autocalibration by Drift in the Zero Voltage Reference

The basis for determining the need for automatic recalibration of the ADC circuits is the drift in the internal zero voltage reference. The zero voltage drift permitted before a recalibration is performed can be defined by the user.

Drift in the internal zero voltage may be in either direction, and is due to warming up of the logger when first powered up, ambient temperature changes, and aging of circuit components.

The internal zero voltage reference is read before each scan of the analog input channels. If the zero voltage has drifted from zero by more than a defined value, then a calibration cycle is initiated.

The default zero voltage drift for autocalibration is ±4 µV, and can be set within the range of 0 to ±10,000 µV by the Parameter0 command (See Section III – Parameter Commands).

Using DeTransfer, the command for example

```
P0=100
```

sets the allowed zero reference voltage drift before autocalibration to ±100 µV. The zero voltage drift is set in increments of 1 µV.

Setting P0=0 will ensure an autocalibration before every scan, while setting P0=10000 (±10 mV) will in practice disable autocalibration.

The current setting for the autocalibration zero voltage drift can be returned into DeTransfer by the command

```
P0
```

Internal Calibration and Checks

There are five calibration standards within the *dataTaker* which determine accuracy of the logger. These standards are used during autocalibration, and are as follows

a 2MHz quartz crystal for frequency measurement

two precision current sources of 250.0 µA and 2.500 mA for resistance measurement

a 2.500 Volt 0.05% precision reference voltage for all voltage and current measurements

a precision 100 Ohm 0.1% reference resistance

The internal reference voltage can be trimmed should it drift with age (see below).

Internal Temperature

The *dataTaker* and the Channel Expansion Modules (CEM-AD) have an internal temperature channel, which is accessible to the user. This temperature channel is used as the default cold junction temperature reference for thermocouple support.

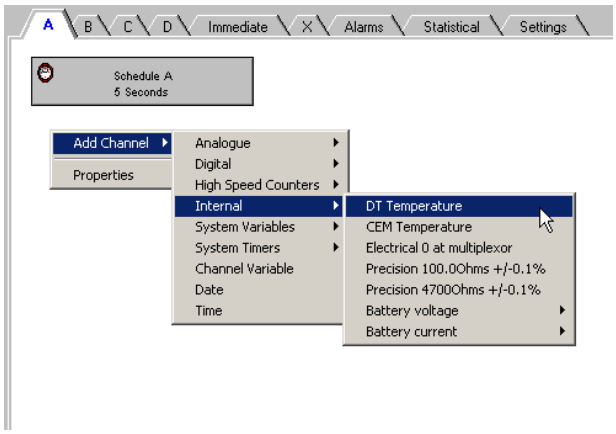
Using DeTransfer, the internal temperature channel of the *dataTaker* can be read at any time by the command

```
1%LM35
```

Using DeLogger, the internal temperature channel of the *dataTaker* can be read

directly, in the Entry Screen of the Text View

by including the Internal : DeTransfer Temperature channel in a program



The channel type LM35 specifies that the internal temperature sensor is an LM35 which is powered by the analog circuits during reading (See Section II – Measuring Temperature with IC Temperature Sensors). The channel number 1% is the internal temperature channel for the *dataTaker*.

The internal temperature channel of the Channel Expansion Module (CEM-AD) can be read in the same way by the command

```
n:1%LM35
```

Channel number 1% is the internal temperature channel for the Channel Expansion Module (CEM-AD), and n: specifies the CEM-AD module address number of 1 or 2.

Trimming Errors in the Reference Voltage

All measuring systems ultimately compare an input voltage with an internal reference voltage. If the internal reference voltage is not accurate, then the reading of the input voltage will be in error.

The *dataTaker* assumes that the internal 2.500 Volt reference voltage is accurate.

However should the internal reference voltage drift with age, then the voltage reference can be trimmed by either of two techniques as follows

the 2.500 Volt voltage reference can be trimmed in hardware by adjusting a trim potentiometer. The trim potentiometer is located on the lower circuit board, and is accessible through a screwdriver hole in the upper circuit board beside the memory card connector. The adjustment range is ±0.1%.

the 2.500 Volt voltage reference can also be trimmed by entering a software trim factor

The internal voltage reference is 2.500 Volt ± 0.05%. Therefore the voltage reference can vary by ±1.25 mV from the nominated value.

The 2.500 Volt reference can be trimmed in software as follows

connect an accurately known precision voltage source to any of the analog input channels of the *dataTaker*, and read the voltage. This precision voltage source should preferably be greater than 1.5 Volt.

using the data for the known voltage and the measured voltage for the precision source, calculate the trim factor by

$$\Delta V = (V_{source} - V_{measured})$$

$$\text{Trim} = (\Delta V \times (V_{ref} / V_{source}))$$

In the following example, the internal reference voltage trim factor is calculated from measurements using a 1 Volt precision source

$$V_{ref} = 2500.00 \text{ mV}$$

$$V_{source} = 1000.00 \text{ mV}$$

$$V_{measured} = 1000.02 \text{ mV}$$

$$\Delta V = (V_{source} - V_{measured})$$
$$= (1000.00 - 1000.02)$$
$$= -0.02 \text{ mV}$$

$$\text{Trim} = (\Delta V \times (V_{ref} / V_{source}))$$
$$= (-0.02 \times (2500.00 / 1000.00))$$
$$= -0.05 \text{ mV}$$
$$= -50 \text{ µV}$$

The magnitude of the reference voltage trim factor is declared to the *dataTaker* in terms of 10's of microvolts. The sign of the calculated trim factor must also be entered.

The reference voltage trim factor is entered into the *dataTaker* using the Parameter1 command (See Section III – Parameter Commands).

Using DeTransfer, the command for example

```
P1=-5
```

loads a trim factor of -50 µV into the *dataTaker*.

The current setting for the reference voltage trim factor can be read by DeTransfer by the command

```
P1
```

Using DeLogger, the trim factor can be set and read using the same commands either

- directly, in the Entry Screen of the Text View
- as part of a program, in the Optional Initialization Commands under the Special Commands button of the Settings tab of the Program Builder

The default setting for P1 is 0 µV.

If a precision voltage source is not available, then the actual internal 2.500 Volt reference of the *dataTaker* can be measured with a precision multimeter, and a software trim factor determined as follows

- if the measured voltage is greater than 2.500 Volt, then the trim factor is +1 for every 10 µV of error
- if the measured voltage is less than 2.500 Volt, then the trim factor is -1 for every 10 µV of error

Calculate the trim factor required, and define this to *dataTaker* using the Parameter1 command.

Connecting Analog Input Signals

The analog input channels of the *dataTaker* data loggers and Channel Expansion Modules (CEM-AD) each have four screw terminals comprising

Positive (+) input terminal for differential or single ended signal measurements

Negative (–) input terminal for differential or single ended signal measurements

Excite (⁕) output terminal for excitation voltages or currents, or input terminal for single ended measurements (*dataTaker 500/600* series loggers only)

Analog Return (R) terminal for grounding and referencing signals, or input terminal (#) for measurement of current and current loop signals using the internal current shunt

Each channel of the *dataTaker* and Channel Expansion Module (CEM-AD) comprises a block of 4 screw terminals, which can be unplugged from the logger. This allows easy temporary disconnection of sensors, etc.

Analog Input Precautions

When installing a data logging system, there are two principal conditions which can become sources of measurement error, and which must be avoided.

These sources of error are

exceeding the common mode voltage - offset voltages present in the signal being measured may exceed the common mode range of the *dataTaker*

ground loops - connections between inputs of a grounded *dataTaker* and other grounded devices such as sensors, test instruments, the host computer, etc. may produce current return paths via ground.

Care must be taken to avoid establishing either of these error producing conditions when installing the *dataTaker*.

Common Mode Voltages

A sensor typically produces an output signal proportional to the magnitude of the physical parameter it is sensing.

However other voltage sources such as DC offset voltages, ground potential differences and electrical noise induced into sensor cables, may contribute additional voltage to the signal that is not due to the magnitude of the physical parameter being measured. This therefore is a source of measurement error.

This additional voltage is referred to as the common mode input voltage, and is defined as the average of the voltages between the measurement system's ground and the two input terminals

$$V_{cm} = (V_{+} + V_{-}) / 2$$

where

V_{cm} is the common mode voltage

V_{+} is the voltage on the +ve input terminal with respect to ground of the measuring instrument

V_{-} is the voltage on the –ve input terminal with respect to ground of the measuring instrument

Some sources of common mode input voltages are summarized in Figure 24 overleaf.

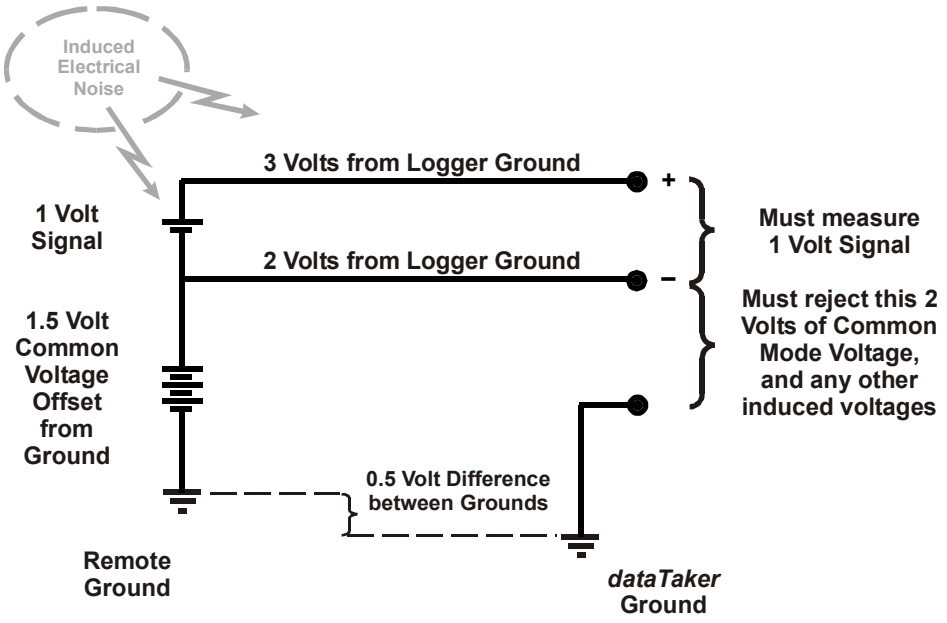


Figure 24 – Sources of Common Mode Input Voltages

A measuring instrument must be able to tolerate or 'reject' the common mode voltages from differential inputs. However the extent to which the measuring instrument can reject common mode voltages is limited by the design of the instrument, and so varies between instruments. The ability to reject common mode voltages is referred to as the 'common mode voltage range', 'common mode range', 'common mode rejection'.

The *dataTaker 50, 500* and *600* data loggers have a common mode input voltage range of –3.5 Volts to +3.5 Volts.

The *dataTaker 505, 515, 605* and *615* data loggers have a common mode input voltage range of –3.5 Volts to +3.5 Volts if the internal attenuators are not selected, or have a common mode input voltage range of –100 Volts to +100 Volts if the internal attenuators are selected.

The Channel Expansion Module (CEM-AD) has a common mode input voltage range which the same as the *dataTaker* model to which it is connected, as follows

- if a Channel Expansion Module (CEM-AD) is connected to either a *dataTaker 500* or *600*, the common mode voltage range is –3.5 Volts to +3.5 Volts
- if a Channel Expansion Module (CEM-AD) is connected to either a *dataTaker 505, 515, 605* or *615*, the common mode voltage range is –3.5 Volts to +3.5 Volts if the internal attenuators are not selected, or –100 Volts to +100 Volts if the internal attenuators are selected

If the total of the signal voltage, common mode voltage and noise voltages of an input are within the common mode range, there should be minimal measurement errors due to common mode voltages.

However if the total of the signal and noise voltages exceeds the common mode range, then measurement errors will occur. Excessive common mode voltages applied to a particular channel may also affect the accuracy of other input channels of the data logger.

In many of the illustrations provided throughout Section II of this manual, which detail connection of analog input signals to the *dataTaker*, the total of all common mode voltage sources is indicated as V_{cm} or NOISE.

Avoiding Excessive Common Mode Voltages

Where the common mode voltages are likely to exceed the common mode voltage range of the *dataTaker*, then steps must be taken to reduce the common mode voltage. This can be done in various ways as follows

the *dataTaker* ground and sensor grounds should be connected to a common ground point, preferably at the data logger (see discussion relating to ground loops below)

where the DC offset voltages are large, the *dataTaker* can be isolated from true ground and the *dataTaker* ground connected to the negative pole of the input signal. This 'floating' of the *dataTaker* may introduce errors on other channels, and is not recommended

where large voltages are likely to be induced into signal cables by environmental noise, then shielded signal cables should be used and the shields grounded.

the input signal and common mode voltages can be attenuated to within the common mode voltage range of the logger, before being applied to an input channel.

Common mode voltages will more likely be a problem with those *dataTaker*s which have solid state multiplexers, or if the internal attenuators are not selected for relay multiplexed *dataTaker*s.

Ground Loops

Ground loops can be inadvertently created in a data logging system in a number of ways. The two most common ways of creating ground loops are

connecting the *dataTaker* to a ground point which has a different potential to the ground of one or more of the sensors connected to the logger

connecting the *dataTaker* to a ground point which is at a different potential to the ground of the host computer (This is generally only a problem with Series 1 *dataTakers*, which did not have an isolated RS232 COMMS interface)

It is often assumed that all 'Ground' or 'Earth' points are always at the same potential of 0 Volts. However this is not the case, and differences of several volts can often occur between two or more ground points.

In some situations where there can be considerable electrical ground noise, due to the large electrical equipment found in factories and high rise buildings, the potential difference between two ground points can even be several tens of volts.

Ground potential differences are also a function of electrical conductivity of the soil. Therefore, in regions where soils are very dry or frozen, ground potential differences are likely to be greater than in wetter regions.

The potential difference between two or more ground points will cause a current to flow via the equipment, producing measurement errors and possibly even equipment failure.

The potential difference between all grounds in the data logging application should not be greater than –3.5 Volts to +3.5 Volts (solid state multiplexer versions), or –100 Volts to +100 Volts (relay multiplexer versions).

Ground Loops via the Communication Cable (Series 1 dataTakers)

Most data logging systems usually include a host computer, which is permanently connected to the serial interface of the *dataTaker* data logger. The host computer is normally connected to the local mains or line power supply ground, which may be different to the ground that the *dataTaker* is connected to.

Series 1 *dataTakers* do not have an electrically isolated RS232 COMMS interface, and so the communication cable can produce a connection or ground loop between the two grounds. If there is a potential difference between the computer ground and logger ground, then current will flow between them

if the ground potential difference is within the common mode range of –3.5 Volts to +3.5 Volts (solid state multiplexer versions), or –100 Volts to +100 Volts (relay multiplexer versions), then there should be no problems

if the ground potential difference is outside the common mode range but within –15 Volts to +15 Volts (solid state multiplexer versions), or –500 Volts to +500 Volts (relay multiplexer versions), then this will cause erroneous signal measurements but no damage to the logger (however the computer may be damaged, depending on what protection its serial interface has)

if the ground potential difference is greater than these levels, then damage of the logger and computer will almost certainly occur.

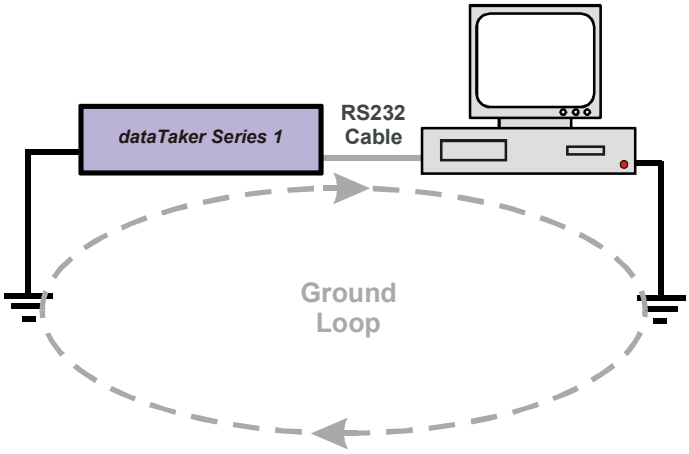


Figure 25 – Ground Loops via the Communications Cable (Series 1 *dataTakers*)

The Series 2 and Series 3 *dataTakers* have an electrically isolated RS232 COMMS interface, and will not suffer these problems.

Ground Loops via the Sensor Cables

Data logging systems include a number of sensors or instruments, from which the readings are being taken. These may be simple sensors such as thermocouples that are grounded to the structure being monitored, or may be various instruments that are grounded via their mains or line power supply ground.

Often these sensors may be connected to a ground point that is at a different potential to that to which the data logger is connected. This ground potential difference can create ground loops via the sensor cables.

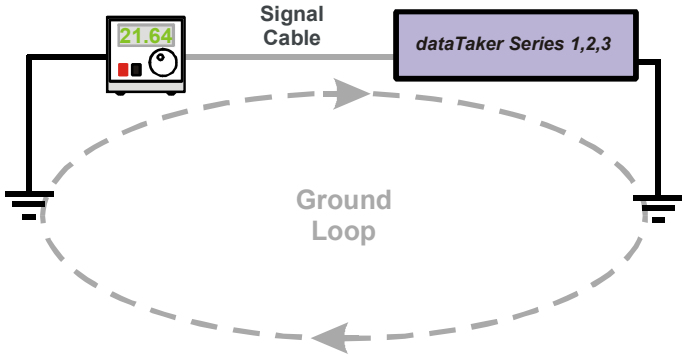


Figure 26 – Ground Loops via the Sensor Cables

Smaller ground potential differences can cause erroneous and/or erratic readings for only some analog input channels, however if the common mode voltage range of the data logger is exceeded then all readings will be effected.

Avoiding Ground Loops

Ground loops can be avoided by ensuring that the *dataTaker* and the sensors are all connected to the same ground point. In most cases this ground point should be at the data logger itself.

However when external and separate grounding of the sensors cannot be avoided, then other strategies can be used to avoid possible ground loops

- signal inputs to the *dataTaker* should be as differential inputs, to take advantage of the logger's common mode rejection range (see Section II – Differential Analog Input Mode)
- a ground which is common to the sensors but separate from logger ground can be connected to the SE Ref terminal of the logger, and single ended measurements made with respect to this reference (see Section II – Single Ended Inputs Referenced to an External Common). This also takes advantage of the logger's common mode rejection range.
- install isolators between the sensors and the logger

Care must be taken to ensure that ground potential differences do not exceed common mode voltage range of the *dataTaker*.

With Series 1 *dataTakers* ground loops via the communication cable are avoided by isolating the RS232 COMMS serial interface, and powering the serial interface of the logger from an external power supply (See Section II – The RS232 COMMS Serial Interface).

Identifying Common Mode Voltages

If common mode or offset voltages are suspected of being the cause of measurement errors, then this can be confirmed and the cause identified as follows

- using a multimeter, measure the voltage between *dataTaker* ground and each of the signal leads for each of the sensors connected to the *dataTaker*.

  Check that these voltages are all within the common mode range of the *dataTaker* you are using.
- follow the procedures described below for identifying ground loops, which may be causing the excessive common mode voltages.

Identifying Ground Loops

If ground loops are suspected of being the cause of measurement errors, then this can be confirmed and the cause identified as follows

- Disconnect all sensors from the *dataTaker*, and perform the logger's diagnostic test. If you are using DeTransfer, then issue a TEST command. If you are using DeLogger, then select dataTaker|Test from the main menu, or click on the Test button on the toolbar.

  If the TEST command returns a PASS result, then the measurement errors are not due to hardware failure of the logger.

  Progressively reconnect the sensors one at a time, checking for a FAIL result by the TEST command after each connection. Sensors with common mode voltage or ground loop problems will produce a TEST command FAIL result when they are reconnected, and may also affect readings for other sensors.

  When causal sensors have been identified, then check these and any sensor power supplies for excessive ground potential differences, etc.
- If you are using a Series 1 logger, then enter a Schedule into the logger and enable data logging. Log channel data for several minutes, then disconnect the computer and continue to log data for several

more minutes.

Reconnect the computer and download the data. If the errors did not occur when the computer was disconnected, then there is a strong possibility that a ground loop exists via the communication cable.

Ground loops in the communication cable of Series 1 loggers can be overcome by electrically isolating the RS232 COMMS serial interface (See Section II – The RS232 COMMS Serial Interface).

The Analog Input Channels

The *dataTaker* range of data loggers have 2 different types of analog input channel multiplexers, solid state (CMOS) multiplexer or relay multiplexer.

The models of the *dataTaker* data loggers which have each of these multiplexer types are as follows

Solid State (CMOS) Multiplexer
- *dataTaker 50, 500* and *600*

Relay Multiplexer
- *dataTaker 505, 605, 515* and *615*
- Channel Expansion Module (CEM-AD)

Throughout this section reference will be made to solid state or relay multiplexing, rather than to individual *dataTaker* models. Confirm from the above list which *dataTaker* model you are using.

The *dataTaker 50* has 5 analog input channels, which can be used for measuring up to 5 analog input signals as differential inputs, or up to 10 analog input signals single ended inputs.

The 5 analog input channels of the *dataTaker 50* provide for channel capacities as follows

15 single ended channels for current, current loop and AD590/AD592 temperature sensors

10 single ended channels for all other analog signal types

10 single ended channels for resistance and RTDs

The *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) have 10 analog input channels, which can be used for measuring up to 10 analog input signals as differential inputs, or up to 30 analog input signals single ended inputs.

The 10 analog input channels of the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) provide for channel capacities as follows

40 single ended channels for current, current loop and AD590/AD592 temperature sensors

30 single ended channels for all other analog signal types.

20 single ended channels for resistance and RTDs

The analog input channels of the *dataTaker* and Channel Expansion Module (CEM-AD) can be used in any mix of differential and single ended channels.

The use of the analog input channels as differential or single ended channels for measuring analog signals is specified as part of the channel specification in the *channel list* of the Schedule (see Section III – Specifying Channels).

The Analog Input Channels

The analog input channels of the *dataTaker* and Channel Expansion Module (CEM-AD) comprise four terminals

the positive (+) input terminal

the negative (–) input terminal

the Excite (✳) terminal

the Analog Return (R) terminal

The screw terminal connections are identical for all analog channels, and are laid out as illustrated below

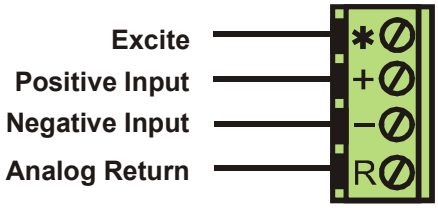


Figure 27 – Analog Input Channel Terminal Layout

The *dataTaker* fundamentally measures the voltage of input signals applied to the analog input channels.

The input channels are configured in various ways to enable measurement of current, resistance, frequency and period signals.

Voltage Ranges – Solid State Multiplexer

The three voltage ranges for the *dataTaker 50*, the *dataTaker 500* and the *dataTaker 600* models which have the solid state (CMOS) multiplexer are

±25.000 mV

±250.00 mV

±2500.0 mV

The analog to digital converter autoranges between these input ranges, with range changing occurring at approximately 120% and 15% of nominal full scale for each range. The highest range allows up to 30% over-range, which allows voltages in the range of approximately ±3250 mV to be measured.

An out of range input signal results in data of ±99999.9 units. An out of range input is also indicated by the error message 'E11-input(s) out of range'.

The operation of the analog to digital converter is discussed in detail in Section II – Analog Processing.

Voltage Ranges – Relay Multiplexer

The voltage ranges for the *dataTaker 505*, the *dataTaker 515*, the *dataTaker 605* and the *dataTaker 615* models which have the relay multiplexer depend on the internal attenuator.

If the internal attenuator is not selected, then the three low voltage ranges are

±25.000 mV

±250.00 mV

±2500.0 mV

If the internal attenuator is selected, then the three high voltage ranges are

±7 Volts

±70 Volts

±100 Volts

The analog to digital converter autoranges between these input ranges, with range changing occurring at approximately 120% and 15% of the nominal full scale for each range. The highest range allows up to 30% over-ranging, allowing voltages in the range of approximately ±3250 mV to be measured in the low voltage range, and ±130 Volts to be measured in the high voltage range.

An out of range input signal results in data of ±99999.9 units. An out of range input is also indicated by the error message 'E11-input(s) out of range'.

The operation of the analog to digital converter is discussed in detail in Section II – Analog Processing.

Termination of Analog Inputs

The analog input channels of the *dataTaker* have an input impedance of >100 MOhm. However the analog input channels can be internally tied to the logger ground or 'terminated' via 1 MOhm resistors. These terminating resistors prevent ungrounded input signals 'floating' out of the common mode range of the *dataTaker*, particularly for the solid state multiplexed models.

This input termination is the default configuration for most of the differential signal types, such as low level voltage, frequency, period, thermocouple and analog state.

The input termination can be enabled or disabled by channel options, which are declared as part of the input signal identification (See Section III – Channel Options).

In DeTransfer, input termination is selected by the channel option T for example

```
8V(T)
```

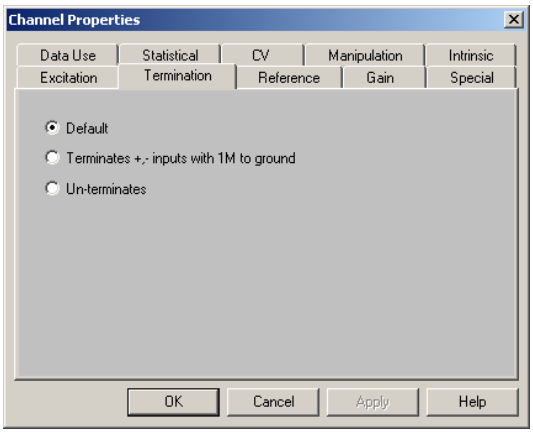
In this case a differential voltage applied to analog channel 8 is to be measured with the inputs terminated to the *dataTaker* ground via the 1 MOhm resistors. This is the default configuration for most differential voltage and thermocouple signal types, and normally does not need to be specified.

The input termination resistors can be disabled if required, for example if differentially connected thermocouples are externally grounded, or if a sensor such as a glass electrode for pH or specific ion measurement has a high output impedance and cannot match the 1 MOhm input impedance of the terminated inputs. Input termination can be disabled by the Unterminate channel option as follows

```
6TT(U)
```

In this case the type T thermocouple connected to analog input channel 6 is to be measured Unterminated. The 1 MOhm resistors are disconnected from the inputs during measurement of the input signal. Note that input termination defaults to terminated for differential thermocouples.

The internal termination 1 MOhm resistors are managed in the DeLogger Program Builder by selecting Channel Options:Termination… for the channel, which opens the Termination dialog box as follows



Differential Analog Input Mode

The differential analog input mode can be used to measure the fundamental signal types of voltage, current, resistance and frequency, and for sensors providing these signals.

In the differential mode, the analog signal is applied between the +ve and –ve terminals of the analog input channel, as illustrated in Figure 28 below

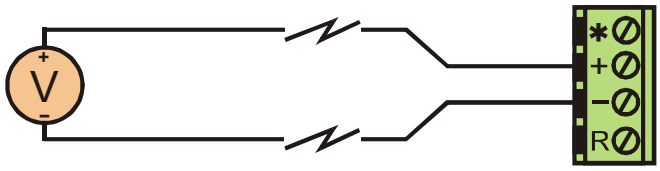


Figure 28 – Differential Analog Input

The unknown voltage V is measured between the +ve and the –ve terminals.

However this is a somewhat simplistic situation which implies that there are no common mode voltages or noise voltages in the measurement system.

Figure 29 below illustrates a more realistic situation, where there are common mode or noise voltages composed of DC offset voltages, ground potential differences and electrical noise voltages which can effect the measurement accuracy.

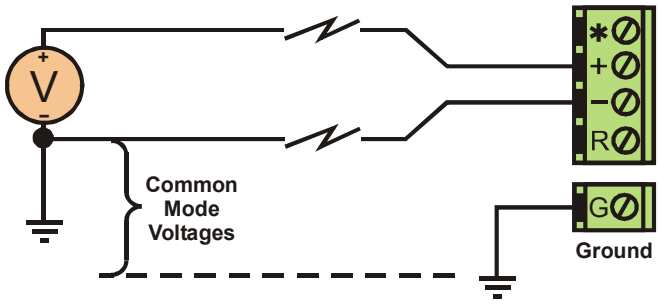


Figure 29 – Differential Analog Input With Common Mode Voltages

The unknown voltage V is measured between the +ve and the –ve terminals. Any common mode voltage or NOISE voltage which is in the common mode range of the *dataTaker* is rejected.

The +ve and the –ve signal lines may float within the *dataTaker* common mode input voltage limits of

solid sate (CMOS) multiplexer
–3.5 Volts to +3.5 Volts with respect to ground

relay multiplexer
–100 Volts to +100 Volts with respect to ground
when the internal attenuators are selected

The voltage inputs are internally tied to ground or terminated via the 1 MOhm resistors, to prevent ungrounded signals 'floating' out of the common mode range.

Differential analog input channels are sampled and the data is returned when Schedules execute or run.

Using DeTransfer, the command for example

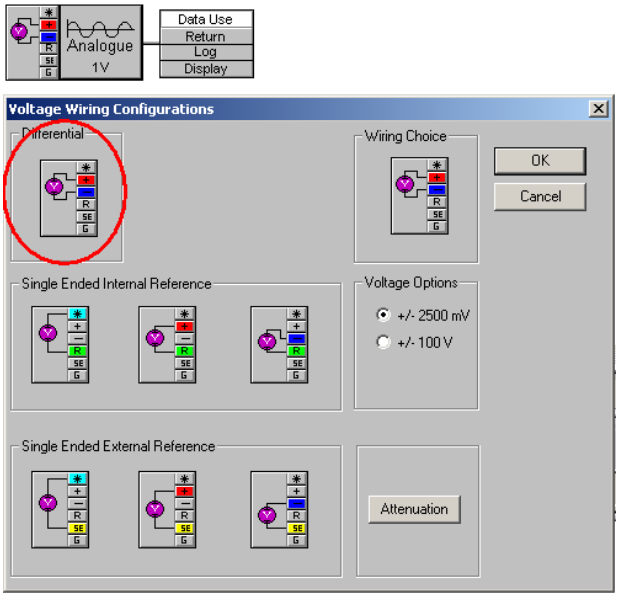
```
1V  5..10V
```

Instructs the *dataTaker* to measure the differential voltages applied to analog input channel 1 and to channels 5 through 10 inclusive. The V indicates that the signal applied to these channels is to be measured as a voltage.

```
2HV 4..6HV
```

Instructs a *dataTaker505* or *605* to measure the differential high level voltages applied to analog input channel 2 and to channels 4, 5 and 6. The HV indicates that the signal applied to these channels is to be measured as a voltage in the high range.

Using DeLogger, differential analog channels are selected from the Voltage Wiring Configurations dialog that opens when you select the analog input channel as follows



Single Ended Analog Input Mode

In the single ended input modes the positive terminal, negative terminal and Excite terminal (*dataTaker 500/600* series loggers and the Channel Expansion Module only) of the analog input channels are used as separate input channels as follows

Positive single ended channel, where the positive line of the signal is connected to the +ve terminal of the analog channel, and the negative line of the signal is connected to a common reference.

Negative single ended channel, where the positive line of the signal is connected to the –ve terminal of the analog channel, and the negative line of the signal is connected to a common reference.

Excite single ended channel, where the positive line of the signal is connected to the ✳ terminal of the analog channel, and the negative line of the signal is connected to a common reference. This mode is supported in the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) only, and is not supported in the *dataTaker 50*.

The negative inputs of the analog channels are non-inverting such that when a positive signal is applied to the negative terminal of a channel, positive data is returned.

The single ended analog inputs can be referenced in two ways as follows

internally referenced single ended inputs, where the input signal is referenced to the Analog Return (R) terminal of each channel.

externally referenced single ended inputs, where the input signal is referenced to a common external point which is connected to the single ended external reference terminal SE REF.

Although these input modes are referred to as the single ended input modes, the signals are actually measured differentially between the indicated terminal of the analog input channel and the indicated reference.

The single ended analog input modes can be used to measure fundamental voltage, current, resistance, frequency and period signals, and sensors providing these signals.

The main advantage of single ended input modes is that the number of measuring channels is increased. However there may be some decrease in accuracy due to noise.

Note : The Excite terminal single ended input is not supported in the *dataTaker 50*. All references to Excite terminal single ended input (✳) modes throughout this manual only apply to the *dataTaker 500 /600* series loggers, and to the Channel Expansion Module (CEM-AD).

Single Ended Inputs Referenced to Analog Return

Single ended analog inputs referenced to Analog Return may be used for measuring voltage, current, frequency and period. Resistance cannot be measured in this mode.

Analog signals are connected for single ended measurement between any of the positive (+) terminal, the negative (–) terminal or the Excite (✳) terminal of the analog channel, and the Analog Return (R) terminal of the same channel.

The connection of single ended inputs referenced to Analog Return is illustrated below

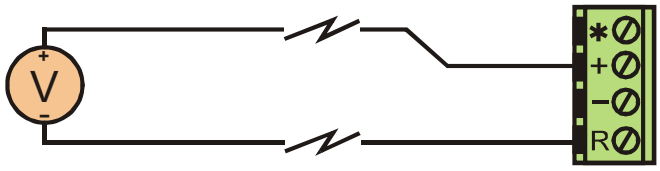


Figure 30 – Positive Terminal Single Ended Input Referenced to Analog Return

The unknown voltage V is measured between the +ve and Analog Return terminals.

However this is a somewhat simplistic situation which implies that there are no common mode voltages or noise voltages in the measurement system.

The input signals connected to the +ve, –ve and ✳ terminals of the *dataTaker* must remain within the common mode limits with respect to the logger ground.

A more realistic situation is shown in Figure 31 below, where the input signal contains common mode or noise voltages composed of DC offset voltages, ground potential differences and electrical noise voltages. These common mode voltages can effect the measurement accuracy.

The single ended input modes are prone to errors due to ground loop noise. Further, excessive ground current into the Analog Return terminal may damage the analog input circuitry. This is due to ground loop currents (which can be large) causing a voltage drop along the return wire.

These problems are largely overcome by not grounding sensors externally, or by externally referencing the single ended inputs (see below).

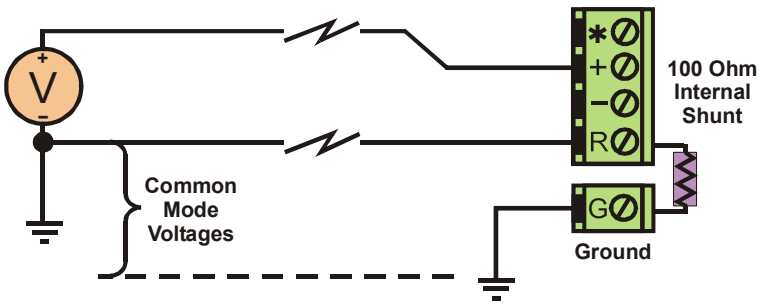


Figure 31 – Positive Terminal Single Ended Input With Common Mode Voltages, Referenced to Analog Return

Single ended inputs referenced to Analog Return are sampled and the data is returned when Schedules execute or run.

Using DeTransfer, single ended inputs referenced to Analog Return are read by the command for example

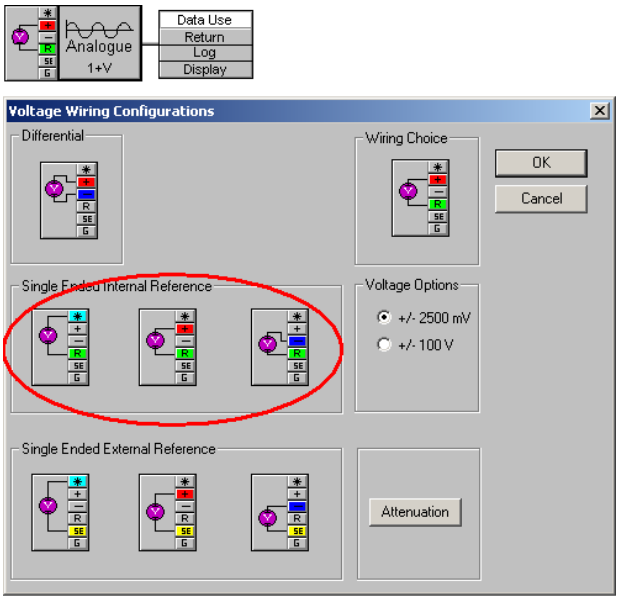
```
1✳V   1+V   1−V
```

which instructs the *dataTaker* to measure the single ended voltages applied between

- the ✳ Excite terminal and the Analog Return terminal of analog input channel 1, which is identified as channel 1✳ (supported by *dataTaker 500/600* series loggers only)
- the +ve terminal and the Analog Return terminal of analog input channel 1, which is identified as channel 1+
- the -ve terminal and the Analog Return terminal of analog input channel 1, which is identified as channel 1–

The V indicates that the signal applied to these channels is to be measured as a voltage.

Using DeLogger, single ended analog channels referenced to Analog Return are selected from the Voltage Wiring Configurations dialog that opens when you have selected the analog input channel as follows



Note : These examples also illustrate that an analog input channel of the *dataTaker 500/600* series loggers and Channel Expansion Module (CEM-AD) can be used to measure three different voltages, and that an analog channel of the *dataTaker 50* can be used to measure two different voltages.

Single Ended Inputs Referenced to an External Common

Single ended analog inputs can be measured with reference to an external reference or 'common'. This mode can be used for measuring voltage, current, current loop, frequency and period. Resistance cannot be measured by this input mode. External references can be used for single ended analog inputs whenever a common reference point can be established in a sensor system remote from the *dataTaker*.

The analog input signal is applied between any of the +ve, –ve or ✳ terminal of the analog input channel, and the established external reference point. The external reference is connected to the single ended external reference input terminal SE REF.

The single ended reference input terminal SE REF has a high input impedance, resulting in very small ground currents. The connection of single ended inputs referenced to an external common is illustrated in Figure 32 below

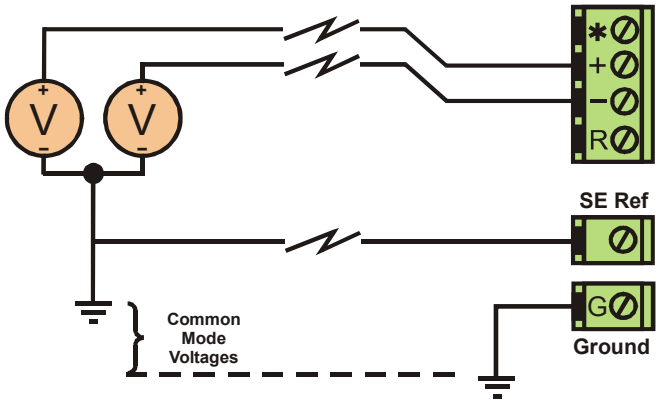


Figure 32 – Single Ended Analog Inputs Referenced to an External Common

The unknown voltages V are measured between the +ve and –ve input terminals respectively, and the SE REF input terminal.

In this configuration, the common mode voltages and the noise voltages occurring between the external common point and the *dataTaker* are rejected.

Single ended measurements which are to be referenced to an external common applied to the SE REF input terminal are specified by the X channel option.

Single ended inputs referenced to an external common connected to SE REF, are sampled and data is returned when Schedules execute or run.

Using DeTransfer, single ended measurements that are to be referenced to an external common connected to SE REF are measured by the command for example

```
1*V(X)   5+V(X)   6-V(X)
```

which instructs the *dataTaker* to measure single ended voltages between

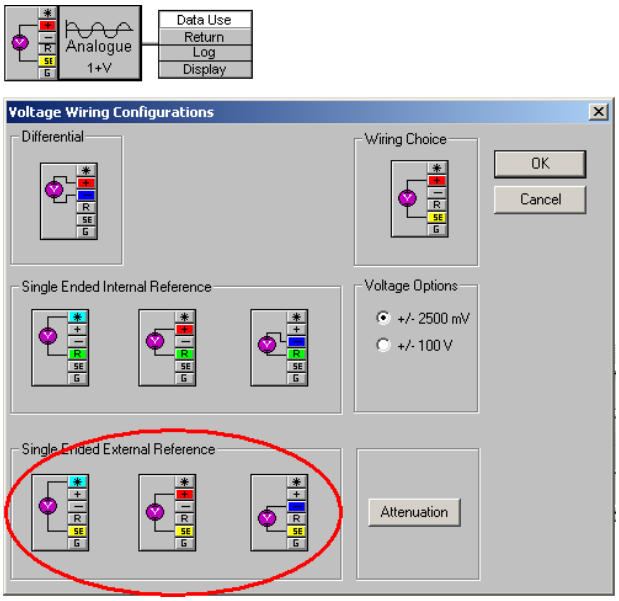
the * Excite terminal of analog input channel 1 (channel 1*) and an external reference input to SE REF

the +ve terminal of analog input channel 5 (channel 5+) and an external reference input to SE REF

the –ve terminal of analog input channel 6 (channel 6–) and an external reference input to SE REF

The V indicates that the signal applied to these channels is to be measured as a voltage. The X channel type option indicates that the single ended voltage is measured between the input terminal of the channel and SE REF.

Using DeLogger, single ended analog channels referenced to SE REF are selected from the Voltage Wiring Configurations dialog that opens when you have selected the analog input channel as follows



Using Externally Referenced Single Ended Inputs to Reduce Cabling

Externally referenced single ended analog inputs can also be used where a number of differential voltages must be measured at some distance from the *dataTaker*.

Figure 33 illustrates cabling required to connect the two differential voltages directly to the *dataTaker* analog input channels, requiring four conductors and two channels.

Figure 34 illustrates the same two sensors connected as single ended inputs referenced to an external common point, requiring only three conductors and one channel.

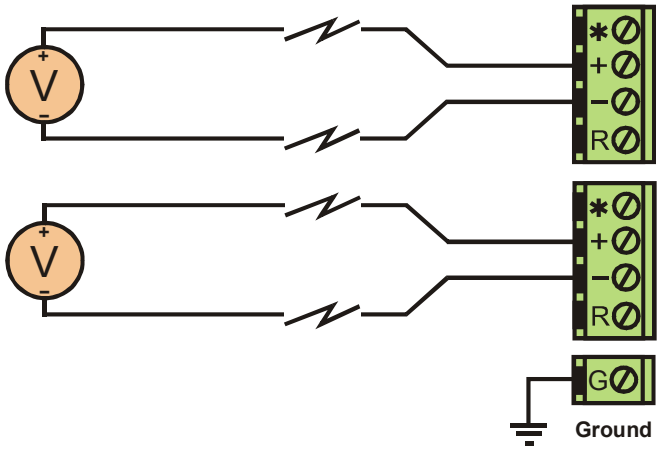


Figure 33 – Measurement of Remote Differential Voltages

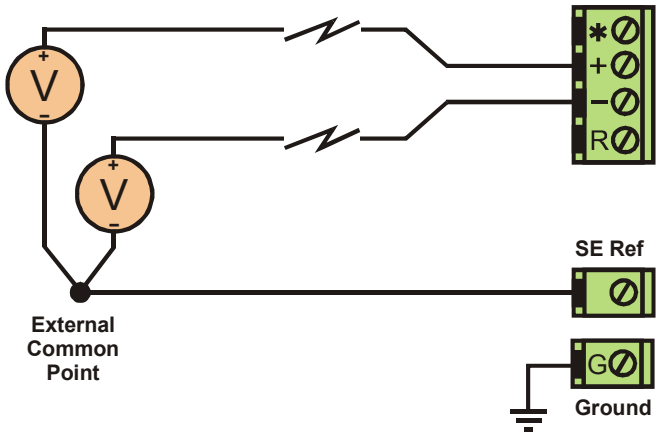


Figure 34 – Measurement of Remote Differential Voltages

The negative outputs of the remote sensors are connected to an external common point adjacent to the sensors, which in turn is connected to the SE REF input terminal. The positive outputs from each sensor are connected to the analog input terminals. The number of cables needed to connect sensors to the *dataTaker* is reduced, and half of the number of analog input channels are used.

Measuring Low Level Voltages

Measuring low level voltages is the simplest measurement to make with the *dataTaker*.

This section describes the measurement of low level differential (double ended) DC voltages and single ended DC voltages, in the range of approximately –3250 mV to +3250 mV.

DataTaker data loggers which have solid state (CMOS) and relay multiplexers are able to measure low level voltages in this range. In the case of loggers with the relay multiplexer, the low level voltage ranges do not select the internal attenuators.

Discussion throughout the remainder of this chapter refers to all models of the *dataTaker*, and is not specific between loggers with solid state (CMOS) or relay multiplexers.

The *dataTaker 50* has 5 differential analog input channels, or 10 single ended input channels, which can be used for low level voltage measurements.

The *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) have 10 differential analog input channels, or 30 single ended input channels, which can be used for low level voltage measurements.

Any combination of these differential and single ended connections can be used for measuring low level voltages.

Low level voltage signals can be measured in a number of ways as follows

as differential inputs on any of the analog input channels.

as single ended inputs referenced to Analog Return.

as single ended inputs referenced to an external common.

All *dataTaker* data loggers have three low level voltage measurement ranges as follows

±25.000 mV full scale

±250.00 mV full scale

±2500.0 mV or ±2.500 Volt full scale

The highest range provides for approximately 30% over-ranging, allowing voltages of –3250.0 mV to +3250.0 mV to be measured.

These voltage measuring ranges are selected automatically by the *dataTaker* to provide the best measurement resolution for a given voltage. Measurement ranges do not have to be selected by the user.

The Channel Expansion Module (CEM-AD) measures voltage in the same ranges as the *dataTaker* to which it is connected.

The *dataTaker 505, dataTaker 515, dataTaker 605* and *dataTaker 615* data loggers also support 3 high level voltage ranges (See Section II – Measuring High Level Voltages).

Differential Measurement of Low Level Voltage

In the differential mode, the signal voltage is applied between the +ve and the –ve terminals of the analog input channels. The differential voltage measured by the *dataTaker* is the difference of the voltages between the +ve and –ve terminals.

The differential mode of voltage measurement is used for

Floating Sources - when neither of the outputs from the signal source can be connected to ground.

Grounded Sources - when one of the outputs from the signal source is grounded independently of the *dataTaker*. Connecting it to the *dataTaker* ground could result in a ground loop.

Noisy Signals - when noise is equally present in both sides of the sensor output (e.g. mains or line hum). Differential mode largely cancels out this common mode noise.

Common Mode Voltage Range

The voltage between either of the +ve or the –ve analog input terminals of the analog input channel and *dataTaker* ground must not exceed ±3.5 Volts (*dataTaker 50, 500, 600*) or ±100 Volts (*dataTaker 505, 515, 605, 615*). This is the maximum allowable common mode voltage for the analog inputs.

Sensors which have no direct connection to ground can be tied to the *dataTaker* ground, via the internal 1 MOhm input termination resistors. When analog channels are used to measure voltage, the channels default to this configuration.

However if there is high common mode noise due to capacitive pickup, these 1 MOhm terminating resistors may not be sufficient to maintain the common mode range.

Under these conditions an additional resistor of 10 KOhm or less must be placed between one input line and the Analog Return terminal or any GND terminal. This is illustrated in Figure 35 below

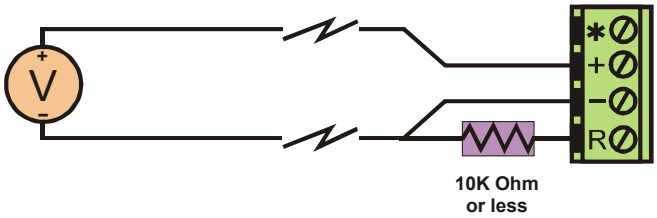


Figure 35 – Grounding of Floating Sensors

When grounding sensors to the *dataTaker* ground, the Ground (GND) terminals can be used either directly or via a suitable resistor.

The *dataTaker* Analog Return (R) terminals can also be used to ground sensors. However if Analog Return terminals are used to ground sensors, the internal 100 Ohm current shunt between the Analog Return terminal and *dataTaker* ground will be in series with the external grounding resistor (see Figure 44).

Differential Measurement

Low level voltages in the range of –2500 mV to +2500 mV can be connected directly to the analog channels of the *dataTaker* data loggers for differential measurement, as follows

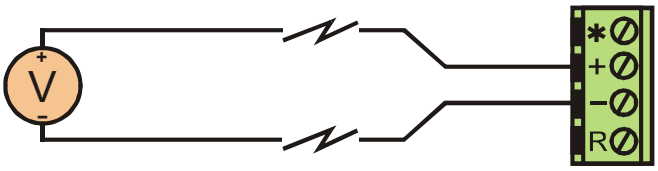


Figure 36 – Connecting Low Level Voltages as Differential Inputs

The differential voltage measurement is made between the +ve and –ve terminals of the analog input channel. The *dataTaker* also measures the polarity of the input voltage.

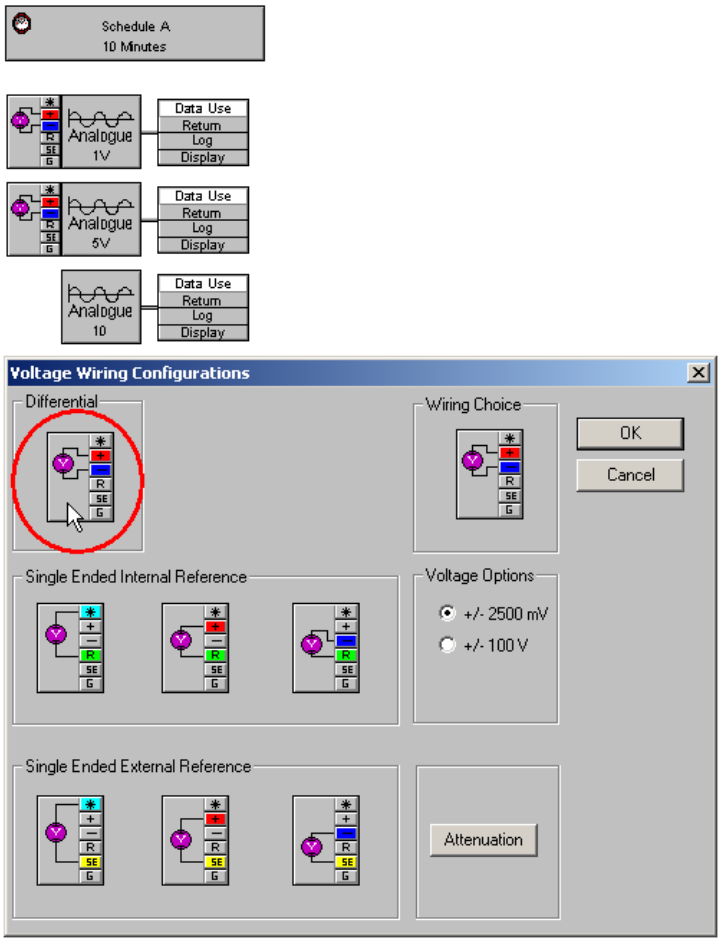
Low level voltages connected to the *dataTaker* data loggers as differential inputs are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, differential low level voltage signals are measured by the commands for example

```
BEGIN
 RA10M
  1V  5V  10V
END
```

which instructs the *dataTaker* to measure the differential voltages applied between the +ve and –ve terminals of analog channel 1, 5 and 10. The V indicates that the signal connected to these channels is to be measured as a low level voltage. The data is returned in units of mV.

Using DeLogger, differential low level voltage signals can be measured by the following Program Builder program. The differential connections are selected from the Voltage Wiring Configurations dialog which opens when you have selected the analog input channel.



By default the 1 MOhm termination resistors will be switched in during measurement of the differential low level voltages.

The *dataTaker* will read the voltage inputs every 10 minutes, and readings are stopped by a H (Halt) command.

Single Ended Measurement of Low Level Voltage

Low level signal voltages can also be measured by the *dataTaker* data loggers as single ended voltages in two ways, which differ in their reference configuration as follows

as single ended inputs to the +ve, –ve or ✳ terminals, referenced to Analog Return

as single ended inputs to the +ve, –ve or ✳ terminals, referenced to an external common point connected to the SE REF terminal

The *dataTaker* treats the voltage signal applied to the +ve, –ve or ✳ input terminal of the analog input channel as the active input, and measures this with respect to Analog Return or SE REF.

The *dataTaker 50* does not support single ended voltage measurement for the Excite (✳) terminal.

Single Ended Measurement Referenced to Analog Return

Low level voltages in the range –2500 mV to +2500 mV can be connected to the analog channels for measurement as single ended voltages referenced to Analog Return

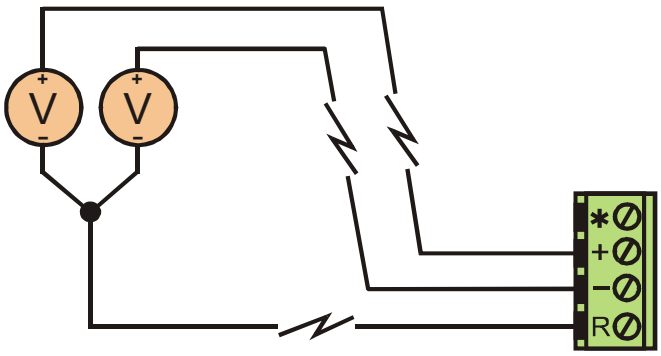


Figure 37 – Connecting Low Level Voltages as Single Ended Inputs Referenced to Analog Return

The single ended voltage is measured between the +ve, –ve or ✳ terminal and the Analog Return terminal of the analog input channel. The *dataTaker* also measures polarity of the input voltage. In the single ended input mode, the 1 MOhm input termination resistors are not switched in during measurement.

Low level voltages connected as single ended inputs and referenced to Analog Return are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, single ended low level voltages are measured by the command

```
BEGIN
 RA5M
  1+V  5  ✳V  8-V
END
```

which instructs the *dataTaker* to measure the single ended voltages as follows

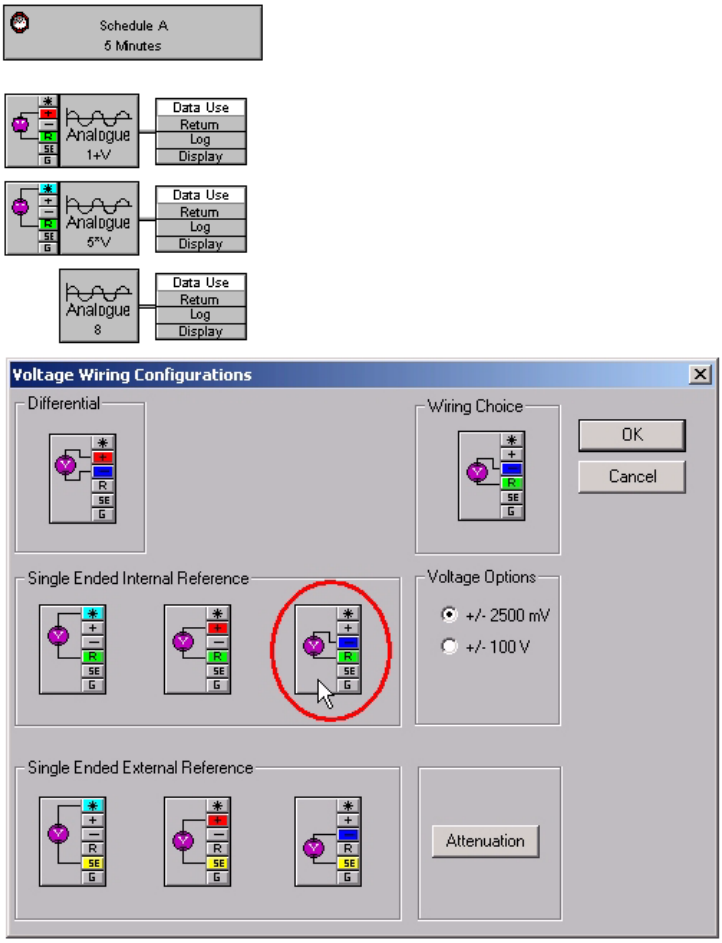
between the +ve terminal of channel 1 (1+) and the Analog Return terminal

between the ✳ terminal of channel 5 (5 ✳ and the Analog Return terminal

between the – terminal of channel 8 (8–) and the Analog Return terminal

The Analog Return terminal of all analog channels is connected internally to *dataTaker* ground. The V indicates that the signal applied to these channels is to be measured as a voltage. The data is returned in units of mV.

Using DeLogger, single ended low level voltage signals are measured by the following Program Builder program. The single ended connections are selected from the Voltage Wiring Configurations dialog which opens when you have selected the analog input channel.



The *dataTaker* will read the voltage inputs every 5 minutes, and readings are stopped by a H (Halt) command.

Single Ended Measurement Referenced to an External Common

Low level voltages in the range of –2500 mV to +2500 mV can be connected directly to the analog channels of the *dataTaker* for measurement as single ended voltages which are referenced to an external common, as illustrated in Figure 38 overleaf.

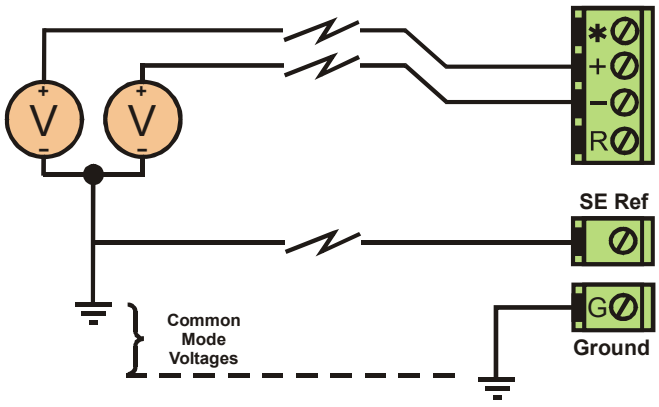


Figure 38 – Connecting Low Level Voltages as Single Ended Inputs Referenced to an External Common

The single ended voltage is measured between the +ve, –ve or ✳ terminal of the analog channel, and the SE REF terminal. Low level voltages connected as single ended inputs referenced to an external common are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, single ended low level voltage signals are measured with respect to the common SE REF terminal by the command for example

```
BEGIN
 RA5M
  1+V(X)   5  ✳V(X)   8-V(X)
END
```

which instructs the *dataTaker* to measure the single ended voltages as follows

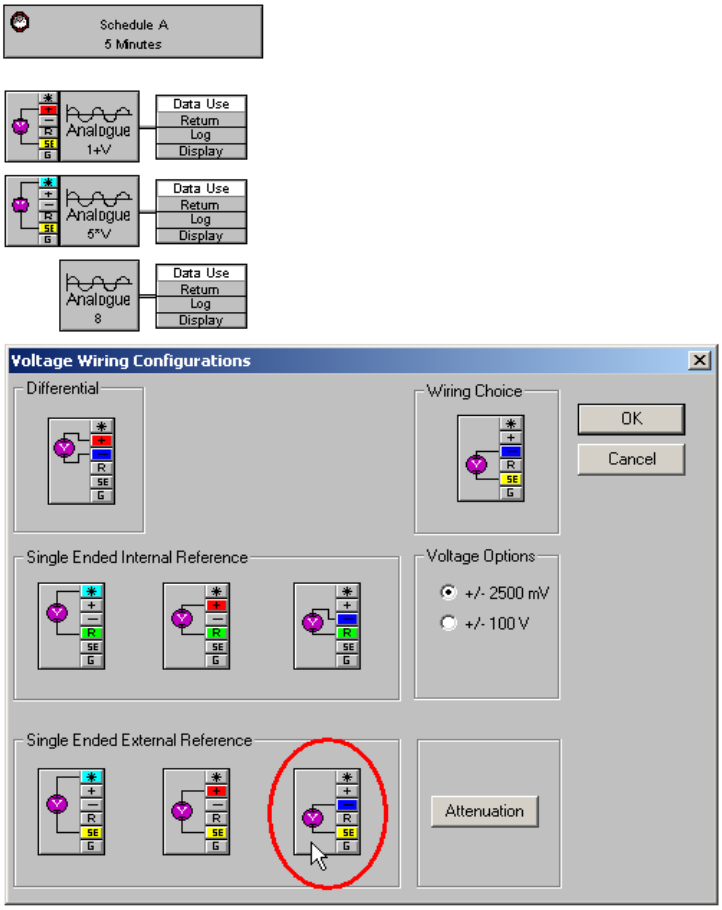
between the +ve terminal of analog input channel 1 (1+) and an external reference point connected to the SE REF terminal

between the ✳ terminal of analog input channel 5 (5 ✳) and an external reference point connected to the SE REF terminal

between the –ve terminal of analog input channel 8 (8–) and an external reference point connected to the SE REF terminal

The V indicates that the signal applied to the channels is to be measured as a voltage. The X channel option indicates that the voltage is to measured with respect to an external reference connected to the SE REF input. The data is returned in units of mV.

Using DeLogger, single ended low level voltage signals are measured by the following Program Builder program. The single ended connections referenced to SE REF terminal are selected from the Voltage Wiring Configurations dialog which opens when you have selected the analog input channel.



The *dataTaker* will read the inputs every 5 minutes, and readings are stopped by entering a H (Halt) command.

Measurement Ranges and Accuracy

All *dataTaker* models have three low level voltage measurement ranges of ±25.000 mV, ±250.00 mV and ±2500.0 mV full scale.

The upper range provides approximately 30% over-ranging, which extends this measurement range to ±3250 mV.

These measurement ranges are selected automatically by the *dataTaker*, to provide the best measurement resolution for a given input voltage.

The low level voltage measurement accuracy is typically better than 0.10%, and can be trimmed via software to greater accuracy by calibrating the *dataTaker* against a standard reference voltage (See Section III – Trimming for Errors in the Internal Reference Voltage).

For differential inputs, the measurement range is determined by the difference of the voltages at the +ve and –ve terminals of the analog input channels.

The absolute value of voltage at the input terminals does not affect the measurement range or accuracy, except for common mode voltage effects.

The common mode rejection ratio of >90 db means that the common mode voltage at the input terminals will have an effect in the input voltage of less than 0.001% of the common mode voltage.

The following table summarizes the measurement ranges, resolution and accuracies for low level voltage inputs.

Input Voltage Range	***Resolution***	***Measurement Accuracy ****
<±25.00mV	1μV	±0.10%
±250.0mV	10μV	±0.10%
±2500.0mV	100μV	±0.02%

* Measurement accuracy can be trimmed

Error Messages

Input voltage signals which fall outside the range the common mode range of the *dataTaker* will produce an over-range reading of –99999.9 mV or +99999.9 mV respectively.

The *dataTaker* also reports the out of range error condition with the error message 'E11–input(s) out of range' if the Messages Switch /M is enabled.

Measuring High Level Voltages

This section describes the measurement of high level differential (double ended) DC voltages and single ended DC voltages with the *dataTaker*.

The *dataTaker* range of data loggers have 2 different types of analog input channel multiplexers, solid state (CMOS) multiplexer or relay multiplexer. The models of the *dataTaker* data loggers which have each of these multiplexer types are as follows

Solid State (CMOS) Multiplexer
- *dataTaker 50, 500* and *600*

Relay Multiplexer
- *dataTaker 505, 605, 515* and *615*
- Channel Expansion Module (CEM-AD)

Throughout this section reference will be made to solid state or relay multiplexing, rather than to individual *dataTaker* models. Confirm from the above list which *dataTaker* model you are using.

The *dataTaker 50, 500* and *600* data loggers have the solid state multiplexer, and have a maximum voltage measurement range of –2500 mV to +2500 mV. However these *dataTaker*s can also measure high level voltages by externally attenuating the signal with attenuation resistors.

The *dataTaker 505, 515, 605* and *615* data loggers have the relay multiplexer, and also have precision internal attenuation resistors as standard. These *dataTaker*s have a maximum input measurement range of –100 Volt to +100 Volt, when the internal attenuators are selected via command. External attenuation of high level voltages is not normally required with these *dataTaker* models.

Note : The Channel Expansion Module (CEM-AD) does not have internal attenuation resistors, and the maximum voltage range of the modules is determined by the maximum voltage range of the *dataTaker* to which it is connected.

The *dataTaker 50* has 5 differential analog input channels, which may also be used as 10 single ended input channels. The *dataTaker 500/600* series loggers and Channel Expansion Module (CEM-AD) have 10 differential analog input channels, which may also be used as 30 single ended input channels. Any combination of differential and single ended analog input channels can be used for measuring high level voltages.

External Attenuation of High Level Voltages

High level voltages outside the range of –2500 mV to +2500 mV can be measured by externally attenuating the signal into this range, and applying the attenuated voltage to the analog input channels as a low level voltage. This applies to all *dataTaker* models which have solid state multiplexers, but can also be used for *dataTaker* models which have the relay multiplexer.

Externally attenuated high level voltage can be measured in several ways as follows

attenuated differential inputs on any of the analog input channels

attenuated single ended inputs referenced to Analog Return

attenuated single ended inputs referenced to an external common

The attenuated voltage input capability of the *dataTaker* in effect provides additional user definable voltage measurement ranges.

The attenuated voltage can be re-scaled after measurement to the magnitude of the original voltage by using

a channel factor, which represents the attenuation factor

a Polynomial or Span (See Section III – Scaling Data, Polynomials, Spans and Functions) which represents the attenuation factor. Polynomials and Spans also allow an offset to be applied to the measured voltage, and allow data to be re-scaled to other units.

Attenuation of voltage inputs also has the additional benefits of providing a greater level of input protection, and of reducing any problem common mode voltages since these are attenuated in the same proportion as the signal voltages.

External Voltage Attenuator Networks

Voltage attenuator networks are simple resistor pairs, which reduce the high level voltage signal to ±2500 mV (or ±3250 mV including the allowable over-ranging) to suit the voltage input range of the *dataTaker* data loggers.

A typical voltage attenuator network is illustrated in Figure 39.

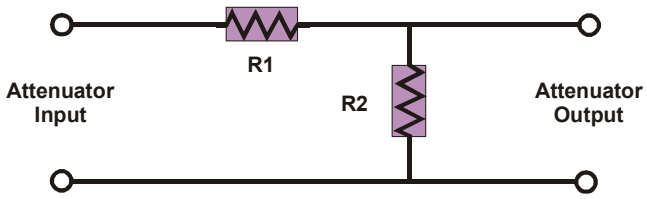


Figure 39 – Schematic of a Voltage Attenuator Network

The voltage reduction by the network is described as the voltage attenuation factor, which is calculated by the relationship

$$\text{Voltage Attenuation Factor} = (R_1 + R_2)/R_2$$

and is declared to the *dataTaker* as a scale factor in the channel options for the input channel in schedule lists (see below).

The voltage attenuation factor can be a floating point decimal value, and is used to directly re-scale the measured low level voltage from the voltage attenuator network to the magnitude of the original high level voltage.

The table below provides suitable preferred resistor values for commonly used attenuation ratios

Attenuation Factor	***R1***	***R2***
5.5:1	10 KOhm	2.2 KOhm
	100 KOhm	22 KOhm
11:1	10 KOhm	1 KOhm
	100 KOhm	10 KOhm
48:1	470 KOhm	10 KOhm
101:1	1 MOhm	10 KOhm

The attenuation ratios in the table above are not exactly 5:1, 10:1, 50:1, 100:1 etc. because this would require non-standard resistors. The resistors used above are standard or preferred magnitudes, which are commonly available.

When building attenuators, the accuracy of voltage measurement will be directly determined by the accuracy and temperature tolerance of the resistors. Where possible use precision 0.1% resistors.

Power rating of the attenuator resistors is not critical, since signals should only be a few milliamps. Generally, 0.5 Watt resistors will be suitable.

Installing Voltage Attenuator Networks

When high level voltages outside the voltage measuring range of the *dataTaker* are to be measured, the voltage signal can be externally attenuated by resistor networks installed in one of the following ways

by installing the voltage attenuator networks directly onto the screw terminals of the analog input channels

by installing the voltage attenuator networks remotely from the *dataTaker*, possibly at the source of the high level voltages

An external voltage attenuator network can be installed as follows

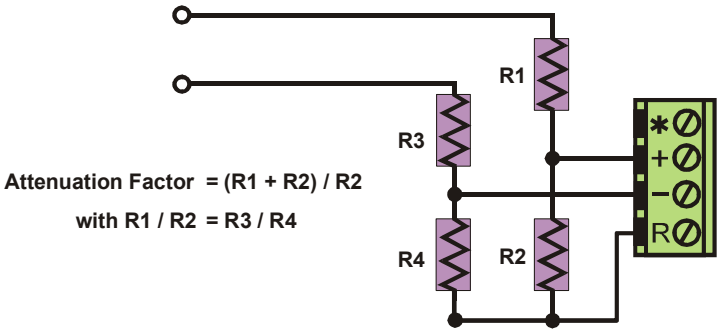


Figure 40 – Installing a Voltage Attenuator Network

Small resistor mounting pads are available from your *dataTaker* supplier. Attenuation resistors can be soldered onto the pad. The pad is directly connected into the screw terminals of input channels, and signal leads connect to the rear edge of the pad.

Differential Measurement of Externally Attenuated Voltage

Whenever externally attenuated high level voltages are measured as differential inputs, then both of the signal lines from the high level voltage signal must be attenuated by separate resistor networks. Each side of the high level voltage is connected via a separate voltage attenuator to the +ve and –ve terminals of an analog input channel.

The differential voltage measured by the *dataTaker* is the difference of the voltages between the +ve and –ve terminal. For a discussion of when to use differential mode see Section II – Measuring Low Level Voltages.

Common Mode Voltage Limits

When using external voltage attenuator networks to attenuate high level voltage signals, then any common mode voltages present in the signal are also attenuated by the voltage attenuator networks.

The maximum allowable common mode voltage for the high level voltage inputs is ±(attenuation factor x 3.5) Volts.

Note : The characteristic of attenuator networks also attenuating the common mode voltages allows low level voltages with high common mode or offset voltages to be measured.

The common mode voltage range of the *dataTaker* can be reduced by attached voltage attenuation networks, if the two attenuation networks of a differential input are not closely matched.

For example if an attenuation ratio of 10:1 is required, which is built from a 100 KOhm and a 10 KOhm resistor for the +ve input, and a 10 KOhm and 1 KOhm resistor for the –ve input, then the common mode voltage range will not be the same for both inputs and the lesser will prevail.

Differential Measurement

When high level voltages are connected to the *dataTaker* for differential measurement, both of the signal lines of the high level voltage must be externally attenuated by separate resistor networks.

High level voltages are connected to the analog input channels as differential inputs as follows

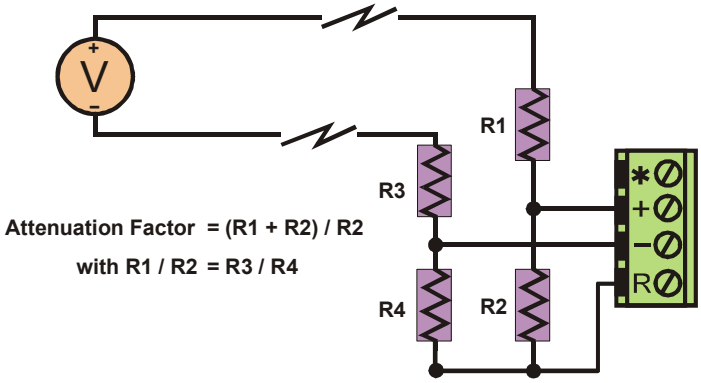


Figure 41 – Connecting Externally Attenuated High Level Voltages as Differential Inputs

Note : Both resistor networks must be of the same attenuation factor. The differential voltage measurement is made between the +ve and –ve terminals of the analog input channel.

High level voltages connected to the *dataTaker* as externally attenuated differential inputs are read when a Schedule containing the channel is executed.

The attenuation factor for the voltage attenuator networks is specified as a scale factor in the channel options for the input channel, and is used by the *dataTaker* to re-scale the measured voltage.

Using DeTransfer, externally attenuated high level voltages are measured differentially by the command for example

```
BEGIN
 RA5M
  1V(10.0,U)   5V(25.2,U)
END
```

which instructs the *dataTaker* to measure these inputs on analog input channels 1 and 5.

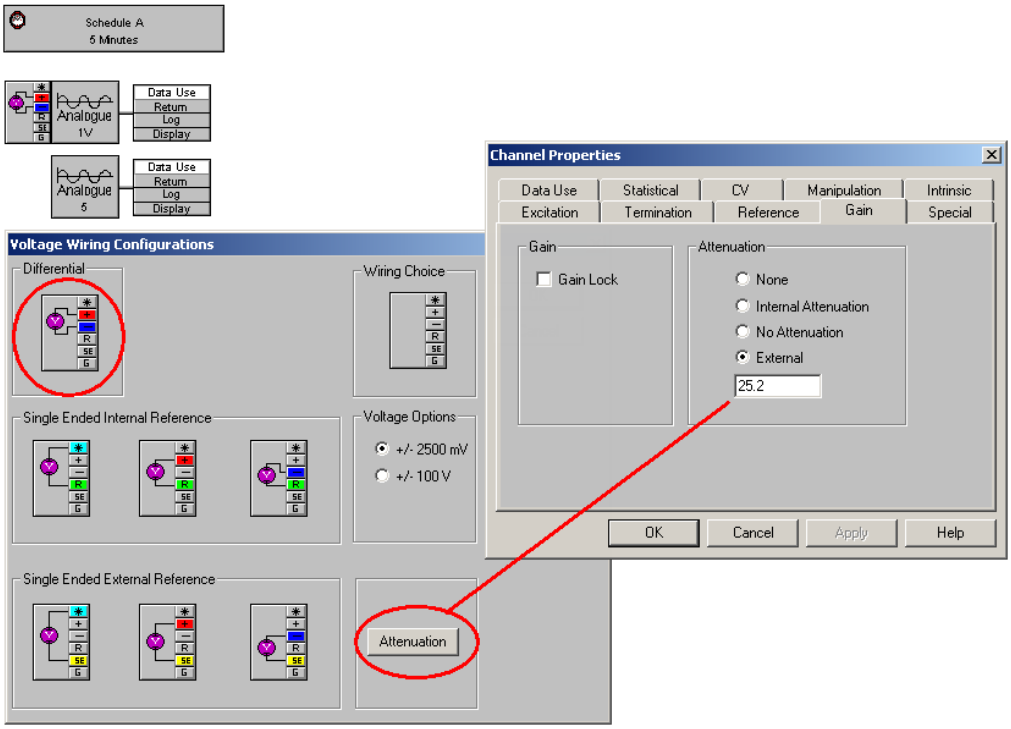
The voltage measured by channel 1 is attenuated by a 10:1 resistor network, and so the data is multiplied by 10.0 before being returned. The voltage measured by channel 5 is attenuated by a 25.2:1 resistor network, and the data is multiplied by 25.2 before being returned.

The V indicates that the signal applied to these channels is to be measured as a differential voltage.

The U channel option disconnects or Unterminates the internal 1 MOhm input termination resistors during measurement of the voltage. If the termination resistors are not disconnected, then these will influence the attenuation factor of the external resistor network.

The data is returned in units of mV.

Using DeLogger, externally attenuated high level voltages can be measured by the following Program Builder program. The differential connections are selected from the Voltage Wiring Configurations dialog which opens when you have selected the analog input channel, and the attenuation factor is specified via the Attenuation button.



The *dataTaker* will read the inputs every 5 minutes, and readings are stopped by entering a H (Halt) command.

Alternatively the data could be scaled using a Polynomial, and assigned an appropriate engineering unit (See Section III – Scaling Data, Polynomials, Spans and Functions).

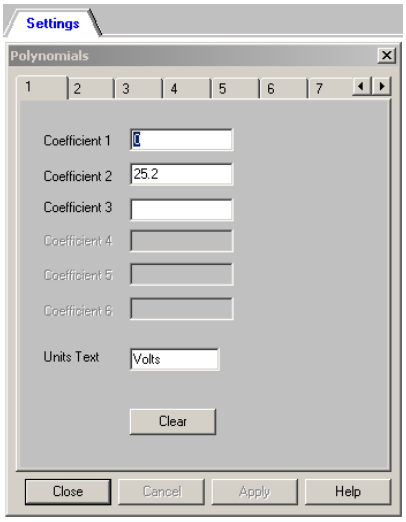
Using DeTransfer, this can be done by the commands for example

```
Y1=0,10.0"Volts"
Y2=0,25.2"Volts"

BEGIN
 RA5M
  1V(Y1,U)   5V(Y2,U)
END
```

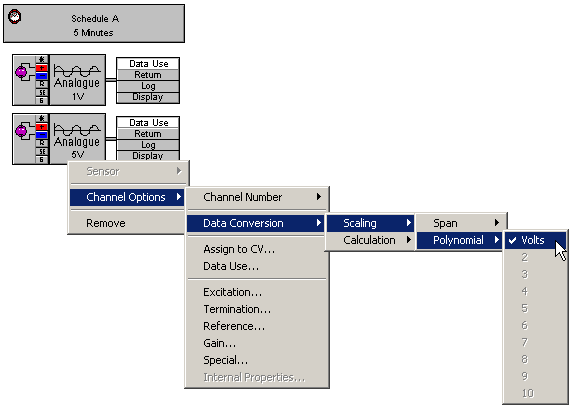
An offset can also be applied to the readings if required, by assigning a value to the first coefficient of the Polynomial.

Using DeLogger, the Polynomials must be defined in the Program Builder under the Settings:Polynomials tab, and attached to the voltage measuring channels as follows



Declare a Polynomial for the attenuator scaling

Attach the Polynomial to the channel



Single Ended Measurement of Attenuated Voltage

Externally attenuated high level voltages can be measured as single ended inputs as follows

as attenuated single ended input to the +ve, –ve or ✶ input terminal, which is referenced to Analog Return

as attenuated single ended input to the +ve, –ve or ✶ input terminal, which is referenced to an external common connected to the SE REF terminal

When high level voltages are measured as single ended inputs, only the positive or non-grounded line of the high level voltage signal is attenuated.

The *dataTaker* treats the voltage signal applied to the +ve, –ve or ✶ input terminal of the analog channel as the active input, and measures this with respect to Analog Return or SE REF.

The *dataTaker 50* does not support single ended voltage measurement for the Excite (✶) terminal.

For a discussion of when to use single ended input modes see Section II – Measuring Low Level Voltages.

Single Ended Measurement Referenced to Analog Return

Whenever high level voltages are connected to the *dataTaker* for measurement as single ended inputs, the positive signal of the high level voltage must be externally attenuated by a resistor network.

High level voltages are connected to the analog input channels as single ended inputs referenced to Analog Return as follows

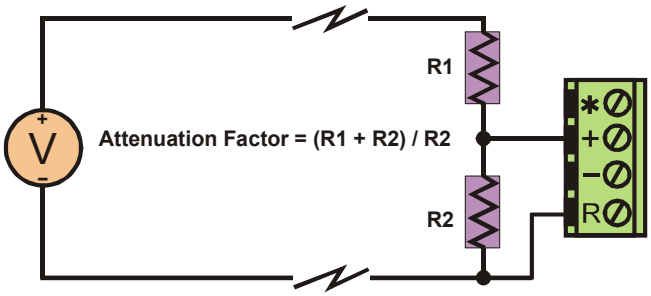


Figure 42 – Connecting Attenuated High Level Voltages as Single Ended Inputs Referenced to Analog Return

The attenuated single ended voltage is measured between the +ve, –ve or ✳ terminal and Analog Return. The *dataTaker* will measure the polarity of the input voltage.

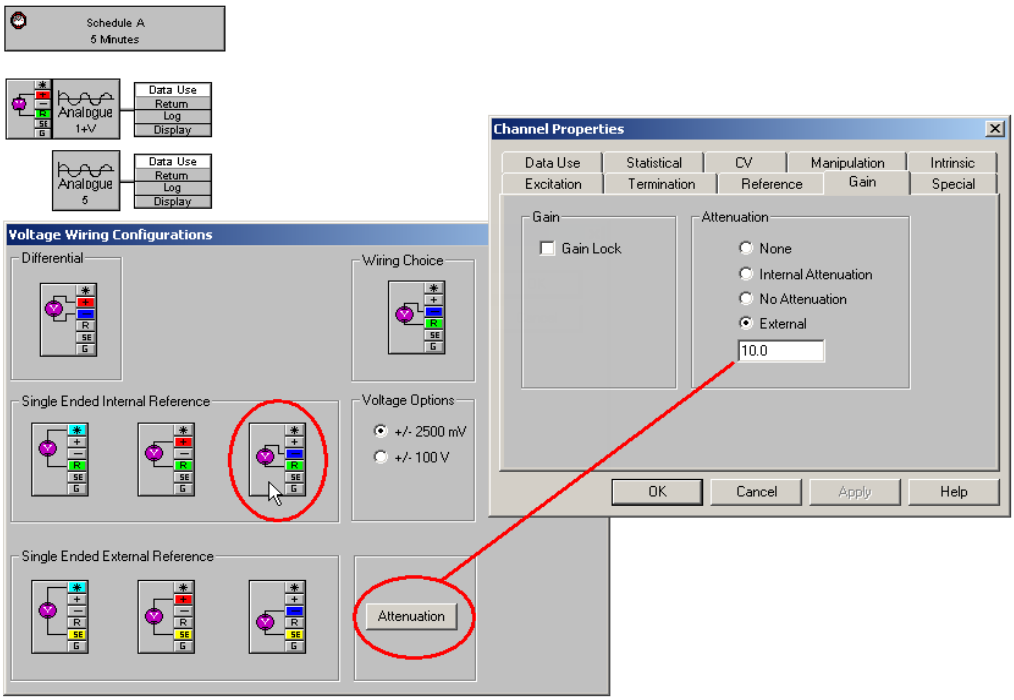
Externally attenuated high level voltages connected to the analog input channels as single ended inputs referenced to Analog Return are sampled and data returned when a Schedule containing the channel is executed. The attenuation factor is specified as a scale factor in the channel options for the input channel, and is used to re-scale the measured voltage.

Using DeTransfer, externally attenuated voltages connected as single ended inputs referenced to Analog Return are measured by the command for example

```
BEGIN
 RA5M
  1+V(5.5)   5-V(10)
END
```

which instructs the *dataTaker* to measure inputs on analog channel 1+ and channel 5–. The voltage measured on channel 1+ is attenuated by 5.5:1, and is multiplied by 5.5. The voltage measured on channel 5– is attenuated by a 10:1, and is multiplied by 10.0.

Using DeLogger,



externally attenuated high level voltages connected as single ended inputs referenced to Analog Return can be measured by the program above.

The single ended connections are selected from the Voltage Wiring Configurations dialog which opens when you have selected the analog input channel, and the attenuation factor is specified via the Attenuation button.

The internal 1 MOhm input termination resistors are not selected during single ended measurement of voltages.

This command will read the inputs every 5 minutes, and readings are stopped by entering a H (Halt) command.

Alternatively the data can be scaled using a Polynomial, and assigned an appropriate engineering unit (See above, and Section III – Scaling Data, Polynomials, Spans and Functions).

Single Ended Measurement Referenced to an External Common

Whenever high level voltages are connected to the *dataTaker* for measurement as single ended inputs, the positive signal of the high level voltage must be externally attenuated by a resistor network.

High level voltages are connected to the analog input channels as single ended inputs referenced to an external common as follows

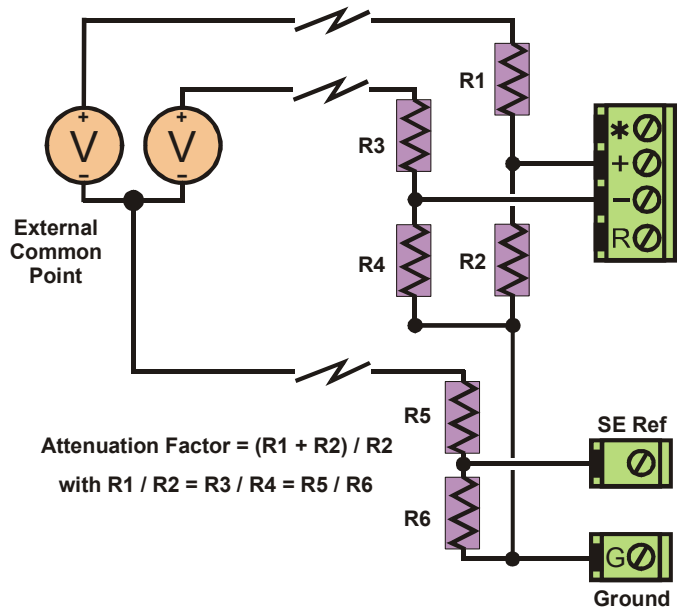


Figure 43 –Connecting Attenuated High Level Voltages as Single Ended Inputs Referenced to an External Common

Note : All of the voltage attenuator networks must have the same attenuation factor.

The attenuated single ended voltage is measured between the +ve, –ve or ✳ terminal of the analog input channel and the SE REF terminal. The *dataTaker* correctly measures the polarity of the input voltage.

Externally attenuated high level voltages connected as single ended inputs referenced to an external common are sampled and the data is returned when a Schedule containing the channel is executed.

The attenuation factor for the voltage attenuator network is specified as a scale factor in the channel options for the input channel, and used to scale the measured voltage.

Using DeTransfer, externally attenuated voltages connected as single ended inputs referenced to SE REF are measured by the command for example

```
BEGIN
 RA15M
  5+V(X,5.5)  10-V(X,12.5)
END
```

which instructs the *dataTaker* to measure these inputs on analog input channel 5+ and channel 10–.

The voltage measured on analog channel 5+ is attenuated by a 5.5:1 external attenuator network, and so is multiplied by 5.5 before being returned. The voltage measured on analog channel 10– is attenuated by a 12.5:1 external attenuator network, and so is multiplied by 12.5 before being returned.

The V indicates that the signal applied to these channels is to be measured as a voltage. The data is returned in units of mV.

The X channel option indicates that the voltage is to measured with respect to an external reference connected to the SE REF input.

Using DeLogger, externally attenuated high level voltages connected as single ended inputs referenced to SE REF can be measured by the same Program Builder program as illustrated above

The single ended connections are selected from the Voltage Wiring Configurations dialog which opens when you have selected the analog input channel, and the attenuation factor is specified via the Attenuation button.

The internal 1 MOhm input termination resistors are not selected during single ended measurement of voltages.

The *dataTaker* will read the inputs every 15 minutes, and readings are stopped by entering a H (Halt) command.

Alternatively the data can be scaled using a Polynomial, and assigned an appropriate engineering unit (See above, and Section III – Scaling Data, Polynomials, Spans and Functions).

Calibrating the External Attenuation Network

It is not essential to accurately know the resistances of the resistors used to build an external voltage attenuation network.

The attenuation factor for the resistor network can be determined by accurately measuring the input and output voltages using a good quality multimeter.

The voltage attenuation factor of the resistor network can then be calculated by

$$\text{Voltage Attenuation Factor} = V_{in} / V_{out}$$

This approach also has the advantage of taking account of the resistor tolerances, and hence producing more accurate data.

This approach also reduces the need for high precision resistors, allowing readily available resistors to be used. However the resistors used should have a low temperature coefficient.

Measurement Ranges and Accuracy for Attenuated Voltages

High level voltages of virtually any magnitude can be measured using external voltage attenuator networks to attenuate the voltage signal. The maximum input voltage range for high level voltage inputs is

$$\pm \text{ (attenuation factor x 3.25) Volts}$$

The measurement accuracy for attenuated voltage signals is a function of the tolerance of resistors used in the attenuator network. If 1% resistors are used for attenuators, then this will result in a worst case measurement accuracy of ±2% if the two resistors are at the opposite ends of their tolerance ranges.

Greater accuracy can be achieved by calibrating the external attenuator network, as described above. This approach also reduces the need for high precision resistors, allowing readily available resistors to be used. However the resistors used should have a low temperature coefficient.

Direct Measurement of High Level Voltages

High level voltages of up to ±100 Volts can be measured directly by the *dataTaker 505*, *dataTaker 605*, *dataTaker 515* and *dataTaker 615*.

These loggers all have the relay multiplexer which tolerates higher level input voltages, and have an internal attenuator network to attenuate the input voltage to the voltage measurement range of the loggers.

The internal voltage attenuator network is as a four wire attenuator, on each of the +ve, –ve, ✳ and SE REF input lines. The attenuator is located between the relay multiplexer and the analog to digital converter. Therefore the same attenuator is available to all analog input channels.

The internal voltage attenuator network has an attenuation factor of 214.6:1. The internal attenuator resistors are precisely matched during manufacture, to ensure high common mode rejection.

The internal attenuator can be selected and deselected by the two channel options A (Attenuation) and NA (No Attenuation).

Voltage measurement ranges for the *dataTaker 505, dataTaker 515, dataTaker 605* and *dataTaker 615* data loggers fitted with the relay multiplexer and internal attenuators are as follows

Input Type	*Differential Channels*	*Single Ended Channels*	*Range*	*Resolution*	*Accuracy at 25 °C*
Voltage ranges without internal attenuator input	10	30	±25mV	1µV	0.1%
			±250mV	10µV	0.1%
			±2500mV	100µV	0.02%
Voltage ranges with internal attenuator input	10	30	±7V	250µV	0.3%
			±70V	2.5mV	0.3%
			±100V	25mV	0.22%

Directly Connecting High Level Voltages

High level voltages of up to ±100 Volts can be directly connected to *the dataTaker 505, dataTaker 605, dataTaker 515* and *dataTaker 615*, and to Channel Expansion Modules (CEM-AD) fitted to these loggers as follows

as differential inputs between the +ve and –ve terminals

as single ended inputs between the +ve, –ve or ✳ terminals and Analog Return

as single ended inputs between the +ve, –ve or ✳ terminals and an external common connected to the SE REF terminal

The direct connections are completed in the same manner as for connecting low level voltages. See Figures 36, 37 and 38 for details.

Reading Directly Connected High Level Voltages

High level voltages directly connected to the analog input channels can be specified in schedule lists in either of two ways in the general formats

nHV
nV(A)

where

n	is the channel number
HV	is the high voltage input type specifier
A	is the channel option to select the internal attenuators

The internal attenuators are automatically selected by the HV input type, and must be selected using the A channel option if the V input type is used.

Using DeTransfer, high level voltages directly connected to the analog input channels are measured by the command for example

```
BEGIN
 RA15M
  1HV   6V(A)   10+HV
END
```

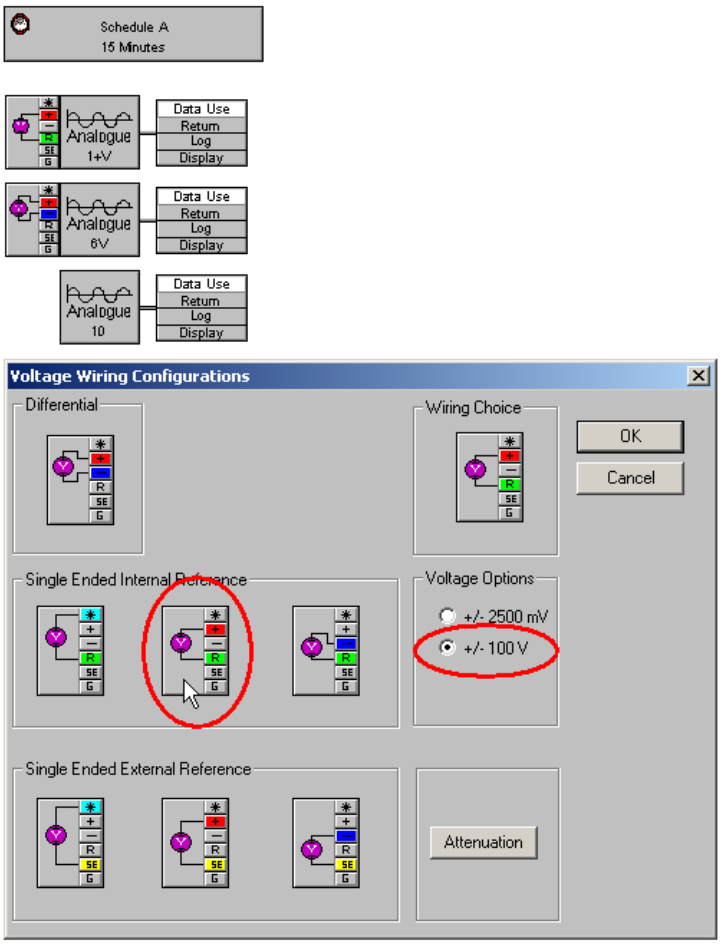
which instructs the *dataTaker* to measure the analog input channels as follows

channel 1 is measured as differential high level voltage

channel 6 is measured as a differential low level voltage with the internal attenuators selected (effectively the same as a HV measurement)

channel 10 is measured as a single ended high voltage between the +ve terminal and Analog Return

Using DeLogger, the high level voltages directly connected to the analog input channels is selected by the radio button ◉ +/- 100 V on the Voltage Wiring Configuration dialog



Other analog input types, such as Current, Current Loop, Frequency and Analog State can select the internal attenuators with channel option A, read inputs with high voltages or high common mode voltages.

The NA channel option will deselect the attenuators for the high voltage (HV) and current loop (L) input types which select the internal attenuators by default.

Over Voltage Protection

The *dataTaker 505, dataTaker 605, dataTaker 515* and *dataTaker 615* do not include built in energy absorbing high voltage or lightning protection.

However the analog input channels and the SE REF input are capable of withstanding input voltages as follows

1500 Volt for 10µS

500 Volt for 50mS

100Volt indefinitely.

The Analog Return input can withstand only 5% of these voltages.

The above withstanding voltages apply only to unselected channels. While a channel is being read (a process which typically takes 30mS), these withstanding voltages are reduced to 100 Volt.

In applications where scanning is infrequent (say not more than every 3 hours), the probability of a scan being coincident with a lightning strike is very low.

Where lightning is frequent, we strongly recommend that external energy absorbing lightning protection be wired to each sensor line.

The *dataTaker 50, dataTaker 500* and *dataTaker 600* can only withstand voltages of 10-15 Volts continuously without damage, and will be damaged by continuous voltages of greater than 20 Volts. However this damage is restricted to failure of the solid state CMOS multiplexer devices, which can easily be replaced.

Error Messages

Voltages at the output of the external attenuator networks (*dataTaker 50,500,600*) or internal attenuator networks (*dataTaker 505, 515, 605, 615*) which fall outside the range of *dataTaker* produce an over-range reading of –99999.9 Volt or +99999.9 Volt respectively.

The *dataTaker* also reports the out of range error condition with the error message 'E11–input(s) out of range' if the Messages Switch /M is enabled.

For differential inputs, an over-range condition also results if the difference of the voltages at the +ve and –ve analog input terminals falls outside the above range.

Measuring Currents

The *dataTaker* measures current signals both as direct currents, and as 4–20 mA current loops. This section describes measurement of direct current (DC) signals with the *dataTaker*. Alternating current (AC) signals cannot be measured directly with the *dataTaker*, but require external rectification and smoothing.

The *dataTaker 50* has 5 differential analog input channels, or 15 single ended input channels, which can be used for current measurements.

The *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) have 10 differential analog input channels, or 40 single ended input channels, which can be used for current measurements.

Any combination of these differential and single ended connections can be used for measuring currents, depending on installation of current measuring shunts.

There are two basic techniques for measuring currents with the *dataTaker*

using the internal current shunts provided within each analog input channel

using external current shunts

The internal current shunts provided within each analog input channel are also used for measuring signals to the 4–20 mA current loop standard.

The Internal Current Shunts

Each analog input channel of the *dataTaker* and the Channel Expansion Module has an internal 100.0 Ohm 0.1% precision current shunt resistor, which is used for the measurement of current signals. Only one current signal can be measured with each analog input channel.

The internal current shunt in each analog input channel is permanently connected between the Analog Return terminal and the *dataTaker* ground (See Figure 45). When currents are measured, the voltage drop across this shunt is selected and measured by the *dataTaker* to determine current flow.

Measuring Current Using the Internal Current Shunts

Currents in the range of ±25 mA can be connected directly to the analog input channels of the *dataTaker*, and measured using the internal current shunts. Connection of current signals using the internal shunt is illustrated below

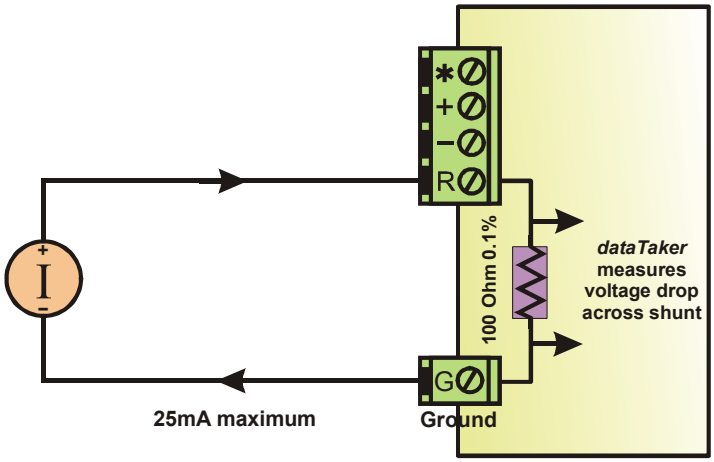


Figure 44 – Connecting Current Signals Using the Internal Current Shunts

The current signal passes into the Analog Return terminal, and flows through the internal current shunt to *dataTaker* ground. The current signal is measured by the voltage drop produced across the internal current shunt.

The current signal being measured flows between the Analog Return terminal Ground, and does not involve any of the +ve, –ve or ✳ terminals of the analog input channel.

Therefore analog channels used to measure currents via the internal current shunts can also be used to measure other differential inputs such as voltage, frequency, period or analog state. Single ended measurement on these channels is also possible if the signal source can tolerate small fluctuations on the Analog Return terminal.

Unused input terminals can be used to monitor more current signals on each channel using external shunts and externally referenced single ended inputs (see below). However analog input channels used to measure internally shunted current signals cannot also be used to measure resistance.

Current signals connected to the *dataTaker* using the internal current shunts are sampled and the data is returned when a Schedule containing the channel is executed.

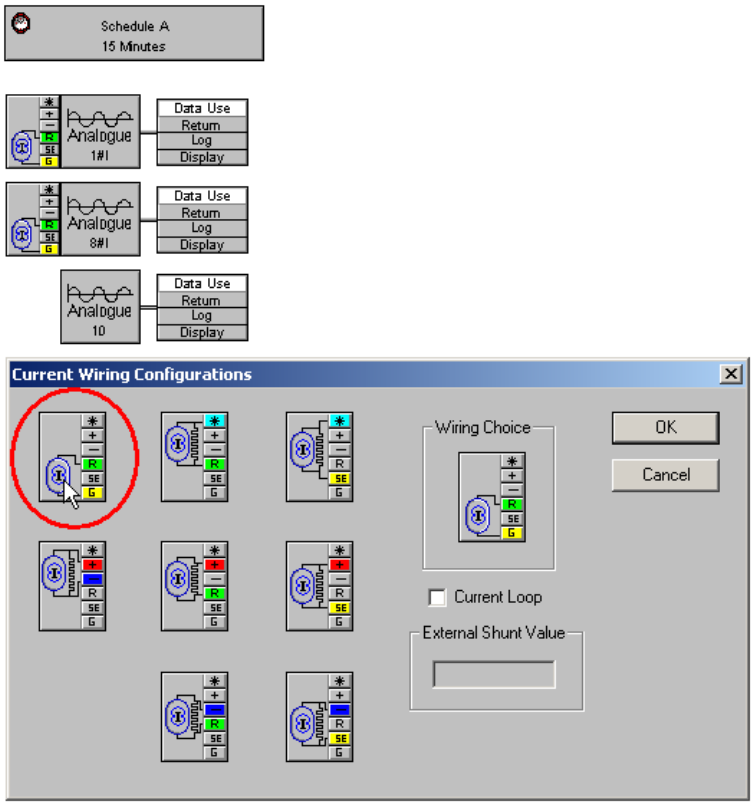
Using DeTransfer, current signals are measured using the internal shunts by the command for example

```
BEGIN
 RA15M
  5#I  8#I  10#I
END
```

which instructs the *dataTaker* to measure the current signals connected to analog input channel 5, channel 8 and channel 10.

The # indicates that the current signals are input to the Analog Return terminal, and measured using the internal current shunt. The I indicates that the signal connected to these channels is to be measured as a current. The data is returned in units of mA.

Using DeLogger, current signals can be measured using the internal shunts by the following Program Builder program.



The current signal connections using internal shunts are selected from the Current Wiring Configurations dialog which opens when you select the analog input channel.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Measuring Current Using External Current Shunts

Current signals can also be measured in both differential and single ended modes using external current shunts. A separate external current shunt is required for each current signal to be measured.

The current shunt can be of any value, and allows very large or very small DC currents to be measured. For example

a current shunt of 0.1 Ohm will allow currents of up to 25 Amp to be measured

a current shunt of 1000 Ohm will allow currents as small as 10 nA to be measured

The important factor when selecting the shunt value is that the voltage drop across the shunt at maximum current to be measured should not exceed 3250mV – the maximum measurable voltage on the ±2500 mV range of the *dataTaker*. According to Ohm's Law

Shunt Resistance (Ohms) = 3250 (mV) / Max Current (mA)

The following table lists some suitable external shunt resistance values for measuring different current ranges

Current Range	***Shunt Resistance***
0 – 25mA	100.0 Ohm
	20.0 Ohm
0 – 1000mA	2.00 Ohm
0 – 10A	200.0 mOhm
0 – 100A	20.00 mOhm

The lower value shunts must be precision resistances, and are available from specialist test and measurement suppliers. The power rating for the shunt must be appropriate for the magnitude of the current to be measured as follows

$$\text{Power Rating (Watts)} = \text{Max Current (mA)}^2 \times \text{Shunt Resistance (Ohms)}$$

The shunt resistor is best installed near the source of the current signal, especially if the current signal is large, or separation between the current signal and the *dataTaker* is appreciable.

In this case the long cable run carries the voltage produced by the current shunt, and signal loss is small because only a small bias current flows in this part of the circuit because of the high input impedance of the analog input channels. Alternatively, the external current shunt could be installed directly across the current signal input lines at the screw terminals of the *dataTaker*. This is appropriate if the signal source is close to the logger.

The current signal produces a voltage across the external current shunt. This voltage is connected to any analog input channel as a low level voltage input. Therefore inputs from external current shunts are connected to the analog input channels in the same manner as differential or single ended low level voltage inputs.

The default resistance for external current shunts for *dataTaker* is 100.0 Ohm If the external shunt resistance is not specified, then a resistance of 100.0 Ohm is assumed.

The resistance of the external current shunt must be declared as a channel option to the current input channel in schedule lists. The resistance of the external current shunt is declared in units of Ohms.

Using DeTransfer, the resistance of the external current shunt is declared in Ohms by the command for example

```
5I(250.0)
6I(249.5)
```

which specifies that the *dataTaker* is to measure

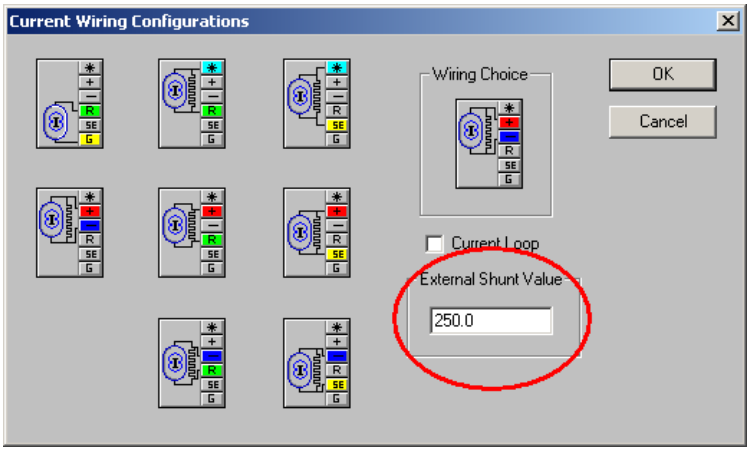
the current signal on analog channel 5, with a current shunt of 250.0 Ohm

the current signal on analog channel 6, with a current shunt of 249.5 Ohm

Note that the current shunts do not have to be the same resistance for all channels. The *dataTaker* can be programmed on each channel for a different shunt resistance.

The I indicates that the signal connected to these channels is to be measured as a current. The data is returned in units of mA.

Using DeLogger, the resistance of the external current shunt is declared in Ohms in the Current Wiring Configurations dialog which opens when you select the Current sensor type for an analog input channel in the Program Builder.



The accuracy of current measurement using external shunts is determined by the accuracy of the current shunt resistors used.

Differential Measurement

Current signals can be measured by passing through external current shunts that are connected to the analog input channels for differential measurement as illustrated in Figure 45 below.

The current signal being measured only flows in the external current shunt circuit (indicated by the red current path in Figure 46). The current signal produces a voltage across the external current shunt, which is measured as a differential voltage between the +ve and –ve terminals of the analog input channel.

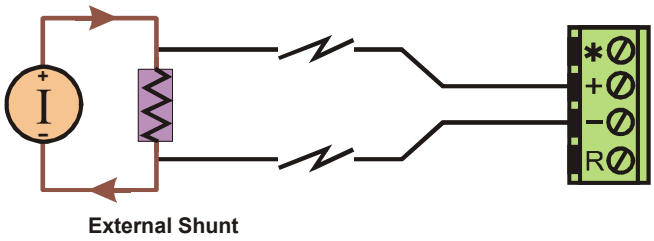


Figure 45 – Connecting Current Signals Using External Current Shunts and Differential Inputs

Current signals connected using external current shunts and differential inputs are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, current signals connected using external current shunts and differential inputs are measured by the command for example

```
BEGIN
 RA2H
  5I(50.0)  8I(100.8)
END
```

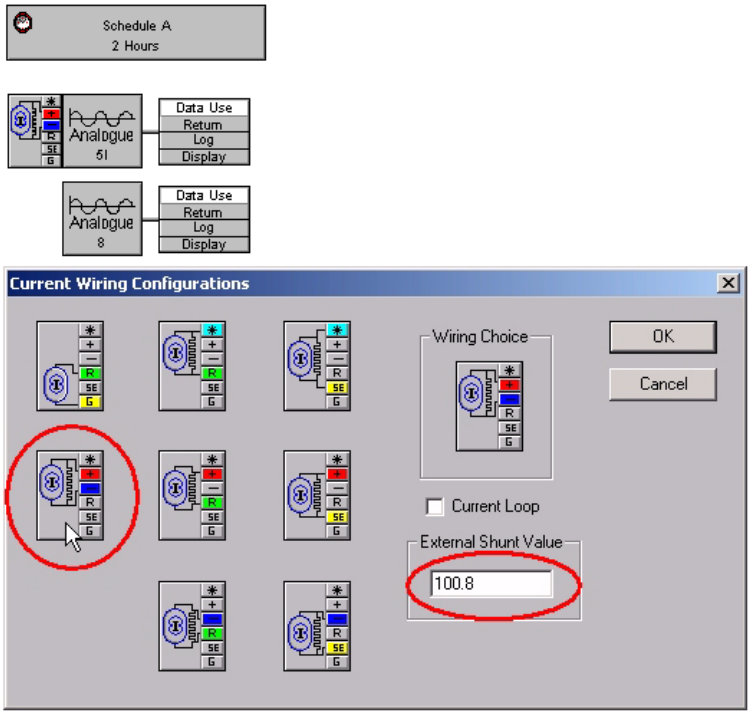
which instructs the *dataTaker* to measure the current signals passing through

an external 50.0 Ohm shunt differentially connected to analog channel 5

an external 100.8 Ohm shunt differentially connected to analog channel 8

The I indicates that the signal connected to these channels is to be measured as a current. The data is returned in units of mA.

Using DeLogger, current signals can be measured using external current shunts and differential inputs by the following Program Builder program.



Differential connections and external shunt values are selected in the Current Wiring Configurations dialog which opens when you have selected the analog input channel.

The *dataTaker* will read the inputs every 2 hours, and readings are stopped by entering a H (Halt) command.

Single Ended Measurement Referenced to Analog Return

Current signals can be measured by passing them through external current shunts that are connected to the analog input channels for single ended measurement referenced to Analog Return as follows

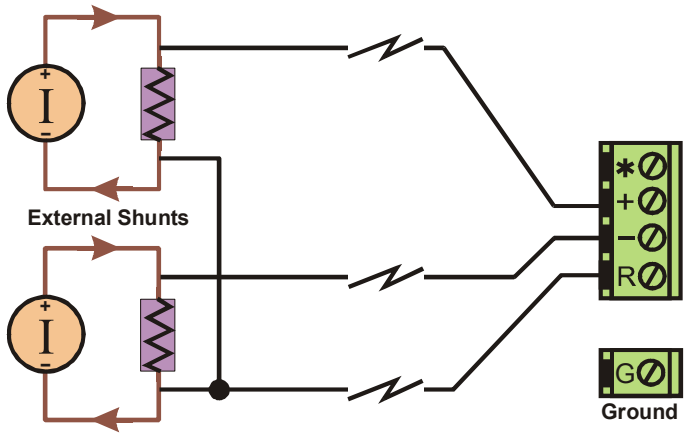


Figure 46 – Connecting Current Signals Using External Current Shunts and Single Ended Inputs Referenced to Analog Return

The current signal being measured only flows in the external current shunt circuit (indicated by the red current paths in Figure 47).

The current signals produce a voltage across the current shunts, which are measured as single ended voltages between the +ve, –ve or ✳ terminals and Analog Return.

There should be no resistance, or least resistance as possible, between the common point of the external shunt resistors, and the Analog Return terminal – especially if the shunt resistances are of smaller value.

Ideally this connection should be a heavy gauge copper wire, or even a bus bar. This will ensure that any current flowing in this path due to any ground potential differences, etc. will not be significant in the overall current measurements.

In this configuration, the *dataTaker* ground is not involved in the measurement circuit, and so a potential difference of up to ±3.5 Volts (*dataTaker 50, 500, 600*) or ±100 Volts (*dataTaker 505, 605, 515, 615*) between the external common point and the *dataTaker* ground can be permitted.

If these limits are exceeded then the *dataTaker* ground must also be connected to the external common point to overcome errors due to common mode voltage problems.

Up to three currents can be measured on each analog channel, using external current shunts and single ended input. A fourth current could still be measured using the internal current shunt.

The *dataTaker 50* does not support single ended current measurement for the Excite (✳) terminal.

Current signals connected using external current shunts and single ended inputs referenced to Analog Return are sampled, and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, current signals that are connected using external current shunts and single ended inputs referenced to Analog Return are measured by the command for example

```
BEGIN
 RA30M
  5+I(70.5)   10  ✳I(101)
END
```

which instructs the *dataTaker* to measure the current signals

passing through an external 70.5 Ohm shunt resistor connected between the +ve terminal of analog input channel 5 (5+) and Analog Return

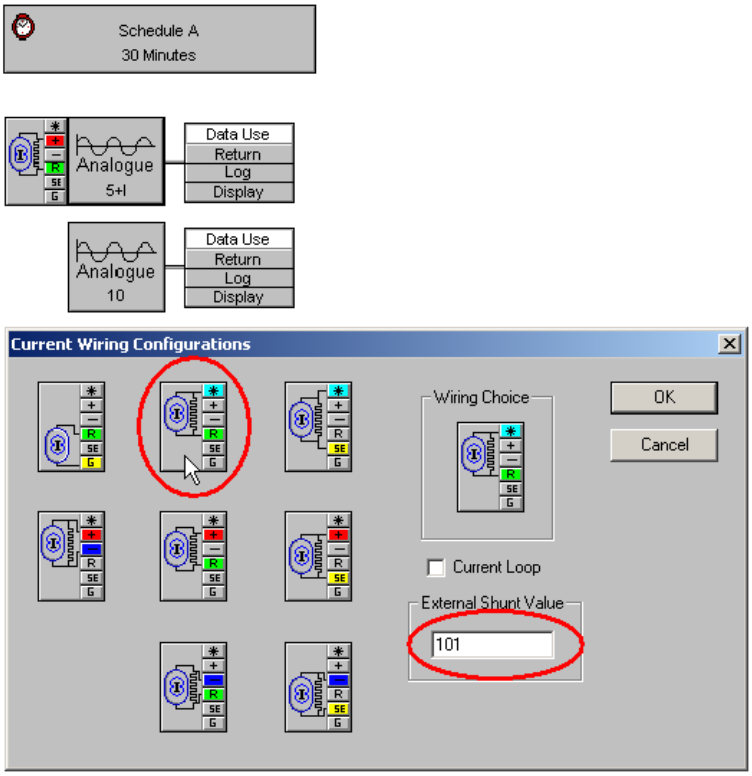
passing through an external 101.0 Ohm shunt resistor connected between the ✳ terminal of analog input channel 10 (10 ✳) and Analog Return

The I indicates that the signal connected to these channels is to be measured as a current. The data is returned in units of mA.

Using DeLogger, current signals can be measured using external current shunts and single ended inputs referenced to Analog Return by the following Program Builder program.

Single ended connections and external shunt values are selected in the Current Wiring Configurations dialog which opens when you have selected the analog input channel.

Be sure not to check the Current Loop check box, otherwise this will result in data not being returned in units of mA (see Section II – Measuring 4-20 mA Current Loops).



The *dataTaker* will read the inputs every 30 minutes, and readings are stopped by entering a H (Halt) command.

Single Ended Measurements Referenced to an External Common

Current signals can be measured by passing them through external current shunts that are connected to the analog input channels for single ended measurement referenced to an external common that is in turn connected to SE REF as follows

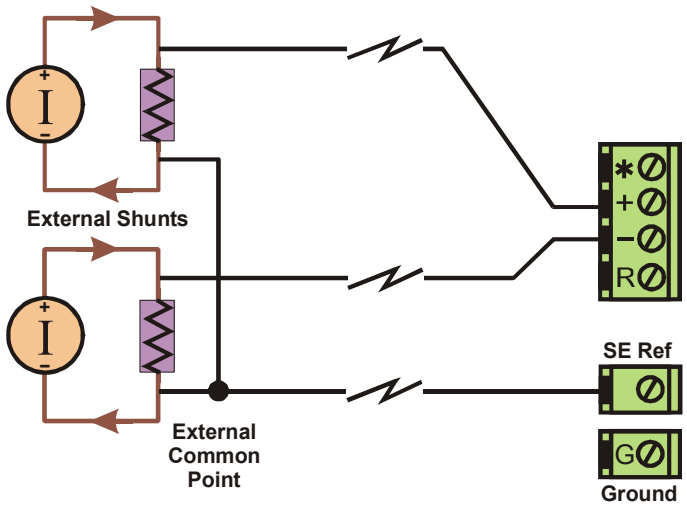


Figure 47 – Connecting Current Signals Using External Shunts and Single Ended Inputs Referenced to an External Common

The current signal being measured only flows in the external current shunt circuit (indicated by the red current paths in Figure 48).

The current signals produce a voltage across the external current shunts, which are measured as single ended voltages between the +ve, –ve or ✳ terminals and SE REF terminal.

There should be no resistance, or least resistance as possible, between the external common point of the external shunt resistors, and the SE REF terminal – especially if the shunt resistances are of smaller value.

Ideally this connection should be a heavy gauge copper wire, or even a bus bar. This will ensure that any current flowing in this path will not be significant in the overall current measurements.

In this configuration, the *dataTaker* ground is not involved in the measurement circuit, and so a potential difference of up to ±3.5 Volts (*dataTaker 50, 500, 600*) or ±100 Volts (*dataTaker 505, 605, 515, 615*) between the common external reference point and the *dataTaker* ground can be permitted.

If these limits are exceeded, then the *dataTaker* ground must also be connected to the external common reference point to overcome errors due to common mode voltage problems.

Up to three currents can be measured on each analog channel, using external current shunts and single ended input. A fourth current could still be measured using the internal current shunt.

The *dataTaker 50* does not support single ended current measurement for the Excite (✳) terminal.

Current signals connected using external current shunts and single ended inputs referenced to an external common reference point connected to SE REF are sampled, and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, current signals that are connected using external current shunts and single ended inputs referenced to SE REF are measured by the command for example

```
BEGIN
 RA1M
  2+I(55.0,X)   5-I(105.0,X)
END
```

which instructs the *dataTaker* to measure the current signals

passing through an external 55.0 Ohm shunt resistor connected between the +ve terminal of analog channel 2 (2+) and SE REF

passing through an external 105.0 Ohm shunt resistor connected between the –ve terminal of analog channel 5 (5–) and SE REF

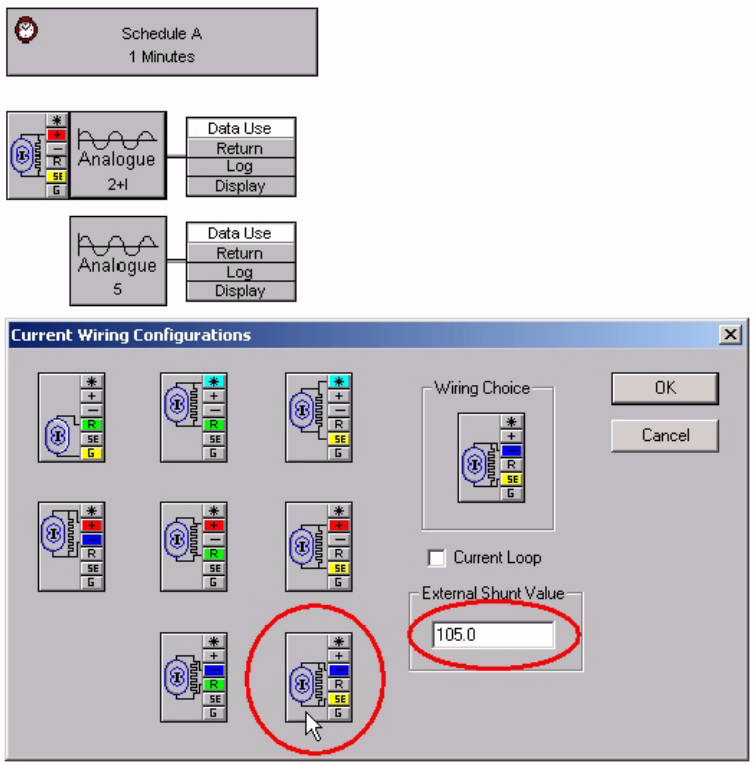
The I indicates that the signal connected to these channels is to be measured as a current. The data is returned in units of mA.

The X channel option indicates that the current is to measured with respect to an external reference connected to the SE REF input.

Using DeLogger, current signals can be measured using external current shunts and single ended inputs referenced to SE REF by the following Program Builder program.

Single ended connections and external shunt values are selected in the Current Wiring Configurations dialog which opens when you have selected the analog input channel.

Be sure not to check the Current Loop check box, otherwise this will result in data not being returned in units of mA (see Section II – Measuring 4-20 mA Current Loops).



The *dataTaker* will read the inputs every minute, and readings are stopped by entering a H (Halt) command.

Measurement Ranges and Accuracy

The measurement range for current signals using 100 Ohm internal current shunts is –25 mA to +25 mA.

The accuracy of current measurement using the 100 Ohm internal current shunts is 0.1%, which is set by the accuracy of the internal shunts.

The measurement range for current signals using external current shunt resistors can be calculated by

±(3250 / shunt resistance) mA

The accuracy of current measurement using external current shunts is determined by the accuracy of the external shunt resistor used. High tolerance resistors can be used as external current shunts if greater accuracies are required.

External current shunts can be calibrated by measuring a steady state current in the external circuit using an accurate multimeter, and adjusting the shunt value declared as a channel option to the input channel until the *dataTaker* returns the correct value.

If there are common mode or offset voltages on the current signals that are causing measurement problems, then the following corrections can be used

- external attenuation of the voltage signal from across external shunts (see Figure 42 for connection details)
- select the internal attenuators of the *dataTaker 505, 515, 605* and *615* by the A channel option in the input channel specification

The following table summarizes the measurement ranges, resolution and accuracies for current measurement using the internal current shunts.

Input Current Range	***Resolution***	***Measurement Accuracy***
±0.25mA	200nA	±0.12%
±2.50mA	1µA	±0.2%
±25.00mA	10µA	±0.12%

Error Messages

Current inputs using the internal current shunts which fall outside of the range of –25 mA to +25 mA produce an over-range reading of -99999.9 mA or +99999.9 mA respectively.

The *dataTaker* also reports the out of range error condition with the error message 'E11–input(s) out of range' if the Messages Switch /M is enabled

Measuring 4–20 mA Current Loops

Current loops provide an interface standard between a wide variety of sensors and test and measurement instruments. This interface standard is widely used in industrial and scientific applications.

The *dataTaker* data loggers support the 4–20 mA current loop standard.

There are many advantages in using current loops to interface sensors to test and measurement equipment, including

- current loop provides a common interface standard to a wide variety of sensors
- wiring requirements for carrying signals between sensors and the test and measurement equipment are simple
- long cables can be used without loss of accuracy, because cable resistance has no influence on the signal
- the interface has good immunity to electrical noise
- the current loop interface can provide isolation barriers between sensors and test and measurement equipment
- only a simple central power supply is required to drive the current loop system
- more than one measuring device can be connected in series in a current loop

The *dataTaker 50* has 5 differential analog input channels, or 15 single ended input channels, which can be used for current loop measurements.

The *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) have 10 differential analog input channels, or 40 single ended input channels, which can be used for current loop measurements.

Any combination of these differential and single ended connections can be used for measuring current loops, depending on installation of current measuring shunts.

Whenever current loops are being measured by the *dataTaker 505*, *dataTaker 515*, *dataTaker 605* and *dataTaker 615* data loggers, the internal attenuators are automatically selected to provide the greatest common mode voltage range.

There are two basic techniques for measuring current loops with the *dataTaker*

- current loop measurement using the internal current shunts provided within each analog input channel
- current loop measurement using external current shunts

Basic Concepts of Current Loops

Current loop transmitters pass or actively drive the current flow in a current loop circuit, such that the current flow is proportional to the magnitude of the physical or electrical parameter being monitored by the associated sensor.

The end points for the 4–20 mA current loop standards are

- a current of 4 mA represents the low scale (which may be a zero or negative value) output from the supported sensor
- a current of 20 mA represents the full scale output from the supported sensor

For example a current loop transmitter supporting an infra red thermometer which has a range of 0 to 500 °C, will pass or drive a current of 4 mA for a temperature reading of 0 °C, and pass or drive a current of 20 mA for a temperature reading of 500 °C.

The relationship between loop current and the physical parameter sensed by the sensor is usually linear. The manufacturer's specifications for the particular current loop device will clarify this relationship.

The current flowing in the loop is independent of the loop voltage, and so unregulated power supplies and long cables can be used with no loss of measurement accuracy.

The current loop transmitter may be powered directly from the loop current if it can operate on 4 mA, or it may be separately powered and perhaps supply the loop current.

Note: The *dataTaker* does not provide power for the current loop. A separate loop power supply must be used.

The block diagram for a current loop transmitter is illustrated in Figure 48 below

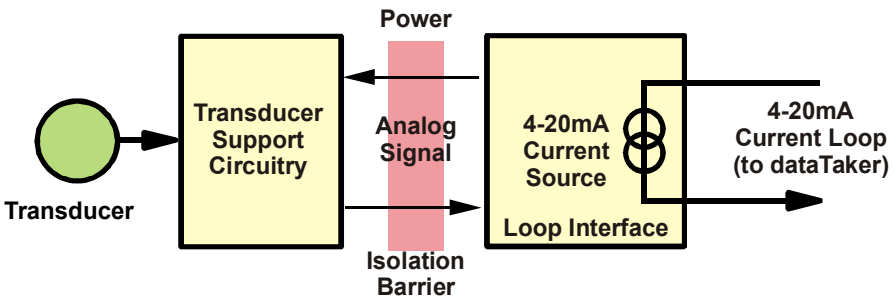


Figure 48 – Block Diagram for an Isolated Current Loop Transmitter

The current loop transmitters can also provide an isolation barrier between the sensor and the test and measurement equipment.

This isolation barrier offers several advantages

- allows the sensor to operate at a different ground potential with respect to the test and measurement instruments (500 – 2500 Volts typical), without common mode voltage or ground loop problems
- protects test and measurement instruments from exposure of the sensor to damaging high voltages.

There are five basic function blocks which make up a complete current loop based measurement system

- a sensor, which senses the magnitude of the physical parameter being measured
- an isolation barrier system
- a current loop transmitter, which translates sensor output to a current loop signal
- a current loop power supply
- a measurement device to measure the loop current

These function blocks are illustrated for a typical current loop system below

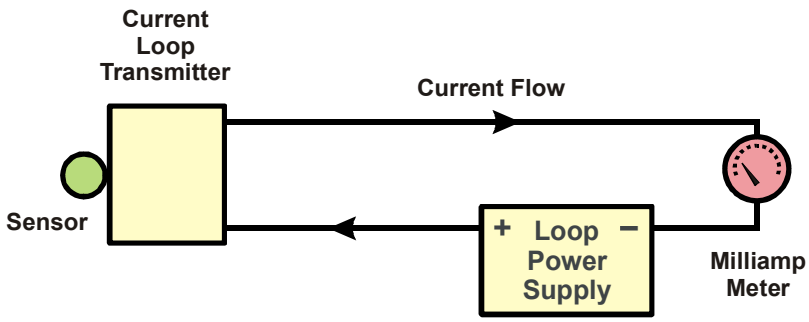


Figure 49 – Current Loop Transmitter System Function Blocks

The current loop power supply is traditionally 24 Volts DC, however 12 Volts DC is becoming more common. The power supply and must have sufficient current capacity to source full scale signal current for all transmitters, plus any power drawn by the transmitters.

For example the power supply current required to support 10 current loop transmitters at full scale (20mA) is at least

Loop Power Supply Current = 10 transmitters x 20 mA

= 200 mA total

It is good policy to over specify the loop power supply by 50 – 100%.

Installation of Current Loop Transmitters

The output of the current loop power supply should be isolated from the ground of its power supply. The negative side of the current loop power supply must be connected to the ground of the *dataTaker*.

The current loop transmitters are usually remotely located adjacent to the sensors they support, and the current loop cable is routed to the power supply and *dataTaker*.

Care must be taken to ensure that remote current loop transmitters are not connected to any other circuit, and that the cabling is not exposed to potential current leaks to ground. Either of these conditions will cause measurement errors and possible equipment damage.

The positive wires of all of the current loop transmitters are connected to the positive output of the current loop power supply.

It is advisable to place a 10 to 100 Ohm current limiting resistor in series with the current shunt in each current loop circuit to protect the *dataTaker* in the event of a short circuit in any of the current loop transmitters.

The *dataTaker* measures the current flow in the return wire of the current loop circuit. This avoids problems of excessive common mode voltage.

The loop current can be measured by the *dataTaker* either using the internal current shunts in each analog input channel, or using external current shunts (See Section II – Measuring Currents).

The Internal Current Shunts

Each analog input channel of the *dataTaker* and of the Channel Expansion Module (CEM-AD) includes an internal 100.0 Ohm 0.1% precision current shunt resistor, which is used for measurement of current and current loop signals.

When the internal current shunts are used for measurement of current loops, only one current loop can be measured with each analog input channel.

The internal current shunt within each analog input channel is internally connected between the Analog Return terminal and the *dataTaker* ground (See Figure 44). When current loops are measured, the voltage across this shunt is selected and measured by the *dataTaker* to determine current flow.

Common Mode Voltage

Current loop transmitters usually operate from a 12 or 24 Volt DC power supply.

Care must be taken when connecting current loops to the *dataTaker*s to ensure that this voltage is not connected across the logger, otherwise the common mode range of the logger could be exceeded. This requirement applies when using either internal or external current shunts.

Exceeding the common mode voltage range will certainly produce measurement errors, and will possibly also damage the logger.

The most common circumstance for the common mode voltage range of the *dataTaker* to be exceeded is where the *dataTaker* is grounded (possibly even to the same ground as the current loop power supply), and the shunt for measuring loop current is located in the positive side of the current loop. This in effect places the current loop power supply across the logger, exceeding the common mode voltage range.

dataTaker 50, 500 and *600*

The *dataTaker 50*, *dataTaker 500* and *dataTaker 600* data loggers have a common mode voltage range of ±3.5 Volts.

Most Important : It is essential when measuring current loops with the *dataTaker 50*, *dataTaker 500* and *dataTaker 600* data loggers that

- the negative side of the current loop power supply must be connected to the *dataTaker* ground
- the current shunt (internal or external) must be located in the current loop return between the negative side of the power supply and the negative terminal of the current loop transmitter
- if a number of current loops are being measured and more than one current loop power supply is used, then the negative side of each of these power supplies and *dataTaker* ground must all be connected to a common grounding point

The voltage across a 100 Ohm current shunt at the maximum loop current (20 mA) is only 2 Volts. However if this voltage is more than 3.5 Volts above or below the *dataTaker 50, 500* or *600* ground potential, then current loop measurement errors will occur.

Where external shunts are used, the common mode voltage can easily be measured with a multimeter between *dataTaker* ground and each side of the current shunt. Neither of these measurements can exceed ±3.5 Volts.

However if excessive common mode voltages cannot be avoided, then current loops can still be measured with the *dataTaker 50, 500* or *600* as follows

- use an external current shunt for each current loop to be measured
- attenuate the voltage produced across each shunt with a voltage attenuator network to reduce the common mode voltage (See Section III – Measuring High Level Voltages). Use at least a 10:1 attenuator for a 24 Volt loop power supply, or a 5:1 attenuator for a 12 Volt loop power supply.
- use data scaling (span, polynomial or direct calculations) to correct the magnitude of the resulting data

dataTaker 505, 515, 605 and 615

Whenever current loops are measured by the *dataTaker 505*, *dataTaker 515*, *dataTaker 605* and *dataTaker 615* data loggers, the internal voltage attenuator is automatically selected which increases the common mode voltage range to ±100 Volts.

Therefore the risk of exceeding the common mode voltage range of these loggers with current loops is virtually eliminated.

Sharing Current Loop Signals

Sometimes it may be necessary to measure and log current loop data from an existing transmitter and indicator/controller current loop. This may be an existing transmitter and associated panel meter readout, or existing transmitter and controller unit, etc.

Most current loop transmitters are designed to handle up to 650 Ohms of resistance in the current loop circuit. Therefore additional current shunts can be inserted into the current loop to support additional measuring devices, so long as this maximum load resistance is not exceeded. Check the specifications for the transmitter in question to determine actual capability.

It is generally acceptable to 'insert' a *dataTaker* into an existing current loop, however

- insert the *dataTaker* between the loop power supply negative terminal and the negative terminal of the transmitter
- ensure that common mode voltages of the *dataTaker* and other readout/controller devices is not exceeded
- ensure that the total loop resistance (the sum of all current shunts) does not exceed the capability of the transmitter

Measuring Current Loops Using the Internal Shunts

Current loops and loop power supplies are connected directly to the analog input channels of the *dataTaker,* when measuring loop currents using the internal current shunts within each of the analog input channels.

The loop current passes into the Analog Return terminal, through the internal current shunt to *dataTaker* ground, and out the GND terminal to the negative terminal of the loop power supply. The current loop signal is measured by the voltage produced across the internal current shunt.

The connection of current loops using the internal shunt is illustrated below

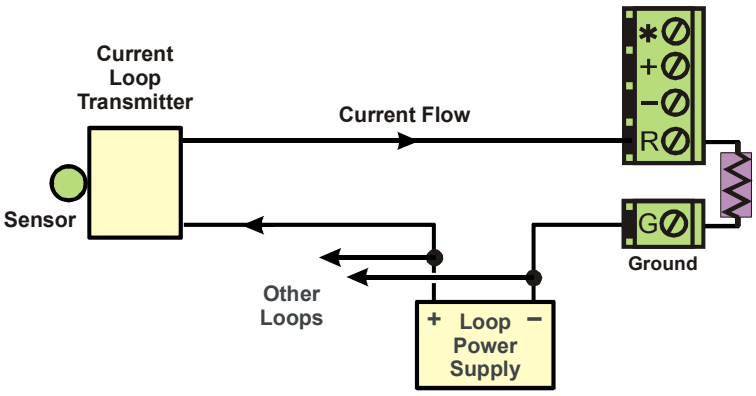


Figure 50 – Connecting Current Loops Using the Internal Current Shunts

If more than one current loop transmitter is to share a common loop power supply, then these should be connected as follows

- the positive wires each current loop transmitter is connected to the positive terminal of the current loop power supply
- the negative wires from each current loop transmitter are connected to the Analog Return of the respective analog input channels
- a single connection is made between the *dataTaker* GND terminal and the negative terminal of the loop power supply. Note that the current for all current loops being measured passes back to the loop power supply via the of the *dataTaker* ground.

Because the current loop being measured flows between the Analog Return terminal and *dataTaker* ground, and does not involve any of the +ve, –ve or ✳ terminals of the analog input channel.

Therefore analog channels used to measure current loops with the internal shunts can also be used to measure other differential inputs such as voltage, period, frequency or analog state. Single ended measurement on these channels is also possible if the signal source can tolerate small fluctuations on the Analog Return terminal.

These unused inputs can be used to monitor additional current loops on each channel using external shunts and single ended inputs referenced to SE REF (see below).

However analog input channels used to measure internally shunted current loops cannot also be used to also measure resistance.

Current loops connected to the *dataTaker* using the internal current shunts are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, current loops are measured using the internal shunts by the command for example

```
BEGIN
 RA15M
  1#L   5#L   10#L
END
```

which instructs the *dataTaker* to measure the current loop signals connected to analog input channel 1, channel 5 and channel 10.

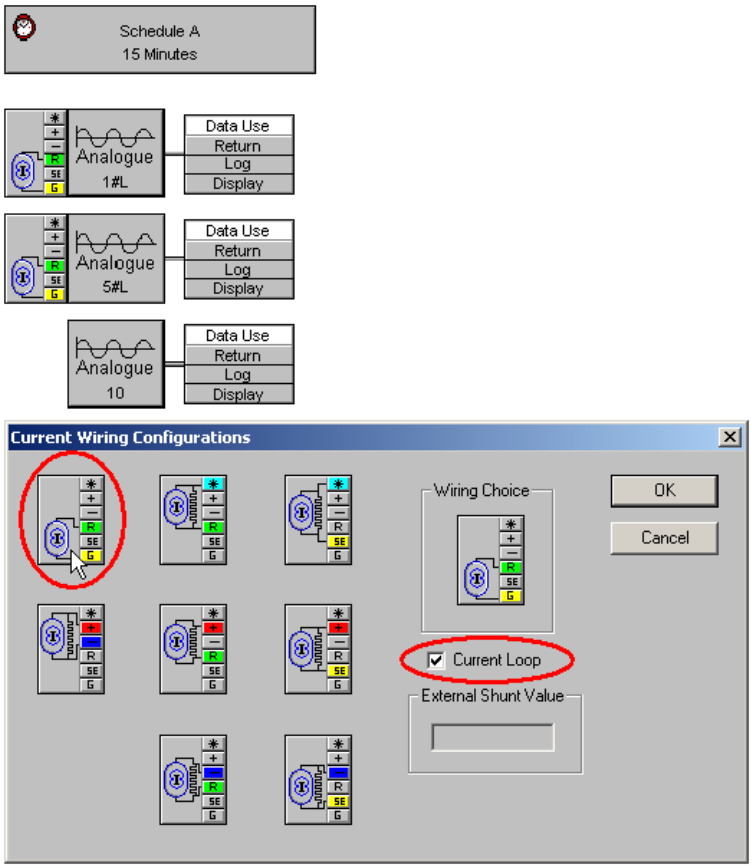
The # indicates that the current signals are input to the Analog Return terminal, and are to be measured using the internal current shunt. The L indicates that the signal connected to these channels is to be measured as a current loop.

Current loop data is returned as the percentage of the 4 – 20 mA range that the signal represents. For example a value of 50% is returned if the current loop signal is 12 mA.

Current loop signals can also be measured as channel type I, in which case data is returned in the range of 4 – 20 mA.

Using DeLogger, current loop signals can be measured using the internal shunts by the following Program Builder program. The current loop connections using the internal shunts are selected from the Current Wiring Configurations dialog which opens when you have selected the analog input channel.

Note that the Current Loop checkbox must be checked. If this is checked then data is returned as 0 – 100% data, otherwise if this is not checked then data is returned as 4 – 20 mA data.



The *dataTaker* will read the current loop signals every 15 minutes, and readings are stopped by entering a H (Halt) command.

Measuring Current Loops Using External Current Shunts

Current loop signals can also be measured by the *dataTaker* using external current shunt resistors, in the same manner as for measuring current signals using external shunts. (See Section II – Measuring Currents Using External Current Shunts).

The external current shunts may be of any appropriate value, however 100 Ohms is the most common. The accuracy of current loop measurements using external current shunts is determined by the accuracy of the current shunt resistors used.

The resistance of the external current shunt is declared as a channel option to the current loop input channels in schedule lists. The external current shunt resistance is declared in units of Ohms.

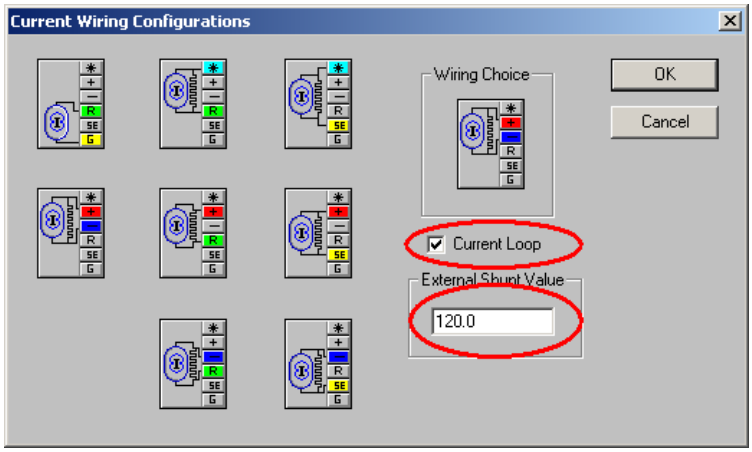
The default external current shunt resistance for the *dataTaker* is 100.0 Ohm. If an external shunt resistance is not specified, then a shunt resistance of 100.0 Ohm is assumed.

Using DeTransfer, the resistance of the external current shunt is declared in Ohms by the command for example

```
6L(120.0)
```

which specifies an external current shunt resistance of 120.0 Ohm, for the current loop signal to be measured on analog input channel 6.

Using DeLogger, the resistance of the external current shunt is declared in Ohms in the Current Wiring Configurations dialog which opens when you select the Current sensor type for an analog input channel in the Program Builder.



Note that the Current Loop checkbox must also be checked. If this is checked then data is returned as 0 – 100% data, otherwise if this is not checked then data is returned as 4 – 20 mA data.

The external current shunt is placed in the current loop return path, between the negative signal output of the current loop transmitter and the negative terminal of the loop power supply.

The loop current produces a voltage across the external current shunt, which is connected to any analog input channel as a low level voltage input. The inputs from external current loop shunts are connected to analog input channels in the same manner as differential or single ended low level voltage signals.

Differential Measurement

Current loops can be measured by passing them through external current shunts that are connected to the analog input channels for differential measurement as follows

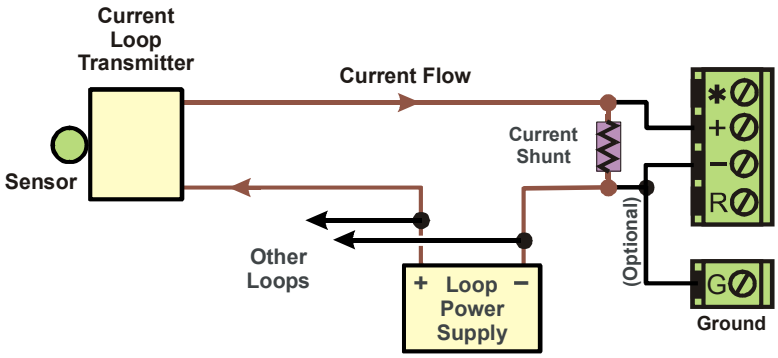


Figure 51 – Connecting Current Loops Using External Current Shunts and Differential Inputs

The loop current produces a voltage across the external current shunt resistor, which is measured as a differential voltage between the +ve and –ve terminals of the analog input channel.

The loop current signal being measured only flows in the external current shunt circuit (indicated by the red current path in Figure 52).

There is no current flow in the ground circuit of the *dataTaker*, contrary to the situation when current signals are measured using the internal current shunts.

Connecting the negative pole of the power supply to *dataTaker* ground is optional. However this is highly recommended, and will ensure that common mode voltage range is not exceeded, especially for *dataTaker 50,500* and *600* series loggers with solid state multiplexers and relatively small common mode voltage ranges.

Current loop signals connected using external current shunts and differential inputs are sampled and data is returned when a Schedule containing the channel is executed.

Using DeTransfer, current loop signals that are connected using external current shunts and differential inputs are measured by the command for example

```
BEGIN
 RA2H
  3L(20.0)  10L(122.5)..8L(122.5)
END
```

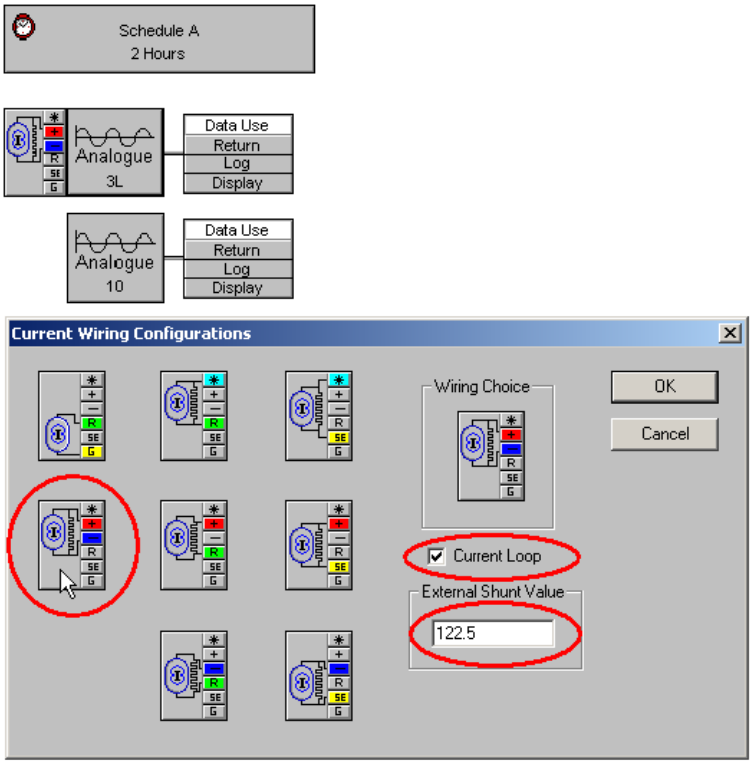
which instructs the *dataTaker* to measure the current loops as follows

- a current loop flowing in a 20.0 Ohm external shunt, which is connected differentially to analog input channel 3
- a current loop flowing in a 122.5 Ohm external shunt, which is connected differentially to analog input channel 10
- a current loop flowing in a 122.5 Ohm external shunt, which is connected differentially to analog input channel 8

The L indicates that the signal connected to these channels is to be measured as a current loop. Current loop data is returned as of the percentage of the 4–20 mA range that the signal represents. For example a value of 50% is returned if the current loop signal is 12 mA.

Using DeLogger, current loops can be measured using external current shunts and differential inputs by the following Program Builder program.

Differential connections and external shunt values are selected in the Current Wiring Configurations dialog which opens when you have selected the analog input channel.



The *dataTaker* will read the inputs every 2 hours, and readings are stopped by entering a H (Halt) command.

Single Ended Measurement Referenced to Analog Return

Current loops can be measured by passing them through external current shunts that are connected to the analog input channels for single ended measurement referenced to Analog Return as follows

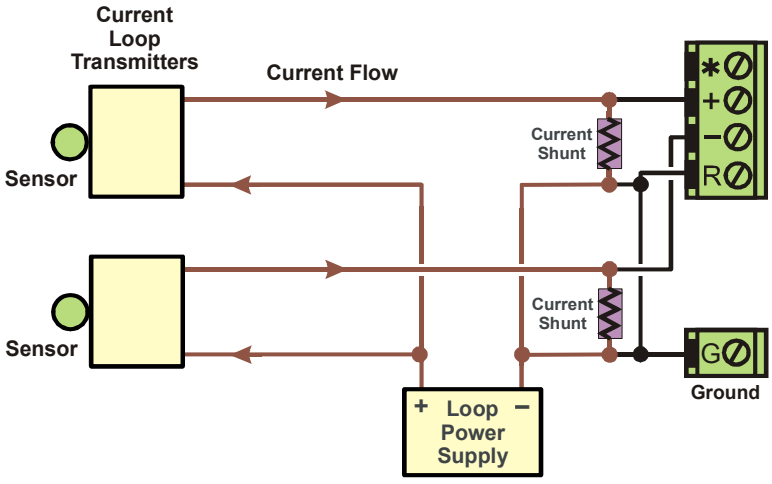


Figure 52 – Connecting Current Loops Using External Current Shunts and Single Ended Inputs Referenced to Analog Return

The current loop signal being measured only flows in the external current shunt circuit (indicated by the red current paths in Figure 53). There is effectively no current flow in the ground circuit of the *dataTaker*,

contrary to the situation when current signals are measured using the internal current shunts. The loop currents produce a voltage across the external current shunts, which are measured as single ended voltages between the +ve, –ve or ✳ terminals and Analog Return.

There must be negligible resistance between the current shunt resistors and Analog Return – especially if the shunt resistances are small in value. If the shunt resistors are located close to the *dataTaker*, then this will not be a concern.

However if the shunt resistors are located at some distance from the *dataTaker*, then these should be connected back to the logger with heavy gauge copper wire. If the remote shunts are close together, then the negative of the shunts and the negative of the loop power supply(s) should connect to a local bus bar, which in turn connects back to the logger via a heavy gauge copper wire.

These measures will ensure that any current flowing in the return path due to any ground potential differences, etc. will not be significant in the overall loop current measurements. This potential source of error can be further minimized by using external current shunts with as large a resistance value as possible – however do not exceed the maximum allowable loop resistance specified for the transmitters.

Up to three loop currents can be measured on each analog channel, using external current shunts and single ended input. A fourth loop current could still be measured using the internal current shunt.

The *dataTaker 50* does not support single ended current measurement for the Excite (✳) terminal.

Current loops connected using external current shunts and single ended inputs referenced to Analog Return are sampled, and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, current loop signals that are connected using external current shunts and single ended inputs referenced to Analog Return are measured by the command for example

```
BEGIN
 RA1H
  4+L(100.0)  5  ✳L(120.0)
END
```

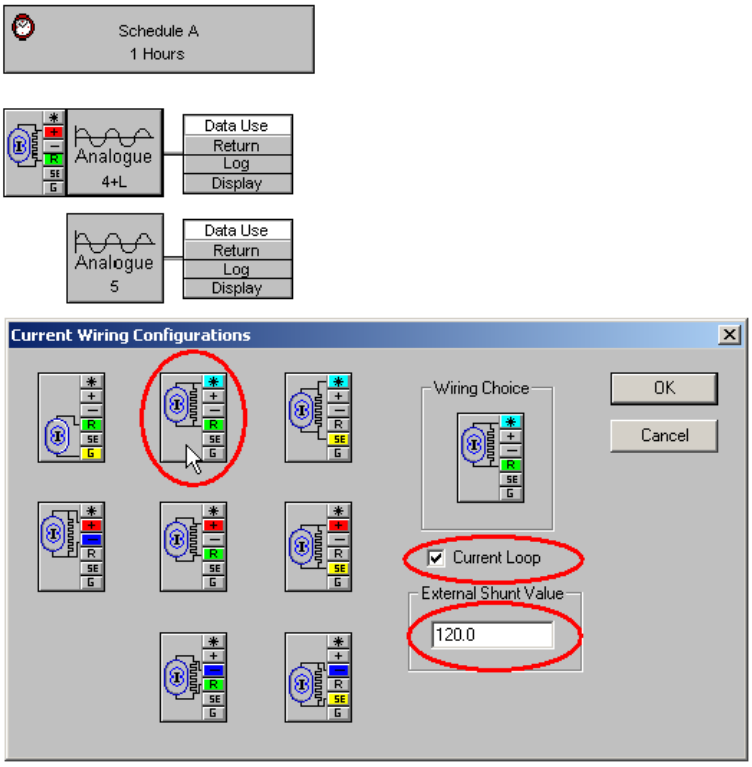
which instructs the *dataTaker* to measure

- a current loop with a 100.0 Ohm external shunt, connected as a single ended input referenced to Analog Return to analog input channel 4+
- a current loop with a 120.0 Ohm external shunt, connected as a single ended input referenced to Analog Return to analog input channel 5 ✳

The L indicates that the signal connected to these channels is to be measured as a current loop. Current loop data is returned as the percentage of the 4 – 20 mA range that the signal represents. For example a value of 50% is returned if the current loop signal is 12 mA.

Using DeLogger, current loops can be measured using external current shunts and single ended inputs referenced to Analog Return by the following Program Builder program.

Single ended connections and external shunt values are selected in the Current Wiring Configurations dialog which opens when you have selected the analog input channel.



The *dataTaker* will read the inputs every hour, and the readings are stopped by entering a H (Halt) command.

Single Ended Measurements Referenced to an External Common

Current loops can be measured by passing them through external current shunts that are connected to the analog input channels for single ended measurement referenced to an external common that is in turn connected to SE REF as follows

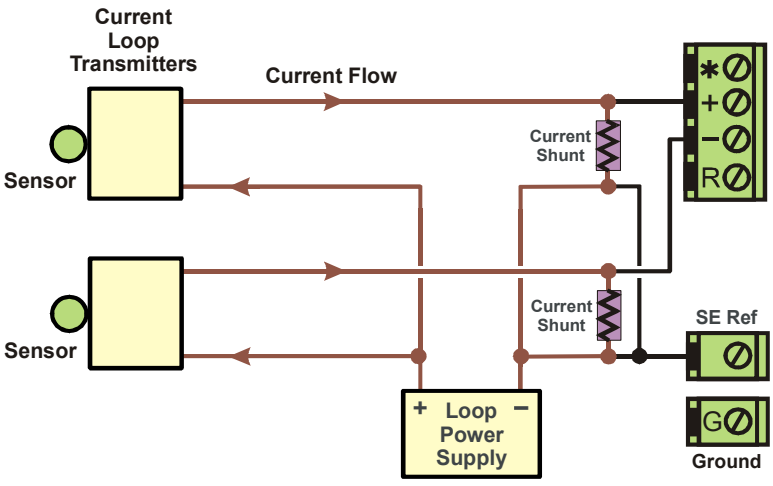


Figure 53 – Connecting Current Loops Using External Shunts and Single Ended Inputs Referenced to an External Common

The loop current being measured only flows in the external current shunt circuit (indicated by the red current paths in Figure 54).

The loop currents produce a voltage across the external current shunts, which are measured as single ended voltages between the +ve, –ve or ✳ terminals and SE REF terminals.

There must be negligible resistance between the current shunt resistors and SE REF terminals – especially if the shunt resistances are small in value. If the shunt resistors are located close to the *dataTaker*, then this will not be a concern.

However if the shunt resistors are located at some distance from the *dataTaker*, then these should be connected back to SE REF with heavy gauge copper wire. If the remote shunts are close together, then the negative of the shunts and the negative of the loop power supply(s) should connect to a local bus bar, which in turn connects back to the logger via a heavy gauge copper wire.

These measures will ensure that any current flowing in the return path due to any ground potential differences, etc. will not be significant in the overall loop current measurements. This potential source of error can be further minimized by using external current shunts with as large a resistance value as possible – however do not exceed the maximum allowable loop resistance specified for the transmitters.

In this configuration, the *dataTaker* ground is not involved in the measurement circuit, and so a potential difference of up to ±3.5 Volts (*dataTaker 50, 500, 600*) or ±100 Volts (*dataTaker 505, 605, 515, 615*) between the common external reference point and the *dataTaker* ground can be permitted. If these limits are exceeded however, then *dataTaker* ground must also be connected to the external common reference point to overcome errors due to common mode voltage problems.

Up to three loop currents can be measured on each analog channel, using external current shunts and single ended input. A fourth loop current could still be measured using the internal current shunt.

The *dataTaker 50* does not support single ended current measurement for the Excite (✳) terminal.

Current loop signals connected using external current shunts and single ended inputs referenced to an external common reference point connected to SE REF are sampled, and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, current loop signals that are connected using external current shunts and single ended inputs referenced to SE REF are measured by the command for example

```
BEGIN
 RA30M
  2+L(X,250) 3+L(X,250)  4+L(X,250)
END
```

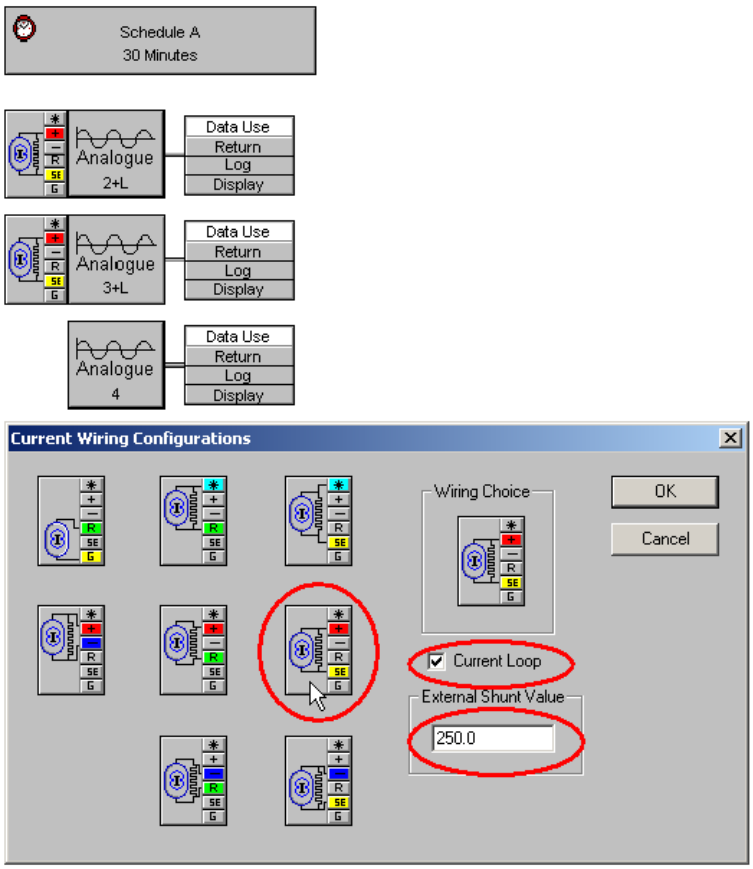
which instructs the *dataTaker* to measure

- a current loop with a 250.0 Ohm external shunt, connected as a single ended input referenced to SE REF to analog input channel 2+
- a current loop with a 250.0 Ohm external shunt, connected as a single ended input referenced to SE REF to analog input channel 3+
- a current loop with a 250.0 Ohm external shunt, connected as a single ended input referenced to SE REF to analog input channel 4+

The L indicates that the signal connected to these channels is to be measured as a current loop. Current loop data is returned as the percentage of the 4 – 20 mA range that the signal represents. For example a value of 50% is returned if the current loop signal is 12 mA.

The X channel option indicates that the current loop is to be measured with respect to an external common connected to the SE REF input.

Using DeLogger, current loops can be measured using external current shunts and single ended inputs referenced to SE REF by the following Program Builder program. Single ended connections and external shunt values are selected in the Current Wiring Configurations dialog which opens when you have selected the analog input channel.



The *dataTaker* will read the inputs every 30 minutes, and the readings are stopped by entering a H (Halt) command.

Converting Current Loop Data to Physical Units

Current loop readings can be linearized in real time to the engineering units of the physical parameter being measured, using a Span function (See Section III – Scaling Data, Polynomials, Spans and Functions).

Spans are defined for current loop inputs in terms of the lower and upper limits of the physical parameter measured.

For example a 4–20 mA thermocouple transmitter measures furnace temperatures over the range of 100 to 1000 Deg C. A loop current of 4 mA (0%) represents a temperature of 100 Deg C, and a loop current of 20 mA (100%) represents a temperature of 1000 Deg C. (The current loop transmitter performs both the cold junction compensation and linearization of the thermocouple signal).

Using DeTransfer, the Span for the thermocouple transmitter can be defined as

```
S1=100,1000,0,100"Deg C"
```

where S1 is the Span being defined.

The defined Span is assigned to the current loop input channels in the schedule lists, for example

```
S1=100,1000,0,100"Deg C"

BEGIN
 RA15M
  1#(S1)   2#L(S1)
END
```

which will return data every 15 minutes from the five 4–20 mA thermocouple transmitters in the units of temperature of Deg C.

Measurement Ranges and Accuracy

The accuracy of current loop measurement using internal current shunts is determined by the accuracy and tolerance of the shunt resistor (±0.1%).

The accuracy of current loop measurement using external current shunts is determined by the accuracy and tolerance of the shunt resistor provided by the user.

The measurement ranges, resolution and accuracy for current loop measurement using the internal current shunts are detailed below.

Input Current Loop Range	***Resolution***	***Measurement Accuracy***
0 – 100% (4 – 20mA)	0.001%	±0.1%

Error Messages

Current loop inputs using the internal current shunts which fall outside of the range of –25 mA to +25 mA produce an over-range reading of –99999.9 % or +99999.9 % respectively.

The *dataTaker* reports the error condition with the error message 'E11–input(s) out of range' if the Messages Switch / M is enabled.

Current loop inputs outside the range the 4–20 mA standard are accepted by the *dataTaker*, but produce data which is outside of the range of 0 to 100 %.

Measuring Resistance

The *dataTaker* supports two wire, three wire and four wire methods for measuring resistance.

The unknown resistance is connected between the Excite (✳) terminal and the Analog Return terminals of the input channel. The *dataTaker* outputs a precise excitation current from the Excite (✳) terminal during resistance measurement, which passes through the unknown resistance a produces a voltage across the unknown resistance.

The voltage produced across the unknown resistance by this excitation current is connected by various methods to the analog input channel. The measured voltage is then used to calculate the value of the unknown resistance for the known excitation current, according to Ohm's Law.

The *dataTaker* can output either of two precise resistance excitation currents from the Excite terminal

a precision 250.0 µA excitation current

a precision 2.500 mA excitation current

These two excitation currents combined with the three low level voltage measurement ranges of the *dataTaker* provide six resistance measurement ranges.

However two of the ranges overlap, such that there are effectively four ranges for measuring resistance as follows

The resistance measurement ranges for the 250.0 µA excitation current are

0 to 100.00 Ohm full scale

100 to 1000.0 Ohm full scale

1000 to 7000.0 Ohm full scale

The resistance measurement ranges for the 2.500 mA excitation current are

0 to 10.000 Ohm full scale

0 to 100.00 Ohm full scale

100 to 500.0 Ohm full scale

The excitation current to be used for resistance measurement is specified as a channel option in the input channel specification. The default excitation current is 250.0 µA, providing a resistance measurement range of 0 – 7000 Ohm.

The resistance ranges for each excitation current are automatically selected by the *dataTaker* to provide the highest measurement resolution for the unknown resistance. The resistance measurement ranges do not have to be selected by the user.

The resistance measuring ranges associated with the 250.0 µA excitation current is used for

measurement of resistances of up to 7 KOhm

measurement of thermistors, etc.

The resistance measuring ranges associated with the 2.500 mA excitation current is used for

precise measurement of resistances of up to 500 Ohm

measurement of Resistance Temperature Detectors (RTDs) whose resistance only changes from 20 to 320 Ohm over their temperature measuring range of –200 to 600 Deg C.

Resistance Measuring Circuit

The resistance measuring circuit of the *dataTaker* is shown below.

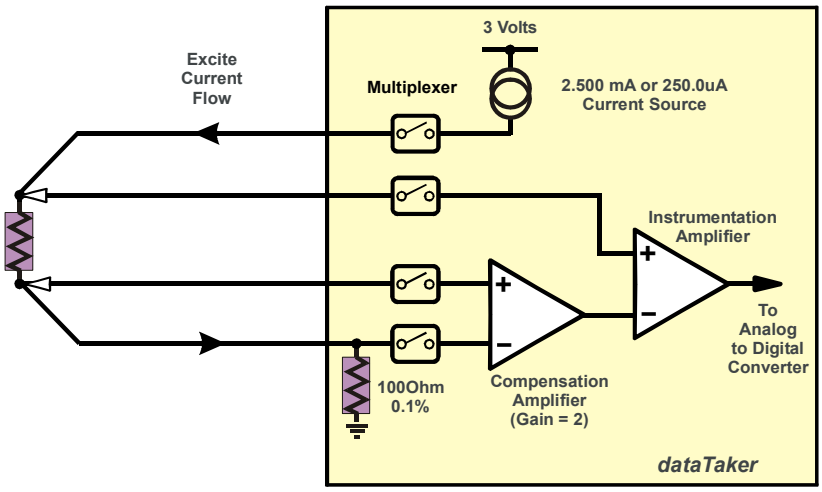


Figure 54 - *dataTaker* Resistance Measuring Circuit

The current source of 250.0 µA or 2.500 mA is output from the Excite terminal.

The voltage produced across the unknown resistance is applied between the +ve and –ve input terminals by the various methods for resistance measurement.

The compensation for lead wire resistance for the 2 wire and 3 wire methods is measured between the –ve terminal and the Analog Return terminal.

The excitation current is switched on when the channel is selected, and remains on until the analog to digital conversion is completed. Normally this is a period of 30 mS.

Specifying Excitation Current

The excitation current to be used for resistance measurement is specified as a channel option in the command to measure the resistance connected to analog input channels. If an excitation current is not specified, then the default 250.0 µA excitation current is used, providing a resistance measurement range of 0 - 7000 Ohm.

Using DeTransfer, the command for example

```
1R
```

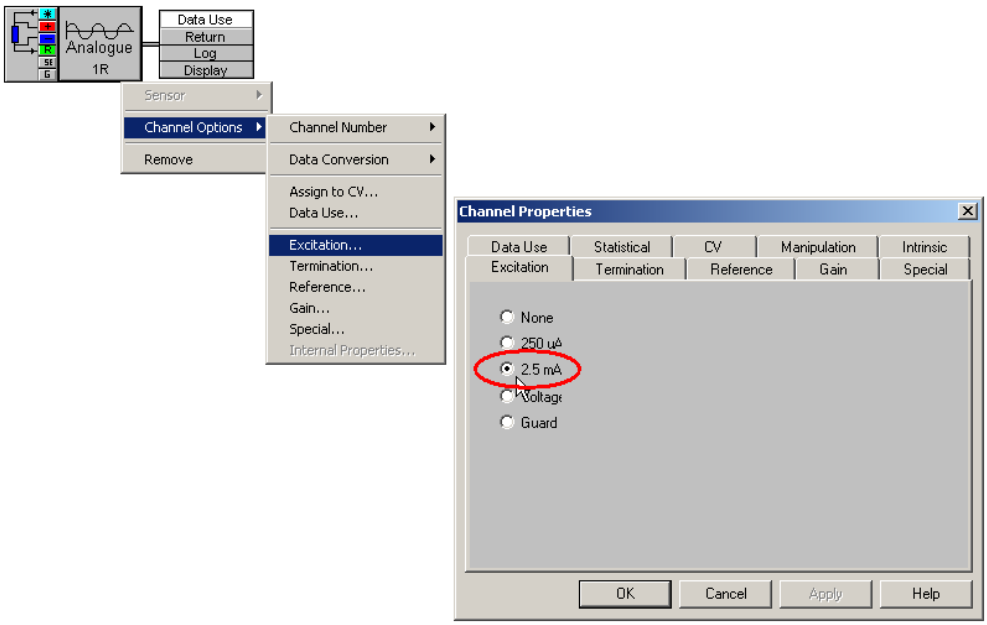
specifies that a resistance connected to analog channel 1 is to be measured, using the default 250.0 µA excitation current. The 250.0 µA excitation current is selected by default whenever a resistance input type is specified without excitation.

The 2.500 mA excitation current is selected by the Excite terminal channel option II. Using DeTransfer, the command for example

```
1R(II)
```

specifies that a resistance connected to analog channel 1 is to be measured using the 2.500 mA excitation current.

In DeLogger, the excitation for all resistance measurements defaults to 250.0 µA. If the 2.500 mA excitation current is required, then this must be enabled for each channel for which it is required. This is specified in the Program Builder via Channel Options : Excitation…



Connection of Resistances

The *dataTaker* supports 4 wire, 3 wire and 2 wire methods of resistance measurement, which are described in the following sections.

All cabling used to connect resistance devices to the *dataTaker* for measurement has innate resistance. If resistance is simply measured between two cable wires at the *dataTaker* end, then this value will include the resistance of the device under test, and the resistance of the cable.

The methods for resistance measurement employed by the *dataTaker* compensate for the resistance in the connecting cables, albeit by various means, thereby allowing use of long cables even when measuring relatively small resistances such as for RTDs.

A fourth resistance measurement method is also supported, where two resistances are placed in series. However this single ended method does not compensate for cable resistance, but allows two resistances to be measured on each analog channel.

Four Wire Resistance Measurement

The 4 wire method of resistance measurement is the most accurate, because there is effectively no current flowing in the measurement cable wires and therefore no added resistance due to the cable wires.

This method requires a 4 core cable, and has two pairs of cable wires connected across the unknown resistance. One pair of cable wires carries the excitation current, and the other pair of cable wires is used to measure the voltage that is produced across the unknown resistance by the excitation current.

This method of resistance measurement should be used where cables are long or if the individual wires are of unequal resistance.

During resistance measurement, the excitation current output from the Excite terminal passes through the unknown resistance, and returns into the Analog Return terminal. The voltage produced across the unknown resistance by the excitation current is then measured differentially between the +ve and –ve terminals.

The excitation current is flowing in the excitation circuit (shown by the red path in Figure 55 below), which is totally separate from the measurement circuit.

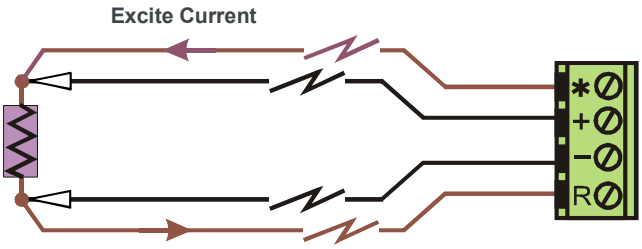


Figure 55 – Four Wire Resistance Measurement

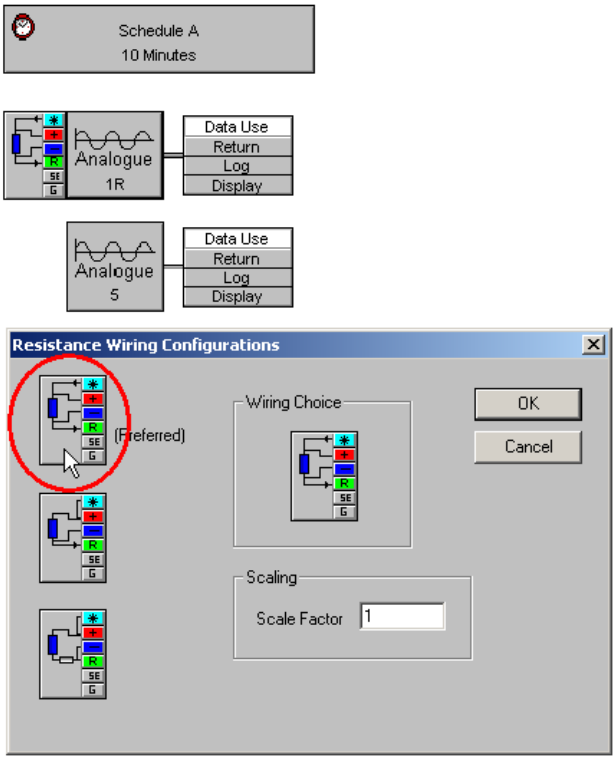
Because of the high input impedance of the instrumentation amplifier and the analog to digital converter of the *dataTaker*, there is negligible current flow in the measurement circuit. Therefore there is a negligible cable wire resistance component in the voltage measured between the +ve and –ve terminals. This measured voltage is then computed to resistance for the known excitation current.

Each resistance measured by the four wire method requires all terminals of the analog input channel. Therefore a maximum of 5 resistances can be measured using this method by the *dataTaker 50*, and a maximum of 10 resistances can be measured using this method by the *dataTaker 500/600* series loggers.

Using DeTransfer, resistance can be measured by the 4 wire method by the command

```
BEGIN
 RA10M
  1R(4W)   5R(4W)
END
```

which instructs the *dataTaker* to measure the resistance connected to analog input channel 1 and channel 5. The R indicates that the signal is to be measured as a resistance. The 4W channel option indicates that the 4 wire method of resistance measurement is to be used.



Using DeLogger, resistance can be measured using the 4 wire method by the Program Builder program above. The 4 wire connections are selected from the Resistance Wiring Configurations dialog which opens when you select the analog input channel.

The resistance data is returned in units of Ohms.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Four Wire Resistance Measurement with External Common

The 4 wire method of resistance measurement can be used with an external common which is common to all of the resistances being measured. The common point is established near the resistances, away from the *dataTaker.*

Each resistance has a pair of cable wires running from the *dataTaker* to one end, and the other end is connected to the local common point. A single pair of cable wires returns from the common point back to the logger.

This method is particularly useful when a number of resistances are to be measured at a point that is some distance away from the *dataTaker*, and almost halves the amount of cabling required.

Resistances are connected to the analog channels for 4 wire resistance measurement referenced to an external common as follows

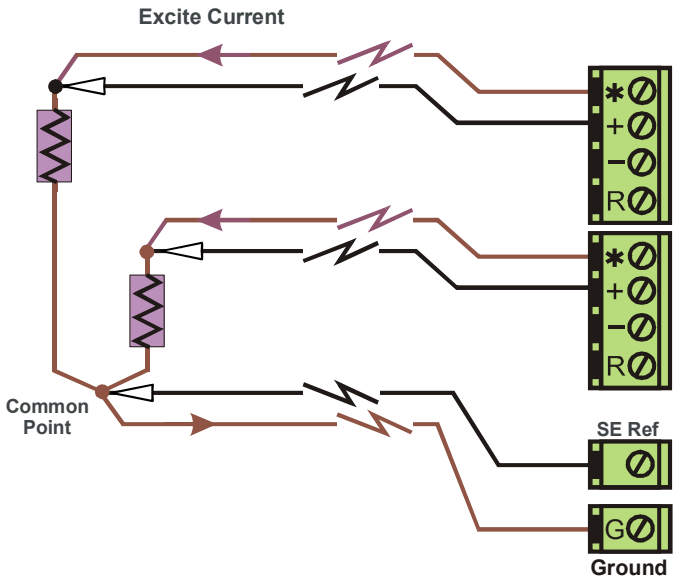


Figure 56 – Four Wire Resistance Measurement With an External Common

During resistance measurement, the excitation current from each Excite terminal passes through the unknown resistance and returns into the GND terminal via the common return wire.

The voltage produced across the unknown resistances is measured between the +ve input terminal and the common SE REF input terminal.

Up to 5 resistances can be measured by the 4 wire method with an external common by the *dataTaker 50*, and up to 10 resistances can be measured by the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) .

Resistances connected to the analog channels using the four wire method with an external common are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA15M
  1+R(4W,X,II)   2+R(4W,X,II)   3+R(4W,X,II)
END
```

instructs the *dataTaker* to measure the resistances connected to analog input channels 1+, 2+ and 3+ using an external common point.

The R indicates that resistance is to be measured. The resistance data is returned in units of Ohms.

The 4W channel option indicates that the resistances are measured by the four wire resistance measurement method for cable compensation.

The X channel option indicates that the resistances are to be measured with respect to the external common connected to the SE REF terminal.

The II channel option indicates that the 2.500 mA excitation current is to be used for measuring the resistance connected to channel 10. The default 250.0 µA excitation current is used for measuring the resistances connected to channel 1.

DeLogger does not directly support this method of resistance measurement. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

The command will read the inputs every 15 minutes, and readings are stopped by entering a H (Halt) command.

Three Wire Resistance Measurement

Most resistance measurements are made using the 3 wire resistance measurement method, which also compensates for cable wire resistances in the measurement circuit provided that the two current carrying leads are of the same resistance.

In this method one cable wire is connected to one end of the unknown resistance, and two cable wires are connected to the other end.

The two cable wires connected to the lower end of the unknown resistance are used to measure the resistance of the cable wire, which is then used to compensate the overall reading of resistance.

Resistances are connected to the analog input channels for 3 wire method of resistance measurement as follows

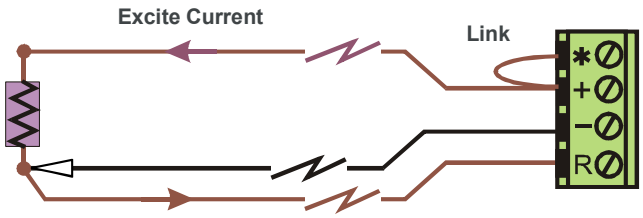


Figure 57 – Three Wire Resistance Measurement

During resistance measurement, the excitation current from the Excite terminal passes via the link and upper cable wire, then through the unknown resistance and returns into the Analog Return terminal (shown by the red path).

The –ve terminal has a much higher input impedance than the Analog Return terminal, and so the excitation current returns via the cable wire connected to Analog Return. There is no appreciable current flowing in the cable wire connected to the –ve terminal.

The voltage is measured between the +ve and –ve terminals, which combines both the voltage produced by the unknown resistance and voltage produced by the cable wire resistance.

The cable wire connecting to the –ve terminal is used to accurately measure the voltage due to the resistance of the return cable wire. The *dataTaker* then uses this voltage to correct the overall voltage measured, such that the result is the voltage produced by the unknown resistance. This is then computed to resistance for the known excitation current.

This technique for correcting for cable wire resistance assumes that the three cable wires are of equal resistance. This in turn implies that cables used for connecting resistance inputs should be of equal length and gauge of wire.

Up to 5 resistances can be measured using the 3 wire method by the *dataTaker* 50, and up to 10 resistances can be measured by the *dataTaker* 500/600 series loggers and the Channel Expansion Module (CEM-AD) .

Resistances connected using the three wire method are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

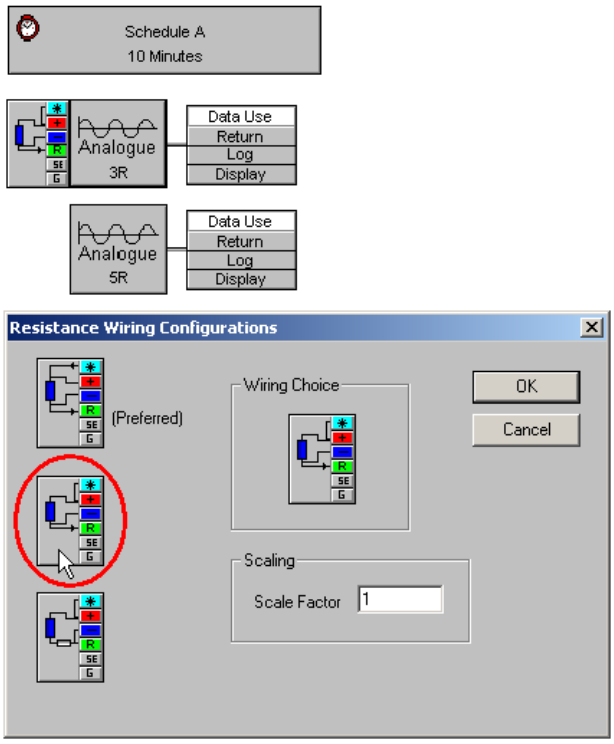
```
BEGIN
 RA10M
  3R(II)   5R
END
```

instructs the *dataTaker* to measure the resistances connected to analog input channels 3 and 5.

The R indicates that the signal applied to these channels is to be measured as a resistance. The resistance data is returned in units of Ohms.

The II channel option indicates that the 2.500 mA excitation current is to be used for measuring the resistances connected to channel 3. The default 250.0 µA excitation current is to be used for measuring the resistance connected to channel 5.

Using DeLogger, resistance can be measured using the 3 wire method by the following Program Builder program. The 3 wire connections are selected from the Resistance Wiring Configurations dialog which opens when you select the analog input channel.



The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Two Wire Resistance Measurement

The 2 wire resistance measurement method is the simplest method for measuring resistance with the *dataTaker*.

Only a single pair cable is required to connect the unknown resistance to the logger, and carries both the excitation current and the signal. Links are installed between the Excite and +ve terminals, and between the Analog Return and –ve terminal, to provide excitation current and return.

This method can be used wherever an existing sensor with just two cable wires must be connected to the *dataTaker*. However this method has the disadvantage of including the cable wire resistances in the measured resistance.

During resistance measurement, the excitation current output from the Excite terminal passes through the unknown resistance R and cable wires, and returns into the Analog Return terminal.

The voltage produced across the unknown resistance and the cable wire resistances is measured between the +ve and –ve terminals.

The connection of resistances to the analog input channels for 2 wire resistance measurement is illustrated in Figure 58a below

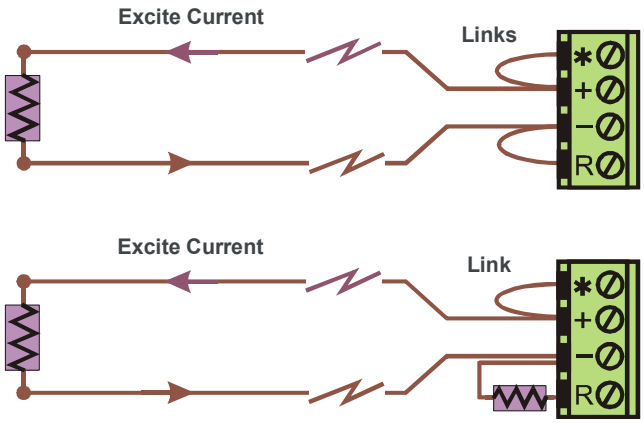


Figure 58 – Two Wire Resistance Measurement

Cable wire compensation is performed during measurement of the resistance, but is of little significance since the resistance of the links is very much less than that of the cable wires.

Since the resistance of the cable wires connecting the unknown resistance to the logger are included in the measurement, the two wire resistance measurement method is best used where the unknown resistance is large relative to the resistance of the cable wires.

However cable wire resistance compensation can be approximated by replacing the link between the –ve terminal and Analog Return with a resistor of equal value to that of the cable resistance, as shown in Figure 58b above.

Alternatively the error produced by the constant cable resistance could be corrected using a Polynomial.

Up to 5 resistances can be measured using the 2 wire method by the *dataTaker* 50, and up to 10 resistances can be measured by the *dataTaker* 500/600 series loggers and the Channel Expansion Module (CEM-AD) .

Resistances connected using the 2 wire method are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  1..5R  10R
END
```

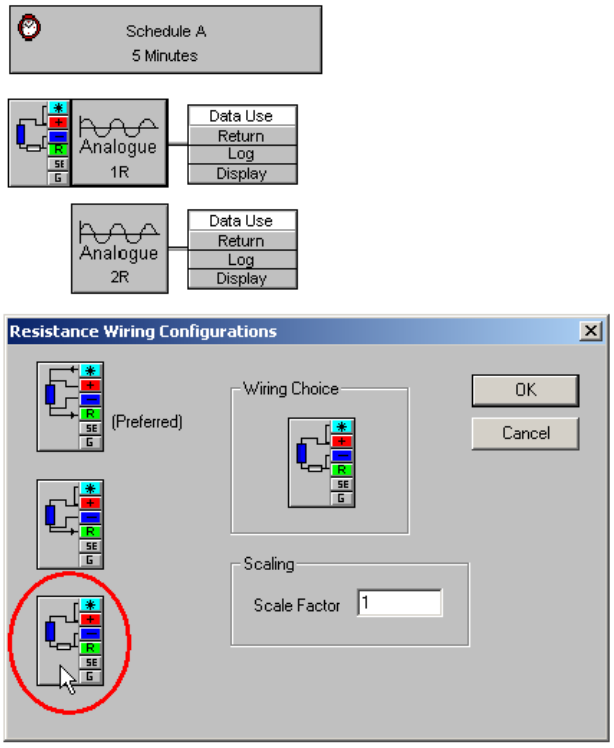
instructs the *dataTaker* to measure the resistances connected to analog input channels 1 through 5 inclusive, and analog input channel 10.

The excitation current option is not specified, and so the default 250.0 µA excitation current is used.

The R indicates that the signal applied to these channels is to be measured as a resistance. The data is returned in units of Ohms.

Using DeLogger, resistance can be measured using the 2 wire method by the following Program Builder program.

The 2 wire connections are selected from the Resistance Wiring Configurations dialog which opens when you select the analog input channel.



The *dataTaker* will read the inputs every 5 minutes, and readings are stopped by entering a H (Halt) command.

Measuring Resistance Using Single Ended Measurement

Two resistances can be measured using a combination of 4 wire input and a single ended input for each of the analog input channels.

Note : This method does not provide cable wire compensation for the resistance connected as a single ended input.

In this method, pairs of unknown resistances are connected to each analog input channel.

Up to 10 resistances can be measured using this method by the *dataTaker* 50, and up to 20 resistances can be measured by the *dataTaker* 500/600 series loggers and the Channel Expansion Module (CEM-AD) .

Resistances are connected to the analog input channels as single ended inputs as follows

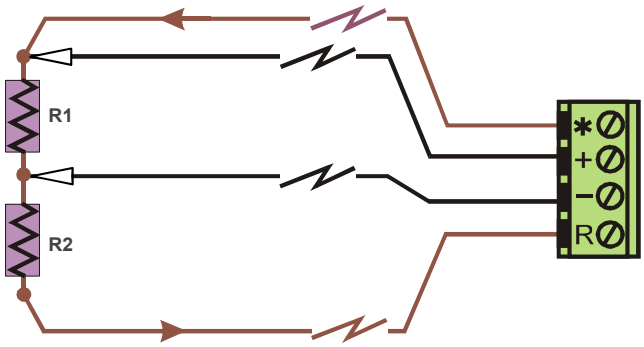


Figure 59 – Connecting Resistances for Single Ended Inputs

During resistance measurement, the excitation current from the Excite terminal passes through the unknown resistances and returns into the Analog Return terminal (shown by the red path).

The voltage produced across the two unknown resistances as a result of this current is measured between the +ve terminal and –ve terminal, and between the –ve terminal and Analog Return.

Resistances connected to an analog input channel as single ended inputs are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1R(4W)   1-R
END
```

instructs the *dataTaker* to measure the two resistances as follows

- the resistance connected between the +ve and the –ve terminal of analog input channel 1 (R1 in Figure 59) is measured by the 4 wire method
- the resistance connected between the –ve terminal of analog input channel 1 and Analog Return 1 (R2 in Figure 59) is measured uncompensated

The R indicates that the signal applied to these channels is to be measured as a resistance. The data is returned in units of Ohms.

The 4W channel option indicates that the resistance is measured by the four wire resistance measurement procedure.

The excitation current channel option is not specified, and so the default 250.0 µA excitation current is used.

The command will read the inputs every 10 minutes, and the readings are stopped by entering a H (Halt) command.

DeLogger does not directly support this method of resistance measurement. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

Measurement Ranges and Accuracy

The *dataTaker* has three resistance measurement ranges for each of the 250.0 µA and 2.500 mA Excite terminal excitation currents.

The resistance measuring ranges for of the each excitation currents are selected automatically by the *dataTaker*.

The following table summarizes the measurement ranges, resolution and accuracy for resistance measurement for each excitation current.

Resistance Range Ohms	***Resolution***	***Measurement Accuracy***
250 µA Excitation		
100 500 7000	5 mOhm 50 mOhm 500 mOhm	±0.1% ±0.2% ±0.2%
2.5 mA Excitation		
10 100 500	0.5 mOhm 5 mOhm 50 mOhm	±0.2% ±0.1% ±0.2%

Error Messages

Resistances which exceed the maximum value of 7000 Ohm produce an over-range reading of 99999.9 Ohm.

The *dataTaker* also reports the error condition with the error message 'E11–input(s) out of range' if the Messages Switch /M is enabled

Measuring Frequency and Period

The *dataTaker* is able to measure both the frequency, and the period of frequency. The period of the frequency is the inverse of the frequency.

The frequency signals are connected to the analog input channels of the *dataTaker* in the same manner as low level voltage signals.

The *dataTaker 50* has 5 differential analog input channels, which can also be used as 10 single ended input channels. The *dataTaker 500/600* series loggers and Channel Expansion Module (CEM-AD) have 10 differential analog input channels, which can also be used as 30 single ended input channels.

Any combination of differential and single ended analog input channels can be used for measuring the frequency or period of signals.

Frequency Measurement

A comparator is used in the frequency measurement circuit of the *dataTaker* to detect the threshold crossings. The amplitude of the frequency signal should be greater than 100 mV peak to peak, but not greater than the input common mode range of the *dataTaker*.

For *dataTaker 50, 500* and *600* the input common mode range is –3.5 Volts to +3.5 Volts.

For *dataTaker 505, 605, 515* and *615* the input common mode range is –100 Volts to +100 Volts if the internal attenuators are selected.

The frequency signal to be measured by-passes the VCO stage of the analog to digital converter, and is fed directly into the counter stage of the analog to digital converter.

Frequency signals are counted by the analog to digital converter counters for between 16 to 40 mS. The frequency of the input signal is calculated from the mean period of the signal over the sampling time.

The period of the input signal is calculated as the inverse of the frequency.

Frequency vs Amplitude

The range of input frequency signals that can be measured by the *dataTaker* data logger is dependent on the amplitude of the signal. As the amplitude of the frequency signal decreases, the upper limit for frequency measurement increases.

This relationship between signal amplitude and the limit of frequency measurement is illustrated below

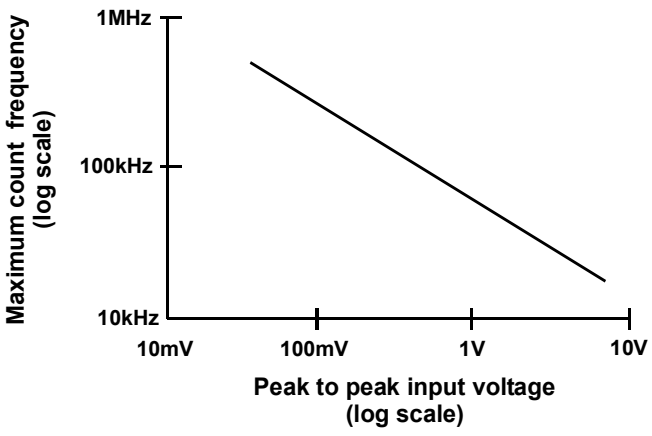


Figure 60 – Upper Limit of Frequency Measurement in Relation to Signal Amplitude

The *dataTaker* can measure frequencies in the range of 0.102 Hz to 20,000 Hz, and can measure periods of frequencies in the range of 50 µSec to 9.8 Sec, depending on the amplitude of the signal.

Data from frequency inputs can be returned either in terms of the frequency (Hz), or in terms of the period of the frequency (µSec).

Range of Frequency Measurement

By default the *dataTaker* measures frequency between 33Hz to 20KHz. However frequency can be measured down to 0.102Hz by changing default settings.

The default minimum frequency is set by the maximum period that the logger can wait until at least two pulses have been received to allow calculation of frequency. The 33Hz default minimum is determined by a default maximum period of 30 mSec.

If frequencies less than 33Hz are to be measured, then the maximum period must be increased accordingly. The maximum period is defined by the channel factor for the frequency and period channel types, which by default is 30mSec.

The maximum allowed period is 9800 mSec, which permits measurement of frequency down to 0.102Hz. Be aware that as the frequency decreases, then the time taken to complete a single measurement increases, such that a frequency of 0.102Hz will take 9.8 seconds to get a single reading. This then has implications on maximum sampling rates of other channels, since the logger waits until the two pulses are received.

The method for setting the minimum frequency measurement is illustrated in the example below for measuring frequency using differential connections.

Measuring Frequency as Differential Inputs

When frequency signals are measured as differential inputs, the frequency signal is connected between the +ve terminal and the –ve terminals of the analog input channels of the *dataTaker*.

The comparator threshold level for frequency measurement is the voltage connected to the –ve terminal of the input channel.

If the frequency of a voltage signal greater than 3 Volts is to be measured using either a *dataTaker 50, 500* or *600* data logger, then the signal must be externally attenuated to the common mode voltage of ±3.5 Volts peak to peak before connecting to the analog input channels. The implementation and use of attenuator networks is described in Section II – Measuring High Level Voltages. Both sides of the high voltage frequency signal must be attenuated for differential inputs.

The frequency of a voltage signals up to ±100 Volts can to be measured directly by the *dataTaker 505, 605, 515* and *615*, if the internal attenuators are selected.

Frequency signals are connected to the analog input channels as differential inputs as follows

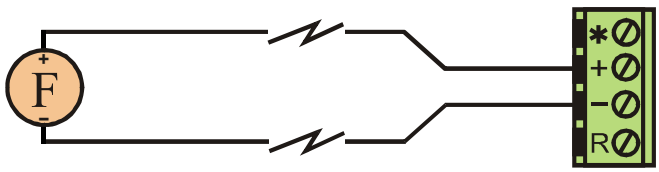


Figure 61 – Connecting Frequency as Differential Inputs

The frequency is measured between the +ve and –ve terminals of the analog input channel.

Frequency inputs connected as differential inputs are read and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1F   5F(1000)   10P(A)
END
```

instructs the *dataTaker* to measure the frequency of signals as follows

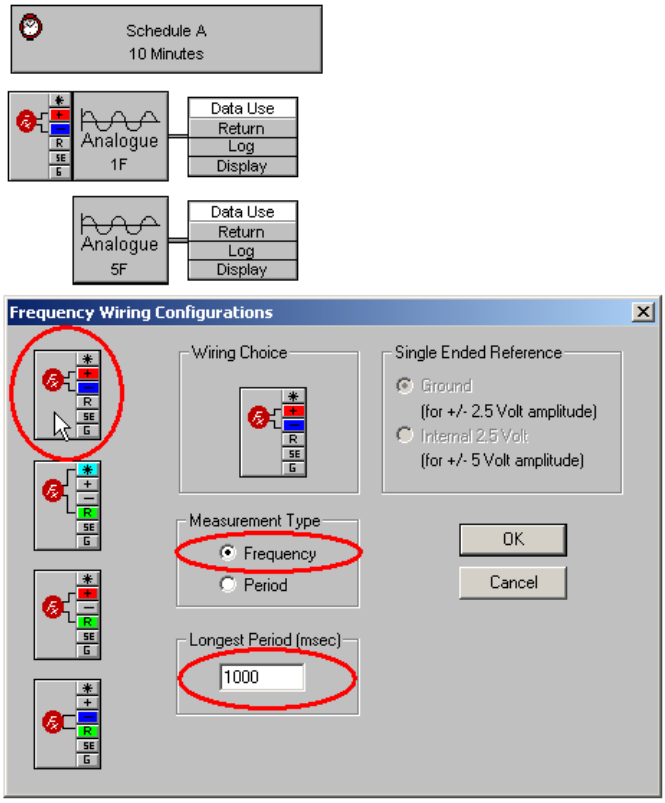
measure the frequency connected to analog input channel 1, which by default has a minimum capability of 33Hz

measure the frequency connected to analog input channel 5, for which minimum capability has been set to 1Hz by setting maximum period to 1000mSec

measure the period of the frequency connected to channel 10, , which by default has a minimum capability of 33Hz. The signal as a higher voltage, and is to be attenuated by selecting the internal attenuator channel option A (*dataTaker 505, 515, 605* and *615* only)

The F indicates that the signal connected to these channels is to be measured as a frequency, and data is returned in units of Hz. The P indicates that the signal is to be measured as the period of the frequency, and data is returned in units of μSec.

Using DeLogger, frequency and period signals that are connected as differential inputs are measured by the following Program Builder program. The differential connection is selected from the Frequency Wiring Configurations dialog which opens when you have selected the analog input channel.



If you are measuring the frequency of the signal, then set Measurement Type to Frequency. If you are measuring the period of the signal, then set Measurement Type to Period.

The Longest Period has also been changed to 1000mSec from the default 30mSec, to allow frequency measurement for this channel down to 1Hz.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Measuring Frequency as Single Ended Inputs

Frequency signals can also be measured as single ended inputs, referenced either to Analog Return or an external common point connected to the SE REF input terminal.

When the single ended frequency signal is connected between the +ve, –ve or ✳ input terminal of the analog input channel and Analog Return, the comparator threshold voltage is the voltage level on the Analog Return terminal of the channel.

When the single ended frequency signal is connected between the +ve, –ve or ✳ input terminal of the analog input channel and the SE REF input, the comparator threshold voltage is the voltage level on the SE REF terminal.

Alternatively for each of the single ended connection methods, frequency inputs can be measured using the internal 2.5 Volt threshold voltage. In this case the voltage level applied to the Analog Return or the SE REF input terminal is not used. This option provides compatibility with 5 Volt TTL and CMOS level inputs.

If the frequency of a voltage signal greater than 3 Volts is to be measured using either a *dataTaker 50, 500* or *600* data logger, then the signal must be externally attenuated to the common mode voltage of ±3.5 Volts peak to peak before connecting to the analog input channels. The implementation and use of attenuator networks is described in Section II – Measuring High Level Voltages.

The frequency of a voltage signals up to ±100 Volts can to be measured directly by the relay multiplexed *dataTaker*s, if the internal attenuators are selected.

Single Ended Frequency Inputs Referenced to Analog Return

Frequency signals are connected to the analog input channels as single ended inputs referenced to Analog Return as follows

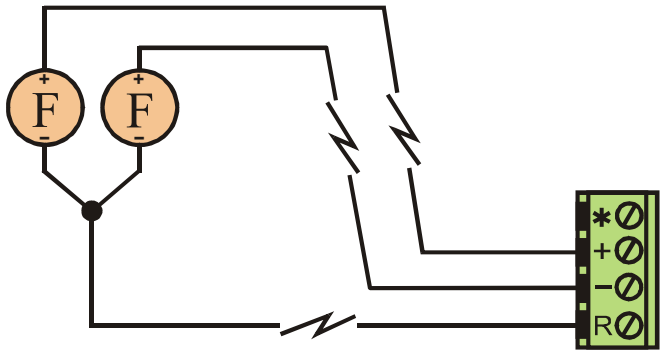


Figure 62 – Connecting Frequency as Single Ended Inputs Referenced to Analog Return

The frequency on the +ve, –ve or ✳ terminal is measured using the voltage on the Analog Return terminal as the comparator threshold voltage.

The *dataTaker 50* does not support single ended frequency measurement on the Excite (✳) terminal.

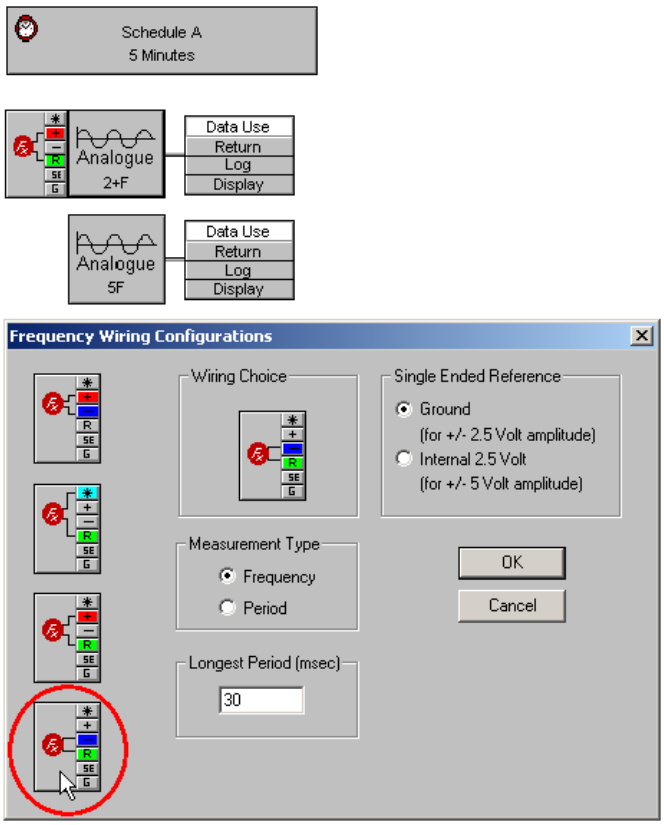
Frequency signals connected to the analog input channels as single ended inputs referenced to Analog Return are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  2+F  5-F
END
```

instructs the *dataTaker* to measure the single ended frequencies connected to the channels 2+ and 5–, and the period connected to channel 10 ✳. The comparator threshold is the voltage level connected to the Analog Return terminal.

The F indicates that the signal connected to these channels is to be measured as a frequency, and data is returned in units of Hz. The P indicates that the signal is to be measured as the period of the frequency, and data is returned in units of μSec.



The *dataTaker* will read the inputs every 5 minutes, and readings are stopped by entering a H (Halt) command.

Single Ended Threshold Voltage

The measurements in the above example would be taken using the voltage on the Analog Return terminal as the comparator threshold voltage.

The single ended frequency signals can also be measured using an internal 2.5 Volt threshold voltage. The amplitude of the single ended frequency signal referenced to Analog Return is compared with this internal comparator threshold voltage to measure frequency. The internal comparator threshold voltage is 2.5 Volt above *dataTaker* ground, and so this single ended frequency measurement of signals 5 Volt peak to peak, which provides compatibility with TTL and CMOS logic.

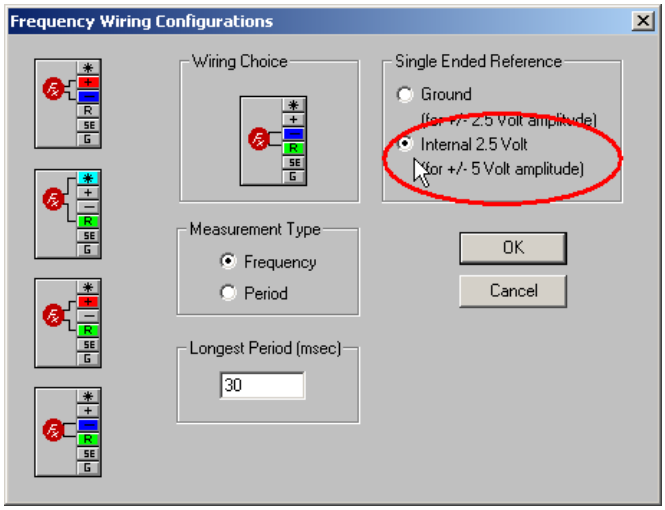
The measurement of single ended frequency signals against the internal 2.5 Volt comparator threshold voltage is specified by the 2V channel option.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  3+F(2V)   5-F(2V)
END
```

instructs the *dataTaker* to measure the frequency on single ended input channels 3+ and 5–. The amplitude of the frequency signals is sensed between the input terminals and Analog Return, and compared with the internal 2.5 Volt comparator threshold voltage. Frequency is counted when the signal amplitude exceeds this 2.5 Volt threshold.

Using DeLogger, the internal 2.5 Volt threshold is specified by setting the Single Ended Reference option to Internal 2.5 Volt.



Single Ended Frequency Inputs Referenced to an External Common

Frequency signals are connected to the analog input channels as single ended inputs referenced to an external common as follows

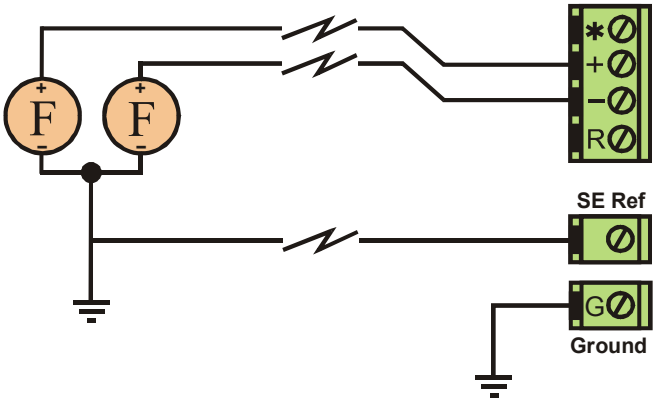


Figure 63 – Connecting Frequency as Single Ended Inputs Referenced to an External Common

The frequency on the +ve, –ve or ✳ terminal is measured using the voltage connected to the external SE REF input terminal as the comparator threshold voltage.

The *dataTaker 50* does not support single ended frequency measurement for the Excite (✳) terminal.

Frequency inputs connected to the analog input channels as single ended inputs referenced to an external common are read and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA15M
  1+F(X)   3-F(X)
END
```

instructs the *dataTaker* to measure the single ended frequencies connected to the channels 1+ and 3–.

The F indicates that the signal connected to these channels is to be measured as a frequency, and data is returned in units of Hz. The P indicates that the signal is to be measured as the period of the frequency, and data is returned in units of µSec.

The X channel option specifies that the single ended frequency inputs are referenced to the SE REF terminal. The comparator threshold is the voltage level connected to the SE REF input terminal.

This command will read the inputs every 15 minutes, and readings are stopped by entering a H (Halt) command.

DeLogger does not support frequency measurement as single ended inputs referenced to an external common. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

Measurement Ranges and Accuracy

The *dataTaker* can measure frequency in the range of 0.102 Hz to 20,000 Hz, and can measure period of the frequency in the range 50 µSec to 9.8 Sec.

The accuracy of frequency and period measurement is 0.0025% of reading.

Error Messages

Frequencies outside the range of 0.102 Hz to 20,000 Hz produce an over-range reading of 0 Hz.

The *dataTaker* also reports the error condition with the error message ‘E11 - Input(s) out of range’ if the Messages Switch /M is enabled.

Measuring Analog Logic State

The *dataTaker* supports an Analog State input type, which allows the analog input channels to be used to monitor logic states and contact closures.

When Analog State inputs are sampled, the voltage on the input channel is measured and compared in software with a user definable threshold voltage.

If the measured voltage is greater than or equal to the threshold, then the *dataTaker* returns a 1 State. If the measured voltage is less than the threshold, then the *dataTaker* returns a 0 State.

The Analog State inputs cannot be used as counters.

The input voltage for Analog State must be within the voltage measurement range, and common mode voltage range, of the *dataTaker*.

For *dataTaker 50, 500* and *600* which have the solid state multiplexer, the voltage measurement range is –2.5 Volts to +2.5 Volts, and the common mode voltage range is –3.5 Volts to +3.5 Volts.

For *dataTaker 505, 605, 515* and *615* which have the relay multiplexer, the voltage measurement range is –100 Volts to +100 Volts, and the common mode voltage range is -100 Volts to +100 Volts.

The *dataTaker 50* has 5 differential analog input channels, which can also be used as 10 single ended input channels. The *dataTaker 500/600* series loggers and Channel Expansion Module (CEM-AD) have 10 differential analog input channels, which may also be used as 30 single ended input channels.

Any combination of differential and single ended analog input modes can be used for Analog State measurements.

The Analog State inputs together with the true digital channels provide a total of 15 digital channels for the *dataTaker 50*, 34 digital channels for the *dataTaker 500/600* series loggers and 50 digital channels for each Channel Expansion Module (CEM-AD).

Analog State Thresholds

The threshold voltage comparison for the Analog State inputs is performed in software by the *dataTaker*.

The threshold voltage is specified as a channel option for the Analog State input channels included in schedule lists.

The threshold voltage must be in the range of –2500 mV to +2500 mV for *dataTaker 50, 500 and 600*, and is declared in units of millivolts.

The threshold voltage must be in the range of –100 V to +100 V for *dataTaker 505, 515, 605 and 615*, and is declared in units of volts.

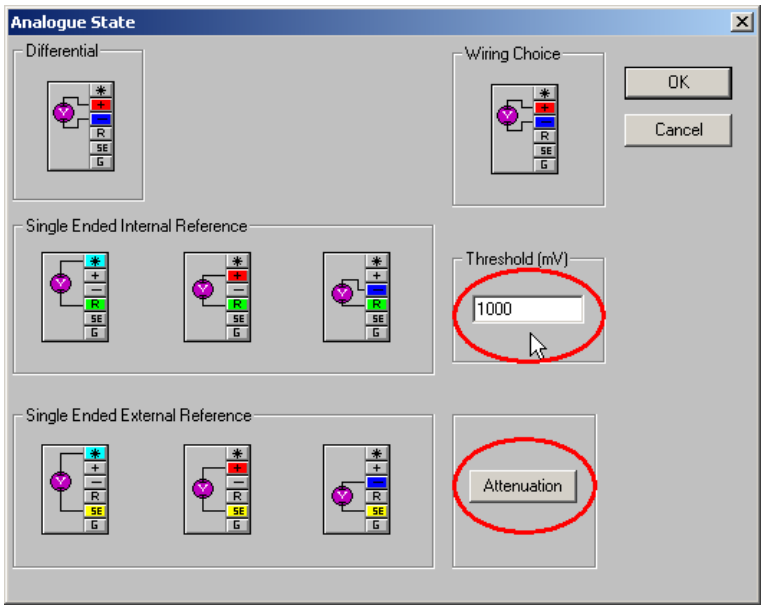
Using DeTransfer, the command for example

```
3AS(1000)  6AS(24.0,A)
```

specifies that the Analog States are to be measured as follows

- the Analog State voltage measured for analog input channel 3 is compared with a threshold voltage of 1000 mV
- the Analog State voltages measured for analog input channels 6 is compared with a threshold voltage of 24.0 V. The channel option A selects in the internal attenuator for the higher voltage range (*dataTaker 505, 605, 515* and *615* only)

Using DeLogger, the Analog State threshold is entered into the Analog State Configuration dialog, and Attenuation for *dataTaker 505, 605, 515* and *615* loggers is set by clicking the Attenuation button.



Each differential or single ended Analog State input signal can be compared to a different threshold if required. The threshold is specified on a channel by channel basis.

If a threshold voltage channel option is not specified, then the threshold voltage defaults to 2500 mV, which is compatible with 5 Volt TTL and CMOS level inputs.

If a threshold voltage of 0 mV is specified, then the Analog State is determined by comparison of the active side of the input signal with its reference as follows

for differential inputs the reference is the voltage applied to the –ve terminal, and the Analog State is determined by the voltage comparison between the +ve and –ve terminals.

for single ended inputs referenced to Analog Return then the reference is the voltage applied to the Analog Return terminal, and the Analog State is determined by a voltage comparison between the input terminal and Analog Return.

for single ended inputs referenced to an external common the reference is the voltage applied to the SE REF terminal, and the Analog State is determined by a voltage comparison between the input terminal and SE REF.

Data is returned for Analog State inputs as digital logic state data as follows

if the voltage applied to the input channel is greater than the specified threshold voltage, then a logic 1 is returned

if the voltage applied to the input channel is less than the specified threshold voltage, then a logic 0 is returned

This logic applies to Analog State voltages which are measured both as differential or single ended inputs.

Measuring Analog State as Differential Inputs

When Analog States are measured as differential inputs, the voltage measured between the +ve and –ve terminals of the analog input channel is compared with the specified threshold voltage.

If the Analog State of high voltage signals is to be measured, then the signal must first be attenuated.

For the *dataTaker 50, 500* and *600*, attenuation is provided by installing an external attenuator on the input channel. The attenuated input must reduce the high voltage signal to within the range of ±3.5 Volts before connecting to the analog input channels.

High voltage Analog State signals are attenuated using voltage attenuator networks which are described in Section II – Measuring High Level Voltages. Both sides of the high voltage signal must be attenuated.

For the *dataTaker 505, 605, 515* and *615*, attenuation is achieved simply by selecting the internal attenuator resistors. When the internal attenuation resistors are selected, the input voltage range can be ±100 Volts.

Measuring Analog States as Differential Inputs

Analog State voltages are connected to the analog input channels as differential inputs as follows

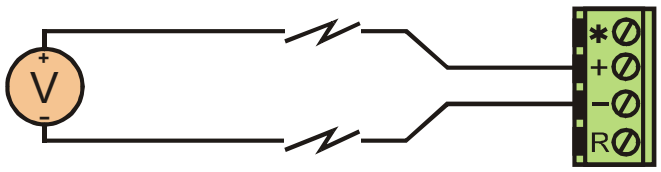


Figure 64 – Connecting Analog State Voltages as Differential Inputs

The differential voltage measured between the +ve and –ve terminals is compared with the specified threshold voltage to determine the analog logic state.

Analog State voltages connected to the analog input channels as differential inputs are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA20M
  1AS  3AS(1500)
END
```

instructs the *dataTaker* to measure Analog States as follows

- the voltage measured between the +ve and –ve terminals of analog channel 1 is compared with the default threshold of 2500 mV
- the voltage measured between the +ve and –ve terminals of analog channel 3 is compared with a threshold of 1500 mV

The AS indicates that the voltages connected to these channels are to be measured for Analog State.

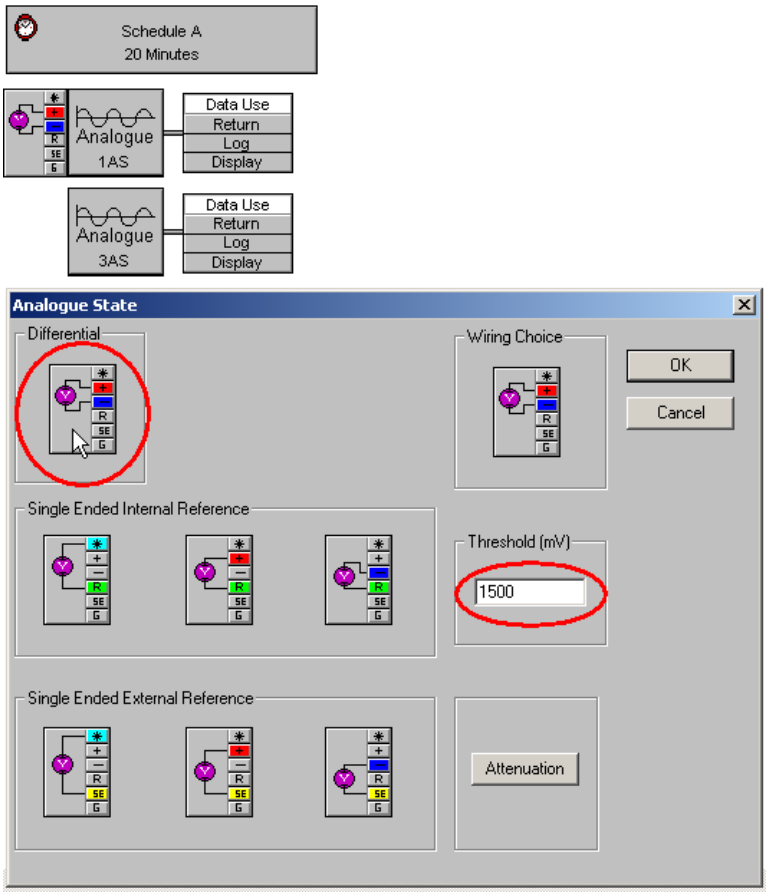
If the threshold voltage is specified as 0 mV, then the Analog State is measured by comparison of the voltage applied to the +ve terminal with the voltage applied to the –ve terminal.

Using DeLogger, differential Analog State signals can be measured by the following Program Builder program.

The differential connections are selected from the Analog State Wiring Configurations dialog which opens when you have selected the analog input channel.

The threshold voltage is entered into the Threshold (mV) field, and applies to this channel only.

If attenuation of the Analog State signal is required, then click the Attenuation button, and enter the appropriate details.



Analog State data is returned in units of State. The *dataTaker* will read the inputs every 20 minutes, and readings are stopped by entering a H (Halt) command.

Measuring Analog State as Single Ended Inputs

Analog State voltages can also be measured as single ended inputs referenced to either Analog Return, or to an external common.

When Analog States are measured as single ended inputs, the voltages measured between the +ve, –ve or ⁇ terminals and the appropriate reference are compared with the specified threshold voltage.

If the Analog State of high voltage signals are to be measured, then the signal must first be attenuated.

For the *dataTaker 50, 500* and *600*, attenuation is provided by installing an external attenuator on the input channel. The attenuated input must reduce the high voltage signal to within the range of ±3.5 Volts before connecting to the analog input channels.

High voltage Analog State signals are attenuated using voltage attenuator networks which are described in Section II – Measuring High Level Voltages. Both sides of the high voltage signal must be attenuated.

For the *dataTaker 505, 605, 515* and *615*, attenuation is achieved simply by selecting the internal attenuator resistors. When the internal attenuation resistors are selected, the input voltage range can be ±100 Volts.

Single Ended Inputs Referenced to Analog Return

Analog State voltages are connected to the analog input channels as single ended inputs referenced to Analog Return as follows

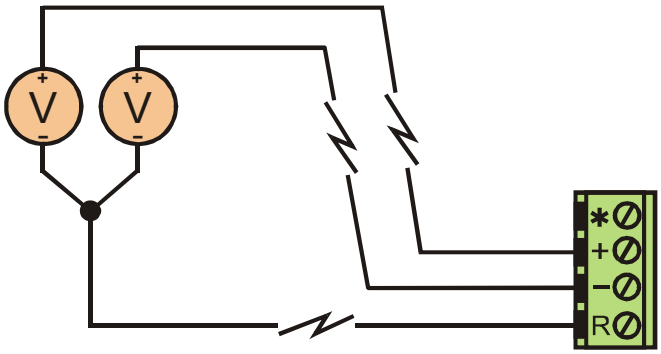


Figure 65 – Connecting Analog State Voltages as Single Ended Inputs Referenced to Analog Return

The voltage measured between the +ve, –ve or ✳ terminals and the Analog Return terminal is compared with the specified threshold voltage to determine the analog logic state.

The *dataTaker 50* does not support single ended Analog State measurement for the Excite (✳) terminal.

Analog State voltages connected to the analog input channels as single ended inputs referenced to Analog Return are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA20S
  1+..2-AS(1000)  8-AS
END
```

instructs the *dataTaker* to measure Analog States as follows

- the voltages measured between single ended analog channels 1+ through 2– and Analog Return, are compared with a threshold of 1000 mV
- the voltage measured between the single ended channel 8– and Analog Return is compared with the default threshold of 2500 mV

The AS indicates that the voltages connected to these channels are to be compared for Analog State.

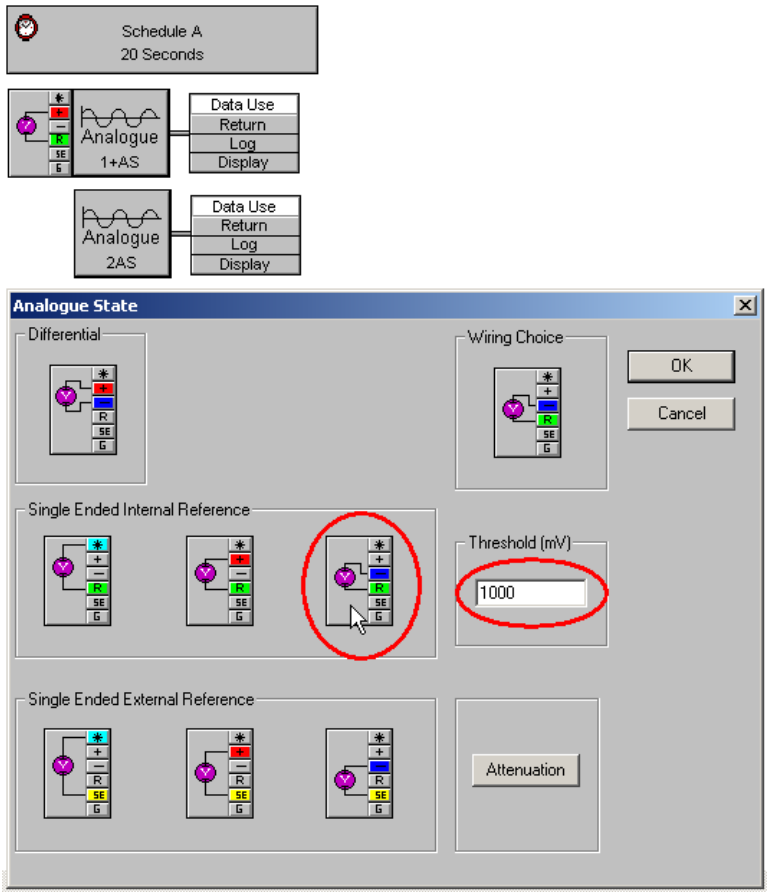
If the threshold voltage is specified as 0 mV, then the Analog State is measured by comparison of the voltage applied to the +ve, –ve or ✳ terminal with the voltage applied to the Analog Return terminal.

Using DeLogger, single ended Analog State signals referenced to Analog Return can be measured by the following Program Builder program.

The single ended connections referenced to Analog Return are selected from the Analog State Wiring Configurations dialog which opens when you have selected the analog input channel.

The threshold voltage is entered into the Threshold (mV) field, and applies to this channel only.

If attenuation of the Analog State signal is required, then click the Attenuation button, and enter the appropriate details.



Analog State data is returned in units of State. The *dataTaker* will read the inputs every 20 seconds, and readings are stopped by entering a H (Halt) command.

Single Ended Inputs Referenced to an External Common

Analog State voltages are connected to the analog input channels as single ended inputs referenced to an external common as follows

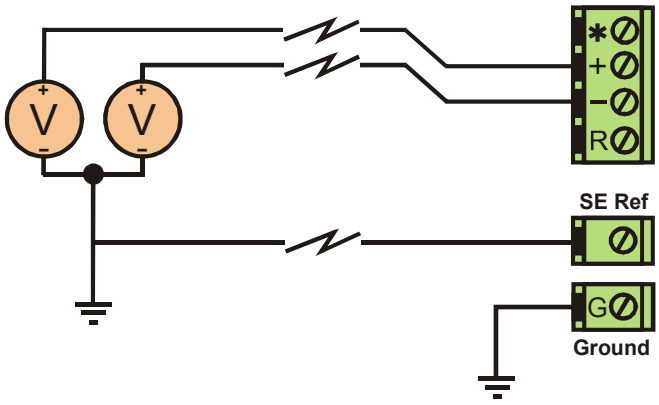


Figure 66 – Connecting Analog State Voltages as Single Ended Inputs Referenced to an External Common

The voltage measured between the +ve, –ve or ✳ terminals and the SE REF terminal is compared with the specified threshold voltage to determine the analog logic state.

The *dataTaker 50* does not support single ended Analog State measurement for the Excite (✳) terminal.

Analog States connected to analog input channels as single ended inputs referenced to an external common are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 R15S
  1+AS(1200,X)   5-AS(20,X,A)
END
```

instructs the *dataTaker* to measure Analog States as follows

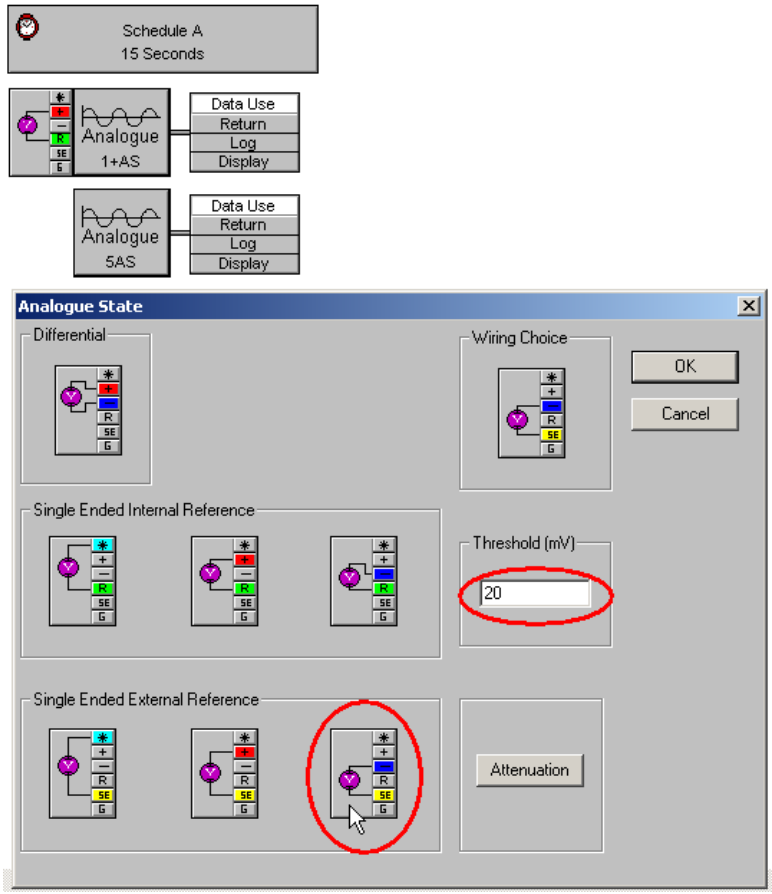
the voltage measured between single ended analog channel 1+ and SE REF is compared with a threshold of 1200 mV

the voltage measured between single ended analog channel 5– and SE REF is compared with a threshold of –20 Volts. The internal attenuator is selected by the channel option A (*dataTaker 505, 605, 515* and *615* only)

The AS indicates that voltages connected to these channels are to be read for Analog State. The X channel option specifies that the Analog State voltage is referenced to the SE REF terminal.

If the threshold is specified as 0 mV, then Analog State is measured by comparing the voltage on the +ve, –ve or ✳ terminal with the voltage on the SE REF terminal.

Using DeLogger, single ended Analog State signals referenced to Analog Return can be measured by the following Program Builder program.



The single ended connections referenced to Analog Return are selected from the Analog State Wiring Configuration dialog which opens when you have selected the analog input channel.

The threshold voltage is entered into the Threshold (mV) field, and applies to this channel only. If attenuation of the Analog State signal is required, then click the Attenuation button, and enter the appropriate details.

Analog State data is returned in units of State. The *dataTaker* will read the inputs every 15 seconds, and readings are stopped by entering a H (Halt) command.

Monitoring Contact Closures by Analog State

The Analog State inputs can be used to monitor switch or contact closures.

Each analog input channel can support one contact closure as a differential Analog State input, and two (*dataTaker 50*) or three (*dataTaker 500* and *600* series) contact closures as single ended Analog State inputs (see the previous sections).

The contact closures must switch a voltage, which is measured and compared with the threshold voltage to determine the state of the switch. Contact closures can be excited by two methods as follows

- the contact closure can be excited from an external 5 Volt power supply, via pull up resistors of 10 – 20 KOhm
- an excitation current or voltage can be output from the Excite terminal when the contact closure is monitored

The excitation voltage holds the analog input channel at a 'high' voltage while the contact closure is open. However when the contact closure closes, the short circuit pulls the input to a 'low' voltage.

The voltage measured for the input channel is either compared to the internal 2.5 Volt default threshold voltage, or a user specified threshold voltage. The voltage comparison is carried out in software.

Contact closures can be connected to the analog input channels as differential or single ended inputs.

An open switch contact returns 1 State, and a closed switch contact returns 0 State.

External Excitation and Differential Inputs

Contact closures excited by an external voltage are connected to the analog input channels as differential inputs as follows

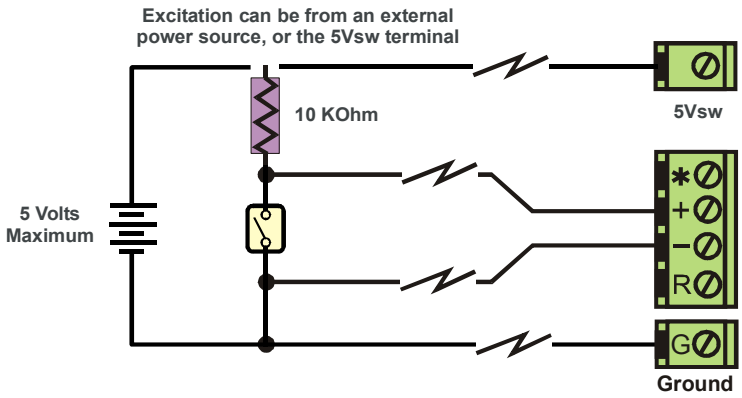


Figure 67 – Connecting Externally Excited Contact Closures as Differential Inputs

Externally excited contact closures connected as differential inputs are read in the same way as described above under Figure 64.

External Excitation and Single Ended Inputs

Contact closures excited by an external voltage are connected to the analog input channels as single ended inputs as follows

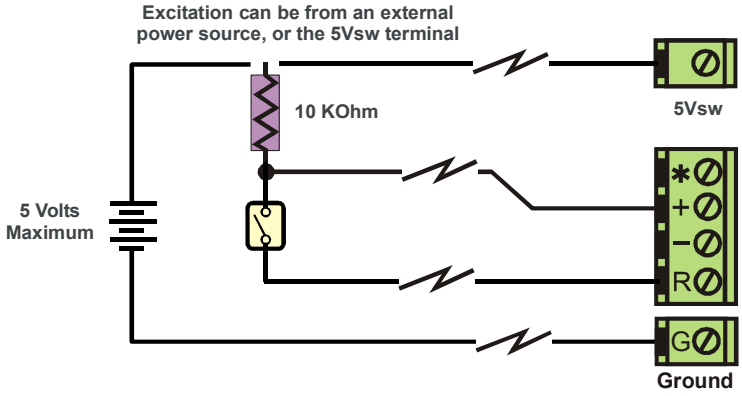


Figure 68 – Connecting Externally Excited Contact Closures as Single Ended Inputs

Externally excited contact closures can be connected as single ended inputs referenced to Analog Return or to an external common connected to SE REF.

Externally excited contact closures connected as single ended inputs are read in the same way as described above under Figure 65 and Figure 66.

Excite Terminal Excitation and Differential Inputs

Contact closures are connected to the analog input channels as differential inputs excited by the Excite terminal as follows

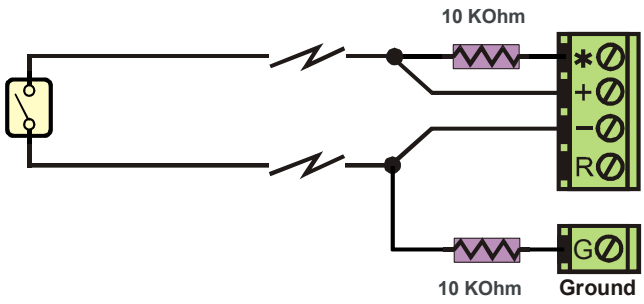


Figure 69 – Connecting Contact Closures Excited from the Excite Terminal as Differential Inputs

Contact closures excited from the Excite terminal and connected as differential inputs are read as described above under Figure 64. However, to turn on the excite terminal at the time of reading the channel, the channel option V must be added. This turns on a 4.5 Volt source from the Excite terminal.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1AS(V)   5AS(2000,V)
END
```

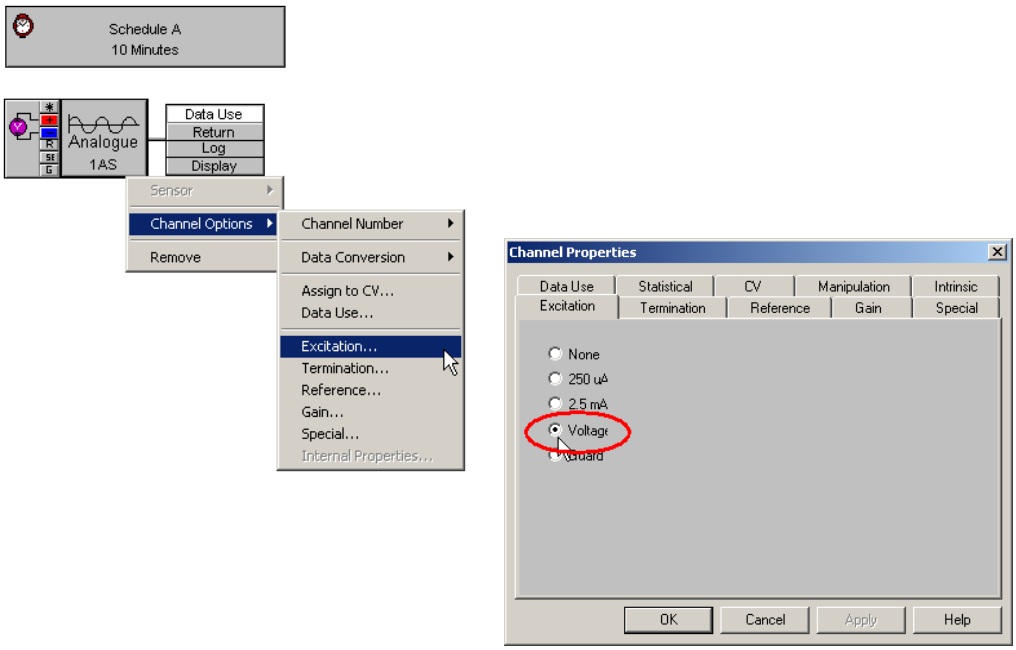
instructs the *dataTaker* to read the contact closures as follows to determine the state of the contacts

the excite terminal is turned on to source 4.5 Volts, and the voltage measured between the +ve and –ve terminals of analog channel 1 is compared with the default threshold of 2500 mV

the excite terminal is turned on to source 4.5 Volts, and the voltage measured between the +ve and –ve terminals of analog channel 5 is compared with a threshold of 2000 mV

The V channel option specifies that the Excite terminal is to output 4.5 Volt when each analog channel is read. Alternatively either the I or II channel option could be used to output a current from the Excite terminal when the contact closure is read.

Using DeLogger, the Excite terminal is programmed to turn on before measurement by selecting Channel Options:Excitation… and clicking Voltage under the Excitation tab.



Excite Terminal Excitation Single Ended Inputs

Contact closures excited by the Excite terminal are connected to the analog input channels as single ended inputs as follows

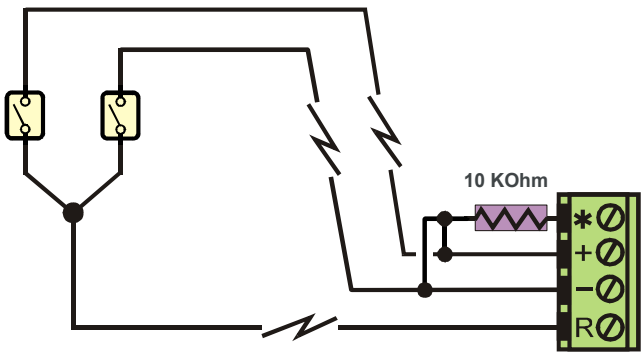


Figure 70 – Connecting Contact Closures Excited from the Excite Terminal as Single Ended Inputs

Contact closures excited from the Excite terminal and connected as differential inputs are read as described above under Figure 65 and Figure 66. However, to turn on the excite terminal at the time of reading the channel, the channel option V must be added. This turns on a 4.5 Volt source from the Excite terminal.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  1+AS(V)   3-AS(1800,V)
END
```

instructs the *dataTaker* to read the contact closures as follows to determine the state of the contacts

- the excite terminal is turned on to source 4.5 Volts, and the voltage measured between the +ve terminal of analog channel 1 and Analog Return is compared with the default threshold of 2500 mV
- the excite terminal is turned on to source 4.5 Volts, and the voltage measured between the –ve terminal of analog channel 3 and Analog Return is compared with a threshold of 1800 mV

The V channel option specifies that the Excite terminal is to output 4.5 Volt when each analog channel is read. Alternatively either the I or II channel option could be used to output a current from the Excite terminal when the contact closure is read.

Using DeLogger, the Excite terminal is programmed to turn on before measurement by selecting Channel Options:Excitation… and clicking Voltage under the Excitation tab as illustrated on the previous page.

Error Messages

There are no specific error messages associated with Analog State inputs. However if excitation is not provided, open circuit contacts will return over-range data of ±99999.9 State.

The *dataTaker* will also report the error condition by the error message ‘E11-input(s) out of range’ if the Messages Switch /M is enabled.

Measuring Temperature

Temperature is the most commonly measured of all physical parameters.

The *dataTaker* data loggers support all of the commonly used temperature sensors including thermocouples, RTDs, solid state or integrated circuit (IC) temperature sensors and thermistors.

Thermocouple types B, C, D, E, G, J, K, N, R, S and T

Platinum Resistance Temperature Detectors with calibration coefficients of
α=0.003850 Ohm/Ohm/Deg C
α=0.003916 Ohm/Ohm/Deg C

Nickel Resistance Temperature Detectors

Copper Resistance Temperature Detectors

AD590, AD592, LM335, LM34 and LM35 solid state or integrated circuit (IC) temperature sensors

Yellow Springs Instrument Inc. 400xx series thermistors

The choice of temperature sensor is determined by a number of factors, including the temperature measurement range, degree of accuracy needed, operating environment, and cost.

For each of these different types of temperature sensors the *dataTaker* performs all of the necessary compensation and linearization calculations.

Units of Temperature

The units of temperature measurement by the *dataTaker* may be degrees Celsius, Fahrenheit, Kelvin or Rankine.

The units of temperature required are selected by the Parameter36 command (See Section III - Parameter Commands).

The permitted settings for Parameter36 and the unit of temperature that each represents are listed below

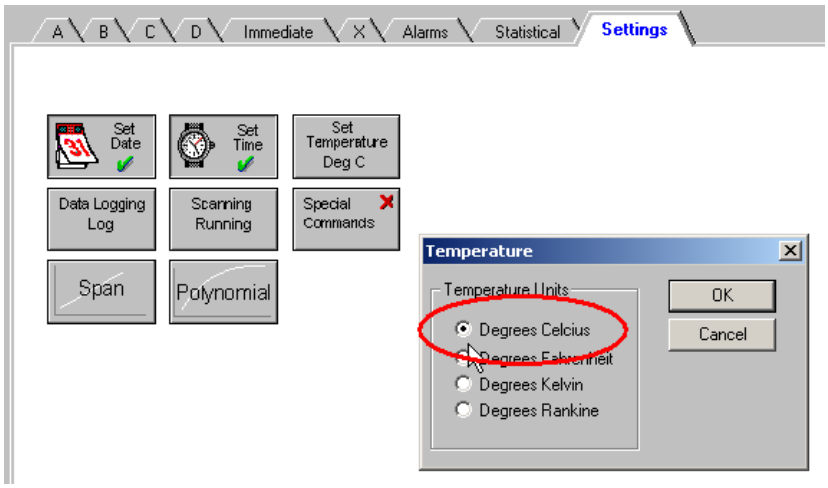
P36=0	Degrees Celsius	(Default)
P36=1	Degrees Fahrenheit	
P36=2	Degrees Kelvin	
P36=3	Degrees Rankine	

Using DeTransfer, the units of temperature are set by the command for example

```
P36=1
```

which in this case defines the units of temperature as Degrees Fahrenheit.

Using DeLogger, the units of temperature can be set in the Program Builder. Click on the Settings tab, click on the Set Temperature button, and select the temperature units you require in the dialog which opens.



This selection is not transferred to the *dataTaker* until the next time that a program is sent to the logger. If you need to change the temperature units in the logger without sending a new program (therefore replacing the program already in the logger), then the Parameter36 commands can be sent directly from the Text View.

Measuring Temperature with Thermocouples

Thermocouples are the most widely used temperature sensors, because of their simplicity, ruggedness, low cost, wide range and interchangability.

The *dataTaker* supports thermocouple types B, C, D, E, G, J, K, N, R, S and T, and provides reference junction compensation and linearization over the temperature range for each of the thermocouple types. Any mix of thermocouples can be used in the same project.

The thermocouple linearizations support the ITS-90 thermocouple standard.

Temperature data from thermocouple inputs is returned in the selected temperature units, with a resolution of 0.1 Deg C and an accuracy of better than 0.5 Deg C.

The accuracy of temperature measurement with thermocouples is a function of the accuracy of the thermocouple, and not of the *dataTaker*.

Thermocouples are available in a wide range of grades, varying from ±0.1 Deg C accuracy to ±5 Deg C accuracy. Thermocouple wire type should be chosen carefully, rather than simply using the cheapest thermocouple wire available which is often only extension grade wire of ±2 Deg C accuracy.

The decreasing price of RTDs is challenging thermocouples as an economical and accurate temperature measuring device. If temperature measurement is performed in the biological or environmental range (-10 to +100 Deg C), then RTDs should preferably be used for greatest accuracy.

Thermocouples are still the cheapest method for measuring high temperatures, and accuracy of even ±5 Deg C is often not a problem when measuring temperatures greater than several hundred Deg C.

Basic Concepts of Thermocouples

A practical thermocouple consists of two wires of dissimilar metals that are electrically joined at one end (the measurement junction), and thermally joined at the other end (the reference junction).

A low DC voltage is produced by the thermocouple when the two junctions are at different temperatures. This voltage is referred to as the thermoelectric voltage.

This thermoelectric voltage is produced by the temperature gradient along the thermocouple wires. The thermoelectric voltage is not produced by the junctions themselves.

It is important that the purity of the two thermocouple wires is rigidly maintained, particularly at points where there are significant temperature gradients.

A typical thermocouple circuit, in this case a type J thermocouple, is illustrated below

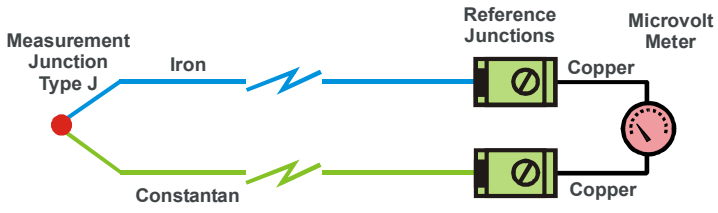


Figure 71 – A Typical Thermocouple Type J Circuit

Thermocouple Alloys and Combinations

The alloy combination, polarity and measurement range for the thermocouple types supported by the *dataTaker* is listed in the table below

Type	Alloy of +ve wire	Alloy of -ve wire	Temperature Range
B	Platinum 30% Rhodium (70% Pt - 30% Rh)	Platinum 6% Rhodium (94% Pt - 6% Rh)	0 to 1700 Deg C 32 to 3100 Deg F
C	Tungsten 5% Rhenium (74% W - 26% Re)	Tungsten 26% Rhenium (95% W - 5% Re)	0 to 2320 Deg C 32 to 4200 Deg F
D	Tungsten 3% Rhenium (97% W - 3% Re)	Tungsten 25% Rhenium (75% W - 25% Re)	0 to 2320 Deg C 32 to 4200 Deg F
E	Chromel (55% Cu - 45% Ni)	Constantan (90% Ni - 10% Cr)	-200 to 900 Deg C -330 to 2280 Deg F
G	Tungsten (100% W)	Tungsten 26% Rhenium (74% W - 26% Re)	0 to 2320 Deg C 32 to 4200 Deg F
J	Iron (100% Fe)	Constantan (55% Cu - 45% Ni)	0 to 750 Deg C 32 to 1380 Deg F
K	Chromel (90% Ni - 10% Cr)	Alumel (96% Ni - 2% Mn - 2% Al)	500 to 1250 Deg C -330 to 2280 Deg F
N	Nicrosil (Ni-Cr-Si)	Nisil (Ni-Si-Mg)	-270 to 1350 Deg C -450 to 2370 Deg F
R	Platinum 13% Rhodium (87% Pt - 13% Rh)	Platinum (100% Pt)	0 to 1450 Deg C 32 to 2640 Deg F
S	Platinum 10% Rhodium (90% Pt - 10% Rh)	Platinum (100% Pt)	0 to 1450 Deg C 32 to 2640 Deg F
T	Copper (100% Cu)	Constantan (55% Cu - 45% Ni)	-500 to 350 Deg C -330 to 660 Deg F

Thermocouple Wire Colour

There are a number of standards for colour coding thermocouple wires. Each standard specifies a colour for each wire of each thermocouple type, and a colour for the outer wrap for each thermocouple type. Consult the standards appropriate to your source of thermocouple wire to identify the positive and negative poles of the wire.

Thermocouple Reference Junction

In practice it is often difficult to identify a single point in the thermocouple circuit as the reference junction. The reference junction is often two junctions as illustrated in Figure 72.

In Figure 72 there are three metals in the thermocouple circuit — iron, constantan and copper. However for practical purposes the Iron–Copper and Constantan–Copper junctions can in fact be considered collectively as the reference junction, provided these are held at the same temperature.

Thermocouple Reference Junction Temperature

The reference junctions of thermocouples are traditionally maintained at 0°C. This is assumed in thermocouple calibration tables.

While this requirement can be provided by maintaining the reference junctions in melting ice baths or electronically cooled enclosures, this is impractical for portable or remote applications.

An alternative and more practical approach is to maintain the thermocouple reference junctions at ambient temperature, and to allow to drift with ambient temperature.

In this approach, the thermocouple reference junctions are all maintained at an equal temperature by placing them in close thermal proximity to a heat conductor, and ideally enclosed in thermal insulation. The heat conductor is generally referred to as an *isothermal block*, and is usually a block of aluminium, copper or similar material with a high thermal conductivity.

This approach requires that the temperature of the thermocouple reference junctions (actually the difference between the temperature of the reference junctions and 0°C) is known. This temperature is then used to correct the temperature measured by the thermocouple.

This correction overcomes the errors produced by a non-zero thermocouple reference junction temperature, referred to as *reference junction temperature compensation*.

The *dataTaker* has a metal chassis around the analog input channel terminals, and optionally a substantial steel case. These contribute towards creating an isothermal environment around the junctions of directly connected thermocouples. However if accurate and stable thermocouple measurements are required, then an external isothermal block should be used.

Thermocouple Zero Voltage

Another potential source of error in measurement of thermocouple inputs are the various thermoelectric voltages produced by the effects of temperature gradients on the mixture of metals in screw terminals, connectors, cables, printed circuit board tracks, etc. in the thermocouple circuit.

Thermoelectric voltages can be produced between the thermocouple reference junctions and the inner circuits of the measuring instrument.

These thermoelectric voltages electrically add to the signal voltage of the thermocouple, producing errors in the measured temperature. Correct thermocouple measurement requires that these voltages be measured and then used to correct the signal voltage read from the thermocouple measurement junction. This correction is referred to as *zero voltage compensation.*

The thermocouple zero voltage is usually measured from a thermocouple mounted in thermal contact with the thermocouple reference junctions.

Voltage to Temperature Conversion

The relationship between temperature and measured voltage for a thermocouple is not linear. The temperature-voltage relationship of all thermocouples has been accurately measured and published. These are used to calculate the thermocouple temperature from a measured voltage by a technique called *thermocouple linearization*.

Thermocouple Support by the *dataTaker*

The *dataTaker* supports the thermocouple types B, C, D, E, G, J, K, N, R, S and T. When a thermocouple type is measured by the *dataTaker*, the *dataTaker* automatically performs the reference junction temperature compensation, zero voltage compensation and linearization calculations.

Readings are returned directly in degrees Celsius, Fahrenheit, Kelvin or Rankine with a resolution of 0.1 Deg C and accuracy of better than 0.5 Deg C, depending on the thermocouple.

Reference Junction Temperature Compensation

The thermocouple support by the *dataTaker* provides for the thermocouple reference junction temperature to be measured with any 'absolute' temperature sensor, such as a AD590, AD592, LM335, LM35, LM34, RTD or thermistor temperature sensor.

The reference junction temperature is used to compensate the thermocouple signal voltage measurements.

The reference junction temperature sensor is normally mounted in thermal contact with the thermocouple reference junction terminals, and is connected to any analog input channel.

Reference junction temperature compensation is accurate to within 0.1 Deg C over the temperature range of –20 to +65 Deg C. Errors due to the thermocouple are in addition to this error.

If a reference junction temperature sensor is not implemented, then the *dataTaker* will use its internal temperature as the default reference junction temperature. The internal temperature is measured by an LM35 sensor, located within the body of the *dataTaker*.

The *dataTaker* also supports thermocouple interfaces which have reference junction compensation, and output a voltage referenced to 0 Deg C. In this case reference junction compensation by the *dataTaker* is not required, and only zero voltage compensation and linearization is performed.

Zero Voltage Compensation

The thermocouple support implemented by the *dataTaker* also provides for the thermocouple zero voltage to be measured.

The zero voltage is usually measured with a thermocouple mounted in thermal contact with the thermocouple terminals, and connected to any analog input channel.

If a zero voltage thermocouple is not implemented, then the *dataTaker* uses the analog to digital converter calibration zero voltage channel as the default thermocouple zero voltage.

The zero voltage is used to correct or compensate the thermocouple signal voltage measurements for any stray thermoelectric voltages in the thermocouple circuit, between the reference junctions and the analog to digital converter.

Linearization of Thermocouple Inputs

Linearization of the measured thermocouple voltage is carried out automatically by the *dataTaker*, by applying polynomial functions for the particular thermocouple type.

The coefficients for these polynomials are stored permanently in the *dataTaker* ROM. The thermocouple linearization errors are less than 0.1 Deg C over the temperature measuring range for each thermocouple type.

The thermocouple linearizations support the ITS-90 thermocouple standard.

Calculation of Thermocouple Temperature

The temperatures sensed by thermocouples connected to the analog input channels of the *dataTaker* are calculated by the following procedure

- the measured reference junction temperature is converted to a voltage equivalent for the thermocouple type being used, and subtracted from the measured thermocouple signal
- the measured thermocouple zero voltage is subtracted from this reference junction temperature compensated thermocouple signal
- the compensated thermocouple signal is linearized by applying the appropriate polynomials to calculate the junction temperature of the thermocouple.

Directly Connecting Thermocouples to the *dataTaker*

The simplest method for monitoring the signal from thermocouples with the *dataTaker* is to connect the thermocouple wires directly to the analog input channel screw terminals.

In this configuration, a number of defaults are implemented as follows

the screw terminals of the analog input channels become the isothermal reference junction

the case temperature of the *dataTaker* and the Channel Expansion Module (CEM-AD) is used as the reference junction temperature. The case temperature is measured by an LM35 temperature sensor located amongst the analog input channel terminals. The accuracy of this sensor is approximately 0.5 Deg C, and may be a limitation to thermocouple measurement accuracy for directly connected thermocouples

the thermocouple zero voltage reference is measured by internal analog channel 2%V of the *dataTaker*, which has its input shorted

This method of thermocouple connection to the *dataTaker* can be subject to small errors due to temperature gradients around the analog input board, and between the analog input channel screw terminals.

Temperature gradients can occur within the *dataTaker* or the Channel Expansion Module (CEM-AD) case as a result of exposing the cases to uneven temperatures.

Thermocouple wires can be connected directly to the screw terminals of the *dataTaker* and the Channel Expansion Module (CEM-AD) analog input channels, either as differential or as single ended inputs.

Installing Thermocouples

When installing thermocouples, you must ensure that the thermocouple tips are not electrically connected to the system or structure being temperature monitored. If the thermocouple tip electrically contacts the system or structure, small noise currents can enter the thermocouple and produce significant measurement errors.

If you need to attach the thermocouple tip to the structure or system, then place an electrical insulation material between the thermocouple tip and the structure.

There are various materials available for this purpose, which have good electrical insulation properties, and goo thermal conduction properties. Consult your thermocouple supplier for details and materials.

Measuring Thermocouples as Differential Inputs

Thermocouples can be connected directly to the analog input channels of the *dataTaker* as differential inputs as follows

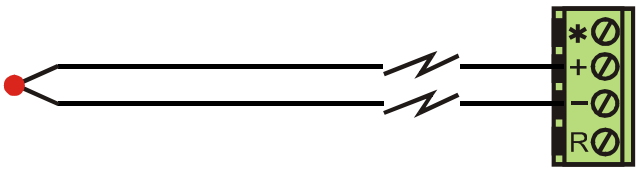


Figure 72 – Connection of Thermocouples as Differential Inputs

The thermocouple measurement junctions must be isolated from ground.

The polarity of the leads for the thermocouple types supported by the *dataTaker* are listed in the Thermocouple Alloys table at the beginning of this chapter.

The differential mode is the preferred method for measuring thermocouples directly connected to the *dataTaker*.

Thermocouples which are directly connected to the analog channels as differential inputs are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  5TT  10TK
END
```

instructs the *dataTaker* to measure thermocouple temperatures as follows

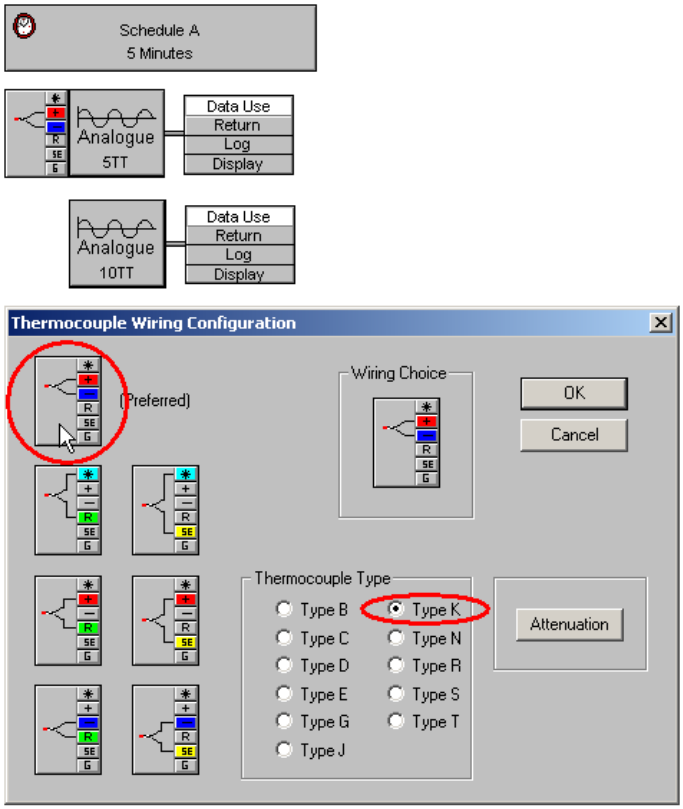
for type T thermocouples connected between +ve and –ve of analog channel 5

for a type K thermocouple connected between +ve and –ve of analog channel 10

the junction temperature and zero voltage for thermocouple compensation is measured from the internal channels

The TT specifies type T thermocouples, and TK specifies type K thermocouples.

Using DeLogger, differential thermocouple signals can be measured by the following Program Builder program. The differential thermocouple connections are selected from the Thermocouple Wiring Configurations dialog which opens when you have selected the analog input channel. Also select the type of thermocouple that is being used.



The temperature data is returned in the units of temperature defined by the Parameter36 command.

The *dataTaker* will read the thermocouples every 5 minutes, and the readings are stopped by entering a H (Halt) command.

Measuring Thermocouples as Single Ended Inputs

Thermocouples can be directly connected to the *dataTaker* analog input channels for measurement as single ended inputs referenced to Analog Return, or as single ended inputs referenced to an external common which is connected to the SE REF input terminal.

Single ended inputs have the advantage of also being able to detect open circuit thermocouples. If a thermocouple becomes open circuit, the data is returned as 99999.9 Deg C and can easily be detected in Alarms.

Single Ended Inputs Referenced to Analog Return

Thermocouples can also be directly connected to the analog input channels as single ended inputs referenced to Analog Return as follows

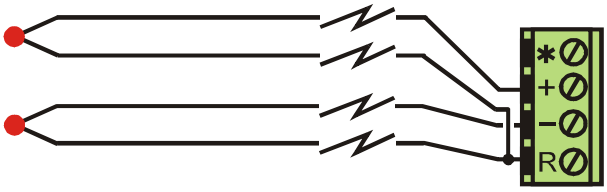


Figure 73 – Connection of Thermocouples as Single Ended Inputs Referenced to Analog Return

The thermocouple measurement junctions must be isolated from ground.

The polarity of the leads for the thermocouple types supported by the *dataTaker* are listed in the Thermocouple Alloys table at the beginning of this chapter.

The *dataTaker 50* does not support single ended thermocouple measurement for the Excite (✳) terminal.

Thermocouples which are directly connected to the analog input channels as single ended inputs referenced to Analog Return are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  1+TK   5✳TK
END
```

instructs the logger to measure thermocouple temperatures as follows

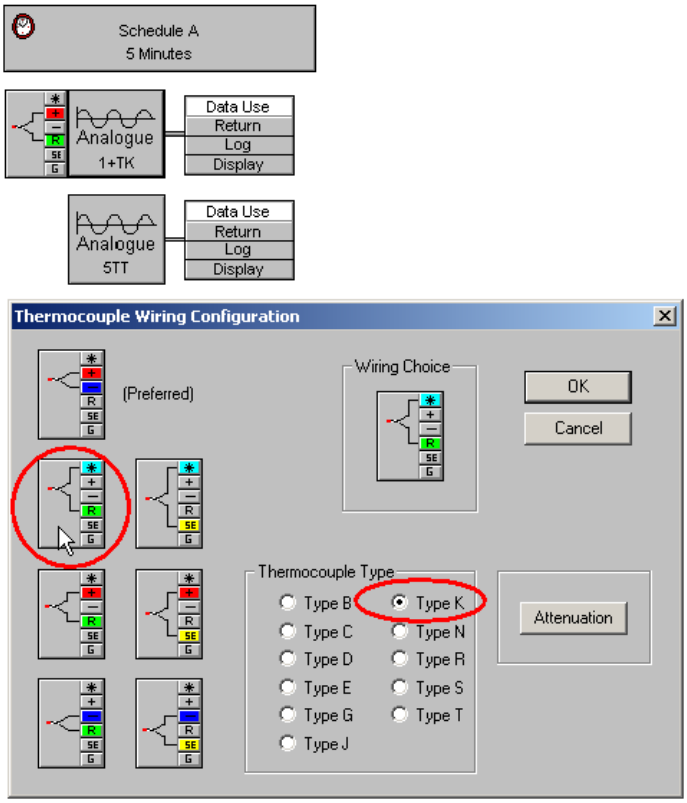
for type K thermocouples connected between channel 1+ and Analog Return

for a type K thermocouple connected between channel 5✳ and Analog Return

the junction temperature and zero voltage for thermocouple compensation is measured from the internal channels

The TK specifies type K thermocouples.

Using DeLogger, single ended thermocouple signals referenced to Analog Return can be measured by the following Program Builder program. The single ended thermocouple connections are selected from the Thermocouple Wiring Configurations dialog which opens when you have selected the analog input channel. Also select the type of thermocouple that is being used. The default is type T.



The temperature data is returned in the units of temperature defined by the Parameter36 command. The *dataTaker* will read the thermocouples every 5 minutes, and the readings are stopped by entering a H (Halt) command.

Single Ended Inputs Referenced to an External Common

Thermocouples can also be directly connected to the analog input channels as single ended inputs referenced to external common as follows

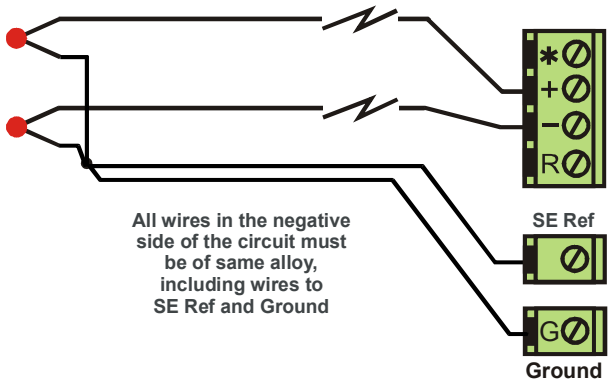


Figure 74 – Connection of Thermocouples as Single Ended Inputs Referenced to an External Common

The thermocouple measurement junctions must be isolated from ground.

This method of connection is only of practical use for external isothermal blocks.

The polarity of the leads for the thermocouple types supported by the *dataTaker* are listed in the Thermocouple Alloys table at the beginning of this chapter.

The *dataTaker 50* does not support single ended thermocouple measurement for the Excite (✳) terminal.

Thermocouples which are directly connected to the analog input channels as single ended inputs referenced to an external common are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA6M
  1+TR(X)   1-TR(X)
END
```

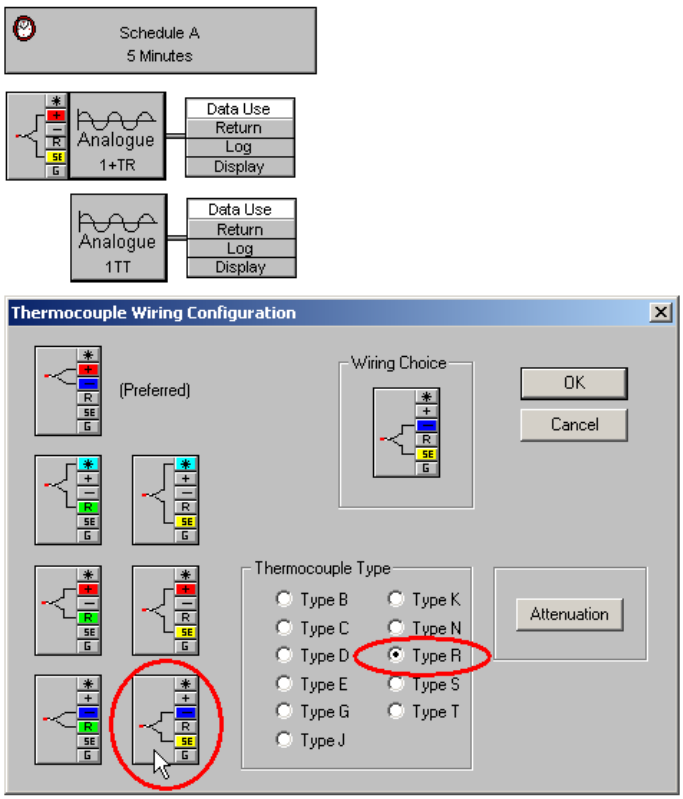
instructs the logger to measure thermocouple temperatures as follows

for type R thermocouples connected between single ended channels 1+ and 1–referenced to SE REF

the junction temperature and zero voltage for thermocouple compensation is measured from the internal channels

The TR specifies type R thermocouples, and TB specifies type B thermocouples. The X channel option indicates that the thermocouple voltage is to be measured with respect to an external common connected to the SE REF input terminal.

Using DeLogger, single ended thermocouple signals referenced to an external common can be measured by the following Program Builder program. The single ended thermocouple connections are selected from the Thermocouple Wiring Configurations dialog which opens when you select the analog input channel.



The temperature data is returned in the units of temperature defined by the Parameter36 command.

The *dataTaker* will read the thermocouples every 6 minutes, and the readings are stopped by entering a H (Halt) command.

Using an External Isothermal Block

An isothermal block is a group of thermocouple terminations which are maintained at the same temperature (isothermal). These terminations are the reference junctions for the connected thermocouples.

The construction of an isothermal block has all of the thermocouple terminations in close thermal contact with a (usually copper) mass of high thermal conductivity.

The temperature of the isothermal block can follow ambient temperature. However the isothermal block should not be exposed to uneven heating (eg. a source of radiant heat) which will heat one side of the block relative to the other.

The temperature of the isothermal block and reference junctions is measured by an RTD or solid state temperature sensor connected to the *dataTaker*. This sensor must be declared to the *dataTaker* with the appropriate commands.

Isothermal blocks can be purpose built for particular thermocouple applications, or can be purchased as an optional accessory from your *dataTaker* supplier.

A correctly constructed and implemented external isothermal block can increase the accuracy of thermocouple measurement to the order of 0.3 Deg C.

Reference Junction Temperature Sensor

The temperature of the isothermal block and thermocouple reference junctions is measured by an 'absolute' type temperature sensor. The *dataTaker* requires that the reference junction temperature is measured with one of AD590, LM335, LM35, LM34, RTD or thermistor temperature sensors. Note that a thermocouple cannot be used as the reference temperature sensor!

The reference junction temperature sensor is specified to the *dataTaker* by a channel option to the particular sensor channel in the general format

```
nsensor(TR)
```

n	is the analog channel for the reference junction temperature sensor
sensor	is the sensor identifier for the sensor type used
(TR)	indicates that the sensor is the reference junction temperature sensor

Temperature sensors that can be used for measuring thermocouple reference junction temperature are listed in the following table

Sensor Type	*Description*
nAD590(TR)	AD590 Kelvin sensor
nLM335(TR)	LM335 Kelvin sensor
nLM34(TR)	LM34 Fahrenheit sensor
nLM35(TR)	LM35 Celsius sensor
nPT385(TR)	Platinum RTD, 0.003850 Ohm/Ohm/Deg C
nPT392(TR)	Platinum RTD, 0.003916 Ohm/Ohm/Deg C
nNI(TR)	Nickel RTD, 0.005001 Ohm/Ohm/Deg C
nCU(TR)	Copper RTD, 0.0039 Ohm/Ohm/Deg C
nYSxx(TR)	YSI thermistor type xx
11SV(TR)	No reference junction temperature compensation

Using DeTransfer, the commands for example

```
10LM35(TR)
```

specifies that the reference junction temperature sensor is an LM35 device, which is differentially connected to analog input channel 10

```
1:5PT392(4W,TR)
```

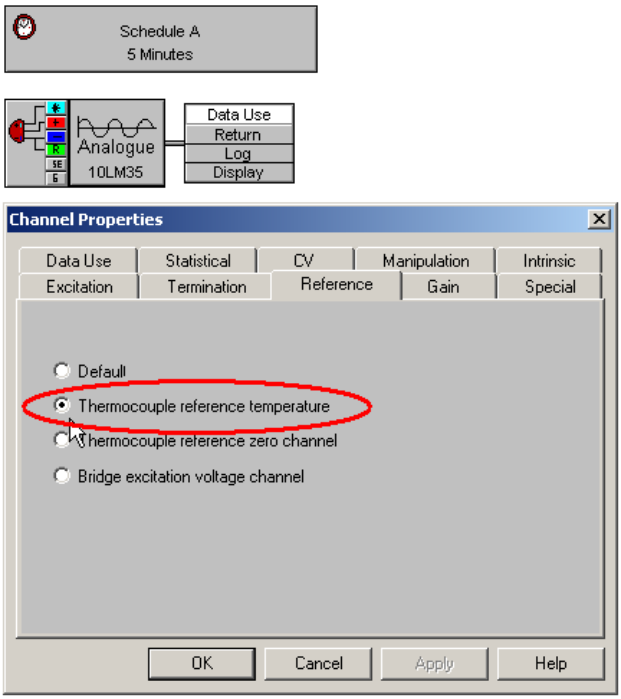
specifies that the reference junction temperature sensor is a platinum RTD, which is measured as a 4 wire input on analog input channel 5 of the Channel Expansion Module 1

```
8YS02(TR,4W)
```

specifies that the reference junction temperature sensor is a YSI thermistor, which is measured as a 4 wire input on analog input channel 8.

Channel options can also be declared in the normal manner to define how the reference junction temperature sensor is to be managed when scanned.

Using DeLogger, firstly create the reference junction temperature sensor channel in the Program Builder, then click Channel Options:Reference… and select Thermocouple Reference Temperature radio button.



The use of each of the temperature sensor types is discussed in detail elsewhere in this manual.

The reference junction temperature sensor channel does not return data to the host when it is read. The data is saved internally for use in subsequent thermocouple compensation.

The reference junction temperature sensor must be declared in the Schedule schedule lists before the thermocouple channels.

If thermocouples from an external isothermal block are included in more than one Schedule, then the reference junction temperature sensor channel must be declared in each Schedule.

Several isothermal blocks can be simultaneously supported by the *dataTaker*. However the reference junction temperature sensor declaration for each isothermal block must precede the thermocouple channels associated with each block.

The *dataTaker* default reference junction temperature sensor is the internal temperature sensor, which is an LM35 device connected to internal channel 1% and is specified as 1%LM35(TR).

The thermocouple reference junction temperature sensor can be connected to any analog input channel as a differential or single ended input.

No Reference Junction Temperature Compensation

A number of thermocouple interfaces are available which internally provide for reference junction compensation and output a corrected thermocouple signal, and so compensation is not required by the *dataTaker*.

The requirement for no reference junction temperature sensor is declared as 11SV(TR), which is a system zero.

If this is not declared, then the *dataTaker* will use the internal temperature as the default reference junction temperature.

Zero Voltage Reference

A thermocouple in close thermal contact with the isothermal block terminations is required for measuring the thermocouple zero voltage reference.

The *dataTaker* uses the zero voltage reference thermocouple to measure and compensate for thermoelectric voltages induced by temperature gradients outside of the thermocouple circuit.

The zero reference voltage thermocouple can be connected as a differential or single ended input to any analog input channel, and is defined by the general format

```
nV(TZ)
```

n is the analog input channel to which the zero voltage reference thermocouple is connected.

V indicates a voltage input

(TZ) channel option which indicates that this sensor is the zero reference voltage thermocouple

Using DeTransfer, the commands for example

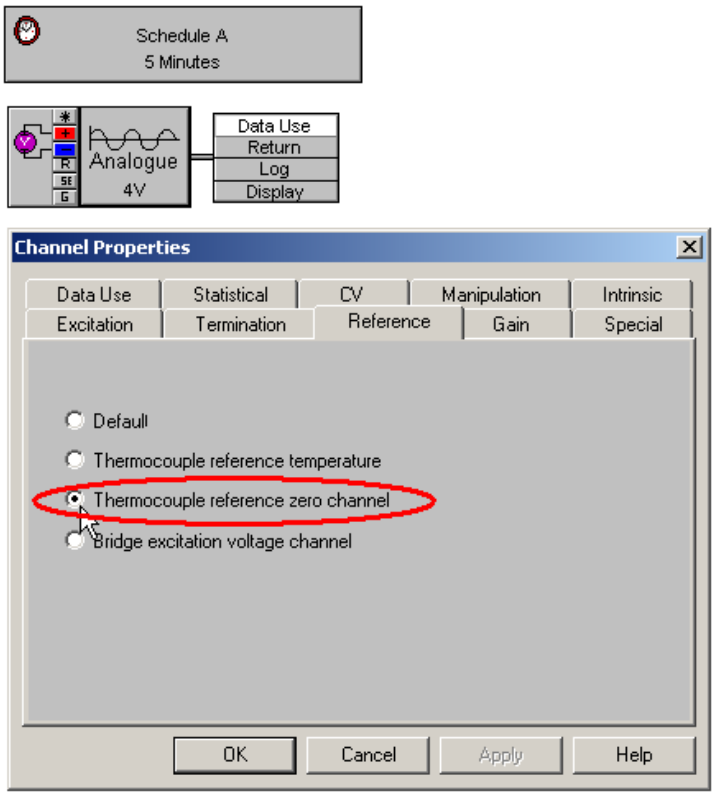
```
4V(TZ)
```

specifies that the zero voltage reference thermocouple is connected as a differential input to analog input channel 4.

```
9+V(TZ)
```

specifies that the zero voltage reference thermocouple is connected as a single ended input to analog input channel 9+

Using DeLogger, firstly create the reference junction temperature sensor channel in the Program Builder, then click Channel Options:Reference… and select Thermocouple Reference Zero radio button.



The zero voltage reference channel does not return data to the host when it is scanned. The data is saved internally for use for subsequent thermocouple compensation.

The zero reference voltage must be declared in the Schedule schedule lists before the thermocouple channels.

Several isothermal blocks can be simultaneously supported by the *dataTaker*. However the zero voltage reference channel for each of the isothermal blocks must precede the thermocouple channels associated with each block.

If thermocouples from an external isothermal block are included in more than one Schedule, the zero reference voltage channel must be declared in each Schedule.

The *dataTaker* default zero voltage reference channel is the internal zero voltage reference, which is connected to internal channel 2% and is specified as 2%V(TZ).

Connecting an Isothermal Block

A schematic for a full isothermal block implementation is illustrated overleaf.

The reference junction temperature sensor is an AD590 powered by the *dataTaker*. Refer to Section II – Measuring Temperature with IC Temperature Sensors for details of the use of this sensor type.

The zero voltage reference is a thermocouple of the same type as the measurement thermocouples, but which is in thermal contact with the thermal mass, and so is at the same temperature as the thermal mass.

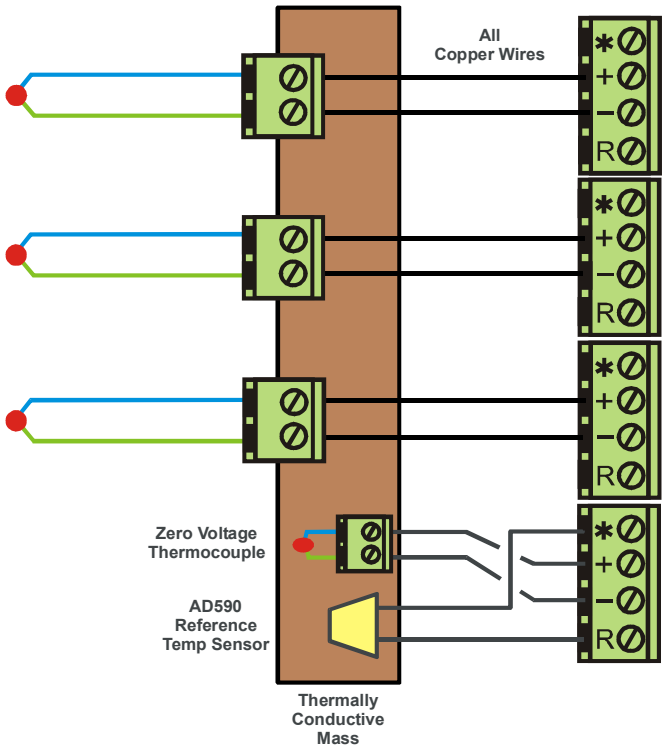


Figure 75 – A Full Isothermal Block Implementation

Reading a Thermocouple Isothermal Block

Using DeTransfer, typical commands for configuring an isothermal block, and reading a group of thermocouples supported by the isothermal block, are as follows

```
P36=0
BEGIN
 RA10M
  9#AD590(TR)   9V(TZ)   1..5TT
END
```

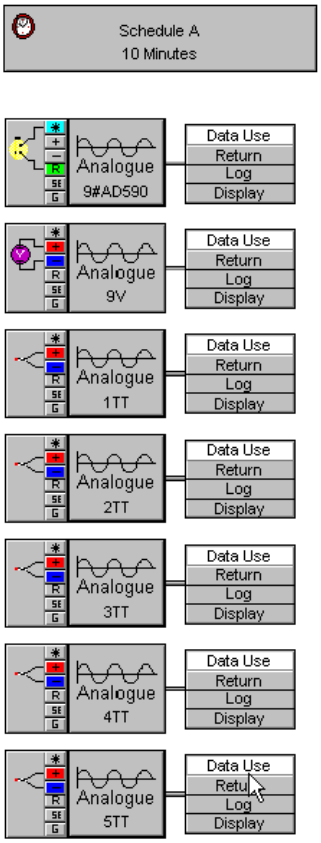
P36=0	temperature date returned in degrees Celsius
9#AD590(TR)	junction temperature sensor is an AD590 powered by the *dataTaker*, measured using the internal shunt of channel 9
9V(TZ)	zero voltage thermocouple is connected differentially to channel 9
1..5TT	five type T thermocouples terminated on the isothermal block, and connected to analog channels 1 through 5

Using DeLogger, the program is created in the Program Builder.

Firstly create the two reference channels, for the thermocouple temperature reference sensor, and the thermocouple zero voltage reference. These two reference channels must be created above the actual thermocouple channels for the isothermal block.

When the reference channels are in place, then add the measurement thermocouple channels.

The finished program should be similar to the following



The reference options have been set for the first two channels, however this does not show on the icons in the Program Builder window.

Grounding of Thermocouples

When thermocouple inputs are measured by the *dataTaker*, the analog input channels are by default internally terminated to the *dataTaker* ground via 1 MOhm resistors. The thermocouple measurement junctions must not be grounded externally.

When externally grounded thermocouples are measured, the internal termination of inputs can be disabled or Un-terminated by the U channel option.

The thermocouple channel specification for example

```
1..5TT(U)
```

specifies the thermocouple inputs are not to be terminated.

Thermocouples connected to the *dataTaker* as single ended inputs are not terminated, because these thermocouples are grounded via the Analog Common.

Thermocouple Measurement Accuracy

The differential thermocouple measurement errors of the *dataTaker* are small. The thermocouple errors are summarized below for all thermocouple types supported. These errors include voltage measurement and linearization errors.

The error envelope does not include errors produced by the method of thermocouple connection to the *dataTaker*, such as implementation and accuracy of the external isothermal reference junctions.

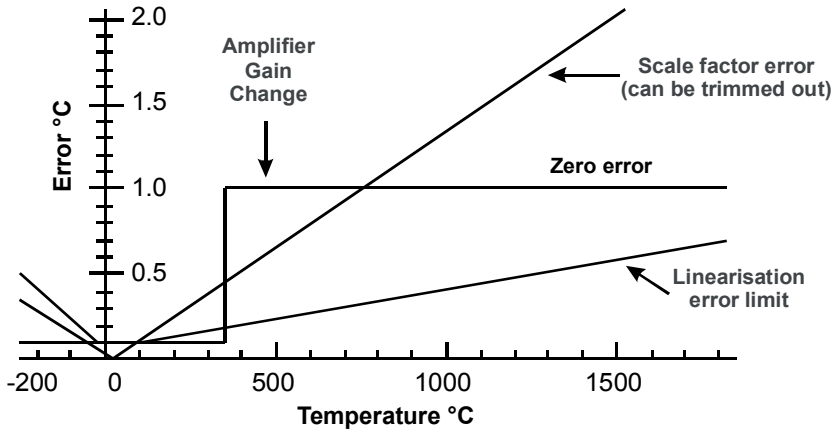


Figure 76 – Thermocouple Measurement Errors

If proper care is taken to implement thermocouples with respect to the various electrical, mechanical and thermal considerations discussed in the preceding sections, thermocouple measurement accuracies of 0.3°C can be achieved.

The thermocouple type G has a ±1 Deg C error band, which is greater than for the other thermocouple types supported.

The temperature measurement accuracy of thermocouples is dependent on a number of other factors including

grade of thermocouple wire – several grades of thermocouple wire are available, ranging from high accuracy measurement wire to low accuracy extension wire

method of junction construction – many methods of junction construction are available, including twisted and soldered junctions, crimped junctions and welded junctions.

Soldering of junctions is not recommended, because this introduces additional metals to the thermocouple junction and produces measurement errors.

Error Messages

Over-ranging may occur if external offset voltages are present in the thermocouple voltage signal.

Thermocouple signals which fall outside the linearization range are returned as temperature data, which is extrapolated from the linearization functions.

However the *dataTaker* also reports the error condition with the error message ‘E16–Linearization error’ if the Messages Switch /M is enabled.

Measuring Temperature with RTDs

The *dataTaker* data loggers support several types of platinum, nickel and copper Resistance Temperature Detectors (RTDs).

These devices provide the most accurate and stable method for measuring temperature with the *dataTaker* data loggers.

The RTD element is a resistive device which changes resistance in an accurately known way with changes in temperature.

The *dataTaker* accurately measures the changes in resistance of the element, and linearizes the resistance to units of temperature.

RTDs have a defined resistance at 0.0 Deg C, which varies between manufacturers and types. The most common are PT100 sensors, which are Platinum RTDs that have a resistance of 100 Ohms at 0.0 Deg C. The *dataTaker* supports all RTDs types which have a resistance at 0.0 Deg C of 10 Ohms to 2000 Ohms.

The linearization errors for RTDs by the *dataTaker* are less than 0.1 Deg C over the range of -200 Deg C to +600 Deg C.

In practice the accuracy of temperature measurement using RTDs is limited by sensor imperfections such as impurities in the element, physical stress in the element, etc.

RTD Support by the *dataTaker*

The *dataTaker* measures RTDs directly as resistive devices. The resistance of the RTD element is measured by any of the 2 wire, 3 wire or 4 wire resistance measurement methods (See Section II – Measuring Resistance).

The 4 wire resistance measurement method for RTDs provides the highest accuracy, since this method fully compensates for cable wire resistances. However the 3 wire method also provides high accuracy.

The resistance of the RTD is measured, and the ratio of this resistance to the specified resistance at 0.0 Deg C is calculated. The ratio is then linearized to units of temperature using the standard RTD calibration tables.

Platinum RTD Types

The *dataTaker* supports two platinum RTD calibration standards, which are specified by the analog input type identifiers as follows

α=0.003850 Ohm/Ohm/Deg C	*dataTaker* channel type PT385
α=0.003916 Ohm/Ohm/Deg C	*dataTaker* channel type PT392

These platinum RTD temperature coefficients comply with the following standards

α=0.003850 Ohm/Ohm/Deg C	DIN43760-1980, IEC751-1983
α=0.003916 Ohm/Ohm/Deg C	US Standard JIS C1604-1981

The platinum RTD types supported by the *dataTaker* are all assumed to have a sensor resistance at 0.0 Deg C of 100.0 Ohms.

However Platinum RTDs with resistances of other than 100.0 Ohms at 0.0 Deg C can also be used. The zero temperature resistance of these RTDs must be declared as a channel option in the channel specification.

Using DeTransfer, the commands for example

```
1PT385
2PT385(120.0)
5PT392(200.0)
```

specifies three different types of platinum RTD sensor as follows

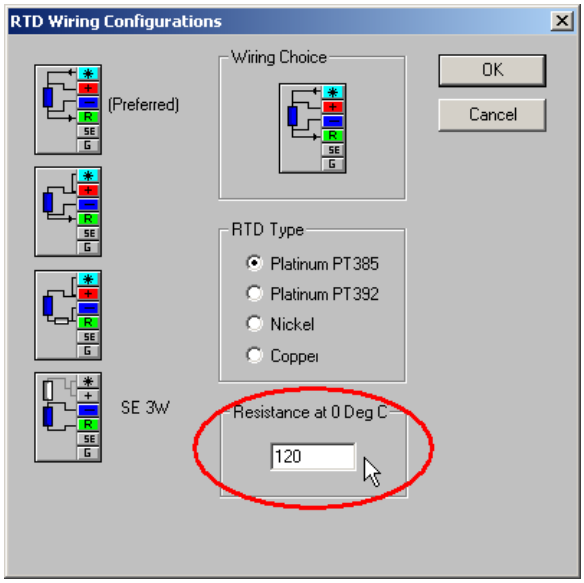
a platinum RTD sensor with a coefficient of α=0.003850 Ohm/Ohm/Deg C, and a resistance of 100.0 Ohms at 0.0 Deg C (default)

a platinum RTD sensor with a coefficient of α=0.003850 Ohm/Ohm/Deg C, and a resistance of 120.0 Ohms at 0.0 Deg C

a platinum RTD sensor with a coefficient of α=0.003916 Ohm/Ohm/Deg C, and a resistance of 200.0 Ohms at 0.0 Deg C

Any number and type of RTD sensors which have different resistances at 0.0 Deg C can be used in the same application.

Using DeLogger, the zero temperature resistance for RTDs is entered into the RTD Wiring Configuration when the program is created in the Program Builder.



If the resistance at 0.0 Deg C for the platinum RTD sensors is not specified, then the default of 100.0 Ohms is applied. This is equivalent to

```
2PT385(100.0)
5PT392(100.0)
```

The resistance for the RTD sensor at 0.0 Deg C is used to calculate the ratio of the measured sensor resistance to the zero temperature resistance (Ohm/Ohm/Deg C).

Nickel RTD Types

The *dataTaker* also supports nickel RTDs, which have a temperature coefficient of 0.005001 Ohm/Ohm/Deg C.

The nickel RTD types supported by the *dataTaker* are all assumed to have a sensor resistance at 0.0 Deg C of 1000.0 Ohms.

However Nickel RTDs with resistances of other than 1000.0 Ohms at 0.0 Deg C can also be used. The zero temperature resistance of the RTD must be declared as a channel option in the channel specification.

Using DeTransfer, the commands for example

```
1NI
3NI(1200.0)
6NI(1500.0)
```

specifies three different types of nickel RTD sensor as follows

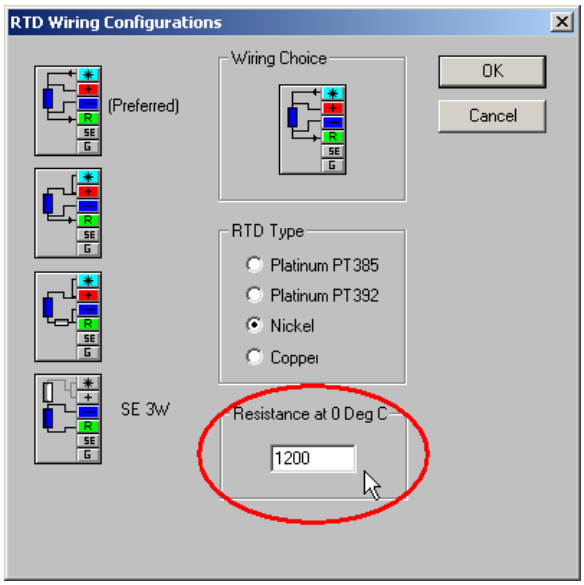
a nickel RTD sensor with a resistance of 1000.0 Ohms at 0.0 Deg C (default)

a nickel RTD sensor with a resistance of 1200.0 Ohms at 0.0 Deg C

a nickel RTD sensor with a resistance of 1500.0 Ohms at 0.0 Deg C

Any number and type of RTD sensors which have different resistances at 0.0 Deg C can be used in the same application.

Using DeLogger, the zero temperature resistance for RTDs is entered into the RTD Wiring Configuration when the program is created in the Program Builder.



If the resistance at 0.0 Deg C for the nickel RTD sensors is not specified, then the default of 1000.0 Ohms is applied. This is equivalent to

```
NI(1000.0)
```

The resistance for the RTD sensor at 0.0 Deg C is used to calculate the ratio of the measured sensor resistance to the zero temperature resistance (Ohm/Ohm/Deg C).

Copper RTD Types

The *dataTaker* also supports copper RTDs, which have a temperature coefficient of 0.0039 Ohm/Ohm/Deg C.

The copper RTD types supported by the *dataTaker* are all assumed to have a sensor resistance at 0.0 Deg C of 100.0 Ohms.

Copper RTDs with resistances of other than 100.0 Ohms at 0.0 Deg C can also be used. The zero temperature resistance of the RTD must be declared as a channel option in the channel specification.

The *dataTaker* support for copper RTDs allows the temperature of copper coils such as those in relays, solenoids, vibrating wire strain gauges, etc. to be measured. The resistance of the coils is declared as the channel option.

Using DeTransfer, the commands for example

```
1CU
3CU(120.0)
7CU(150.5)
```

specifies three different types of copper RTD sensor as follows

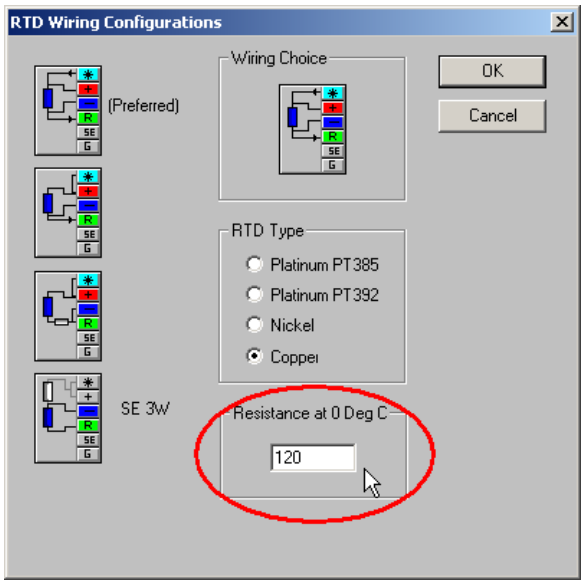
a copper RTD sensor with a resistance of 100.0 Ohms at 0.0 Deg C (default)

a copper RTD sensor with a resistance of 120.0 Ohms at 0.0 Deg C

a copper RTD sensor with a resistance of 150.5 Ohms at 0.0 Deg C

Any number and type of RTD sensors which have different resistances at 0.0 Deg C can be used in the same application.

Using DeLogger, the zero temperature resistance for RTDs is entered into the RTD Wiring Configuration when the program is created in the Program Builder.



If the resistance at 0.0 Deg C for the copper RTD sensors is not specified, then the default of 100.0 Ohms is applied. This is equivalent to

```
3CU(100.0)
```

The resistance for the RTD sensor at 0.0 Deg C is used to calculate the ratio of the measured sensor resistance to the zero temperature resistance (Ohm/Ohm/Deg C).

Excitation Current

The *dataTaker* outputs a precise constant current from the Excite terminal of the analog channel during measurement of the RTD.

This current is turned on when the channel is selected, and remains on until after the analog to digital conversion is completed. This is usually for a period of 30 mS, and so there is little self heating of the RTD.

The RTD temperature sensors are connected between the Excite terminal of the analog input channels and the Analog Return or Ground terminals, such that this excitation current passes through the sensor element.

The voltage produced across the RTD sensor element by the excitation current is connected by various methods to the analog input channels. The measured voltage drop is then used to calculate the value of the resistance of the RTD element.

The *dataTaker* can output either a 250.0 µA or a 2.500 mA excitation current from the Excite terminal (See Section II – Measuring Resistance).

The 2.500 mA excitation current is the most appropriate for RTD measurement, although the 250.0 µA excitation current can be used if desired.

While the temperature resolution obtained for the two currents is generally the same, the higher current is preferred because any input noise which may be present is reduced. However the trade-off is self heating, because the 30 mS pulse of the higher current during measurement can heat the sensing element.

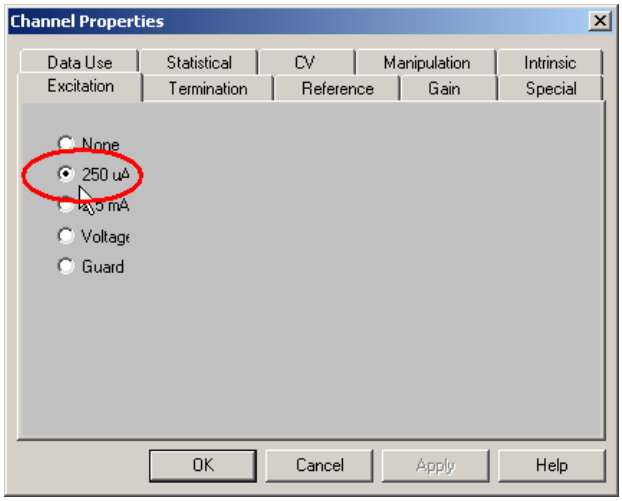
If the non default 250.0 µA excitation current is to be used for RTD measurement, then this is specified as part of the channel specification in the schedule lists.

Using DeTransfer, the commands for example

```
3PT385(I)
```

specifies that the platinum RTD sensor connected to analog channel 3 is to be measured using a 250.0 µA (channel option I) excitation current from the Excite terminal.

Using DeLogger, click Channel Options:Excitation to open the Channel Properties dialog, and select the 250 µA radio button under the Excitation tab.



The 2.500 mA excitation current is selected by default when an RTD input type is specified, and does not need to be specifically selected.

Connection and Measurement of RTDs

RTDs can be connected to the analog channels in three different ways, which provide varying degrees of compensation for cable wire resistance. The connection methods are the 4 wire, 3 wire and 2 wire methods of resistance measurement, and are described in the following sections.

These RTD connection methods use all four terminals of the analog input channels. Therefore a total of 5 RTDs can be monitored by the *dataTaker 50*, and a total of 10 RTDs can be monitored by the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD).

A fourth RTD measurement method is also supported, where the RTDs are connected as single ended inputs to the analog input channels. The single ended input methods do not perform cable wire compensation, but allow up to twice as many RTDs to be measured.

Four Wire RTD Measurement

The 4 wire method of RTD measurement is the most accurate, since there is no current flowing in the signal cable wires.

In this method two cable wires are connected to each end of the RTD. One pair of cable wires carries the excitation current, the other pair of cable wires is used to measure the voltage across the RTD.

The 4 wire method provides accurate compensation for cable wire resistance, and should be used where cables are long or the individual cable wires are of unequal resistance

RTDs are directly connected to the analog input channels for 4 wire resistance measurement as follows

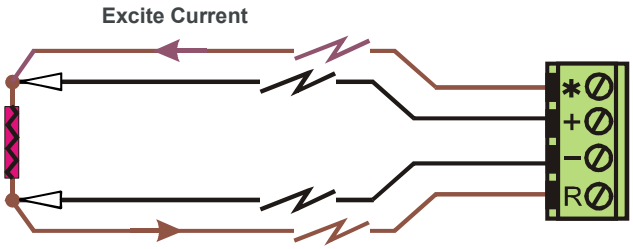


Figure 77 – Connection for Four Wire RTD Measurement

During RTD resistance measurement, the excitation current output from the Excite terminal passes through the RTD and returns into the Analog Return terminal.

The voltage produced across the RTD sensor is measured differentially between the +ve and –ve terminals.

The excitation current flows in the excitation circuit (shown by the red path), which is totally separate from the measurement circuit. Because of the high input impedance of the analog to digital converter, there is negligible current flow in the measurement circuit. Therefore there is a negligible cable resistance component in the voltage drop measured across the RTD.

Each RTD measured by the four wire method requires all terminals of the analog input channel.

Therefore a maximum of 5 RTDs can be measured using this method by the *dataTaker 50*, and a maximum of 10 RTDs can be measured using this method by the *dataTaker 500/600* series loggers and the Channel Expansion Module.

RTDs connected to the analog input channels using the 4 wire method are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1PT385(4W)   3NI(1200.0,4W)
END
```

instructs the *dataTaker* to measure the RTDs connected to analog input channel 1 and channel 3.

The excitation current channel option is not specified, and so the default 2.500 mA excitation current is used.

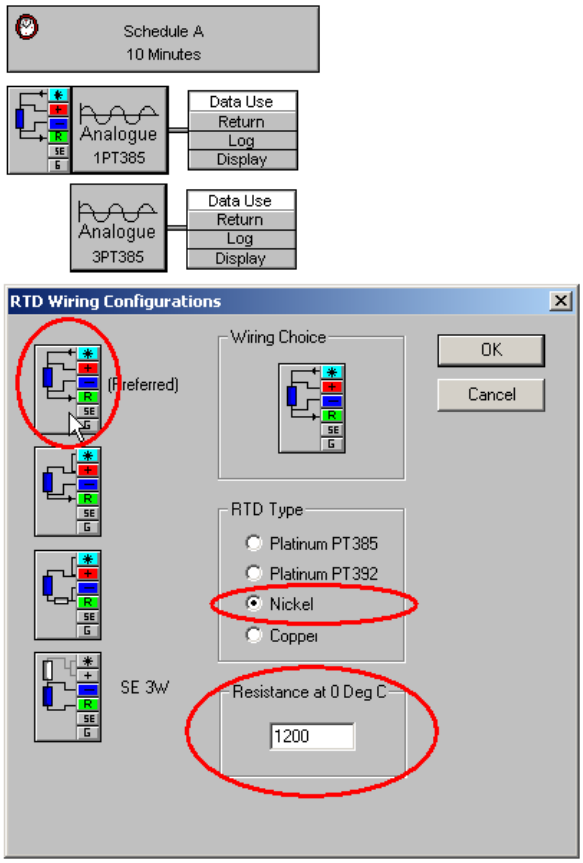
The platinum RTD connected to channel 1 is assumed to have a resistance of 100.0 Ohms at 0.0 Deg C. The nickel RTD connected to channel 3 has a declared resistance of 1200.0 Ohms at 0.0 Deg C

The PT385 indicates that the signal applied to these channels is from a platinum RTD, with a temperature coefficient of α=0.003850 Ohm/Ohm/Deg C. The resistance is linearized.

The NI indicates that the signal applied to this channels is from an nickel RTD. The resistance is linearized.

The 4W channel option indicates that the 4 wire method of resistance measurement is to be used.

Using DeLogger, RTDs connected by the 4 wire method can be measured by the following Program Builder program. The 4 wire connections are selected from the RTD Wiring Configurations dialog which opens when you have selected the analog input channel.



The data is returned in units of temperature.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Three Wire RTD Measurement

Most RTD measurements are made using the 3 wire resistance measurement method, which compensates for cable wire resistances in the measurement circuit (assuming that the two current carrying cable wires are of the same resistance).

In this method of measurement, one cable wire is connected to one end of the RTD, and two cable wires are connected to the other end.

The two cable wires connected to the lower end of the RTD are used to compensate the measured resistance for cable wire resistance.

RTDs are connected to the analog input channels for 3 wire measurement as follows

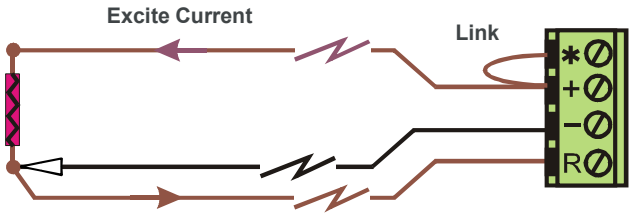


Figure 78 – Connection for Three Wire RTD Measurement

During RTD resistance measurement, the excitation current from the Excite terminal of the analog channel passes through the RTD and returns into the Analog Return terminal.

The voltage produced across the RTD sensor and the cable wire resistance is measured differentially between the +ve and –ve terminals.

Since all cable wires have resistance, the voltage produced across these is included in the total voltage measured. This introduces resistance offset errors, especially when long cables are used.

There is a much higher input impedance for the –ve input terminal than for the Analog Return terminal. This causes the excitation current to return in the cable wire connected to Analog Return. This ensures that there is no appreciable current flowing in the third cable wire which is connected to the –ve terminal.

The third cable wire is used to accurately measure the voltage due to the resistance of the return cable in the resistance measurement circuit. The *dataTaker* doubles this voltage, and subtracts it from the total voltage drop measured across the RTD and the cable wires.

This technique assumes that the two current carrying cable wires are of equal resistance. This in turn implies that cables used to connect RTDs to the *dataTaker* should be of equal length, gauge and type of wire.

Up to 5 RTDs can be measured using the 3 wire method by the *dataTaker 50*, and up to 10 RTDs can be measured by the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD).

RTDs connected using the 3 wire method are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1..2PT392
END
```

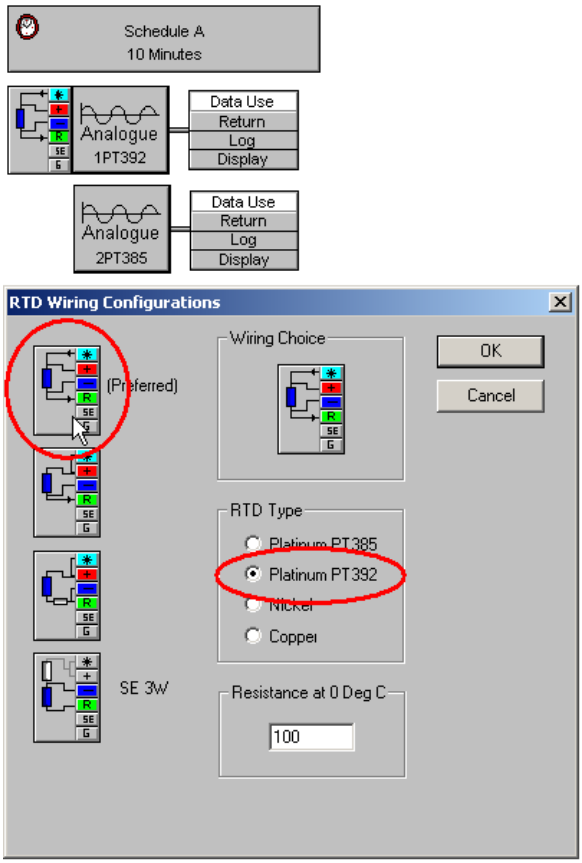
instructs the *dataTaker* to measure the two platinum RTDs connected to analog channels 1 and 2.

The excitation current channel option is not specified, and so the default 2.500 mA excitation current is used. The platinum RTDs are all assumed to have a resistance of 100.0 Ohms at 0.0 Deg C.

The PT392 specifies that the signal connected to these channels is from a platinum RTD, with a temperature coefficient of α=0.003916 Ohm/Ohm/Deg C. The resistance is linearized.

Using DeLogger, RTDs connected by the 3 wire method can be measured by the following Program Builder program.

The 3 wire connections are selected from the RTD Wiring Configurations dialog which opens when you have selected the analog input channel.



The data is returned in units of temperature.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Two Wire RTD Measurement

The 2 wire RTD measurement method is the simplest method for measuring RTDs with the *dataTaker*. This method only requires a single pair cable to connect each RTD to the logger. The cable pair carries both the excitation current and signal voltage.

The 2 wire RTD measurement method has the advantage of less cabling, which may be a consideration in some applications.

However this method also has the disadvantage of including the cable wire resistances in the measured resistance, which can produce significant errors. Measurement errors will also be produced by temperature effects on the cable wires, particularly if the cable wires are subject to temperature variations.

RTDs are connected to the analog input channels for 2 wire measurement as illustrated below.

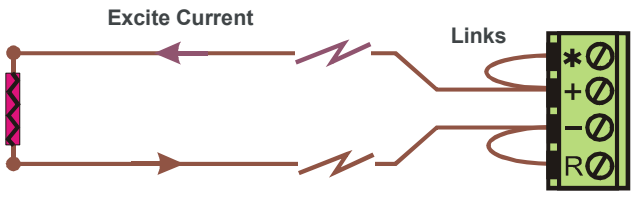


Figure 79a – Connection for Two Wire RTD Measurement

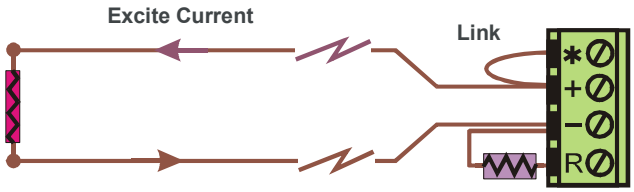


Figure 79b – Connection for Two Wire RTD Measurement

During RTD resistance measurement, the excitation current output from the Excite terminal passes through the RTD and cable wires, and returns into the Analog Return terminal.

The voltage produced by the excitation current across the RTD and the cable wires is measured between the +ve and –ve terminals.

Cable wire compensation is performed during measurement of the RTD, but is of little significance since the resistance of the links is very much less than that of the cable wires (Figure 79a).

Where cables with only two cable wires are already installed and must be used, cable wire resistance compensation can be obtained by replacing the link between the –ve terminal and Analog Return with a resistor (Figure 79b) of value equal to that of the total cable resistance.

However using a compensation resistor does not compensate for temperature effects on the cables.

RTDs connected using the 2 wire method are read and data returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1..3PT385(120.0)
END
```

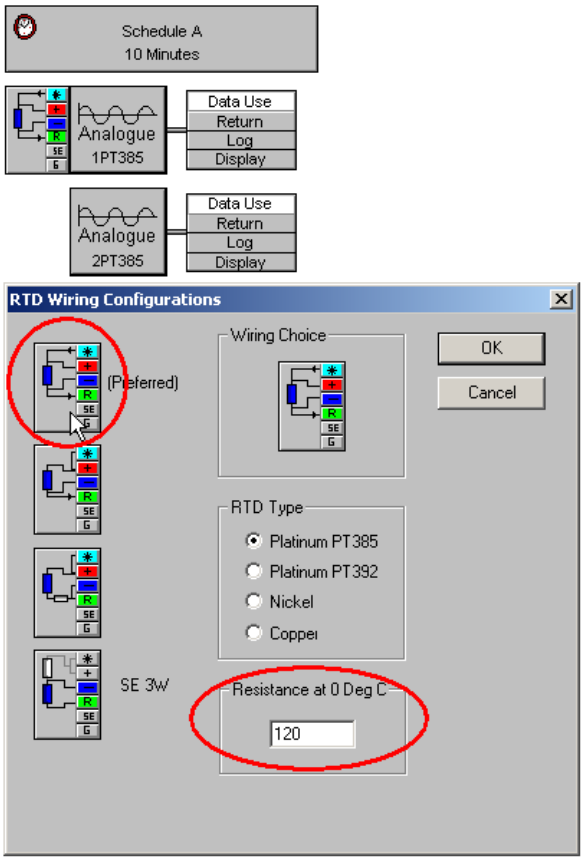
instructs the *dataTaker* to measure the three platinum RTDs connected to analog channels 1, 2 and 3.

The excitation current channel option is not specified, and the default 2.500 mA excitation current is used. The platinum RTDs all have a resistance of 120.0 Ohms at 0.0 Deg C.

The PT385 specifies that the signal applied to these channels is from a platinum RTD, with a temperature coefficient of α=0.003850 Ohm/Ohm/Deg C. The resistance is linearized.

Using DeLogger, RTDs connected by the 2 wire method can be measured by the following Program Builder program.

The 2 wire connections are selected from the RTD Wiring Configurations dialog which opens when you have selected the analog input channel.



The data is returned in units of temperature. The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Measuring RTDs Using Single Ended Inputs

Two RTDs can be measured using a combination of 4 wire input and a single ended input for each of the analog input channels. Note : This method does not provide cable wire compensation for the RTD connected as a single ended input.

Referenced to Analog Return

Pairs of RTDs are connected to the analog input channels with the second RTD referenced to Analog Return as follows

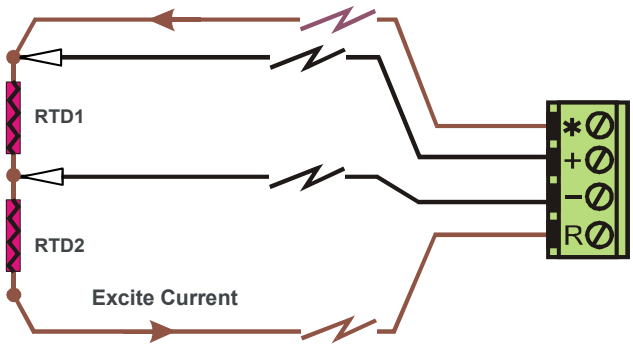


Figure 80 – Connecting RTDs for Single Ended Input

Up to 10 RTDs can be measured using this method by the *dataTaker 50*, and up to 20 RTDs can be measured by the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) .

During resistance measurement, the excitation current output from the Excite terminal of the analog channel passes through the two RTDs and cable wires, and returns into the Analog Return terminal.

The voltage produced across the RTD1 is measured as a differential input between the +ve and –ve terminals.

The voltage produced across the RTD2 is measured as a single ended input between the –ve terminal and Analog Return.

The two RTDs connected to the analog input channels by this method are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1PT385(4W)  1-PT385
END
```

instructs the *dataTaker* to measure the two platinum RTDs

connected between the +ve and –ve terminals of analog input channel 1. The 4W channel option indicates that the 4 wire method of resistance measurement is to be used. This measurement will be cable compensated.

connected between the –ve terminal of analog input channel 1 and Analog Return. This measurement will not be cable compensated.

The excitation current channel option is not specified, and the default 2.500 mA excitation current is used. The two platinum RTDs are assumed to have a resistance of 100.0 Ohms at 0.0 Deg C.

The PT385 specifies that the signal applied to these channels is from an platinum RTD, with a temperature coefficient of α=0.003850 Ohm/Ohm/Deg C. The resistance is linearized, and the data is returned in units of temperature.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Measurement Ranges and Accuracy

The measurement accuracy for RTD sensors is the same as that for standard resistance measurement, and is discussed in Section II – Resistance Measurement.

Greater accuracy of RTD measurement is achieved when 3 wire or 4 wire methods providing cable wire compensation are used, and the 2.500 mA RTD excitation current is used.

The temperature resolution is approximately 0.025 Deg C for 100 Ohms RTD sensors. However the temperature resolution can be increased by a factor of ten if the RTDs are measured as a half bridge with constant current excitation (See Section II – Measuring Bridges and Strain Gauges).

The disadvantage of this method however is that the *dataTaker* will not convert the resistance to units of temperature. This must be done with a user defined polynomial, or later by the host computer.

Sources of Error

There are two sources of error in RTD measurements by the *dataTaker*.

The first is due to errors in resistance measurement (excluding cable wire effects), which is specified at ±0.1% over the logger temperature range of 10 to 40 Deg C. This systematic error can however be calibrated out.

The second source of error is linearization error. The graph below illustrates the *dataTaker* linearization error limits along with the various tolerances.

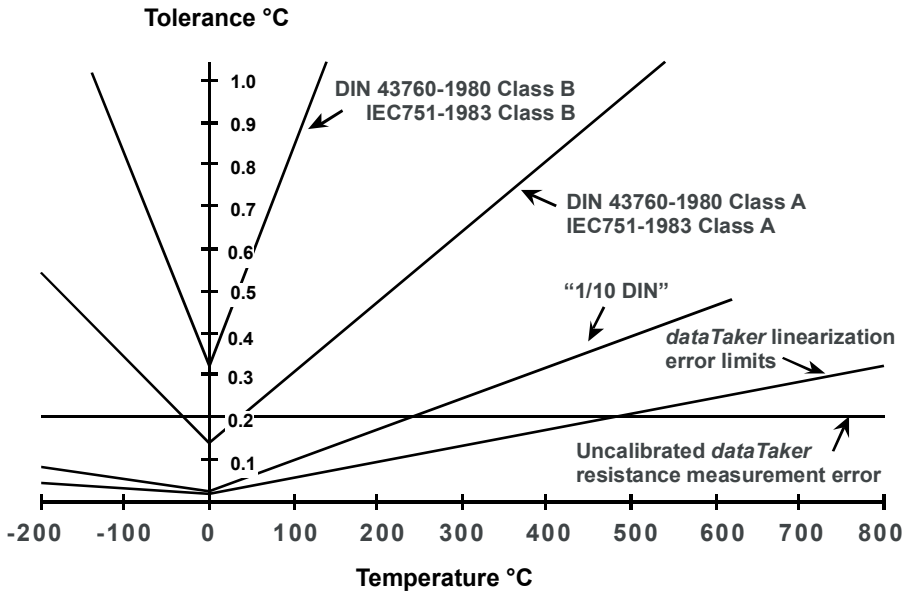


Figure 81 – RTD Linearization Error Limits

Error Messages

RTD resistances which fall outside of the linearization range of –200 to 600 Deg C are returned as temperature data, which is extrapolated from the linearization functions.

When this occurs, the *dataTaker* reports the error by returning the error message 'E16 - Linearization error' if the Messages Switch /M is enabled.

Measuring Temperature with Integrated Circuit Temperature Sensors

The *dataTaker* directly supports the range of integrated circuit (IC) temperature sensors.

IC temperature sensors can be used for measuring temperatures in the environmental and biological range of –50 Deg C to +150 Deg C.

There are four basic types of IC temperature sensors, according to the output signal produced in response to sensed temperature, as follows

Analog Devices AD590/592 series temperature sensor, which produces a linear current output of 1 µA/Deg K

National Semiconductor LM335 temperature sensor, which produces a linear voltage output of 10 mV/Deg

National Semiconductor LM34 temperature sensor, which produces a linear voltage output of 10 mV/Deg F

National Semiconductor LM35 temperature sensor, which produces a linear voltage output of 10 mV/Deg C

The *dataTaker 50* can measure up to 5 IC temperature sensors as differential inputs, and up to 10 IC temperature sensors as single ended inputs.

The *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) can measure up to 10 of the IC temperature sensors as differential inputs, and up to 30 IC temperature sensors as single ended inputs.

Powering and Connecting IC Temperature Sensors

The IC temperature sensors can either be powered from external power sources, or can be powered directly from the Excite terminals of the analog input channels of the *dataTaker*.

When powered from the Excite terminal, the sensors are powered for 30 mS during the period of measurement. This has the benefit of reducing self heating effects on the sensors by continuous currents flowing through them from external power supplies.

IC temperature sensors can be connected to the analog input channels of the *dataTaker* in a number of different ways depending on powering arrangement, as follows

when the sensors are powered from the Excite terminals, the signal can be connected to the analog channels as differential inputs or as +ve and –ve terminal single ended inputs

when the sensors are powered from external sources, the signal can be connected to the analog input channels as differential inputs or as +ve, –ve and ✶ terminal single ended inputs

Each of these methods of connection and powering is discussed for each of the IC temperature sensors in the sections below.

Readings from the IC temperature sensors is returned in units of temperature defined by the setting of the Parameter36 command (See Section II – Measuring Temperature).

The Analog Devices AD590/592 Sensor

The Analog Devices AD590/592 is a two terminal IC temperature transducer which produces an output current proportional to absolute temperature. The AD590 is rated for use over the temperature range of –55 Deg C to +150 Deg C.

At power supply voltages between 4 to 12 Volt, the AD590 acts as a high impedance constant current regulator, and passes a current of 1 µA/Deg K. The thin film resistors on the IC device's chip are laser trimmed to calibrate the device to 298.2 µA output at 298.2 Deg K (25 Deg C).

The AD590 is particularly useful for remote temperature sensing applications, because the sensor delivers an output current which is insensitive to voltage drops over long cables. Any well insulated twisted pair cable is sufficient for operation hundreds of feet from the *dataTaker*.

For detailed specifications of the AD590, refer to the *Analog Devices Data Acquisition Handbook — Volume 1 Integrated Circuits* (1984).

The current signal from the AD590 temperature sensor can be connected to the analog input channels of the *dataTaker* in any of the configurations for current measurement described in Section II – Measuring Current.

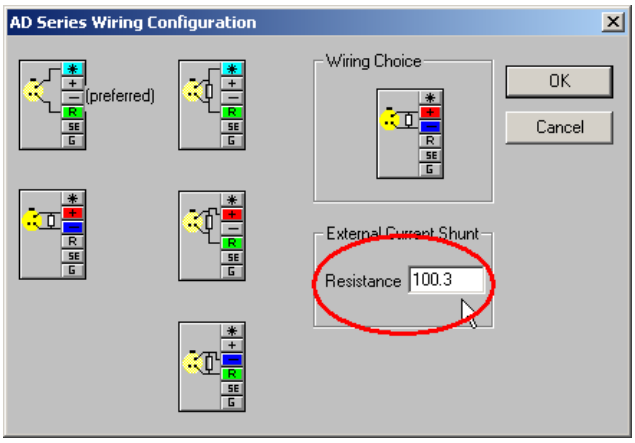
The AD590 is available in a number of tolerances or grades. The accuracy of different grades varies between ±1 Deg C and ±5 Deg C. A calibration factor to correct sensor accuracy can be specified by adjusting the shunt resistance channel option.

Using DeTransfer, the specification for example

```
8#AD590(100.3)
```

instructs the *dataTaker* to apply a calibration factor of 0.3 to the shunt resistance to correct the AD590 sensor measured on channel 8.

Using DeLogger, the calibration factor/shunt resistor adjustment can be entered into the AD Series Wiring Configuration which opens when you have selected the analog channel



The AD590 is supported by the *dataTaker* in a number of ways as follows

- the sensor can be powered from the Excite terminal of the analog input channels, or be externally powered
- the current signal from the sensor can be measured using the internal current shunt within the analog channels, or by using external current shunts.

When using external current shunts, the voltage across the shunt can be connected to the analog input channels as differential or single ended inputs.

These options provide a variety of ways in which the AD590 temperature sensor can be connected to the *dataTaker*. The most commonly used connections are described in the sections below.

AD590 Directly Powered and Measured Using Internal Shunts

The AD590 temperature sensor can be powered from the Excite terminal of the analog input channels, and the current signal from the sensor measured across the internal current shunt between Analog Return and GND.

This is the most commonly used and simplest connection method.

The AD590 sensors are connected directly to the analog input channels between the Excite terminal and the Analog Return terminal.

The connection for AD590 sensors powered from the Excite terminal and measured using the internal current shunts is illustrated below.

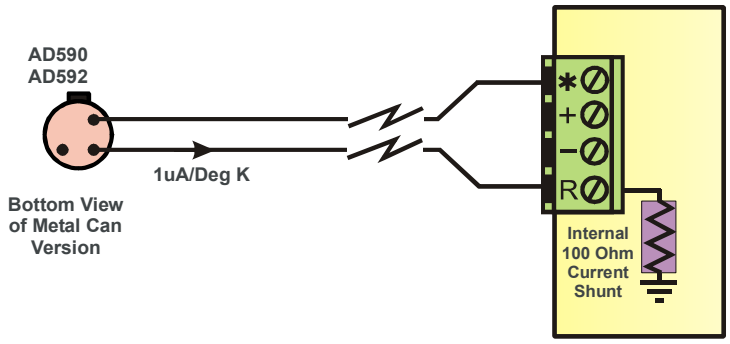


Figure 82 – Connection of AD590 Sensors Powered from the Excite Terminal and Measured Using Internal Shunts

The unused +ve and –ve terminals of channels monitoring the AD590 sensors can be used for other differential and single ended inputs.

AD590 temperature sensors powered directly from the Excite terminals and measured using the internal current shunts are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  1#AD590  5#AD590
END
```

instructs the *dataTaker* to measure the current signals from the AD590 sensors connected to analog input channels 1 and 5, using the internal shunts.

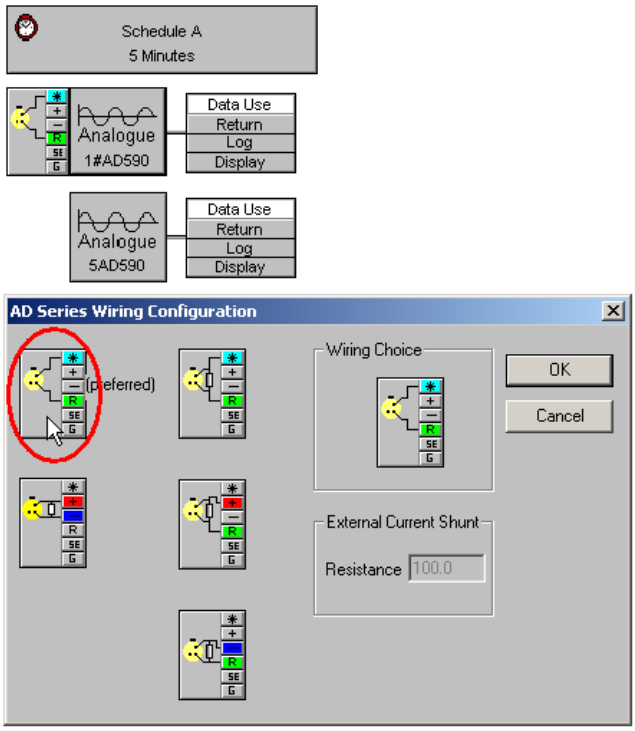
The AD590 specifies that the signals applied to these channels are from AD590 sensors.

The AD590 channel type defaults to Excite terminal powering, and so an Excite terminal channel option does not have to be specified.

The # indicates that the current signal is input to the Analog Return terminal, and is to be measured across the internal current shunt.

Using DeLogger, AD590 sensors that are connected for direct powering are measured by the following Program Builder program.

The connection is selected from the AD Series Wiring Configurations dialog which opens when you have selected the analog input channel.



The data is returned in units of temperature.

The *dataTaker* will read the sensors every 5 minutes, and readings are stopped by entering a H (Halt) command.

AD590 Directly Powered and Measured Using External Shunts

The AD590 temperature sensor can be powered from the Excite terminal, and the current signal from the sensor measured across external current shunts.

The voltage produced across the external shunts by the sensor output current can be measured as differential or single ended inputs (See Section III – Measuring Currents).

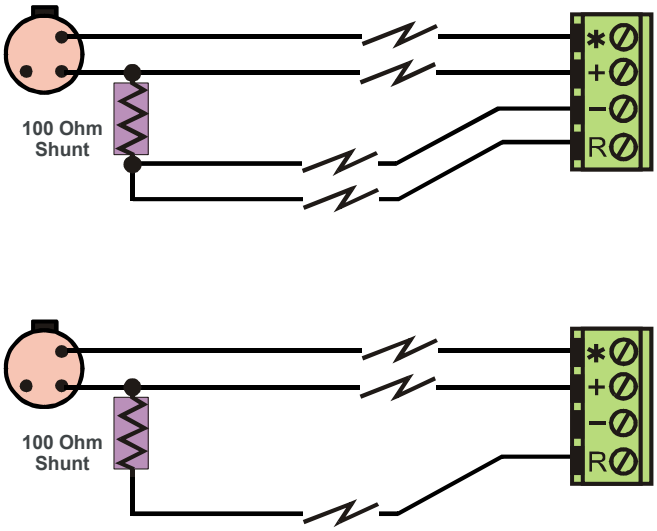


Figure 83 – Connection of AD590 Sensors Powered from the Excite Terminal and Measured Using External Shunts

The connection for AD590 sensors which are powered from the Excite terminal and are measured using external current shunts connected as differential and single ended inputs are illustrated above. The upper schematic is for a differential connection of the shunt resistor, and the lower is for a single ended connection of the shunt resistor.

The external shunt is assumed to have a resistance of 100 Ohms. If a different shunt resistance is used, then the resistance is declared as a channel option (see above).

AD590 sensors powered directly from the Excite terminals and measured using the external current shunts are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  1AD590  3+AD590  3-AD590
END
```

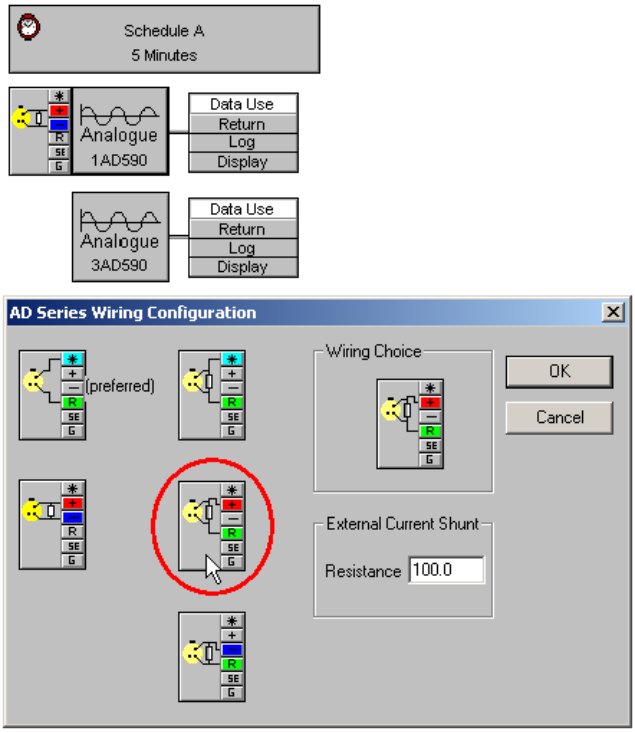
instructs the *dataTaker* to measure

- the current signal from the AD590 sensor connected as a differential input to the +ve and –ve terminals of channel 1
- the current signal from the AD590 sensor connected as a single ended input to channel 3+ and Analog Return
- the current signal from the AD590 sensor connected as a single ended input to channel 3– and Analog Return

using external shunts

The AD590 specifies that the signals applied to these channels are from AD590 sensors. The AD590 channel type defaults to Excite terminal powering, and so powering does not have to be specified.

Using DeLogger, AD590 sensors that are connected for direct powering and external shunts are measured by the following Program Builder program. The connection is selected from the AD Series Wiring Configurations dialog which opens when you have selected the analog input channel.



Note: The sensor powering from the Excite terminal is not shown in these icons.

The data is returned in units of temperature.

The *dataTaker* will read the sensors every 5 minutes, and readings are stopped by entering a H (Halt) command.

AD590 Externally Powered and Measured Using External Shunts

The AD590 temperature sensors can be powered from an external power supply of 4 to 12 Volt, or from the 5 Volt switched sensor power supply of the *dataTaker*.

The current signal from the sensor can be measured using external current shunts that are connected to the analog input channels as differential or single ended inputs.

Single ended inputs can be referenced to Analog Return, or can be referenced to an external common which is connected to the SE REF input terminal.

Differential Inputs

The AD590 temperature sensors can be powered from an external 4 to 12 Volt power supply. The current signal from the sensor is measured using external current shunts connected to the analog input channels as differential inputs.

The current shunt is assumed to have a resistance of 100 Ohms. If a different shunt resistance is used, then this must be declared as a channel option (see above).

The connection for externally powered AD590 sensors using external current shunts and differential inputs is illustrated below

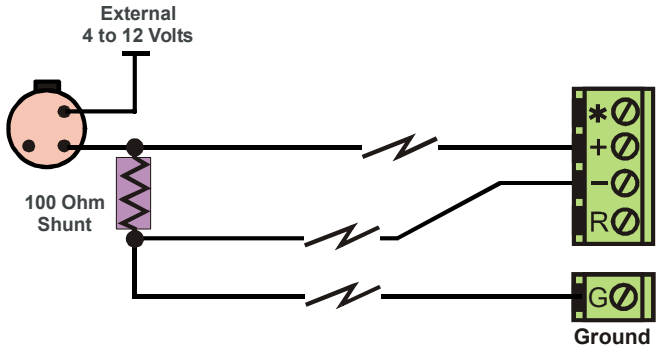


Figure 84 – Connection of Externally Powered AD590 Sensors Using External Shunts and Differential Inputs

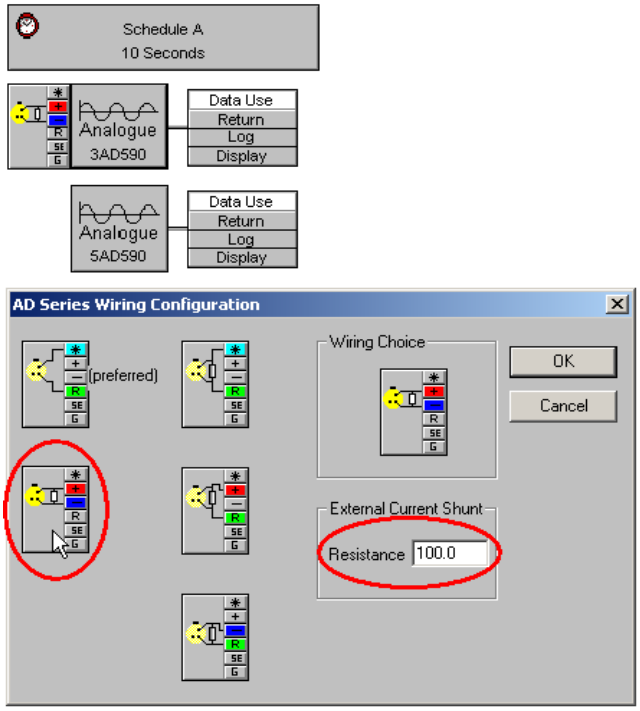
Using DeTransfer, the command for example

```
BEGIN
 RA10S
  3AD590(N)   5AD590(N)
END
```

instructs the *dataTaker* to measure the current signals from AD590 sensors connected to analog input channels 3 and 5, from external shunts.

The AD590 specifies that the signals applied to these channels are from AD590 sensors. The AD590 channel type defaults to Excite terminal powering, and so the channel option N should be specified to disable output from the Excite terminal.

Using DeLogger, AD590 sensors connected for external powering and differentially connected external shunts are measured by the following Program Builder program. The differential connection is selected from the AD Series Wiring Configurations dialog which opens when you select the analog input channel.



Note: The sensor powering from the Excite terminal is not shown in these icons.

The data is returned in units of temperature.

The *dataTaker* reads sensors every 10 seconds, and readings are stopped by a H (Halt) command.

Single Ended Inputs Referenced to Analog Return

The AD590 temperature sensor can be externally powered from an external 4 to 12 Volt power supply, and the current signal from the sensor measured using external current shunts connected to the analog input channels as single ended inputs. The single ended inputs can be referenced to Analog Return (See Section II – Measuring Currents).

The current shunt is assumed to have a resistance of 100 Ohms. If a different shunt resistance is used, then this must be declared as a channel option (see above).

The connection of externally powered AD590 sensors using external shunts and single ended inputs referenced to Analog Return is illustrated below

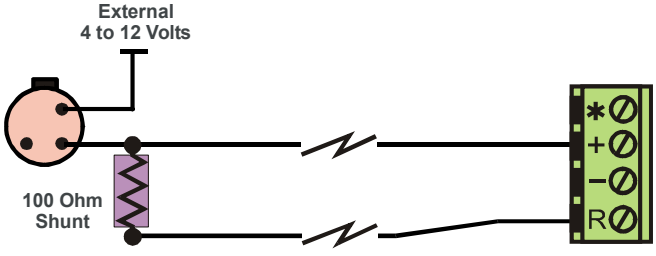


Figure 85 – Connection of Externally Powered AD590 Sensors Using External Shunts and Single Ended Inputs Referenced to Analog Return

By using the Excite terminal as an input, the *dataTaker 500/600* series loggers are able to monitor up to 30 AD590 sensors connected as single ended inputs referenced to Analog Return.

The *dataTaker 50* does not support the Excite terminal as an input, and can only monitor up to 10 sensors.

Externally powered AD590 temperature sensors using external shunts connected to the analog input channels as single ended inputs referenced to Analog Return are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

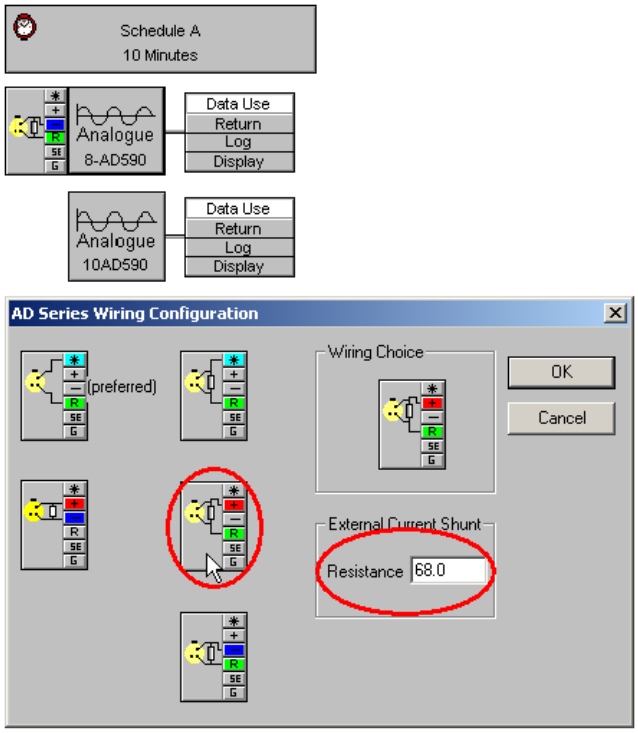
```
BEGIN
 RA10M
  8-AD590(N)   10+AD590(68,N)
END
```

instructs the *dataTaker* to measure the current signal from the AD590 sensors

- using 100 Ohm external shunts connected between single ended analog channels 8– and Analog Return
- using a 68 Ohm external shunt connected between single ended analog channel 10+ and Analog Return.

The AD590 specifies that the signals applied to these channels are from AD590 sensors. The AD590 channel type defaults to Excite terminal powering, and so the channel option N should be specified to disable output from the Excite terminal.

Using DeLogger, AD590 sensors connected for external powering and external shunts connected as single ended inputs are measured by the following Program Builder program. The single ended connection is selected from the AD Series Wiring Configurations dialog which opens when you have selected the analog input channel.



Note: The sensor powering from the Excite terminal is not shown in these icons.

The data is returned in units of temperature.

The *dataTaker* will read the sensors every 10 minutes, and readings are stopped by entering a H (Halt) command.

Single Ended Inputs Referenced to an External Common

Remotely located AD590 temperature sensors powered from an external power supply and using external current shunts, can be measured with respect to an external common reference point which is connected to the SE REF terminal.

The external current shunt is assumed to have a resistance of 100 Ohms. If a different shunt resistance is used, then the shunt resistance must be declared as a channel option (see above).

The connection of externally powered AD590 sensors using external current shunts and single ended inputs referenced to an external common is illustrated below

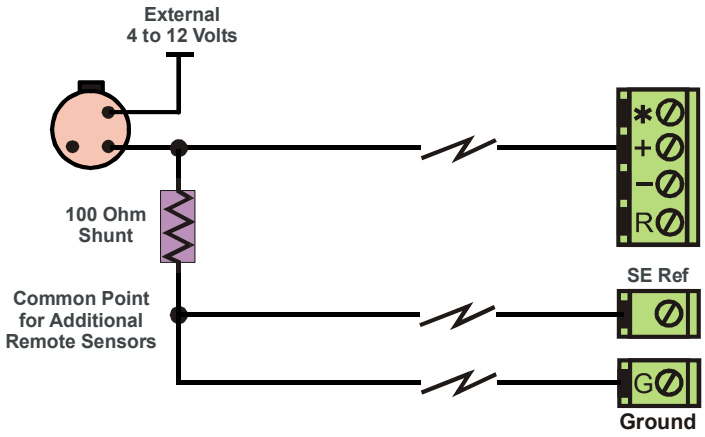


Figure 86 – Connection of Externally Powered AD590 Sensors Using External Shunts and Single Ended Inputs Referenced to an External Common

Up to 15 AD590 sensors can be connected to the *dataTaker 50* in this way, and up to 40 AD590 sensors can be connected to the *dataTaker 500/600* series loggers.

Externally powered AD590 temperature sensors using external shunts connected to the analog input channels as single ended inputs referenced to an external common are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  6-AD590(N,X)
END
```

instructs the *dataTaker* to measure the current from the AD590 sensors using an external 100 Ohm shunt connected between single ended analog input 6– and SE REF.

The AD590 specifies that the signals applied to these channels are from AD590 sensors. The AD590 channel type defaults to Excite terminal powering, and so the channel option N should be specified to disable output from the Excite terminal. The X channel option indicates that the single ended inputs are to be measured between the input terminal and SE REF. The data is returned in units of temperature.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

DeLogger does not directly support this method for measuring AD590 temperature sensors. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

The National Semiconductor LM335 Sensor

The National Semiconductor LM135/LM235/LM335 series of IC temperature sensors operate as 2 terminal zener diodes with a breakdown voltage directly proportional to absolute temperature at +10 mV/Deg K. These sensors measure temperatures within the range –55 Deg C to +150 Deg C.

When calibrated at 25 Deg C the LM335 series typically has an error of up to ±3 Deg C, depending on the grade of the sensor.

However the sensor has provision for calibration adjustment. A trim potentiometer of 20 - 50 KOhm can be connected across the LM335, with the wiper connected to the Adjust pin. This allows for a single point calibration of the device.

For detailed specifications of the LM335 series, refer to the *National Semiconductor Linear Databook (Vol 2)*.

The LM335 temperature sensor outputs +10 mV/Deg K, and so at the maximum temperature range of 150 Deg C (423 Deg K) the sensor will output 4.23 Volts. This exceeds the voltage input range of the *dataTaker*, and so the signal must be attenuated before input to the analog channels if the full temperature range is to be used. The *dataTaker* assumes that the LM335 voltage signal is attenuated by a factor of 2.

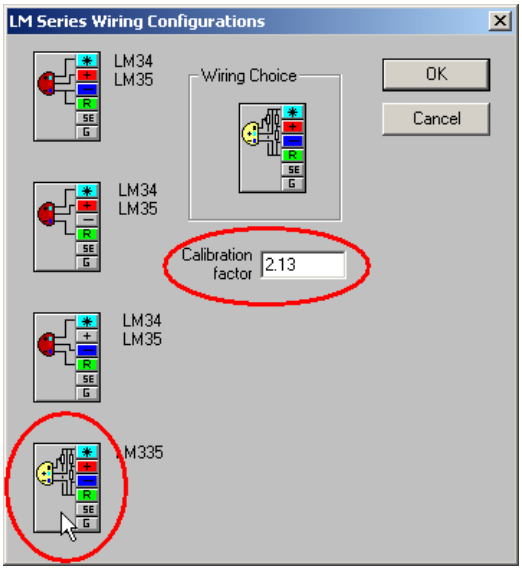
A calibration factor to correct sensor accuracy and tolerance of the attenuation resistors can be specified as a channel option.

Using DeTransfer, the command for example

```
6LM335(2.13)
```

instructs the *dataTaker* to apply a calibration factor of 0.13 to the LM335 sensor measured on analog channel 6. The calibration factor defaults to 2.0 which is the required input attenuation.

Using DeLogger, the calibration factor can be entered into the LM Series Wiring Configuration which opens when you have selected the analog channel



The LM335 is supported by the *dataTaker* in a number of ways as follows

the sensor can be powered from the Excite terminal of the analog input channels, or can be externally powered

the voltage signal from the sensor can be connected to the analog input channels as differential or single ended inputs

The voltage signal from the LM335 temperature sensor can be connected to the analog channels of the *dataTaker* in any configuration for high level voltage measurement described in Section II – Measuring High Level Voltages.

The most commonly used connections are described in the sections below.

LM335 Directly Powered

LM335 temperature sensors can be powered directly from the Excite terminal of the *dataTaker*, and the output voltage from the sensor measured either as differential or single ended inputs.

The single ended inputs can be referenced to Analog Return or to an external common connected to SE REF.

The upper temperature measurement range for LM335 sensors powered from the Excite terminal is limited to 70 Deg C because the voltage output from the Excite terminal is not sufficient to drive the sensor over its full temperature measurement range.

Up to 10 LM335 sensors can be connected to the *dataTaker 500/600* series loggers as differential inputs, and up to 30 as single ended inputs.

Up to 5 LM335 sensors can be connected to the *dataTaker 50* as differential inputs, and up to 10 as single ended inputs

Differential Input

The connection for LM335 sensors powered from the Excite terminal and measured as differential inputs is illustrated below.

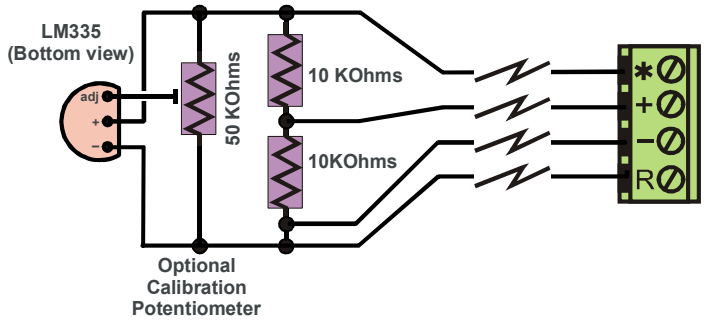


Figure 87 – Connection of LM335 Sensors Powered from the Excite Terminal and Differential Inputs

The attenuation resistors can either be located with the sensor, or connected across the terminals of the analog channel.

The fourth wire between Analog Return and the bottom of the sensor can be replaced by a link between the Analog Return and –ve terminals. However if cable resistance is significant, then accuracy will be reduced.

LM335 temperature sensors powered directly from the Excite terminals and measured as differential inputs are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

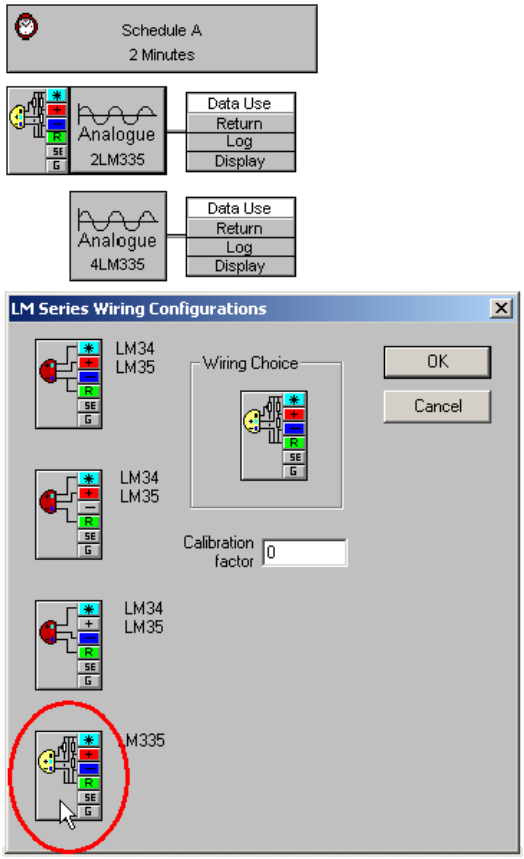
```
BEGIN
 RA2M
  2LM335  4LM335
END
```

instructs the *dataTaker* to measure the voltage signals from an LM335 sensor differentially connected to analog channels 2 and 4.

The LM335 specifies that the voltage applied to this channel is from an LM335 sensor.

The LM335 channel type defaults to Excite terminal powering, and so powering does not have to be specified.

Using DeLogger, LM335 sensors powered from the Excite terminal and connected as differential inputs are measured by the following Program Builder program. The single ended connection is selected from the LM Series Wiring Configurations dialog which opens when you have selected the analog input channel.



The data is returned in units of temperature.

The *dataTaker* will read the sensors every 2 minutes, and readings are stopped by entering a H (Halt) command.

Single Ended Inputs

LM335 sensors powered from the Excite terminals can be measured as single ended inputs referenced to Analog Return or to an external common connected to SE REF.

However these connections are not normally used because of the limited maximum temperature range due to the limited drive voltage available from the Excite terminal, and so are not described here.

LM335 Externally Powered

The LM335 temperature sensor can be externally powered from a power supply greater than 4 Volts.

The output signal voltage can be measured as a differential input, or as a single ended input referenced to Analog Return or external common connected to SE REF terminal.

To reduce self heating of the sensor, a series resistor should be included in the power supply circuit to limit current to between 0.5 and 1.0 mA. The resistance of the current limiting resistor depends on the voltage of the external power supply.

The external power can be switched off between readings by a relay that is controlled by the Schedule used to scan the LM335 sensors.

Up to 10 LM335 sensors can be connected to the *dataTaker 500/600* series loggers as differential inputs, and up to 30 as single ended inputs.

Up to 5 LM335 sensors can be connected to the *dataTaker 50* as differential inputs, and up to 10 as single ended inputs

Differential Input

The LM335 temperature sensor can be externally powered from any power supply greater than 4 Volts, and the output signal voltage can be measured as a differential input.

The connection of externally powered LM335 sensors measured as differential inputs is illustrated below

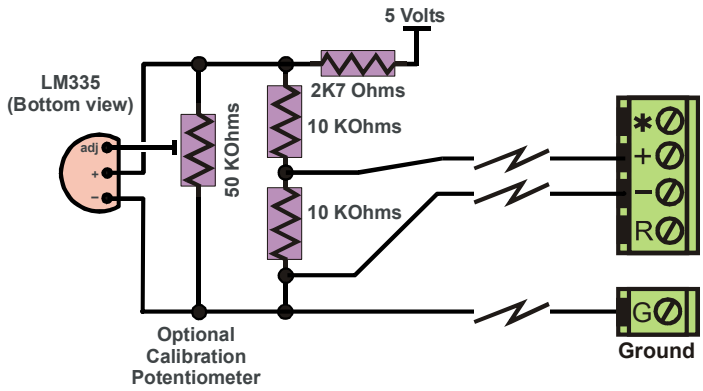


Figure 88 – Connection of Externally Powered LM335 Sensors as Differential Inputs

Externally powered LM335 temperature sensors connected as differential inputs are sampled and the data is returned when a Schedule containing the channel is executed.

The command for example

```
BEGIN
 RA10S
  3LM335(N)   7LM335(N)
END
```

instructs the *dataTaker* to measure the voltage signals from the LM335 sensors differentially connected to analog channels 3 and 7.

The LM335 specifies that the signals applied to these channels are from LM335 sensors.

The LM335 channel type defaults to Excite terminal powering, and so the channel option N should be specified to disable output from the Excite terminal.

The data is returned in units of temperature.

The *dataTaker* will read the sensors every 10 seconds, and readings are stopped by entering a H (Halt) command.

DeLogger does not directly support this method for measuring LM335 temperature sensors. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

Single Ended Inputs Referenced to Analog Return

The LM335 temperature sensor can be externally powered from a power supply greater than 4 Volts, and the voltage signals measured as single ended inputs referenced to Analog Common.

The connection of externally powered LM335 sensors as single ended inputs referenced to Analog Return is illustrated below

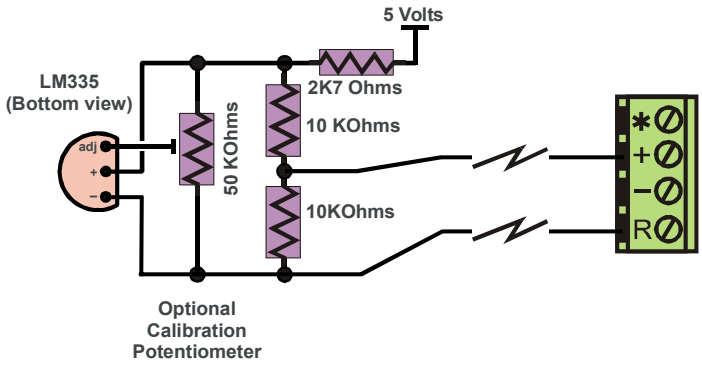


Figure 89 – Connection of Externally Powered LM335 Sensors as Single Ended Inputs Referenced to Analog Return

Externally powered LM335 temperature sensors connected as single ended inputs referenced to Analog Return are sampled and data returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA8M
  3+LM335(N)   3-LM335(N)
END
```

instructs the *dataTaker* to measure the voltage signals from the LM335 sensors which are connected as single ended inputs between analog channel 3+ and Analog Return, and channel 3– and Analog Return.

The LM335 specifies that the signals applied to these channels are from LM335 sensors. The LM335 channel type defaults to Excite terminal powering, and so the channel option N should be specified to disable output from the Excite terminal.

The data is returned in units of temperature. The *dataTaker* will read the sensors every 8 minutes, and readings are stopped by entering a H (Halt) command.

DeLogger does not directly support this method for measuring LM335 temperature sensors. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

Single Ended Inputs Referenced to an External Common

The LM335 temperature sensor can be externally powered from any power supply greater than 4 Volts, and the voltage signals measured as single ended inputs referenced to an external common which is connected to the SE REF input terminal.

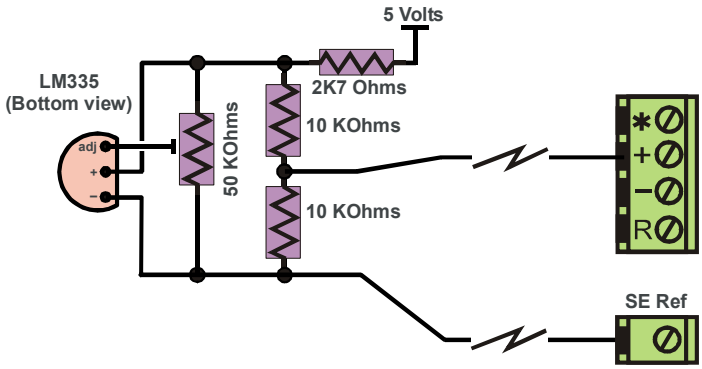


Figure 90 – Connection of Externally Powered LM335 Sensors as Single Ended Inputs Referenced to an External Common

The connection of externally powered LM335 sensors as single ended inputs referenced to an external common is illustrated above. The grounded ends of the LM335 temperature sensor voltage signals are connected to a common external reference point, which is in turn connected to the SE REF terminal.

Externally powered LM335 sensors connected as single ended inputs referenced to an external common are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA2M
  1-LM335(N,X)
END
```

instructs the *dataTaker* to measure the voltage signal from the LM335 sensor which is connected as a single ended input between analog channels 1– and SE REF.

The LM335 specifies that the signals applied to these channels are from LM335 sensors. The LM335 channel type defaults to Excite terminal powering, and so the channel option N should be specified to disable output from the Excite terminal. The X channel option indicates that the single ended inputs are to be measured between the input terminal and SE REF.

The data is returned in units of temperature. The *dataTaker* will read the sensors every 2 minutes, and readings are stopped by entering a H (Halt) command.

DeLogger does not directly support this method for measuring LM335 temperature sensors. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

The National Semiconductor LM34 and LM35 Sensors

The National Semiconductor LM34 and LM35 temperature sensors are precision integrated circuit devices, with output voltages which are linearly proportional to temperature as follows

LM34	10 mV/Deg F
LM35	10 mV/Deg C

The LM34 temperature sensor measures temperatures in the range of –70 Deg F to +300 Deg F.

The LM35 temperature sensor measures temperatures in the range of –55 Deg C to +155 Deg C.

The LM34 and LM35 sensors do not have facilities to directly adjust calibration accuracy. These sensors are available in several grades of different accuracy, varying from ±0.2 Degrees to ±2.0 Degrees.

For detailed specifications of the LM34 and LM35 series of *IC* temperature sensors, refer to the *National Semiconductor Linear Supplement (1984)*.

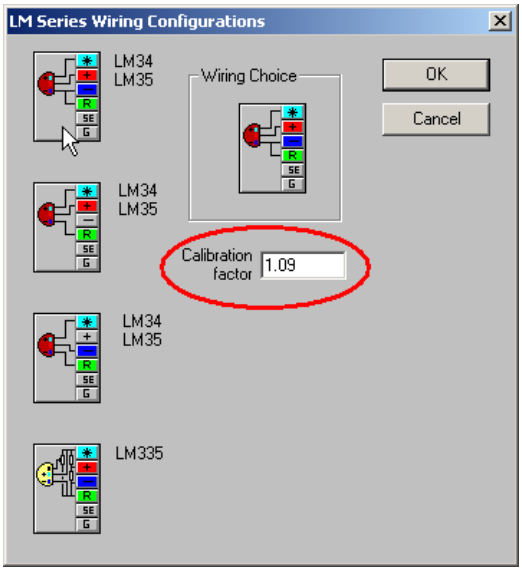
A calibration factor to correct sensor accuracy can be specified as a channel option.

Using DeTransfer, the command for example

```
2LM35(1.09)
```

instructs the *dataTaker* to apply a calibration factor of 1.09 to the LM35 sensor measured on channel 2.

Using DeLogger, the calibration factor can be entered into the LM Series Wiring Configuration which opens when you have selected the analog channel



The LM34 and LM35 temperature sensors are supported by the *dataTaker* as follows

powered from the Excite terminal of the analog channels, or be externally powered

voltage signal can be connected to analog channels as differential or single ended

These options provide a variety of ways in which the LM34 and LM35 temperature sensors can be connected to the *dataTaker*. The most commonly used connections are described in the sections below.

LM34 and LM35 Directly Powered

LM34 and LM35 temperature sensors can be powered from the Excite terminal, and the output voltage measured either as differential or single ended inputs. Single ended inputs can be referenced to Analog Return or external common connected to SE Ref.

However the temperature measurement range for the LM34 and LM35 sensors is limited to approximately 5 Deg F and 10 Deg C respectively by this configuration, due to a lack of pull down voltage capacity.

Differential Input

LM34 and LM35 temperature sensors can be powered directly from the Excite terminal, and the output voltage from the sensors measured as differential inputs.

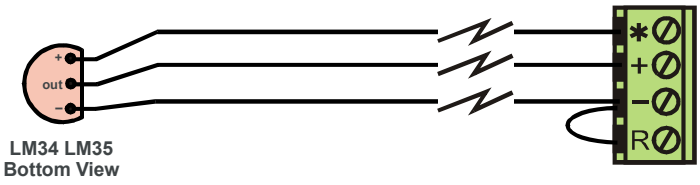


Figure 91 – Connection of LM34 and LM35 Sensors Powered from the Excite Terminal as Differential Inputs

The link between Analog Return and the –ve terminal can be replaced by a fourth lead wire connected between the sensor and the –ve terminal, to improve accuracy.

LM34 and LM35 temperature sensors powered directly from the Excite terminals and measured as differential inputs are sampled and data returned when a Schedule containing the channel is executed.

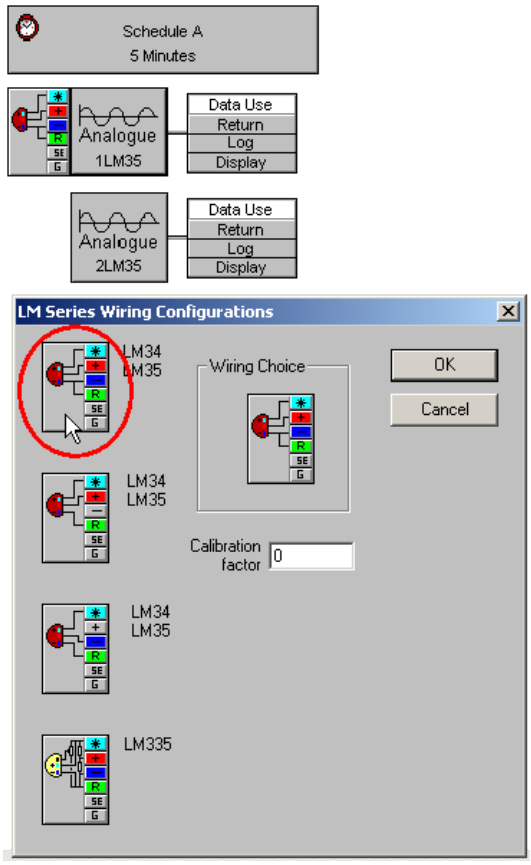
Using DeTransfer, the command for example

```
BEGIN
 RA5M
  1..2LM35
END
```

instructs the *dataTaker* to measure the voltage from LM35 sensors connected to channels 1 and 2.

The LM35 specifies the sensor attached to these channels. The LM35 channel type defaults to Excite terminal powering, and so powering does not have to be specified.

Using DeLogger,



LM34 and LM35 temperature sensors powered from the Excite terminal and connected as differential inputs are measured by the Program Builder program below.

The differential connection is selected from the LM Series Wiring Configurations dialog which opens when you have selected the analog input channel.

The data is returned in units of temperature. The *dataTaker* will read the sensors every 5 minutes, and readings are stopped by entering a H (Halt) command.

Single Ended Inputs Referenced to Analog Return

LM34 and LM35 temperature sensors can be powered directly from the Excite terminal of the *dataTaker*, and the output voltage from the sensors measured as single ended inputs referenced to Analog Return.

The connection is illustrated below

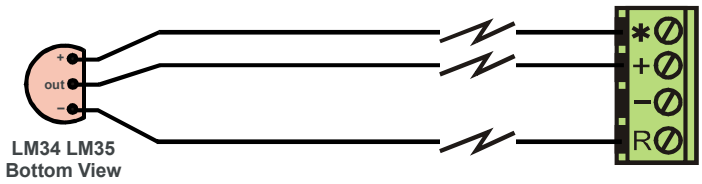


Figure 92 – Connection of LM34 and LM35 Sensors Powered from the Excite Terminal as Single Ended Inputs Referenced to Analog Return

LM34 and LM35 temperature sensors powered directly from the Excite terminals and measured as single ended inputs referenced to Analog Return are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  1+..2-LM35
END
```

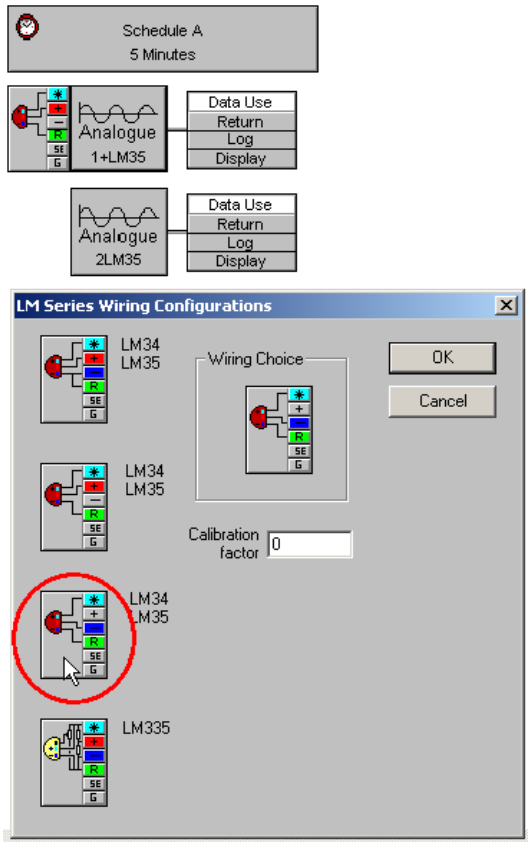
instructs the *dataTaker* to measure the voltage signals from LM35 sensors differentially connected to channels 1+ through 2–.

The LM35 specifies that the signals applied to these channels are from LM35 sensors respectively.

The LM35 channel type defaults to Excite terminal powering, and so powering does not have to be specified.

Using DeLogger, LM34 and LM35 sensors powered from the Excite terminal and connected as single ended inputs referenced to Analog Common are measured by the following Program Builder program.

The single ended connection is selected from the LM Series Wiring Configurations dialog which opens when you have selected the analog input channel.



The data is returned in units of temperature.

The *dataTaker* will read the sensors every 5 minutes, and readings are stopped by entering a H (Halt) command.

Single Ended Inputs Referenced to an External Common

LM34 and LM35 temperature sensors can be powered directly from the Excite terminal of the *dataTaker*, and the output voltage from the sensors measured as single ended inputs referenced to an external common connected to SE REF.

The connection is as follows

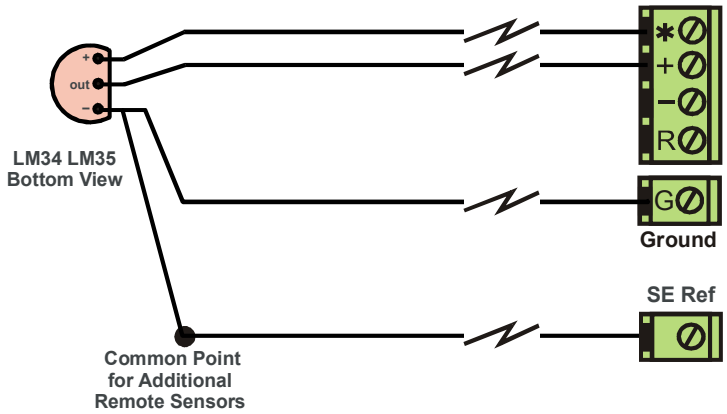


Figure 93 – Connection of LM34 and LM35 Sensors Powered from the Excite Terminal as Single Ended Inputs Referenced to an External Common

LM34 and LM35 sensors connected as single ended inputs referenced to an external common, and powered from the Excite terminal, are sampled and data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA5M
  1+..2-LM35(X)
END
```

instructs the *dataTaker* to measure the voltage signals from LM35 sensors differentially connected to channels 1+ through 2–.

The LM35 specifies the signals are from LM35 sensors. The LM35 channel type defaults to Excite terminal powering, and so powering does not have to be specified. The X channel option indicates that the single ended inputs are to be measured between the input terminal and SE REF.

The data is returned in units of temperature. The *dataTaker* will read the sensors every 5 minutes, and readings are stopped by entering a H (Halt) command.

DeLogger does not support this method of measuring LM34 and LM335 IC temperature sensors. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

LM34 and LM35 Externally Powered

The LM34 and LM35 temperature sensors can be externally powered from any 4 to 30 Volt power supply, and the output voltage measured as differential inputs, or as single ended inputs referenced to Analog Return or to an external common connected to SE REF.

The power supply can be an external power source, or can be from the 5 Volt switched sensor power supply of the *dataTaker*. The LM34 and LM35 temperature sensors cannot drive voltages near or below their ground pin (corresponding to 0 Deg F for the LM34 and –18 Deg C for the LM35).

Therefore the pull down arrangement shown in Figure 94 must be used. This functions by elevating the negative pin voltage by the two diode drops (approximately 1.2 Volts), and pulling the output below the negative pin when necessary.

Differential Input

The LM34 and LM35 temperature sensors can be externally powered from any 4 to 30 Volt power supply, and the output voltage measured as differential inputs.

The connection is illustrated below

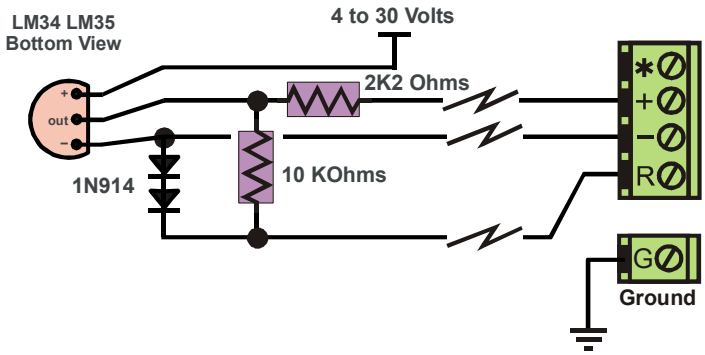


Figure 94 – Connection of Externally Powered LM34 and LM35 Sensors as Differential Inputs

Note : The circuit also shows a 2K2 series resistor, which may be necessary on long cable runs to prevent oscillation.

Externally powered LM34 or LM35 temperature sensors which are connected as differential inputs are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10S
  3LM34(N)..5LM34(N)
END
```

instructs the *dataTaker* to measure the voltage signals from the LM34 sensors differentially connected to channel 3 and channel 5.

The LM34 specifies that the signals applied to these channels are from LM34 sensors respectively, and the data is returned in units of temperature. The LM34 channel type defaults to Excite terminal powering, and so the channel option N should be specified to disable output from the Excite terminal.

The *dataTaker* will read the sensors every 10 seconds, and readings are stopped by entering a H (Halt) command.

DeLogger does not support this method of measuring LM34 and LM335 IC temperature sensors. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

Single Ended Inputs Referenced to an External Common

Remotely located LM34 and LM35 temperature sensors which are powered from an external power supply can be measured as single ended inputs referenced to an external common.

The grounded ends of the LM34 and LM35 temperature sensor voltage signals are connected to a common external reference point, which is in turn connected to the SE REF terminal.

The connection of externally powered LM34 and LM35 sensors as single ended inputs referenced to an external common is illustrated below

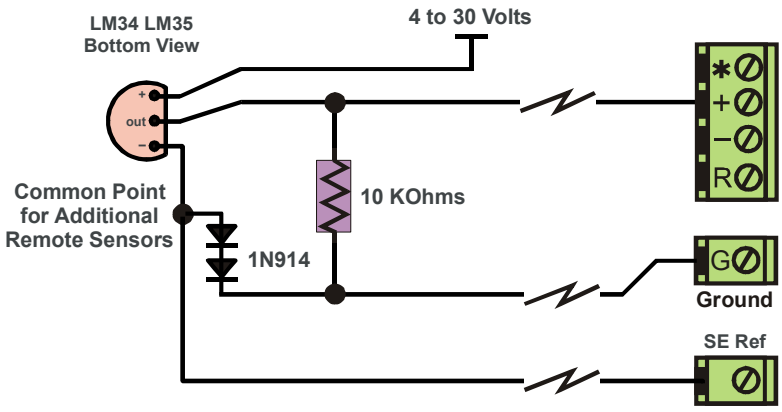


Figure 95 – Connection of Externally Powered LM34 and LM35 Sensors as Single Ended Inputs Referenced to an External Common

Externally powered LM34 or LM35 temperature sensors which are connected as single ended inputs referenced to an external common are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA30M
  5+LM35(N,X)
END
```

instructs the *dataTaker* to measure the voltage signal from the LM35 sensor connected between channels 5+ and an external common reference connected to SE REF.

The LM35 specifies that the signals applied to these channels are from LM35 sensors respectively. The LM35 channel type defaults to Excite terminal powering, and so the channel option N should be specified to disable output from the Excite terminal. The X channel option indicates that the single ended inputs are to be measured between the input terminal and SE REF.

The data is returned in units of temperature. The *dataTaker* will read the sensors every 30 minutes, and readings are stopped by entering a H (Halt) command.

DeLogger does not support this method of measuring LM34 and LM335 IC temperature sensors. However the User channel type (DeLogger Version 4.2.15 or later) can be used, and the channel specifications shown above for DeTransfer can be entered.

Error Messages

There are no specific error messages for the IC temperature sensors supported by the *dataTaker*. Nonetheless, temperature readings outside of the temperature range of –55 Deg C to +150 Deg C can be considered suspect.

Input voltage signals which fall outside the range of ±3250 mV, or current signals which fall outside the range of ±25 mA, will produce the error message 'E11–input(s) out of range' if the Messages Switch /M is enabled.

Measuring Temperature with Thermistors

The *dataTaker* data loggers support several types of the Yellow Springs Instrument Inc. range of thermistors.

The thermistor is a thermo-resistive device, which changes resistance with changes in temperature. The thermistors have a negative temperature coefficient, such that the resistance of the device decreases as temperature increases.

The *dataTaker* accurately measures the changes in resistance of the element, and linearizes the measured resistance to units of temperature.

Thermistor Support by the *dataTaker*

The *dataTaker* supports a number thermistors manufactured by Yellow Springs Instrument Inc. The YSI thermistors supported are essentially two wire types, which have a nominal resistance of 10 KOhm or less at 25 Deg C.

The YSI Thermilinear range of thermistors is not supported by the *dataTaker*, because of increased wiring complexity and the requirement for a precision resistor network. However because of their linear response, these thermistors can be supported by the polynomials or span limits (See Section III – Scaling Data, Polynomials, Spans and Functions).

There is no direct support provided for thermistors from other manufacturers. However other thermistors can be used with the *dataTaker* if the appropriate polynomials are available, and applied to the thermistor readings in the normal manner (See Section III – Scaling Data, Polynomials, Spans and Functions).

The *dataTaker* measures YSI thermistors directly as resistive devices. The resistance of the thermistor element is measured by any of the 2 wire, 3 wire or 4 wire resistance measurement methods (See Section II – Measuring Resistance).

The lower resistance types of thermistors, and those being used near the upper end of their respective temperature range (small resistances), will generally benefit from the 4 wire and 3 wire methods of resistance measurement. This is particularly true if long cables are being used.

However because thermistors have a relatively high resistance and sensitivity, satisfactory measurements can also be made using the 2 wire method of resistance measurement. This is because the resistance of the thermistor is generally very much greater than the resistance of the cables, particularly if shorter cables are used.

The resistance of the thermistor element is measured, and then linearized to units of temperature using internal linearization functions.

Minimum Temperature Measured

Thermistors have a negative temperature coefficient, in that the resistance of the device decreases as temperature increases. The resistance of thermistors is highest at lower temperatures.

Therefore the minimum temperatures which can be measured using thermistors with the *dataTaker* data loggers is limited by the maximum resistance that the logger can measure. The *dataTaker* can measure resistance to approximately 7000 Ohm (See Section II – Measuring Resistance).

This maximum resistance that can be measured translates to a different minimum temperature for the different types of YSI thermistors supported. The minimum temperatures which can be measured using the *dataTaker* for the different types of YSI thermistors supported are summarized in the table overleaf.

In contrast the maximum temperatures that can be measured using the YSI thermistors with the *dataTaker* are limited only by the manufacturers specification.

Extending the Minimum Temperature

The YSI thermistors which exceed the resistance measurement limit of approximately 7000 Ohm for the *dataTaker*, can however be supported by placing a 'linearizing' resistor in parallel with the thermistor. This fixed resistor reduces the total resistance from the thermistor, and allows the *dataTaker* to measure lower temperatures.

The connection of this optional parallel resistor is illustrated below

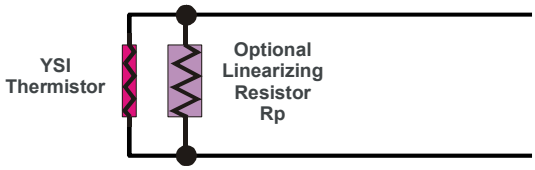


Figure 96 – Connection of Optional Parallel Resistor Rp

This fixed resistor can be located at the sensor, or be installed across the analog input channel screw terminals of the *dataTaker*.

The optimal value for the linearizing resistor is approximately equal to the resistance of the thermistor in the middle of the expected temperature range to be measured. However any value is allowable.

If the resistor value is too high, it will have little effect. If the resistor value is too low, it will reduce the sensitivity of the thermistor at lower temperatures.

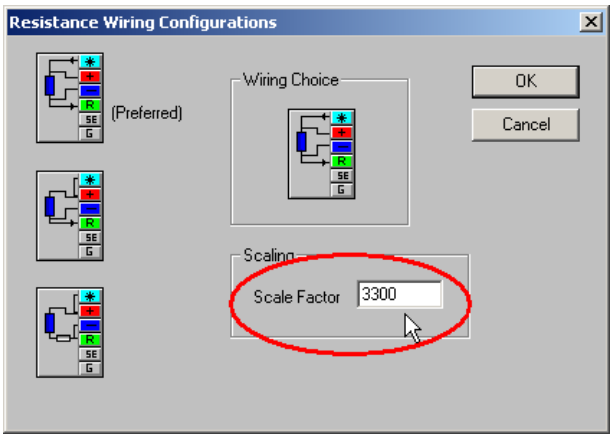
When a linearizing resistor is used to extend the minimum temperature measurement capability for thermistors, then the value of the resistor is declared as a scaling factor to the analog signal type.

Using DeTransfer, the command for example

```
2YS05(3300.0)
```

specifies that a YSI type 5 thermistor (see table below) is connected to analog input channel 2, which has a 3300 Ohm linearizing resistor Rp connected in parallel.

Using DeLogger, resistance of the linearizing resistor is specified as the Scale Factor in the Resistance Wiring Configurations dialog, which opens after you select the YS temperature sensor type



YSI Thermistor Types

The YSI thermistor types supported by the *dataTaker* are listed in the table on the next page. The YSI thermistor types have been grouped into nine different analog channel types according to thermistor sensitivity, and the minimum measurable temperature.

When any of these YSI thermistor types are used, they are specified to the *dataTaker* by the analog channel type identifier. For example the YSI thermistor types 44003A, 44035 and 44101A are all identified to the *dataTaker* as analog input type YS03.

The table also lists

the minimum temperature that can be measured for each thermistor type with the *dataTaker* when no linearizing resistor is used

the maximum temperature that can be measured with the *dataTaker*

The *dataTaker* measures YSI thermistors by any of the 4 wire, 3 wire or 2 wire methods of resistance measurement. The method of resistance measurement recommended for each of the YSI thermistor types are also listed in the table.

Excitation Current

The *dataTaker* outputs a precise constant current from the Excite terminal of the analog channel during measurement of a resistance. This excitation or measurement current is pulsed on for 30 mS during the measurement of thermistors.

The thermistors are connected between the Excite terminal of the analog input channels and the Analog Return terminals, such that this excitation current passes through the sensor element.

The voltage produced across the thermistor element by the excitation current is then connected by various methods to the analog input channels. The measured voltage is used to calculate the value of the resistance of the thermistor element.

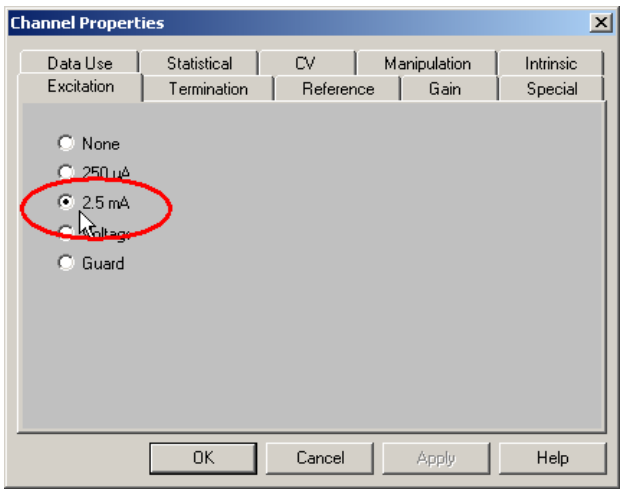
The *dataTaker* can output a 250.0 µA or a 2.500 mA excitation current from the Excite terminal (See Section II – Measuring Resistance). The 250.0 µA excitation current is used by default for thermistors, although the 2.500 mA excitation current can be used.

While the temperature resolution obtained for the two currents is generally the same, the lower current is preferred because of potential errors due to self heating at the higher current.

Using DeTransfer, the channel specification for example

```
2YS01(II)
```

specifies that the thermistor is to be measured using a 2.500 mA excitation current from the Excite terminal, as defined by the ***II*** channel option



Using DeLogger, the excitation current can be specified in the Channel Properties dialog which opens when you select Channel Options:Excitation… as shown above

The 250.0 µA excitation current is selected by default whenever a thermistor input type is specified, and does not need to be specifically selected.

Yellow Springs Inc Thermistors

Yellow Springs Inc Thermistors					
dataTaker Channel Type	YSI Sensor Catalogue Number		Minimum Temp (no Rp) Deg C	Maximum Temp Deg C	Preferred Measuring Method
YS01	44001A		–65	100	3, 4 wire
YS02	44002A	44102A	–45	100	3, 4 wire
YS03	44003A 44035	44101A	–20 –20	100	3, 4 wire
YS04	44004 44033 45004 46033 44901 44902	44104 46005 46040	1 1 1 1 1 1	150 75 200 200 90 70	2, 3, 4 wire 2, 3, 4 wire 3, 4 wire 3, 4 wire 2, 3, 4 wire 2, 3, 4 wire
YS05	44005 44030 45005 46030 44903 44904	44105 46005 46040	7 7 7 7 7 7	150 75 200 200 90 70	2, 3, 4 wire 2, 3, 4 wire 3, 4 wire 3, 4 wire 2, 3, 4 wire 2, 3, 4 wire
YS07	44007 44034 45007 46034 44905 44906	44107 46007 46044	18 18 18 18 18 18	150 75 200 200 90 70	2, 3, 4 wire 2, 3, 4 wire 3, 4 wire 3, 4 wire 2, 3, 4 wire 2, 3, 4 wire
YS17	44017 45017 46017 46037	 46047	22 22 22 22	150 250 200 200	2, 3, 4 wire 3, 4 wire 3, 4 wire 3, 4 wire
YS16	44016 44036 46036		34 34 34	150 75 200	3, 4 wire 2, 3, 4 wire 3, 4 wire
YS06	44006 44031 45006 46006 46031 44907 44908	44106 46041	35 35 35 35 35 35 35	150 75 250 200 200 90 70	3, 4 wire 2, 3, 4 wire 3, 4 wire 3, 4 wire 3, 4 wire 2, 3, 4 wire 2, 3, 4 wire

Connection and Measurement of Thermistors

YSI thermistors can be connected to the analog input channels by three similar methods, which provide for varying degrees of compensation for cable wire resistance.

The three connection methods are the 4 wire, 3 wire and 2 wire methods of resistance measurement, and are described in the following sections.

These thermistor measurement methods use all four terminals of the analog input channels. Therefore a total of 5 thermistors can be connected to the *dataTaker 50*, and a total of 10 thermistors can be connected to the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD).

Four Wire Thermistor Measurement

The 4 wire method of thermistor measurement is the most accurate, because there is effectively no current flowing in either of the measurement cable wires and therefore no added resistance due to the cable wires.

This method has two cable wires are connected to each end of the thermistor. One pair of cable wires carries the excitation current, and the other pair of cable wires is used to measure the voltage across the unknown resistance.

This method of thermistor measurement should be used where cables are necessarily long or of unequal resistance.

During resistance measurement of the thermistor, the excitation current output from the Excite terminal of the analog channel passes through the thermistor and cable wires, and returns into the Analog Return terminal.

The voltage produced across the thermistor as a result of the excitation current is measured differentially between the +ve and –ve terminals.

Thermistor are connected to the analog input channels for 4 wire resistance measurement as follows

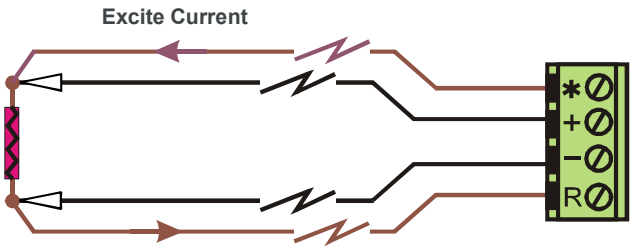


Figure 97 – Connection for Four Wire Thermistor Measurement

The excitation current is flowing in the excitation circuit (shown above by the red path), which is totally separate from the measurement circuit. Because of the high input impedance of the analog to digital converter, there is negligible current flow in the measurement circuit. Therefore there is a negligible cable wire resistance component in the voltage measured across the thermistor.

Each thermistor measured by the four wire method requires all terminals of the analog input channel. Therefore a maximum of 5 thermistor can be measured using this method by the *dataTaker 50*, and a maximum of 10 thermistor can be measured using this method by the *dataTaker 500/600* series loggers.

Thermistors connected to the analog channels using the 4 wire method are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1YS01(4W)   2YS03(2200.0,4W)
END
```

instructs the *dataTaker* to measure the thermistor connected to each analog input channel as follows

a YSI type 44001A thermistor connected to analog channel 1

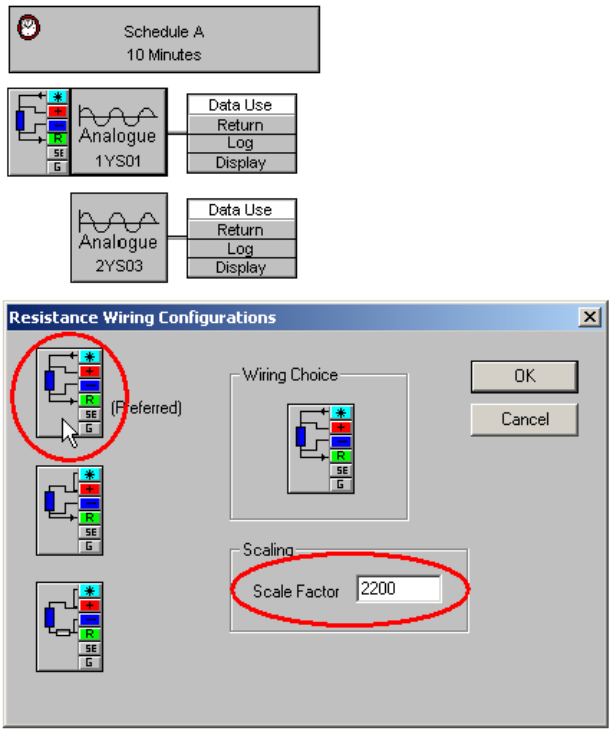
a YSI type 44035 thermistor connected to analog channel 2. This thermistor has a 2200 Ohm resistor connected in parallel across it

The excitation current option is not specified, and so the default 250.0 µA excitation current is used.

The YS01 and YS03 specifies that the signals applied to these channels are from YSI thermistors. The 4W channel option indicates that the 4 wire method of resistance measurement is to be used.

Using DeLogger, thermistors connected by the 4 wire method can be measured by the following Program Builder program.

The 4 wire connections are selected from the Resistance Wiring Configurations dialog which opens when you have selected the analog input channel.



The measured resistance is linearized, and data is returned in units of temperature.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Three Wire Thermistor Measurement

Most thermistor measurements are made using the 3 wire resistance measurement method, which compensates for cable wire resistances in the measurement circuit (assuming that the two current carrying wires are of the same resistance).

In this method one cable wire is connected to the upper end of the thermistor, and two cable wires are connected to the lower end. The third cable wire is used to compensate for cable wire resistance.

Thermistors are connected to the analog input channels for 3 wire measurement as follows

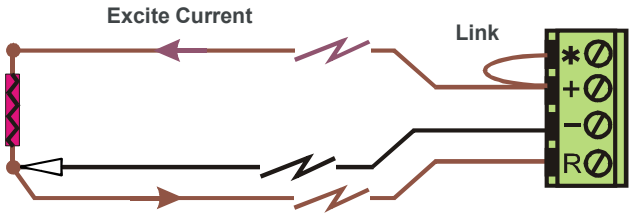


Figure 98 – Connection for Three Wire Thermistor Measurement

During resistance measurement, the excitation current output from the Excite terminal of the analog channel passes through the thermistor and cable wires, and returns into the Analog Return terminal (shown above by the red path).

The voltage produced across the thermistor and the cable wire resistance as a result of the excitation current is measured between the +ve and –ve terminals.

Since all cable wires have resistance, the voltage produced across these is included in the total voltage drop measured. This introduces resistance offset errors, especially when long cable wires are used.

There is a much higher impedance for the –ve input terminal than for the Analog Return terminal, causing the excitation current to return via the cable wire connected to Analog Return. As a result there is no appreciable current flowing the third cable wire connected to the –ve terminal.

The third cable wire is used to accurately measure the voltage drop due to the resistance of the return cable wire in the resistance measurement circuit. The *dataTaker* doubles this voltage and subtracts it from the voltage measured across the thermistor and the cable wires, before calculating the temperature.

This technique for correcting for cable wire resistance assumes that the two current carrying cable wires are of equal resistance. This in turn implies that cables used for connecting thermistors to the *dataTaker* should be of equal length, gauge and type of wire.

Up to 5 thermistors can be measured using the 3 wire method by the *dataTaker 50*, and up to 10 thermistors can be measured by the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD).

Thermistors connected using the 3 wire method are sampled and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  3..4YS06
END
```

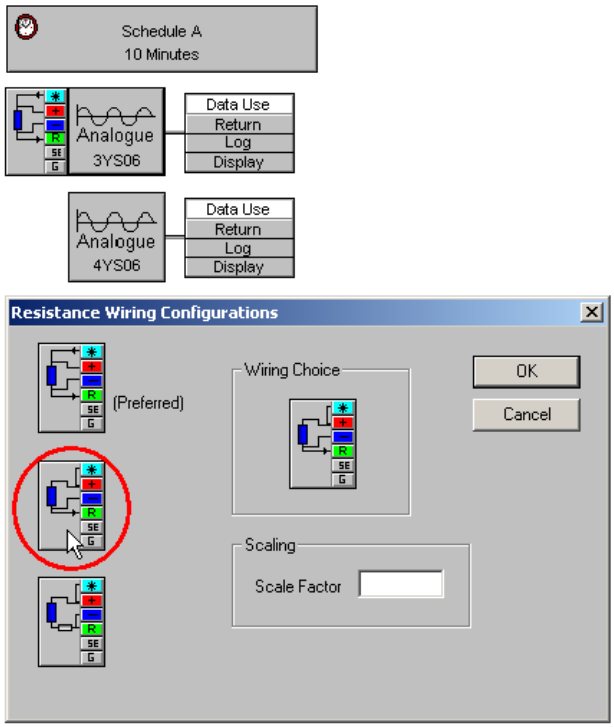
instructs the *dataTaker* to measure the two YSI thermistors connected to analog channels 3 and 4.

The excitation current option is not specified, and so the default 250.0 µA excitation current is used.

The YS06 specifies that the signal applied to these channels is from YSI thermistors.

Using DeLogger, Thermistors connected by the 3 wire method can be measured by the following Program Builder program.

The 3 wire connections are selected from the Resistance Wiring Configurations dialog which opens when you have selected the analog input channel.



The measured resistance is linearized and data is returned in units of temperature.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Two Wire Thermistor Measurement

The 2 wire thermistor measurement method is the simplest method for measuring thermistors with the *dataTaker*. The 2 wire method only requires a single pair cable to connect the thermistor to the logger. This cable pair carries both the excitation current and the signal current.

However this method also has the disadvantage of including the effects of cable wire resistances in the measured temperature.

Thermistors are connected to the analog input channels for 2 wire measurement by installing links between Excite and +ve terminals, and between Analog Return and –ve terminal. The two cable wires connect the thermistor to the +ve and –ve terminals.

During measurement, the excitation current passes from the Excite terminal through the thermistor and cable wires, and returns into the Analog Return terminal. The voltage produced across the thermistor and the cable wire resistances is measured between the +ve and –ve terminals.

This configuration is only recommended for measuring thermistors with resistances greater than 500 Ohm.

Thermistors are connected to analog input channels for 2 wire measurement as follows

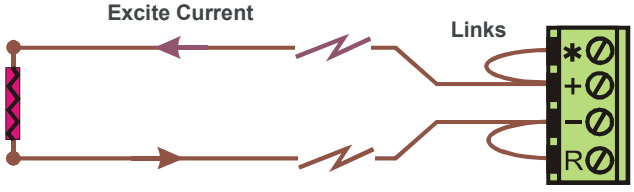


Figure 99a – Connection for Two Wire Thermistor Measurement

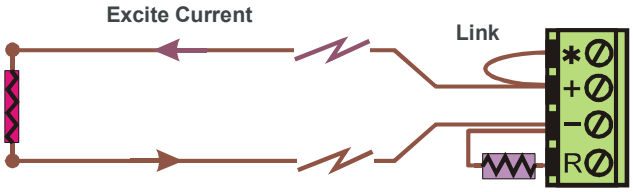


Figure 99b – Connection for Two Wire Thermistor Measurement

Cable wire compensation is performed during measurement of the thermistor, but is of little significance since the resistance of the links is very much less than that of the cable wires.

Since the resistance of the cable wires connecting the thermistor to the analog input channel are included in the measurement, the two wire resistance measurement method is best used where the thermistor resistance is large relative to the resistance of the cable wires.

Cable wire compensation is performed during measurement of the RTD, but is of little significance since the resistance of the links is very much less than that of the cable wires (Figure 99a).

Where cables with only two cable wires are already installed and must be used, cable wire resistance compensation can be obtained by replacing the link between the –ve terminal and Analog Return with a resistor (Figure 99b) of value equal to that of the total cable resistance.

However using a compensation resistor does not compensate for temperature effects on the cables.

Alternatively the error produced by the constant cable wire resistance could be corrected using a Polynomial.

Up to 5 thermistors can be measured using the 2 wire method by the *dataTaker 50*, and up to 10 thermistors can be measured by the *dataTaker 500/600* series loggers and the Channel Expansion Module (CEM-AD) .

Thermistors connected using the 2 wire method are sampled and the data is returned when a Schedule is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1..2YS04
END
```

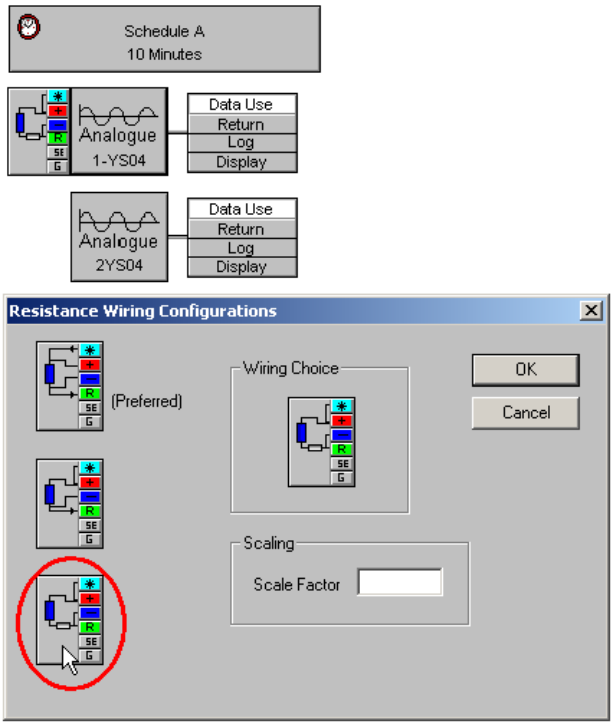
instructs the *dataTaker* to measure the three YSI thermistors connected to analog channels 1 and 2.

The excitation current option is not specified, and so the default 250.0 µA excitation current is used.

The YS03 specifies that the signal applied to these channels is from YSI thermistors

Using DeLogger, thermistors connected by the 2 wire method can be measured by the following Program Builder program.

The 2 wire connections are selected from the Resistance Wiring Configurations dialog which opens when you have selected the analog input channel.



The measured resistance is linearized, and data is returned in units of temperature.

The *dataTaker* reads the sensors every 10 minutes, and readings are stopped by a H (Halt) command.

Measurement Ranges and Accuracy

The measurement accuracy for thermistors is the same as that for standard resistance measurement, and is discussed in Section II – Resistance Measurement.

Greater accuracy of thermistor measurement is achieved when 3 or 4 wire methods are used, and the 250.0 µA thermistor excitation current is used.

Sources of Error

There are three sources of error in thermistor measurements by the *dataTaker*.

The first is due to errors in resistance measurement (excluding cable wire effects), which are ±0.1% over the logger temperature range of 10 to 40 Deg C. This systematic error can however be calibrated out.

The second source of error is due to self heating of thermistors during measurement. The 250.0 µA measurement current will heat the thermistor bead. The worst case power that can be dissipated in the thermistor is approximately 450µW (a 7 KOhm thermistor without a parallel resistor). In open air this can result in a 0.45 Deg C error if the thermistor is continually excited. In practice a thermistor is normally excited for 30 mS each reading, resulting in an error of less than 0.007 Deg C in still air.

The third source of error is linearization error. The *dataTaker* uses the Steinhart and Hart equation recommended by YSI, with coefficients as derived from resistances measured at 0, 50 and 100 Deg C. Over the –20 to +120 Deg C range the deviation is quoted as +0.83, –0.067 Deg C. The higher value only occurs at the lower extreme.

Error Messages

Thermistor resistances which fall outside of the linearization range of –80 to 275 Deg C are returned as -99999.9 Deg C or +99999.9 Deg C.

When this occurs, the *dataTaker* reports the error condition by returning the error message 'E16 - linearization error' if the Messages Switch /M is enabled

Measuring Bridges and Strain Gauges

The *dataTaker* provides comprehensive support for Wheatstone bridge circuits in full, half and quarter bridge configurations.

Wheatstone bridge circuits are extensively used for measuring the output of strain gauges, and for measuring the output of other sensors where a relatively small change in resistance must be detected.

Bridge circuits have the advantage of high measurement sensitivity, and also provide a significant degree of temperature compensation.

The *dataTaker* supports two methods of Wheatstone bridge measurement

- excitation of sensors is by a constant current
- excitation of sensors is by a voltage

Both of these methods of bridge support can have a number of options, depending the number of active arms in the bridge and the number of wires used to connect bridges to the *dataTaker*.

Constant Current Excitation of Bridges

The constant current excitation method of bridge measurement has a constant current of 2.500 mA or 250.0 µA flowing in the bridge circuit for excitation.

The bridge sensitivity and zero is independent of the length of leads used to connect the bridge to the *dataTaker*.

In some cases the bridge output can have greater linearity and reduced temperature sensitivity for constant current excitation, than for voltage excitation.

The Bridge Excitation Current

The bridge excitation current is supplied by the Excite terminal of the analog input channel during measurement. The bridge excitation current is 2.500 mA by default, but may be set for 250.0 µA if required.

If the 250.0 µA excitation current is required, then this is specified as a channel option in the channel specification.

Using DeTransfer, 250.0 µA excitation current is specified as follows

```
5BGI(I)
```

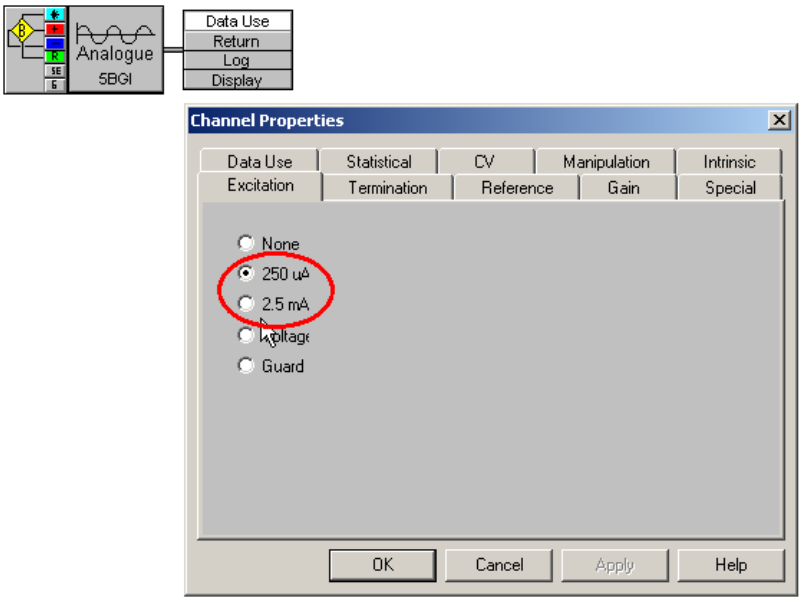
where the channel option I specifies that a 250.0 µA excitation current is output from the Excite terminal during bridge measurement.

The default excitation current of 2.500 mA is equivalent to

```
5BGI(II)
```

where the channel option II specifies that a 2.500 mA excitation current is output from the Excite terminal during bridge measurement.

Using DeLogger, the excitation current can be selected in the Channel Properties dialog box of the Program Builder. When the bridge channel has been created, right click on the Data Use icon and select Properties... Click on the Excitation tab, and select the excitation current required.



Both the 250.0 µA and 2.500 mA excitation currents can be used within the same application, for different bridges connected to different channels.

The bridge excitation current is supplied from the Excite terminal for a period of 30 mS during measurement of the bridge circuit.

The Arm Resistance

The data that is returned by the *dataTaker* from bridge circuits excited by constant current is the ratio of the change in arm resistance to the nominated arm resistance. The data is returned in units of ppm.

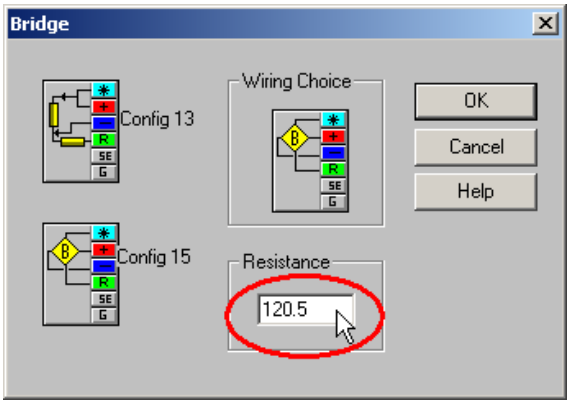
The arm resistance for the bridge being measured must be known by the *dataTaker*, and is specified as a channel option in the channel specification. The default arm resistance is 350 Ohm, which is typical for many types of strain gauges.

Using DeTransfer, the arm resistance is specified as follows

```
5BGI(120.5)
```

In this example the bridge circuit connected to analog channel 5 has an arm resistance of 120.5 Ohm. The arm resistance is specified in Ohms.

Using DeLogger, the arm resistance for constant current excited bridges can be defined in the Resistance Wiring Configurations dialog of the Program Builder, which opens when you select the Bridge input channel type.



Full Bridge with Constant Current Excitation

The *dataTaker* can provide excitation and measure the output from a full bridge of devices such as strain gauges, pressure cells, load cells, etc. This configuration is a 4 wire input, and supports 1, 2 or 4 active arms. Any of the bridge arms can be active arms. The configuration also provides compensation for cable wire resistance, allowing long cable wires to be used.

Bridge arms which are not active must have bridge completion resistances. These can be an inactive device of the same type as the devices on the active arms, or these can be a resistor with the same resistance value as the active devices at rest, and ideally have a temperature coefficient that is similar to that of the active devices.

The entire bridge circuit is external to the *dataTaker* – the logger does not provide any bridge completion for partial bridges.

The full bridge configuration with constant current excitation is connected to the *dataTaker* as a 4 wire input as follows

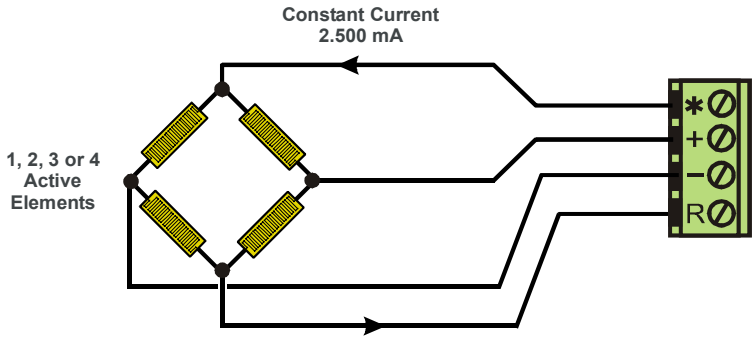


Figure 100 –Full Bridge with Constant Current Excitation

where the Excite terminal provides current excitation of 250.0 µA and 2.500 mA, which returns via the Analog Return terminal. The bridge output is read between the +ve and – ve terminals.

Full bridges with constant current excitation are sampled, and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, full bridges with constant current excitation are measured by the command for example

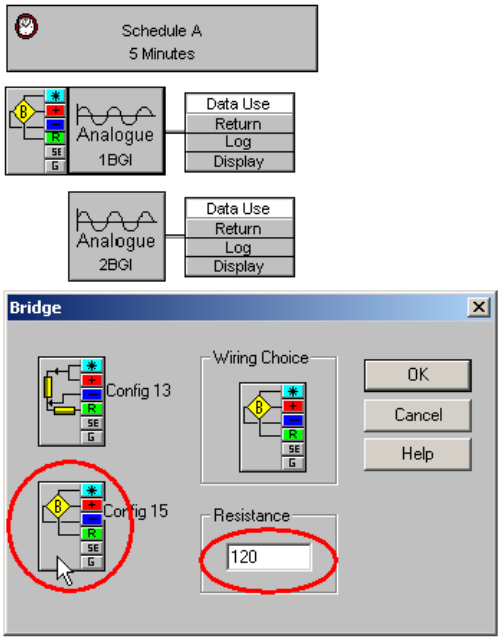
```
BEGIN
 RA5M
  1BGI(4W,120)   2BGI(4W,120)
END
```

which instructs the *dataTaker* to measure the output from full bridges connected to analog input channels 1 and 2. The full bridge circuits have an arm resistance of 120.0 Ohm.

The BGI specifies that the signals applied to these channels are from constant current excited bridge circuits. The 4W channel option indicates that the bridge is connected in a 4 wire configuration, where the excite terminal provides excitation. This option must be specified for all four wire bridge connections.

The excitation current channel option is not specified, and so the default 2.500 mA excitation current is used.

Using DeLogger, full bridges with constant current excitation can be measured by the following Program Builder program. The 4 wire connection is selected from the Bridge Wiring Configurations dialog which opens when you have selected the analog input channel.



The *dataTaker* will read the inputs every 5 minutes, and readings are stopped by entering a H (Halt) command.

Interpreting the Data from a Full Bridge with Current Excitation

The data returned from full bridges is the ratio of change in measured resistance to the arm resistance, expressed in parts per million as follows

$$\text{Reading} = \frac{(\Delta R * 10^6)}{R_{arm}} \quad \text{ppm}$$

where

^{a}R is the sum of changes in arm resistances taking into account sign of the changes

R_{arm} is the nominated arm resistance, default is 350 Ohm

Calculating Microstrain for Full Strain Gauge Bridges

When using stain gauges in full bridges, it may be desirable to convert the returned data from units of ppm to units of Microstrain. This can be done by the following formula

$$\text{Microstrain} = \left(\frac{4}{\text{No.Gauges} * \text{Gauge Factor}}\right) * \text{ppm}$$

This full bridge method of strain gauge measurement has a measurement resolution of approximately 0.2 Microstrain.

Using DeTransfer, the output from a full bridge with constant current excitation can be calculated to units of microstrain by the program for example

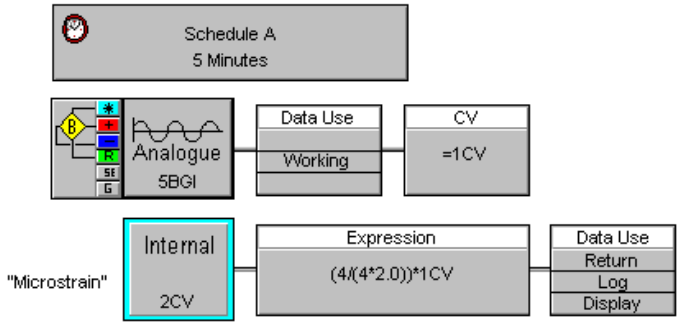
```
BEGIN
 RA5M
  5BGI(4W,120,=1CV,W)
  2CV("Microstrain =")=(4/(4*2.0))*1CV
END
```

which instructs the *dataTaker* to firstly read the bridge output in ppm, and save this in Channel Variable 1 (1CV), then calculate microstrain from the reading using the formula above.

Here it is assumed that the gauge factor is 2.0 for the strain gauges used (check your strain gauge supplier or manufacturer for details of the gauge factor).

The use of calculations in the *dataTaker* are discussed in detail in Section III – Channel Variables and Calculations.

Using DeLogger, the calculation can be entered in the Program Builder as follows



Refer to your DeLogger Manual for details of using calculations in the Program Builder.

Half Bridge with Constant Current Excitation

The *dataTaker* can also provide excitation and measure the output from a half bridge configuration of strain gauges, pressure cells, etc. This configuration is a 3 wire input, and supports 2 active arms.

This configuration compensates for cable wire resistance and temperature difference.

The half bridge configuration with constant current excitation is connected to the *dataTaker* as a 3 wire input as follows

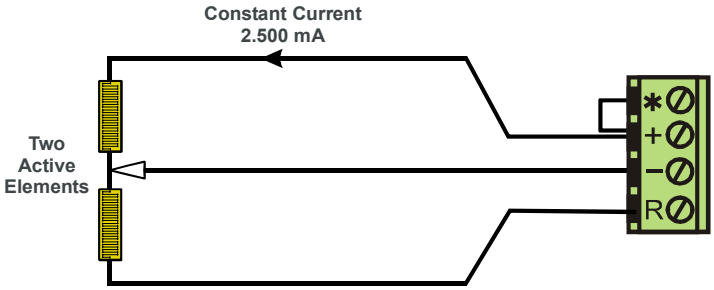


Figure 101 – Half Bridge with Constant Current Excitation and Two Active Arms

When using stain gauges in half bridges and current, it may be desirable to convert the returned data from units of ppm to units of Microstrain. This can be done by the following formula

$$Microstrain = \left(\frac{2}{No.Gauges * Gauge\ Factor}\right) * ppm$$

This bridge configuration can be used over a wide variation of resistance.

The half bridge configuration can be used to measure the position of the wiper of a potentiometer (<5 KOhm) - the ends of the potentiometer are connected between Excite / +ve terminals and Analog Return, and the wiper is connected to the – ve terminal. The arm resistance is set to the total resistance of the potentiometer.

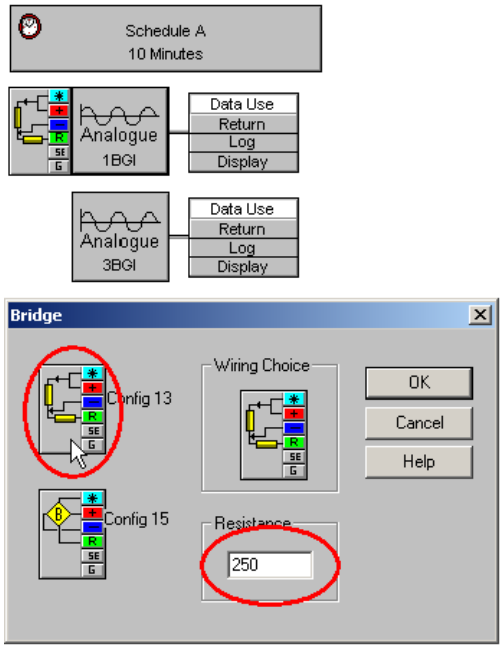
Using DeTransfer, half bridges with constant current excitation can be measured by the command for example

```
BEGIN
 RA10M
  1BGI(250)   3BGI(250)
END
```

which instructs the *dataTaker* to measure the half bridges connected to analog input channels 1 and 3. These half bridge circuits all have an arm resistance of 250.0 Ohm.

The BGI specifies that the signals applied to these channels are from constant current excited bridge circuits. The default configuration for this type of bridge input is a 3 wire connection, and so no connection needs to be specified.

Using DeLogger, half bridges with constant current excitation can be measured by the following Program Builder program. The 3 wire connection is selected from the Bridge Wiring Configurations dialog which opens when you select the analog input channel.



The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Interpreting the Data from a Half Bridge with Current Excitation

The data returned from half bridges is the ratio of change in measured resistance to the arm resistance, expressed in parts per million as follows

$$\text{Reading} = \frac{(\Delta R * 10^6)}{R_{arm}} \quad \text{ppm}$$

where

^{a}R	is the sum of changes in arm resistances taking into account sign of the changes
R_{arm}	is the nominated arm resistance, default is 350 Ohm

Calculating Microstrain for Half Strain Gauge Bridges

When using stain gauges in half bridges, it may be desirable to convert the data from units of ppm to units of Microstrain. This can be done using the standard formula

$$Microstrain = \left(\frac{2}{2 * Gauge\ Factor}\right) * ppm$$

or

$$Microstrain = \left(\frac{1}{Gauge\ Factor}\right) * ppm$$

This half bridge method of strain gauge measurement has a measurement resolution of approximately 0.2 Microstrain.

Using DeTransfer, the output from a half bridge with constant current excitation can be calculated to units of microstrain by the program for example

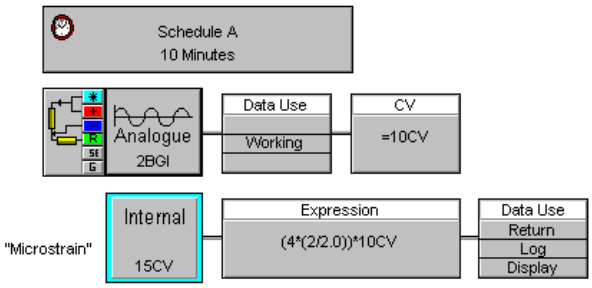
```
BEGIN
 RA5M
  7BGI(=10CV,W)
  15CV("Microstrain =")=(4/(2*2.0))*10CV
END
```

which instructs the *dataTaker* to read the bridge output in ppm, save this in Channel Variable 10 (10CV), then calculate microstrain from the reading using the formula.

Here it is assumed that the gauge factor is 2.0 for the strain gauges used (check your strain gauge supplier or manufacturer for details of the gauge factor).

The use of calculations in the *dataTaker* are discussed in detail in Section III – Channel Variables and Calculations.

Using DeLogger, the calculation can be entered in the Program Builder as follows



Refer to your DeLogger Manual for details of using calculations in the Program Builder.

Quarter Bridge with Constant Current Excitation

The quarter bridge configuration for measuring bridges is a variation of the half bridge configuration, where there is one active device such as a strain gauge, and a bridge completion resistance to balance the bridge.

The bridge completion resistance can be an inactive device of the same type as the active device, or can be a resistor with the same resistance value as the active device, and ideally has a temperature coefficient similar to that of the active device.

The entire bridge circuit is external to the *dataTaker* – the logger does not provide any bridge completion for partial bridges.

Quarter bridges with constant current excitation are connected to the *dataTaker* as follows

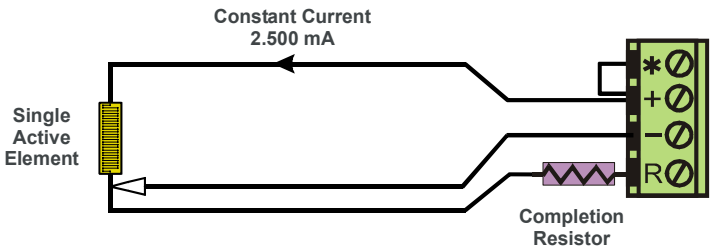


Figure 102 – Quarter Bridge with Constant Current Excitation

This 3 wire configuration provides compensation for cable wire resistance, allowing long cable runs to be used.

The basic quarter bridge configuration can be used for multiple quarter bridges, with a shared bridge completion resistor. The shared bridge completion resistor should be adjacent to the *dataTaker* to ensure accurate lead wire compensation.

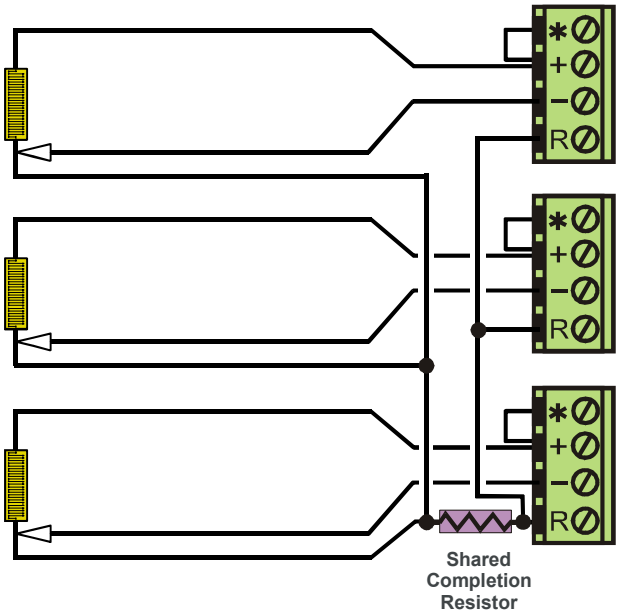


Figure 103 – Multiple Quarter Bridge with Shared Bridge Completion

The quarter bridge configuration with constant current excitation is sampled, and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, quarter bridges with constant current excitation can be measured by the command for example

```
BEGIN
 RA30S
  1..3BGI(120)
END
```

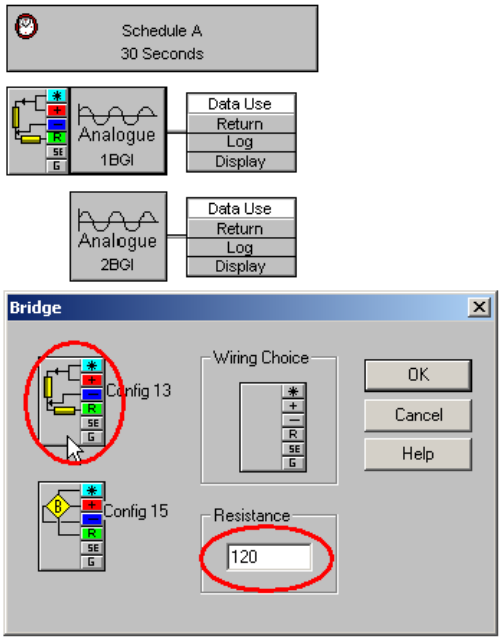
which instructs the *dataTaker* to measure the output from quarter bridges connected to the analog input channels 1, 2 and 3. These quarter bridge active arms all have a resistance of 120.0 Ohm.

The BGI specifies that the signals applied to these channels are from constant current excited bridge circuits. The default configuration for this type of bridge input is a 3 wire connection, and so no connection needs to be specified.

The excitation current channel option is not specified, and so the default 2.500 mA excitation current is used.

Using DeLogger, quarter bridges with constant current excitation can be measured by the following Program Builder program.

The 3 wire connection is selected from the Bridge Wiring Configurations dialog which opens when you select the analog input channel.



The *dataTaker* will read the inputs every 30 seconds, and readings are stopped by entering a H (Halt) command.

Interpreting the Data from a Quarter Bridge with Current Excitation

Quarter bridge data is returned in units of ppm, and is ratio of the change in measured resistance to the arm resistance as follows

$$\text{Reading} = \frac{(R_{act} - R_c) * 10^6}{R_{arm}} \quad \text{ppm}$$

or

$$\text{Reading} = \frac{\Delta R * 10^6}{R_{arm}} \quad \text{ppm}$$

where

R_{act}	is the active arm resistance
R_c	is the bridge completion resistance
ΔR	is the change in bridge resistance
R_{arm}	is the nominated arm resistance, defaults to 350 Ohm

The bridge completion resistor R_c must have a resistance equal to that of the active device at rest, for the bridge to be properly balanced.

Calculating Microstrain for Quarter Strain Gauge Bridges

When using stain gauges in quarter bridges, it may be desirable to convert data from units of ppm to units of Microstrain. This can be done by the standard formula

$$Microstrain = \left(\frac{2}{1 * Gauge\ Factor}\right) * ppm$$

or

$$Microstrain = \left(\frac{1}{Gauge\ Factor}\right) * ppm$$

This quarter bridge method of strain gauge measurement has a resolution of approximately 0.2 Microstrain.

Using DeTransfer, the output from a quarter bridge with constant current excitation can be calculated to units of microstrain by the program for example

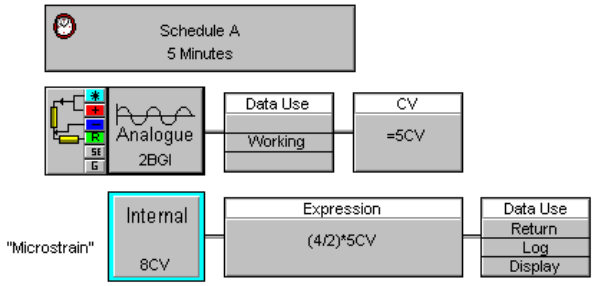
```
BEGIN
 RA5M
  2BGI(=5CV,W)
  8CV("Microstrain =")=(4/2.0))*5CV
END
```

which instructs the *dataTaker* to firstly read the bridge output in ppm, and save this in Channel Variable 5 (5CV), then calculate microstrain from the reading using the formula above.

Here it is assumed that the gauge factor is 2.0 for the strain gauges used (check your strain gauge supplier or manufacturer for details of the gauge factor).

The use of calculations in the *dataTaker* are discussed in detail in Section III – Channel Variables and Calculations.

Using DeLogger, the calculation can be entered in the Program Builder as follows



Refer to your DeLogger Manual for details of using calculations in the Program Builder.

Voltage Excitation of Bridges

The alternative method for measuring bridge circuits with the *dataTaker* is the voltage excitation with ratiometric measurement. The principal of the method is that the bridge is excited by a constant voltage source, and the bridge output voltage is measured as a ratio of the measured excitation voltage.

In practice, resistance of the cable wires connecting the bridge to the logger reduces the excitation voltage that is actually applied to the bridge, which in turn results in a proportionate loss of output signal voltage from the bridge.

To correct for this error the actual voltage applied across the bridge is measured using a second channel.

The Bridge Excitation Voltage Source

The bridge excitation voltage, also often referred to as the bridge power supply, can be supplied from a number of sources

- the Excite terminal of the analog channel, which can output a nominal 5 Volts (actually nearer 4.5 Volts)
- the Excite terminal of the analog channel, which can output a 2.500 mA or 250.0 µA precision current
- the switched 5 Volt sensor power supply terminal of the *dataTaker*, which is limited to 100 mA total current draw
- an external voltage source

The bridge excitation voltage must be switched on during the period of measurement

- if excitation from the *dataTaker* is used, then excitation can be switched on by the logger at appropriate times
- if an external voltage is used for excitation, the bridges can be either permanently powered or can be powered only during measurement by using a digital output channel to control a relay which switches power to the bridges

The default bridge excitation voltage is the 5 Volt supply from the Excite terminal, and is automatically selected when bridge inputs with voltage excitation are specified.

However if the bridges are powered from external sources, then the Excite terminal voltage should be disabled.

Using DeTransfer, this is done by the command for example

```
2BGV(N)
```

where BGV specifies that a bridge voltage is to be measured, and the N channel option specifies no voltage excitation from the Excite terminal.

Alternatively if the bridge is to be excited by either of the Excite terminal current sources then channel option I for 250.0 µA current, or channel option II for 2.500 mA current, should be used.

Using DeTransfer, this is done by the command for example

```
1BGV(I)
8BGV(II)
```

where BGV specifies that a bridge voltage is to be measured, and the ***I*** and ***II*** channel options specify current excitation from the Excite terminal.

DeLogger does not directly support the measurement of bridges that are excited by a voltage. However bridge measurements, including controlling the method of powering, can be programmed into the *dataTaker* via the User channel type (DeLogger Ver 4.2.15 or later) in the Program Builder. This procedure is illustrated in the following topics.

Measuring the Bridge Excitation Voltage

In practice the resistance of the cable wires connecting bridges to the *dataTaker* reduce the excitation voltage that is actually applied to the bridge.

This results in a proportionate loss of output signal from the bridge. To correct for this error, the actual excitation voltage across the bridges is also measured.

The bridge excitation voltage is connected as a differential or single ended voltage input to any analog input channel, and must be measured immediately before the output of any bridge is measured.

This measurement is referred to as the 'bridge reference voltage', and is measured on the bridge reference channel that is identified to the *dataTaker* by the BR channel option for the particular channel.

Using DeTransfer, the command for example

```
1V(BR)
```

identifies that the bridge reference voltage is to be measured as a differential voltage connected to analog input channel 1.

dataTaker 50,500,600 series loggers : The bridge reference channel has a maximum input voltage of 2.5 Volts for *dataTaker 50,500,600* series loggers. Therefore if the bridge excitation voltage is greater than 2.5 Volts, then this must be externally attenuated (see Section II – Measuring High Level Voltages) before input to the bridge reference channel on these loggers.

Whenever the bridge excitation voltage must be attenuated, the attenuation factor is also declared as a channel option to the bridge reference voltage channel as follows

```
1V(2.0,BR)
```

which declares an attenuation of 2.0:1, that is appropriate for bridges powered by an external 5 Volt supply.

dataTaker 505,605 series loggers : If a *dataTaker 505,605* series logger is being used, then the bridge excitation voltage can be measured directly as a High Level Voltage (see Section II – Measuring High Level Voltages) on the bridge reference channel, for example

```
2HV(BR)
```

The bridge reference channel does not return any data when it is scanned. The data is retained for subsequent use in bridge measurements and calculations.

However if you want to include the bridge power supply or excitation measurements in your data, then the bridge reference voltage can be returned by a second command for example

```
1V(BR)   1V
```

where the channel 1V will return the bridge reference voltage.

Note : The bridge reference channel must precede the bridge measurement channel(s) in the *dataTaker* program, because the bridge reference voltage is used to calculate the bridge data for the subsequent bridge measurement channels.

Note : If bridge measurements are included in more than one Schedule, then the bridge reference channel(s) must be declared in each Schedule.

If a bridge reference channel is not declared, then the bridge reference voltage defaults to 5 Volts. This is based on the assumption that most voltage excited bridges will be powered from the *dataTaker* 5 Volt sensor power supply.

DeLogger does not directly support the measurement of bridges that are excited by a voltage. However the bridge reference channel, and bridge measurements channels can be programmed into the *dataTaker* via the User channel type (DeLogger Ver 4.2.15 or later) in the Program Builder. This procedure is illustrated in the following topics.

Full Bridge with Voltage Excitation

The full bridge with voltage excitation configuration is the more traditional method for the measurement of bridge outputs. However a full implementation of requires more resources than any of the constant current methods, requiring

- two channels for the each bridge, if each bridge has a separate bridge excitation
- two channels for the first bridge, and one channel for each additional bridge that is excited by the same bridge power supply. This configuration is only appropriate if all cable wires are the same length, such that all bridges receive the same voltage excitation as measured for the first bridge

This configuration supports 1, 2 or 4 active arms. Any of the bridge arms can be active arms.

Bridge arms which do not have active devices must have bridge completion resistances to balance the bridge. These can be inactive devices of the same type as the active devices, or can be a resistor with the same resistance value as the active devices at rest, and ideally have a temperature coefficient that is similar to that of the active devices.

Where the bridge power supply and bridge output are measured for each bridge, this is referred to as a six wire connection as illustrated below

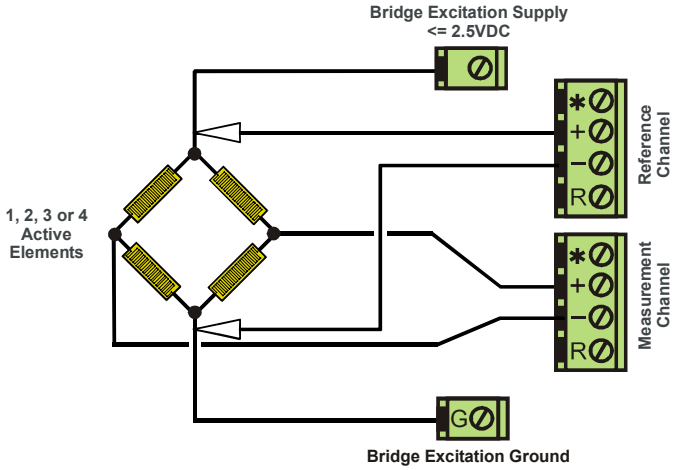


Figure 104 – Full Bridge with Voltage Excitation

The entire bridge circuit is external to the *dataTaker* – the logger does not provide any bridge completion for partial bridges.

Full bridges with voltage excitation are sampled, and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, full bridges with voltage excitation are measured by the commands for example

```
BEGIN
 RA5M
  1V(BR)        'bridge reference channel
  2BGV(4W,N)    'bridge measurement channel
END
```

which instructs the *dataTaker* to measure the bridge excitation voltage connected to analog input channel 1 (bridge reference channel), and the bridge output connected to the analog input channel 2.

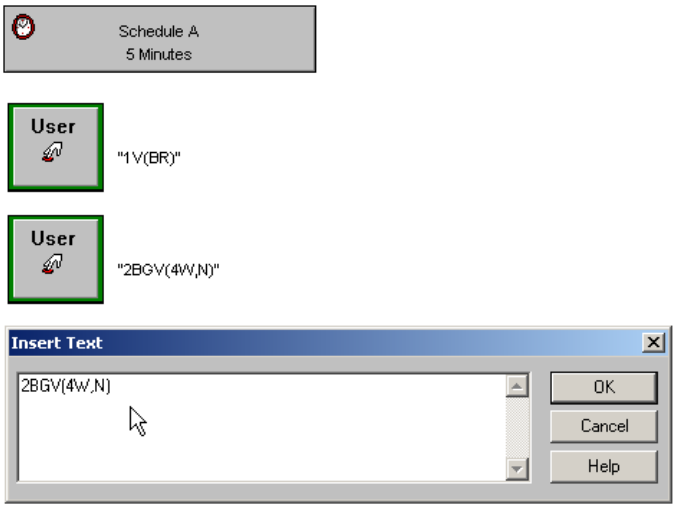
The BR indicates which analog channel the bridge excitation voltage is connected to for measurement. Note that the bridge reference channel is measured before the bridge output channel is measured.

The BGV specifies that the signal applied to this channel is from a bridge that is excited by a voltage. The bridge output data is returned in units of ppm.

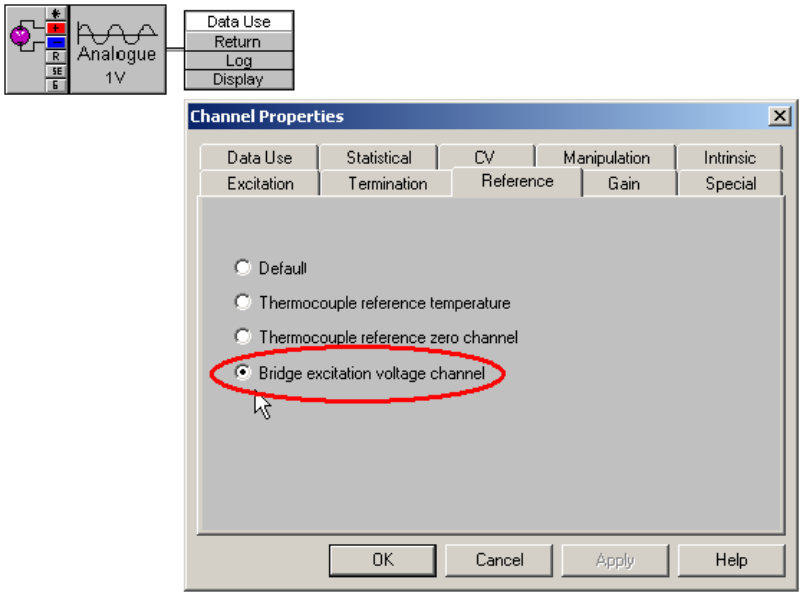
The 4W channel option indicates that the 4 wire measurement method is to be used. This option must be specified for all full bridge inputs.

The bridge excitation voltage is supplied from an external source in this example, and so the Excite terminal is disabled by the N channel option.

DeLogger does not directly support full bridges with voltage excitation. However, full bridges with voltage excitation can still be measured with DeLogger (Ver 4.2.15 or later) by using the User channel as illustrated by the following Program Builder program.



The bridge reference channel could also be entered as a low level voltage channel, and the Channel Properties set to Bridge excitation voltage channel in the Reference tab as follows



The *dataTaker* will read the inputs every 5 minutes, and readings are stopped by entering a H (Halt) command.

Various compromises are possible with full bridges with voltage excitation as follows

the number of cable wires can be reduced from six to four by measuring the bridge excitation voltage at the *dataTaker*, rather than at the bridge. However this does not take into account reduction of the excitation voltage at the bridge due to cable resistance.

the bridge reference channel can be shared by a number of bridge measurement channels. No errors will be introduced provided all of the bridges have the same excitation voltage. This can be done by close proximity of the bridges to a shared supply, or the use of cable wires of the same type and length.

Interpreting the Data from a Full Bridge with Voltage Excitation

Data returned from full bridges with voltage excitation is calculated as the ratio of the change in bridge output voltage to bridge excitation voltage, expressed in parts per million as follows

$$\text{Reading} = \frac{(\Delta V * 10^6)}{V_{excite}} \text{ ppm}$$

where

^aV	is the change in bridge output voltage
V_{excite}	is the bridge excitation voltage, measured by the bridge reference channel

Calculating Microstrain for Full Strain Gauge Bridges

When using stain gauges in full bridges, it may be desirable to convert the returned data from units of ppm to units of Microstrain.

This can be done by the following formula

$$\text{Microstrain} = \left(\frac{4}{\text{No.Gauges} * \text{Gauge Factor}}\right) * \text{ppm}$$

This full bridge method of strain gauge measurement has a resolution of approximately 0.2 Microstrain.

Using DeTransfer, output from a full bridge with voltage excitation can be calculated to units of microstrain by the program for example

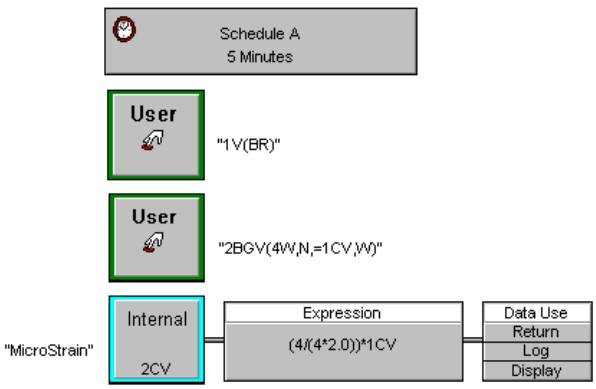
```
BEGIN
 RA5M
  1V(BR)
  2BGV(4W,N,=1CV,W)
  2CV("Microstrain =")=(4/(4*2.0))*1CV
END
```

which instructs the *dataTaker* to read the bridge excitation and bridge output voltages, calculate the ratio in ppm and save in Channel Variable 1 (1CV), and calculate the microstrain from the reading in ppm using the formula above.

Here it is assumed that the gauge factor is 2.0 for the strain gauges used (check your strain gauge supplier or manufacturer for details of the gauge factor).

The use of calculations in the *dataTaker* are discussed in detail in Section II – Channel Variables and Calculations.

Using DeLogger (Ver 4.2.15 or later), the calculation can be entered into the Program Builder as follows



The bridge reference channel could alternatively be entered as a low level voltage channel, and the Channel Properties set to Bridge excitation voltage channel in the Reference tab as illustrated on the previous page.

Refer to your DeLogger Manual for details of using calculations in the Program Builder.

Half Bridge with Voltage Excitation

Half bridges with two active arms and voltage excitation are commonly used when a large number of bridges need to be located in close proximity.

The *dataTaker* supports this configuration by using single ended inputs and the single ended reference SE Ref.

Half bridges with two active arms require two bridge completion resistances to balance the bridge. The two bridge completion resistances can be either inactive devices of the same type as the active device, or can be a resistor with the same resistance value as the active devices, and ideally have a temperature coefficient similar to that of the active devices.

This half bridge configuration with voltage excitation can be used to measure a single half bridge, or to measure a number of half bridges which share the same bridge excitation voltage supply, and share the same set of bridge completion resistors.

Multiple half bridges that are excited from a single excitation voltage source and share bridge completion resistors are illustrated below in Figure 105. The configuration for a single half bridge is that for the innermost half bridge.

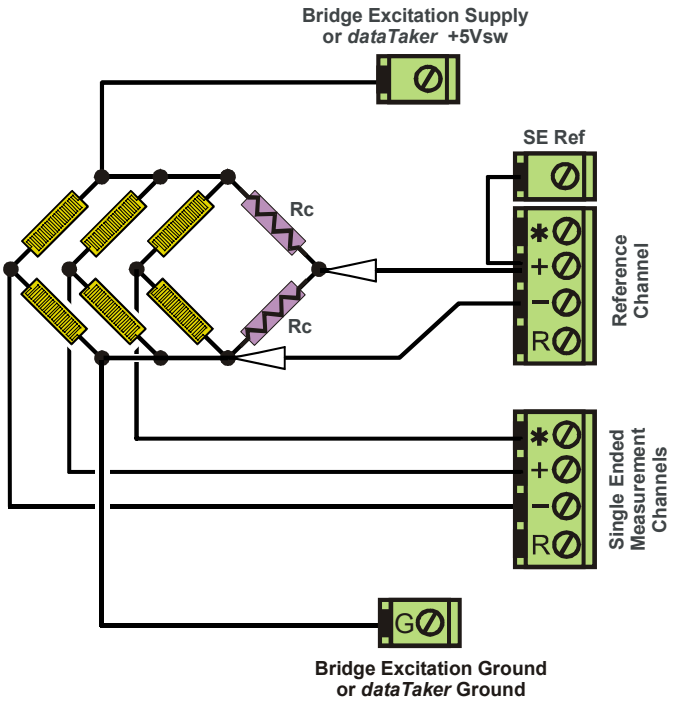


Figure 105 – Half Bridges with Voltage Excitation

dataTaker 50,500,600 series loggers : If this bridge configuration is connected to a *dataTaker 50,500,600* series logger, then the bridge completion resistors must provide a 2:1 attenuation of the 5 Volt bridge excitation voltage, to a reduce the signal suitable for input to the bridge reference channel.

dataTaker 505,605 series loggers : If this bridge configuration is connected to a *dataTaker 505,605* series logger, then the bridge excitation voltage can be measured as a High Level Voltage (see Section II – Measuring High Level Voltages) on the bridge reference channel.

The half bridges should preferably all be in close proximity to the bridge completion resistors. However if this is not possible, then the bridge completion resistors can be located at the *dataTaker*, and each half bridge connected by three leads. This will provide lead compensation for zero, but no scale compensation.

Half bridges with voltage excitation are sampled, and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, half bridges with voltage excitation are measured by the commands for example

```
BEGIN
 RA10M
  1V(BR,5.0)
  2*BGV(N,X)
  2+BGV(N,X)
  2-BGV(N,X)
END
```

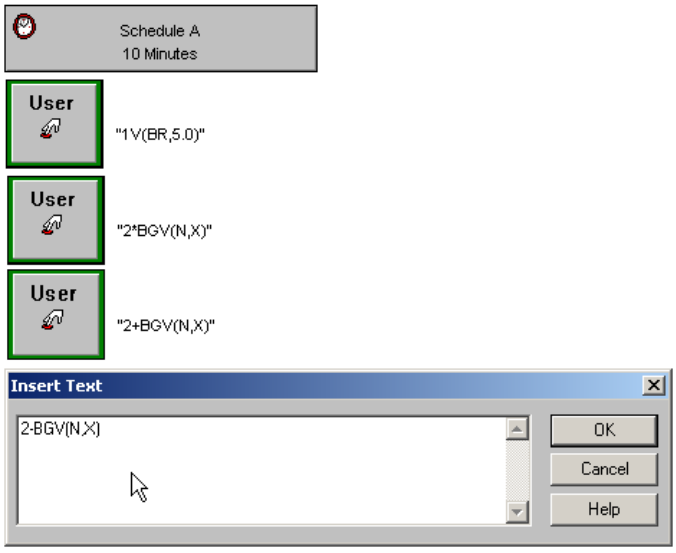
which instructs the logger to measure the bridge reference voltage that is differentially connected to analog channel 1, and measure the output from half bridges connected as single ended inputs to analog channels 2* through 2– .

Assuming that the excitation voltage is supplied from an external 10 Volts source, the default Excite terminal voltage output is disabled by the N channel option.

The BR channel option indicates the analog channel to which the bridge excitation voltage is connected for measurement, and is attenuated by a factor of 5.0:1 to reduce the 10 Volt excitation voltage into range for a *dataTaker* 50,500,600 series logger. The reference channel must be read before the measurement channel.

The BGV specifies that the signals applied to these channels come from voltage excited half bridges. The X channel option indicates that the single ended inputs are to be measured with reference to SE REF terminal.

DeLogger does not directly support half bridges with voltage excitation. However, half bridges with voltage excitation can still be measured with DeLogger (Ver 4.2.15 or later) using the User channel type as illustrated by the following Program Builder program.



The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Interpreting the Data from a Half Bridge with Voltage Excitation

Data returned from half bridges with voltage excitation is calculated as the ratio of the change in bridge output voltage to bridge excitation voltage, expressed in parts per million as follows

$$\text{Reading} = \frac{(\Delta V * 10^6)}{V_{excite}} \text{ ppm}$$

where

^{a}V	is the change in bridge output voltage
V_{excite}	is the bridge excitation voltage, measured by the bridge reference channel

Calculating Microstrain for Half Strain Gauge Bridges

When using stain gauges in half bridges, it may be desirable to convert the data from units of ppm to units of Microstrain. This can be done by the standard formula

$$\text{Microstrain} = \left(\frac{4}{2 * \text{Gauge Factor}}\right) * \text{ppm}$$

or

$$\text{Microstrain} = \left(\frac{2}{\text{Gauge Factor}}\right) * \text{ppm}$$

This full bridge method of strain gauge measurement has a measurement resolution of approximately 0.2 Microstrain.

Using DeTransfer, output from a half bridge with voltage excitation can be calculated to units of microstrain by the program for example

```
BEGIN
 RA10M
  1V(BR,5.0)
  2*BGV(N,X,=1CV,W)
  2CV("Microstrain 1 =")=(4/(2*2.0))*1CV
  2+BGV(N,X,=3CV,W)
  4CV("Microstrain 2 =")=(4/(2*2.0))*3CV
  2-BGV(N,X,=5CV,W)
  6CV("Microstrain 3 =")=(4/(2*2.0))*5CV
END
```

which instructs the *dataTaker* to

read the bridge excitation and bridge output voltages

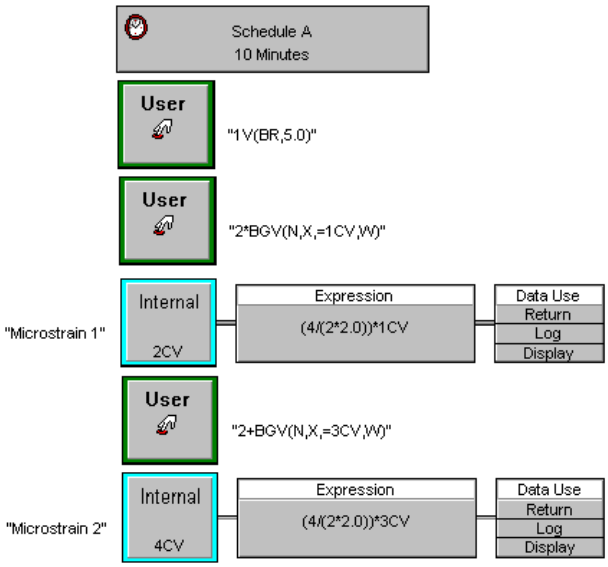
calculate the ratio in ppm and save in Channel Variables

calculate the microstrain from the readings in ppm using the formula above

Here it is assumed that the gauge factor is 2.0 for the strain gauges used (check your strain gauge supplier or manufacturer for details of the gauge factor).

The use of calculations in the *dataTaker* are discussed in detail in Section II – Channel Variables and Calculations.

Using DeLogger (Ver 4.2.15 or later), the calculation can be entered into the Program Builder as follows



etc, etc

Refer to your DeLogger Manual for details of using calculations in the Program Builder.

Converting Bridge Outputs to Engineering Units

This chapter has provided methods to convert measured bridge output in ppm to units of Microstrain for the various bridge configurations. However units of Microstrain apply to strain gauge bridges which are measuring deformation.

Many sensors available today employ a bridge circuit to sense the parameter they are designed to measure. For example some pressure cells, load cells, micro-displacement transducers, etc. in fact contain a diaphragm or similar structure which has a full strain gauge bridge bonded to one surface. The diaphragm is mechanically distorted by the pressure or load, which is measured by the strain gauge bridge. This distortion is calibrated to units of pressure, or load, etc. by the manufacturer.

Supporting these types of sensors with the *dataTaker* is quite simple, as shown by the following examples.

Pressure Transducer

A pressure transducer that is constructed as a full bridge device with a 4 wire connection, is connected to the *dataTaker* as a full bridge with constant current excitation (type BGI) as illustrated in Figure 100.

The transducer has an output of 0.05 V full scale at 10 VDC excitation. The *dataTaker* will measure

$$\text{Minimum Bridge Output} = 0.0 \text{ ppm at } 0.0 \text{ kPa}$$

and

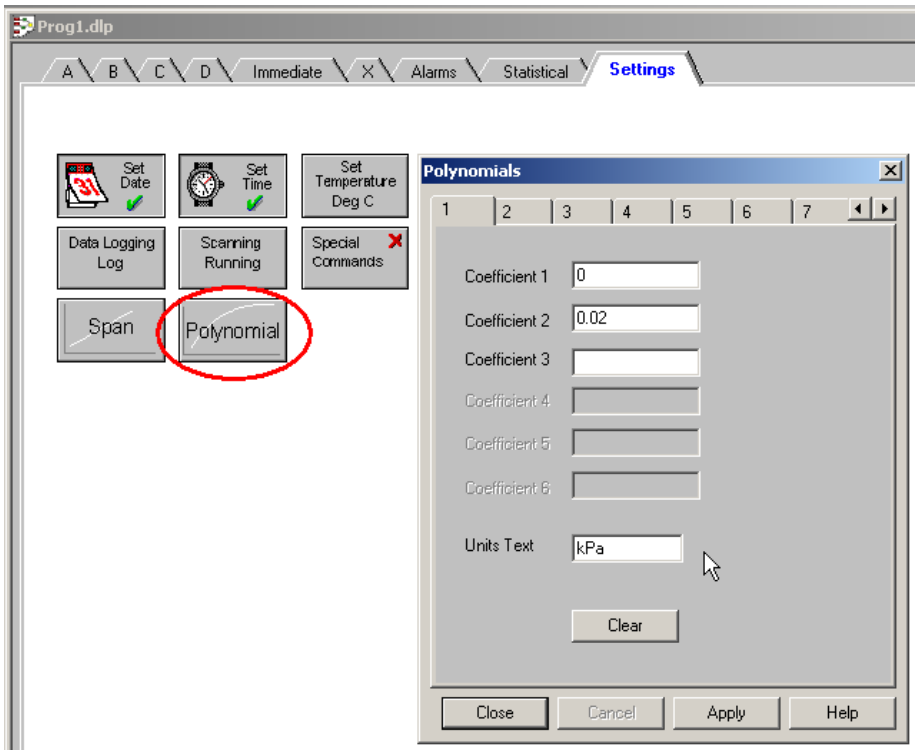
$$\text{Maximum Bridge Output} = \frac{(0.05 * 10^6)}{10.0} = 5000 \text{ ppm at } 100 \text{kPa full scale}$$

Therefore 1ppm = 100 kPa / 5000 ppm = 0.02 kPa. This transducer calibration can be used in a *dataTaker* program to return the data in units of kPa.

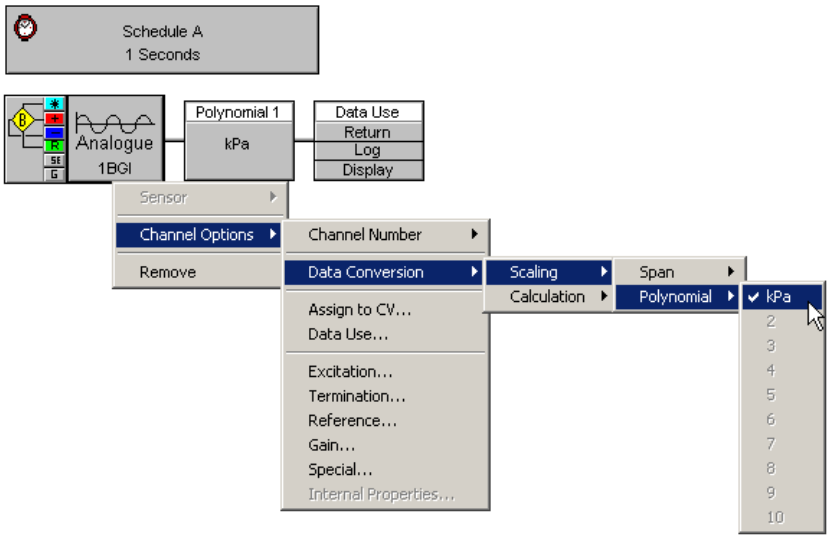
Using DeTransfer, the program will be similar to

```
BEGIN
Y1=0,0.02”kPa”
 RA1S
  1BGI(Y1)
END
```

Using DeLogger, the calibration for the pressure transducer must be entered as a polynomial into the Polynomials dialog under the Settings tab of the Program Builder as follows



The polynomial is then attached to the bridge input channel in the program to convert the raw data to units of lbs as follows



For further discussion of polynomials, see Section III – Polynomials and Spans of this manual, and the DeLogger Users Manual.

Load Cell

A load cell that is constructed as a full bridge device with a 4 wire connection, is connected to the *dataTaker* as a full bridge with constant current excitation (type BGI) as illustrated in Figure 100.

The load cell measures a load of 100 lbs full scale, and has an output of 2.0006 mv/V at full scale. The *dataTaker* will measure

Minimum Output = 0.0 ppm at 0.0 lbs

and

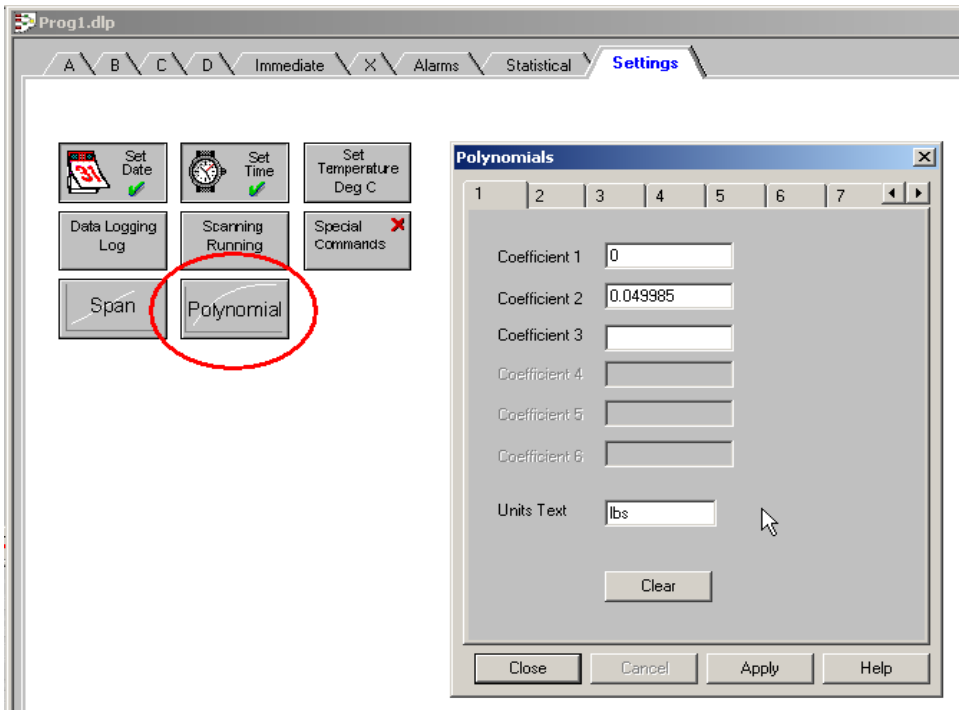
$$\text{Maximum Output} = \frac{(2.0006 * 10^6)}{1000.0} = 2000.6 \text{ ppm at 100 lbs full scale}$$

Therefore 1ppm = 100 lbs / 2000.6 ppm = 0.049985 lbs. This transducer calibration can be used in a *dataTaker* program to return the data in units of lbs.

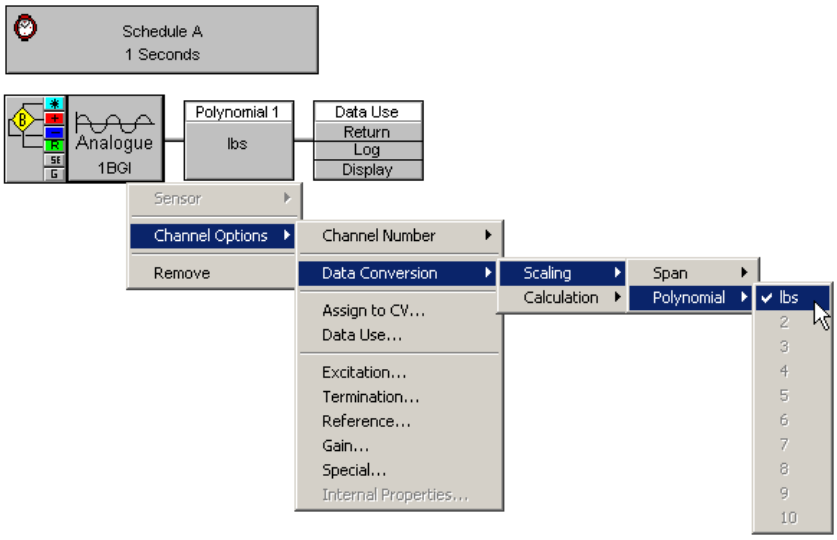
Using DeTransfer, the program will be similar to

```
BEGIN
Y1=0,0.049985"lbs"
 RA1S
  1BGI(Y1)
END
```

Using DeLogger, the calibration for the load cell must be entered as a polynomial into the Polynomials dialog under the Settings tab of the Program Builder as follows



The polynomial is then attached to the bridge input channel in the program to convert the raw data to units of lbs as follows



For further discussion of polynomials, see Section III – Polynomials and Spans of this manual, and the DeLogger Users Manual.

Measurement Ranges and Accuracy

The *dataTaker* measures all bridge inputs as a low level voltage, with a resolution of 1 µV, and a nominal accuracy of 0.1%.

The accuracy for particular applications can be calculated from this information, and the excitation current or voltage used.

Error Messages

There are no specific error messages for bridge inputs. However input voltage signals which fall outside the voltage range of the *dataTaker* will produce an over-range reading of –99999.9 ppm or +99999.9 ppm.

The *dataTaker* also reports the error condition with the error message ‘E11–input(s) out of range’ if the Messages Switch /M is enabled.

Measuring Vibrating Wire Strain Gauges

Vibrating wire sensors are used in the mining, civil and hydrological engineering, and other geophysical disciplines.

There are two versions of the *dataTaker* data loggers which support the vibrating wire strain gauges as follows

dataTaker 515 Geologger

dataTaker 615 Geologger

The Geologgers are functionally the same as the *dataTaker 505* and *dataTaker 605* data loggers, with the addition of an internal vibrating wire strain gauge support module and associated commands for reading vibrating wire sensors.

The Geologgers support all the signal types and functions available in the *dataTaker 505* and the *dataTaker 605* data loggers.

Vibrating wire strain gauges are essentially taut wires which change their natural resonant frequency in proportion to the square root of tension placed on them.

In practice these elements are used in various sensors designed to measure soil pore pressure, strain in structure, rock stress, overburden pressure, etc.

The Geologgers support vibrating wire strain gauges with resonance between the range of 600 Hz and 4.5 KHz.

Vibration Wire Support by the Geologgers

The Geologgers generate a 'current pulse' to excite or 'pluck' the wire in the vibrating wire gauge. Immediately following excitation, the resonant frequency of the vibrating wire is measured. The advantage of the plucking pulse method is that a fixed pulse is able to stimulate a wide range of gauges. This greatly simplifies channel programming for the user.

The balanced plucking pulse is approximately 200 µS long and up to 36 Volts in amplitude. The pulse has a current source characteristic that provides automatic cable length compensation. Sensors on long cables will be pulsed with the similar energy as those on shorter cables.

The Geologger has a high gain low noise signal amplifier with transformer coupling on the input.

The amplified signal is filtered using band pass filters (500Hz to 5KHz) and a phase lock loop to reduce frequency noise before the frequency is measured by a precision frequency counter.

Signals of the order of tens of microvolts can provide useful readings. Transformer coupling ensures very high common mode rejection, a characteristic that is needed to reject 50/60 Hz mains noise and other interfering noise.

A block diagram of the pulse pluck method is illustrated in below

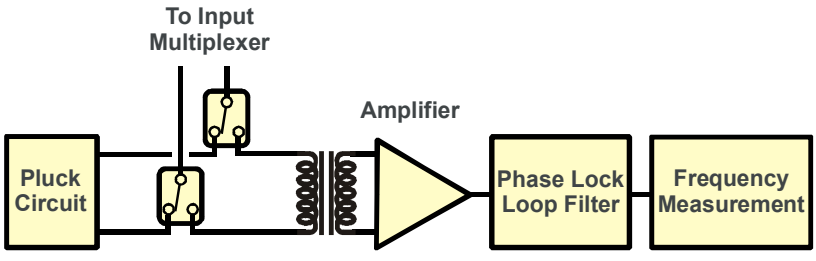


Figure 106 - Block Diagram of Vibrating Wire Strain Gauge Support

Differential Measurement of Vibrating Wire Strain Gauges

The preferred method for connecting vibrating wire strain gauges to the *dataTaker* is differential connection, where the sensor is connected between the +ve and –ve terminals of the analog input channels.

Differential connection of a Vibration Wire sensor is illustrated below

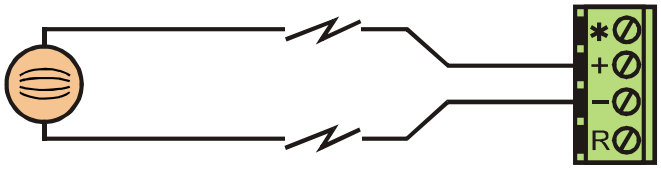


Figure 107a - Differential Vibrating Wire Strain Gauge Connection

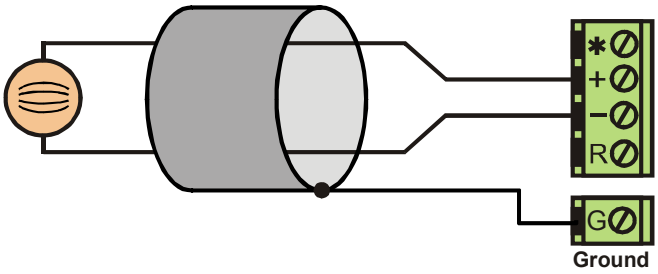


Figure 107b - Differential Vibrating Wire Strain Gauge Connection with Shield

A shielded signal cable is optional, however it will often be found necessary when noise pick-up is a problem. The preferred shield connection point is either one of the Geologger ground GND terminals or a case ground terminal strip.

If the Analog Return terminal of the channel is not used for other purposes, it can be used as a shield terminal. However because the Analog Return terminal is internally connected to ground via a 100 Ohm resistor, its effectiveness is not as great as a direct connection to ground. (Also if lightning strike is possible, then the resistor may be destroyed).

Vibrating wire strain gauges which are directly connected to the analog input channels as differential inputs are sampled, and the data is returned to the host computer when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
BEGIN
 RA3M
  1FW
  5FW
END
```

instructs the Geologger to measure the vibrating wire strain gauges

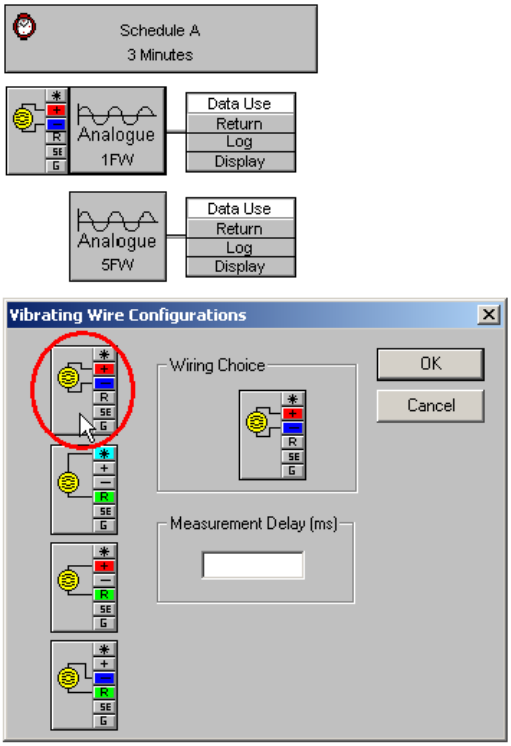
connected between the +ve and –ve terminals of analog input channel 1

connected between the +ve and –ve terminals of analog input channel 5

The FW specifies a vibrating wire strain gauge input, and configures the channel to pluck the sensor and measure the frequency returned.

Using DeLogger, vibrating wire strain gauges that are connected differentially can be measured by the following Program Builder program.

The differential connections are selected from the Vibrating Wire Configurations dialog which opens when you have selected the analog input channel.



Data is returned in units of Hz, and can be scaled to engineering units using either Polynomials or Spans (See Section III – Scaling Data, Polynomials, Spans and Functions), or using calculations. (See Section III – Channel Variables and Calculations).

The *dataTaker* reads the inputs every 3 minutes, and readings are stopped by a H (Halt) command.

Single Ended Measurement of Vibrating Wire Strain Gauges

Vibrating wire strain gauges can also be connected to the Geologger as single ended analog inputs. The signal from the sensor is connected between the +ve, –ve or ✳ terminal, and Analog Return terminal.

Single ended connection for vibrating wire strain gauges is illustrated below

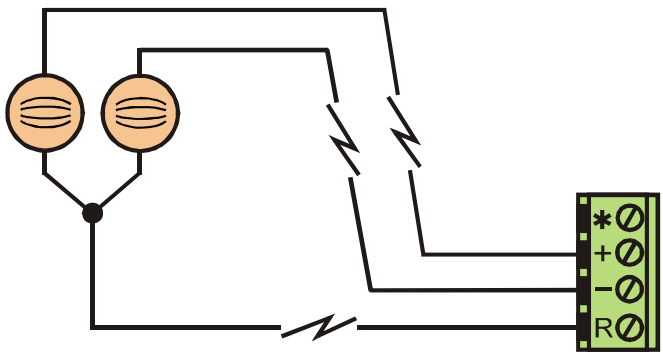


Figure 108 - Single Ended Vibrating Wire Strain Gauge Connection

The connection of vibrating wire strain gauges as single ended inputs referenced to Analog Return is best used where

cable lengths are relatively short (< 100 meters)

vibrating wire strain gauges have good sensitivity (high signal to pluck ratio)

Because of the great range in vibrating wire strain gauge sensitivity, it is difficult to predict the operating limits. It is suggested that where cable lengths are in excess of 100 meters, a test be conducted with the gauges to be deployed.

Single ended connection of vibrating wire strain gauges allows up to three sensors to be connected to each analog input channel, and up to 30 sensors to be connected to a Geologger 515 or Geologger 615.

Vibrating wire strain gauges which are directly connected to the analog channels as single ended inputs are sampled and the data returned when a Schedule containing the channel is executed.

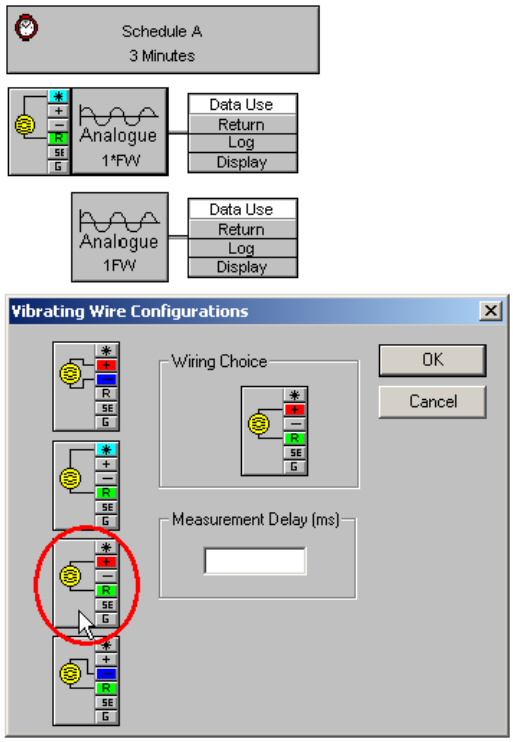
Using DeTransfer, the command for example

```
BEGIN
 RA3M
  1  ?FW   1+FW   1-FW
END
```

instructs the Geologger to measure 3 vibrating wire strain gauges connected between channel 1+ and Analog Return, channel 1– and Analog Return, and channel 1 ? and Analog Return.

The FW specifies a vibrating wire strain gauge input, and configures the channel to pluck the sensor and measure the frequency returned.

Using DeLogger, vibrating wire strain gauges that are connected as single ended inputs can be measured by the following Program Builder program. Single ended connections are selected from the Vibrating Wire Configurations dialog which opens when you have selected the analog input channel.



Data is returned in units of Hz, and can be scaled to engineering units using Polynomials or Spans (See Section III – Scaling Data, Polynomials, Spans and Functions), or by using calculations. (See Section III – Channel Variables and Calculations).

The *dataTaker* will read the inputs every 3 minutes, and readings are stopped by entering a H (Halt) command.

Speaker Output

The Geologger has a built in speaker, and 3.5mm mono or stereo headphone jack for 8 Ohm headphones. The speaker or headphones are used for fault diagnosis.

The speaker output is optional, and is determined by the Speaker Switch as follows

/V Enable output from the speaker

/v Disable output from the speaker (Default)

The Speaker Switch defaults to /v when the Geologger is initially powered up, hardware reset, or executes a RESET command.

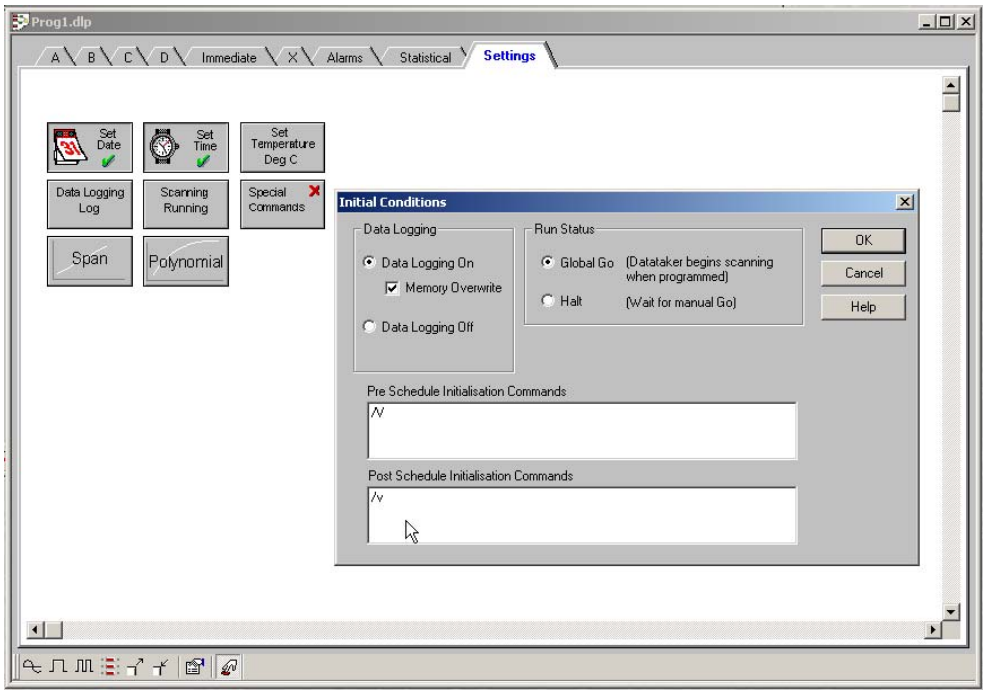
Using DeTransfer, the speaker can be turned on by the command

```
/V
```

and off by the command

```
/v
```

Using DeLogger, the speaker can be turned on by including the Speaker Switch Enabled command in the Pre Schedule Initialization Commands under Settings : Special Commands in the Program Builder.



The speaker can be turned off again here if required by placing a Speaker Switch Disabled command in the Post Schedule Initialization Commands

The speaker is connected to the high gain amplifier's output. The frequency response of the small speaker is not flat, and the use of headphones is preferred.

Diagnosing Faults with the Speaker

The speaker or headphones output can be used to diagnose faults with the vibrating wire strain gauges, the sensor cabling and sensor connection.

A good vibrating wire strain gauge which is correctly installed will produce a clean "ping" sound decaying over a period of a few seconds, when the sensor is sampled.

Note : The full decay can only be heard for a single channel, or the last channel of a channel sequence. Other channels can be heard but only for approximately half a second.

If a clean pinging sound is not heard when the vibrating wire strain gauge is sampled, then the following trouble shooting guide will help diagnose the problem

If there is only random noise, check the channel type, wiring and resistance (see below).

If a ping can be heard but it is faint or buried in random noise, then the cable is too long or is "leaky", or the gauge sensitivity is too low.

If the ping is not clean and pure, then the gauge is possibly faulty. The gauge may have been mechanically damaged during installation.

If you can hear a low frequency hum, then noise pick is a problem. If the gauge is placed near a transformer, electric motor, high current power cables, etc, then relocate or reorient the gauge for minimum pickup. Ensure that the cable is shielded to prevent capacitive pickup.

Measuring Gauge Resistance

The vibrating wire strain gauge and cable integrity is best determined by measuring the circuit resistance.

This can be done using a multimeter, or using the Geologger's analog input channels to measure the resistance (See Section II - Measuring Resistance).

The vibrating wire strain gauge and cable resistance should be stable, and should not drift with time.

Measurement Delay

The Geologger inserts a delay period between stimulating or plucking the vibrating wire strain gauge, and reading the resonant frequency of the wire.

If the data returned from vibrating wire strain gauge is unstable, to the extent that it varies by ±20Hz even though the speaker indicates a strong signal, then the gauge signal may contain harmonics.

The harmonics generally decay more rapidly than the fundamental frequency, and so increasing the time between stimulation and the frequency measurement can improve the results.

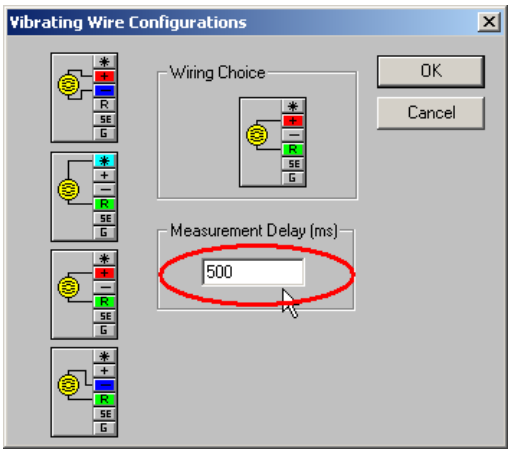
The measurement delay can be adjusted by setting the channel factor which specifies a delay in milliseconds (see Section III - Channel Options). The default measurement delay is 200 mS, which is suitable for most gauges.

Using DeTransfer, the measurement delay is set by the command for example

```
1FW(500)
```

will increase the delay from the default 200 mS to 500 mS.

Using DeLogger, the measurement delay can be set for each channel in the Vibrating Wire Configurations dialog which opens when you have selected the analog input channel.



Extra Samples

The Geologger normally measures a vibrating wire frequency over a period of 10 line periods (167mS at 60Hz, or 200mS at 50Hz). The Geologger samples the vibrating wire strain gauge frequency 10 times, and returns data which is the average of these readings.

Sampling the vibrating wire strain gauge inputs 10 times has been found optimal for most gauge types. However for gauges with a rapid signal decay, this period can be reduced so that the measurement window does not extend into the noise which may be present.

The Extra Samples (ES) channel option can be used to control the number of samples taken by the Geologger to determine the vibrating wire strain gauge frequency.

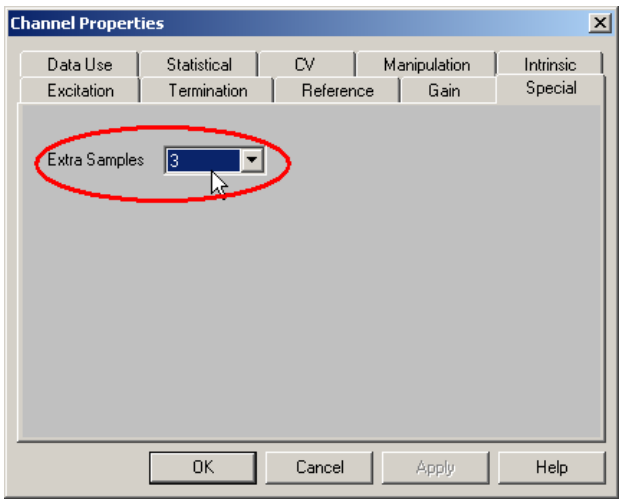
Using DeTransfer, the number of samples is set by the command for example

```
1FW(ES4)
```

specifies that 5 samples (the original sample + 4 extra samples) are to be taken when the vibrating wire strain gauge is read.

The ES channel option and the measurement delay channel option can be used in the same channel specification to optimise gauge measurement.

Using DeLogger, the number of extra samples can be set for each channel by clicking Channel Options:Special... to open the Channel Properties dialog. Select the Special tab to enter the number of Extra Samples.



The measurement delay and the number of extra samples can be combined.

Using DeTransfer, the command for example

```
3FW(ES6,100)
```

specifies a measurement delay of 100 mS between plucking the sensor and beginning sampling of the frequency, and specifies that 6 extra samples are to be taken.

Measuring Gauge Temperature

Most vibrating wire strain gauges are sensitive to temperature fluctuations. Where a vibrating wire strain gauge temperature is likely to change significantly, the gauge temperature should be measured.

The vibrating wire strain gauge temperature can be measured using IC temperature sensors, RTDs or thermistors supported by the logger.

Depending on the internal wiring of the vibrating wire strain gauge, it is often possible to measure the vibrating wire frequency, and a resistive temperature sensor (RTD or thermistor), on a single analog input channel.

High Resistance Nickel RTDs

High resistance RTDs can be wired as single ended inputs, and the vibrating wire strain gauge sensor wired as a differential input.

There is no compensation for cable wire resistance on the measured RTD temperature. Therefore the RTD sensor should be a relatively high resistance type (>1000 Ohm), so that temperature measurement errors due to cable resistance are minimised.

However the differential input for the gauge allows maximum accuracy.

The following configuration is recommended

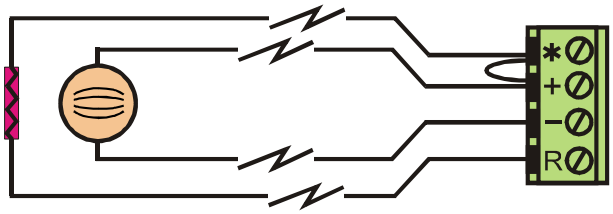


Figure 109 - Connecting a Vibrating Wire Gauge and 2 Wire RTD

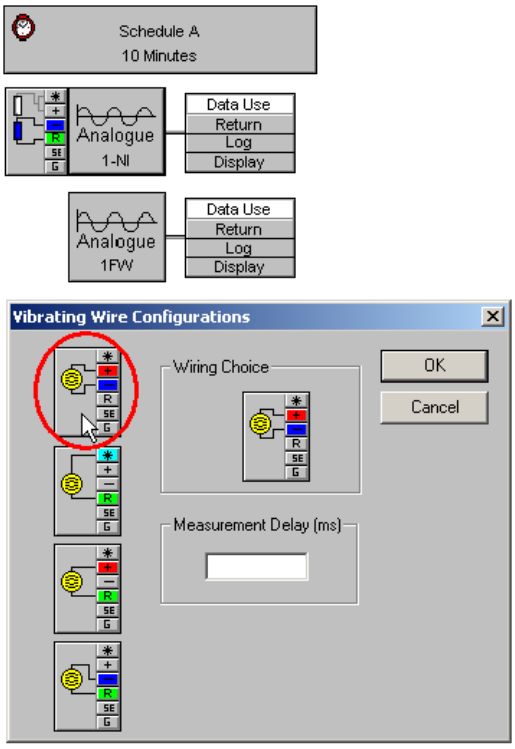
The RTD temperature is read as a single ended input, and the vibrating wire frequency is read as a differential input.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1-NI  1FW
END
```

instructs the Geologger to read the nickel RTD connected as a single ended input to channel 1, and the vibrating wire frequency connected differentially to channel 1.

Using DeLogger, the vibrating wire sensor and RTD can be measured by the following Program Builder program.



Low Resistance Platinum RTDs

If the RTD temperature sensor is a low resistance platinum type, then it can be measured by the more accurate 3 wire method, and the vibrating wire sensor measured by the single ended method. This compromise still allows the gauge temperature and gauge signal to be measured on the same channel.

However this configuration has the disadvantage that the single ended vibrating wire strain gauge connection will be a little less accurate than a differential connection.

The following configuration is recommended

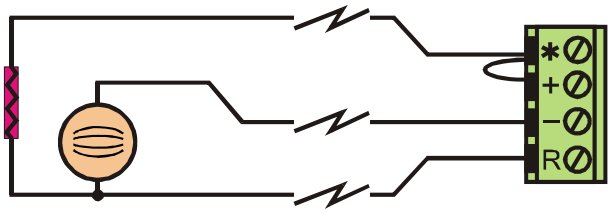


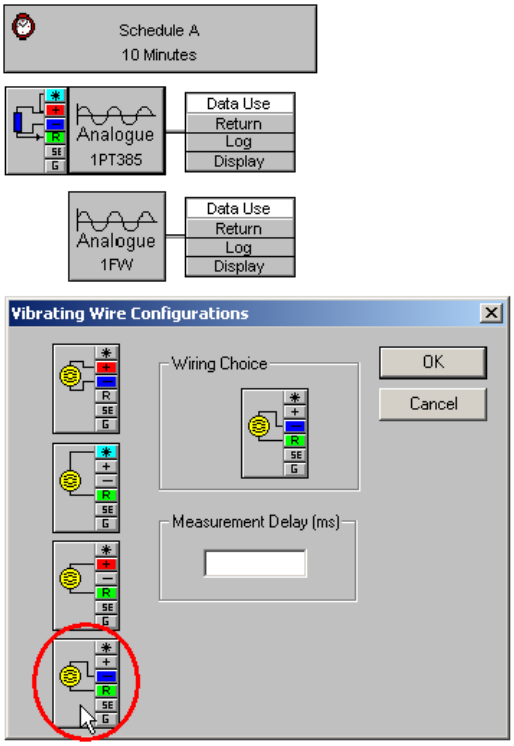
Figure 110 - Connecting a Vibrating Wire Gauge and 3 Wire RTD

The temperature channel is read as a 3 wire RTD, and the vibrating wire strain gauge is read as a single ended input.

Using DeTransfer, the command for example

```
BEGIN
 RA10M
  1PT385  1-FW
END
```

Using DeLogger, the vibrating wire sensor and low resistance RTD can be measured by the following Program Builder program.



Copper RTDs

It is possible to use the copper coil in the vibrating wire strain gauge as a temperature sensor, provided that a three wire connection is used. The vibrating wire strain gauge is measured as a differential connection.

The following configuration is recommended

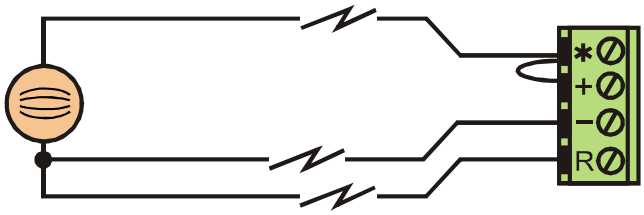


Figure 111 - Connecting a Vibrating Wire Gauge and 3 Wire Copper RTD

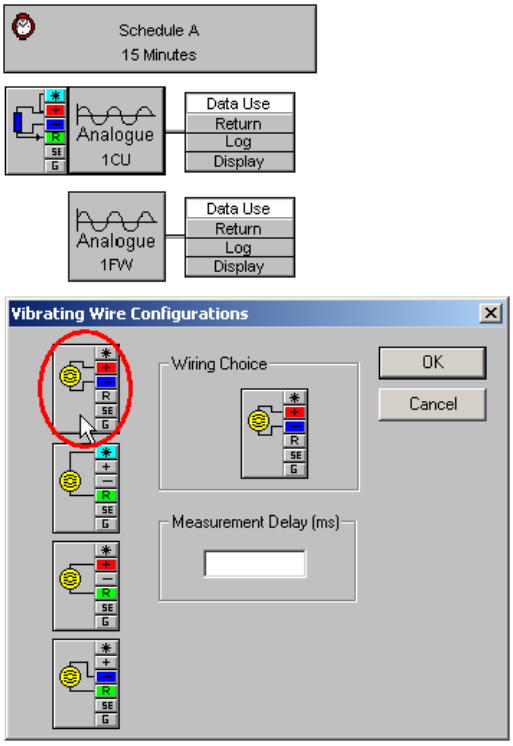
Using DeTransfer, the following command for example

```
BEGIN
 RA15M
  1CU(135)   1FW
END
```

uses the copper coil of the vibrating wire strain gauge as a copper RTD to measure the gauge temperature, and the gauge signal is measured as a differential connection.

The channel factor for the copper RTD is 135 Ohms of coil resistance at 0°C.

Using DeLogger the vibrating wire sensor, and the sensor coil used as a copper RTD, can be measured by the following Program Builder program.



The Digital Input Channels

The *dataTaker 50* data logger has 5 digital input channels, and the *dataTaker 500/600* series data loggers have 4 digital input channels.

The digital input channels of the *dataTaker* share with the digital output channels.

The digital input channels are used for

monitoring bit wide and byte wide logic state inputs

low speed counting to 10 Hz

for detecting digital and counter events to trigger Schedules

The Channel Expansion Module (CEM-AD) has 20 digital input channels which can only be used for monitoring bit wide and byte wide logic state inputs. The digital input channels of the CEM-AD cannot be used as counter inputs or digital outputs.

The Analog Frequency, Analog Period, and Analog State input types can also be considered as digital inputs.

The input type identifier, units text and number of channels for the digital inputs are summarized in the table below.

Digital Input Channels			Number of Channels		
	Channel Type	Units	DT50	DT500 DT600	CEM
Logic State					
Logic state bit	DS	State	5	4	20
Logic state byte	DS(mask)	Byte	1	1	3
Low Speed Counter *					
Accumulating	C	Counts	5	4	0
Resetting	C(R)	Counts	5	4	0
Phase Encoder *					
Accumulating	PE	Counts	1	1	0
Resetting	PE(R)	Counts	1	1	0

* these counters share the digital input channels

Note : The digital input channels of the *dataTaker* and the Channel Expansion Module (CEM-AD) are powered down whenever the logger is operating in low power mode. The digital input channels will not register or count changes of state while powered down.

The high speed counters of the *dataTaker* remain powered at all times, and count pulses while the *dataTaker* is in low power mode.

The digital channels can also be used as open collector outputs to switch external devices.

The Digital Input Circuit

The digital input channels of the *dataTaker* and the Channel Expansion Module (CEM-AD) are TTL and CMOS compatible.

The digital input circuits have a 10 KOhm pull up resistor to 5 Volt, as illustrated in Figure 112. This forces the digital inputs to be normally high, and allows contact closures to be connected directly between the digital input channels and GND which will pull the channel low when contacts are closed.

A series 100 KOhm resistor provides input protection on the digital channels against over voltages being applied.

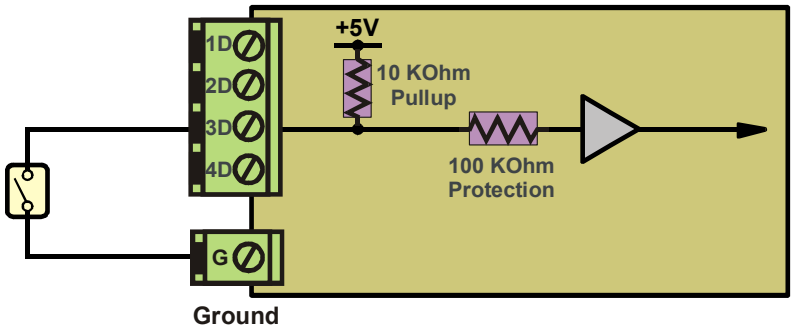


Figure 112 – Digital Input Circuit

Digital Input Sample Period

Whenever the *dataTaker* is operating in normal power mode, the digital input channels (and keys on the display) are read every 50 mS to detect any changes of state.

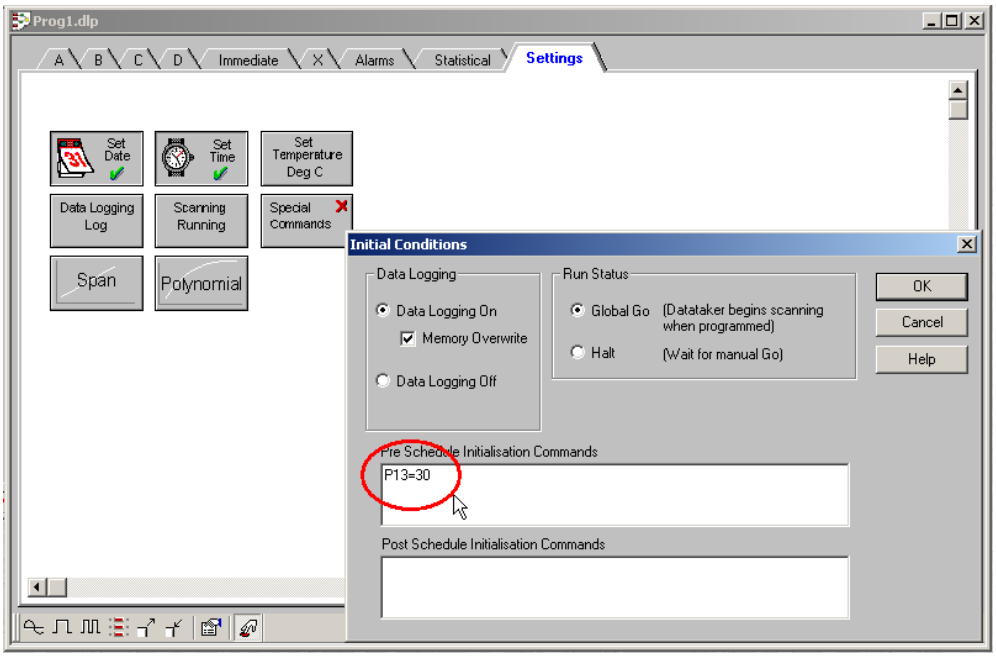
This period is referred to as the digital sample interval, and determines the minimum detectable pulse width. This facility has importance when monitoring large contacts, switches, etc. which can bounce for intervals of 10's to 100's of milliseconds on a closure.

The digital input sample period may be changed by the Parameter13 command (See Section III – Parameter Commands).

Using DeTransfer, the digital sample period is set by the command for example

```
P13=30
```

sets the digital input sample period to 30 mS. The digital input sample period may be set within the range of 1 to 100 mS. The default digital input sample period is 50 mS.



Using DeLogger, the digital sample period is set in the Program Builder under the Settings tab. Right click the Special Commands button to open the dialog, and enter the Parameter13 command in the Pre Schedule Initialization Commands window, as shown above.

Digital Input Modes

The digital input channels of the *dataTaker* are multipurpose, and can be used in a number of modes as follows

used alone for monitoring logic state inputs

combined into groups of sequential channels for monitoring logic byte (nibble) inputs

used alone as low speed counters (except CEM-AD). The low speed counters may be defined as accumulating counters, resetting counters, or up down counters.

The digital inputs can be simultaneously used in more than one of these modes.

For example a digital channel can be used to periodically read the current state of a contact closure, and to count the number of times that the closure was opened or closed over a period of time.

However the individual digital input channels of the *dataTaker* cannot be used as both accumulating and resetting low speed counters in the same task.

Triggering Schedules

The digital input channels can also be used to detect digital events (for example pulses, contact closures, etc) and trigger Schedules to read input channels and execute calculations as follows

detect external digital events for triggering Schedules
(See Section III – Schedules Triggered by Digital Events)

detect counter events resulting from external pulse inputs, for Schedules
(See Section III – Schedules Triggered by Counter Events)

detect external conditions under which Schedules can proceed
(See Section III – Schedules Triggered While Condition)

Note : These triggers can only be used while the *dataTaker* is operating in the normal power mode.

Connecting to Digital Input Channels

Input signals are connected to the digital input channels of the *dataTaker* via the screw terminals as shown in Figure 112.

Digital logic state and low speed counter inputs are connected to the digital input channels between the digital channel terminal and any GND terminal or any unused Analog Return terminals.

Logic inputs from solid state circuits can be connected directly to the digital input channels. The ground of the logic circuit must be connected to any GND terminal of the *dataTaker*.

Contact closures can be connected directly to the digital input channels, between the digital input channel screw terminals and any GND terminal

Monitoring Digital State

The *dataTaker 50* has 5 digital input channels, and the *dataTaker 500/600* series loggers have 4 digital input channels. The digital input channels can be used to monitor digital state.

The Channel Expansion Module (CEM-AD) has 20 digital input channels, which can also be used to monitor digital state.

Monitoring digital state can be implemented in 2 ways

digital bit states, where the digital state of a single digital signal is applied to a single digital input channel. The digital bit data is returned as 0 or 1 with units of State

digital byte state, where

dataTaker 50 - the digital state of 5 parallel signals are applied to the digital input channels. The digital byte data is returned as a decimal value in the range of 0 to 31

***dataTaker 500/600* series loggers** - the digital state of 4 parallel signals are applied to the digital input channels. The digital byte data is returned as a decimal value in the range of 0 to 15

Channel Expansion Module (CEM-AD) - can have multiple digital state inputs of up to 8 parallel signals which are applied to the digital input channels. The digital byte data is returned as a decimal value in the range of 0 to 255

The data reflects the bit pattern of the digital input channels.

Digital state signals may be from a variety of sources, such as contact closures, switches, optical sensors, logic circuits, etc.

The digital input channels are normally high or true when open circuit, because each is internally connected to the internal 5 Volt supply via a 10 KOhm pull up resistor.

Since the digital input channels of the *dataTaker* are bi-directional, the present output state of the channels can be determined by reading the channels as digital bit or byte inputs.

Connecting Digital Bit Inputs

The digital bit inputs are connected to the digital input channels between the selected digital input channel terminal and any GND or unused Analog Return terminal as follows

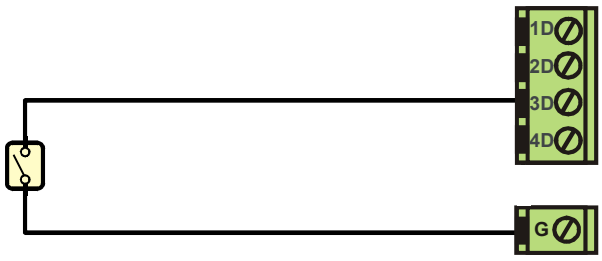


Figure 113 – Connecting Digital Bit Inputs

Note : If the Analog Return terminal is used as the return connection, be aware that this is 100 Ohm above the *dataTaker* ground.

Reading Digital Bit Inputs

Digital bit signals connected to the *dataTaker* are sampled and the data is returned when a Schedule containing the channel is executed.

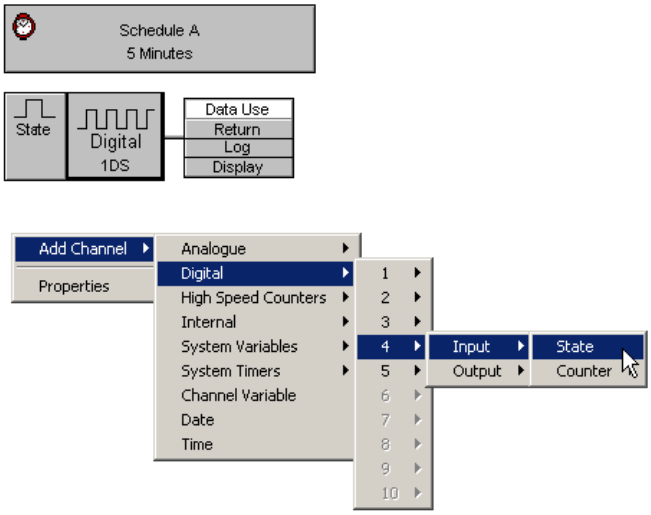
Using DeTransfer, the command for example

```
R5M  1DS  4DS
```

instructs the *dataTaker* to measure the digital state of the signals applied to the digital input channels 1 and 4.

The DS specifies a digital bit signal applied to the digital input channel.

Using DeLogger, digital bit inputs connected to digital channels can be measured by the following Program Builder program. Digital inputs are selected directly from the channel selection walking menu.



Data is returned as 0 or 1 with units of State, depending on the state of the digital bit input.

The *dataTaker* will read the inputs every 5 minutes, and readings are stopped by entering a H (Halt) command.

Connecting Digital Byte Inputs

The individual bits of the digital byte inputs are connected to successive digital input channels between the digital input channel terminal and any GND or unused Analog Return terminal as follows

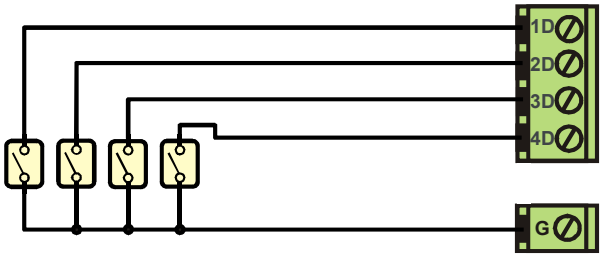


Figure 114 – Connecting Digital Byte Inputs

Digital byte measurement *per se* has limited value with *dataTaker*, because there is only a limited number of bits (channels). However the digital byte is useful if you need to read several digital bit states simultaneously. The result is a decimal value which will indicate the pattern of set and unset bits.

Reading Digital Byte Inputs

With digital byte inputs, where there are a number of digital channels involved, the channel number for the byte is the first channel of the byte. For example, if a 3 bit byte is connected to digital channels 2, 3 and 4, then the channel number for the byte is 2.

The *dataTaker 500/600* series loggers can read up to a 4 bit byte, the *dataTaker 50* can read up to a 5 bit byte, and the CEM-AD can read up to a full 8 bit byte.

The digital byte is assumed to include all digital channels from the starting channel nominated (as the byte channel number) to the last available channel or the eighth channel along, whichever applies. Therefore the default byte from say channel 2 will be a 3 bit byte for *dataTaker 500/600* series loggers, a 4 bit byte for *dataTaker 50*, and an 8 bit byte for the CEM-AD.

If you do not need all subsequent channels from the first channel, then a mask can be set to indicate which channels are in the byte. The mask is a decimal value which is the binary pattern of the channels in the byte, and is entered as a channel option. For example a mask value of 6 (binary 0110) would mean that only channels 2 and 3 are in the byte.

The default mask value is 255, which means all channels up the next eighth.

The digital byte inputs are sampled and the resulting data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the commands for example

```
R10M  1DB

R5S  1DB(7)
```

The first command instructs the *dataTaker* to measure the digital byte of the signals applied to the digital input channels 1 through 4 inclusive (*dataTaker 500/600* series) or input channels 1 through 5 inclusive (*dataTaker 50*).

The second command instructs the *dataTaker* to measure the digital byte of the signals applied to the digital input channels 1 through 3 (binary 11100) of any logger.

The DB specifies that the digital state signals applied to the digital input channels are to be read as a byte.

DeLogger does not support digital byte measurements. However the User channel type (DeLogger Version 4.2.15 or later) can be used, entering the DeTransfer commands described above.

Data from digital byte inputs is returned as a decimal value in the following ranges

dataTaker 50	- 0 to 31
dataTaker 500/600 series loggers	- 0 to 15
Channel Expansion Module (CEM-AD)	- 0 to 255

with units of Byte, representing the bit pattern of the status of consecutive digital input channels.

The *dataTaker* will read the inputs every 10 minutes, and readings are stopped by entering a H (Halt) command.

Low Speed Counters

The *dataTaker 50* data logger has 5 digital input channels, and the *dataTaker 500/600* series loggers have 4 digital input channels, which can be used as counters.

The Channel Expansion Module (CEM-AD) have 20 digital input channels which however cannot be used as counters.

The digital input channels of the *dataTaker* can be used to count digital pulses in any of four modes

accumulating counters, which totalise digital pulses forever. When accumulating counters are read, the current contents of the counter registers are returned

resetting counters, which totalise digital pulses during the interval between readings. When resetting counters are read, the current contents of the counter registers are returned, and the register is reset to zero

accumulating Phase Encoder counter, which increments or decrements counts depending on the phase relationship between pulses input on two digital channels. When the Phase Encoder counter is read as an accumulating counter, the contents of the counter register is returned.

resetting Phase Encoder counter, which increments or decrements counts depending on the phase relationship between pulses input on two digital channels. When the Phase Encoder counter is read as a resetting counter, the contents of the counter registers are returned and the register is reset to zero

Individual counters cannot be simultaneously used as accumulating and resetting counters– if you need total and interval counts, then input the same pulse train into two channels, and run one as an accumulating counter and the other as a resetting counter.

The counters associated with the digital channels of the *dataTaker* are all software counters.

The low speed counters are 16 bit up counters, and have a count range of 0 to 65535. The counters begin at zero counts, count up to 65535, 'roll over' to 0 on the next count, and then continue to count upwards.

The Phase Encoder counter is a 16 bit up down counter, and has a count range of –32768 to 0 to 32767. The counter begins at zero counts, and counts upwards or downwards depending on the phasic relationship of the two input pulse trains.

The low speed counters increment when the state of the digital input switches from high (logic 1) to low (logic 0).

Maximum Counting Rate

All digital counters have 10 Hz count rate capability by default. The maximum count rate is determined by the digital input channel sample period Parameter13, and the number of counters in use.

By default, Parameter13 is set to a sample period of 50 mSec. This means that the digital inputs are sampled every 50 mSec, and so the maximum discernable count rate is 1000/(50 x 2) = 10Hz. If a pulse rise and fall occurs in less than 50 mSec (ie. in the same sample period), then the pulse will not be registered or counted.

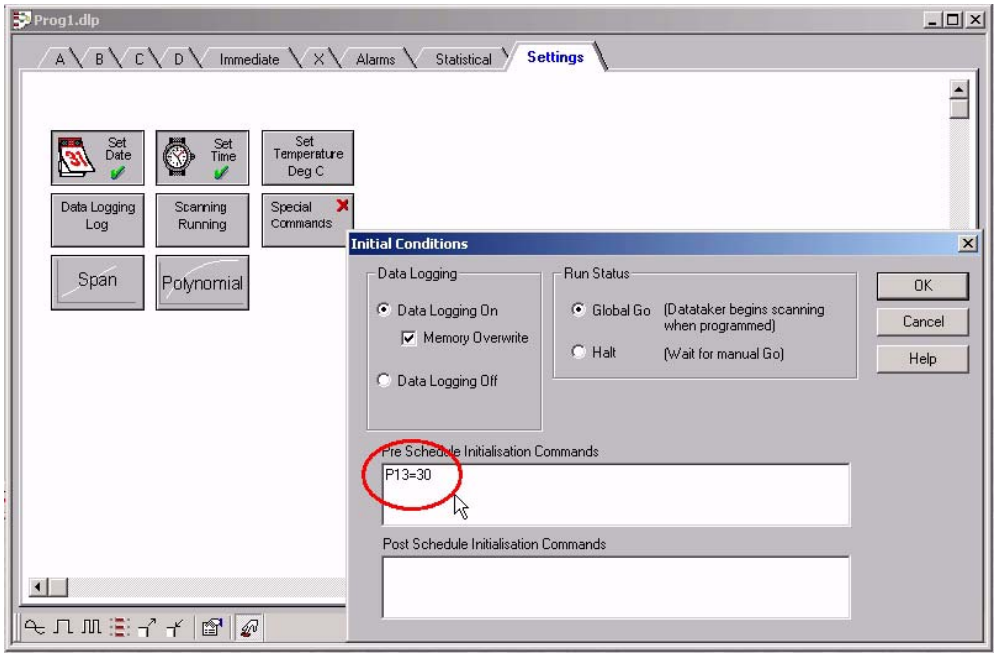
Higher count rates of up to 500 Hz can be achieved if the digital input sample period reduced (digital sampling rate is increased) and other time consuming tasks such as communications, analog to digital conversion, etc. are kept to a minimum.

Using DeTransfer, the digital sampling period is set by the command for example

```
P13=10
```

sets the period to 10 mSec, and translates to a maximum count rate of 1000/(10 x 2) = 50Hz.

Using DeLogger, the digital sampling period is set in the Program Builder under the Settings tab,. Right click the Special Commands button to open the dialog, and enter the Parameter13 command in the Optional Initialization Commands window.



Low Power Mode

The digital input channels of the *dataTaker* are powered down whenever the logger is operating in the low power mode. The digital channels will not register or count any changes of state on the inputs while powered down.

Therefore the low speed counters cannot be used if the logger is to be operated in the low power mode. However, the high speed counters can still be used.

Counts received by the low speed counters while the logger is awake are registered, counted and saved. Although counts are not registered while the logger is asleep, new counts will continue to be counted from the current value of the counters when the logger next wakes.

Connecting Low Speed Counter Inputs

The low speed count inputs are connected to the digital input channels between the selected digital input channel terminal, and any GND or unused Analog Return terminal as follows

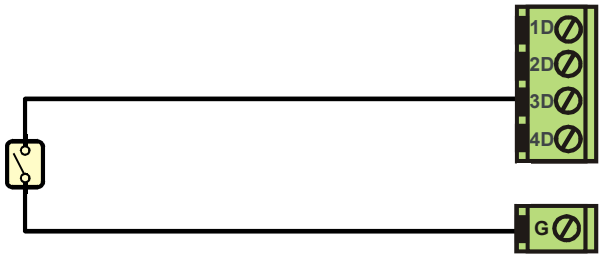


Figure 115 – Connecting a Low Speed Count Signal

Caution : If the Analog Return terminal is used, this is 100 ª above the *dataTaker* ground.

Reading as Accumulating Counters

The low speed counters of the digital input channels are operated as accumulating counters, and the count data is returned to the host, when a Schedule containing the channel is executed.

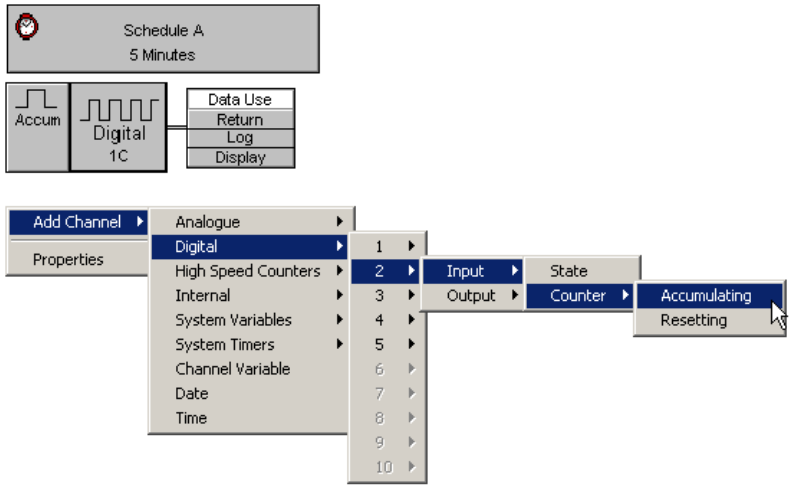
Using DeTransfer, the command for example

```
R5M  1C  3C
```

instructs the *dataTaker* to read the current value for the low speed counters for digital channels 1 and 3.

The C specifies an accumulating low speed counter.

Using DeLogger, the low speed counters can be read as accumulating counters by the following Program Builder program. The accumulating counter mode is selected directly from the channel selection walking menu.



The data is returned in units of Counts. Each successive reading is the accumulation of counts since the command / program was entered into the *dataTaker*.

The *dataTaker* will read the counter every 5 minutes, and readings are stopped by the H (Halt) command.

Reading as Resetting Counters

The low speed counters of the digital input channels are operated as resetting counters, and the count data is returned to the host, when a Schedule containing the channel is executed. The counter is reset each time that it is read.

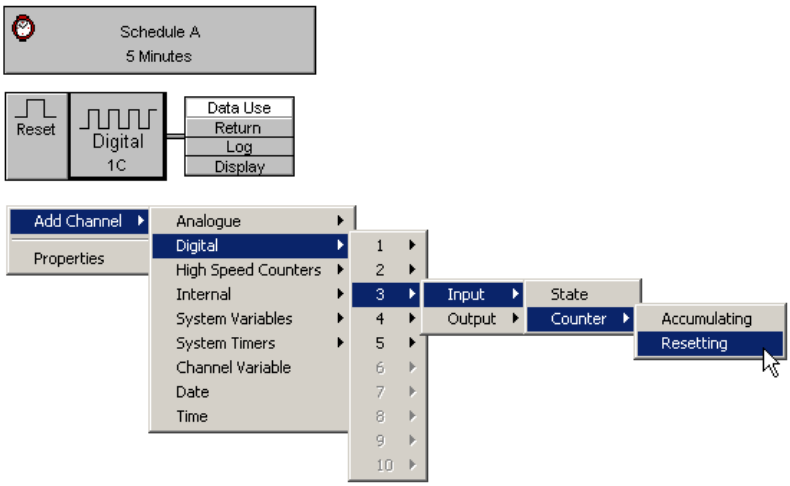
Using DeTransfer, the command for example

```
R5M  1C(R)  2C(R)
```

instructs the *dataTaker* to read the current value for the low speed counter channels 1 and 2.

The C specifies a low speed counter. The R channel option indicates the counter is to be reset to zero when read.

Using DeLogger, the low speed counters can be read as resetting counters by the following Program Builder program. The resetting counter mode is selected directly from the channel selection walking menu.



The data is returned in units of Counts. Each successive reading is the accumulation of counts since the command / program was entered into the *dataTaker*.

The *dataTaker* will read the counters every 5 minutes, and readings are stopped by the H (Halt) command.

Programming and operation of the low speed counters is described in detail in Section III [a]– The Counter Channels.

The Phase Encoder Counter

The *dataTaker*s have a single Phase Encoder counter channel, which increments or decrements counts according to the phase relationship between two pulse trains from a Phase Encoder device.

The Channel Expansion Module (CEM-AD) does not support a Phase Encoder input.

The phase encoder is an implementation of the low speed counters of the digital input channels.

The phase encoder has a variety of applications, for counting bi-phasic pulse trains from linear and rotary motion sensors. These types of sensors are collectively referred to as phase encoders.

The phase encoders output two digital pulse trains, each with the same period but with a phase difference of approximately 90°.

The two digital signals are applied to digital input channels 3 and 4 (3DS and 4DS). The phase encoder support provided by the *dataTaker* is implemented as counter 1PE.

The phase encoder is able to count at up to 10 Hz, and may be used

as an accumulating counter, which totalises the digital pulses forever. When the accumulating counter is read, the current contents of the counter register is returned

as a resetting counter, which totalises digital pulses during the interval between readings. When the resetting counter is read, the current contents of the counter register is returned, and the register is reset to zero

The phase encoder is a 16 bit up down counter, and counts over the range of -32768 to 0 to 32767. The counter begins from zero and counts up or down depending on the phase relationship of the input pulses.

Basic Concepts of Phase Encoding

As an example of a phase encoding device, consider a half opaque disk with two light sensors placed at 90 degrees to one another as illustrated in Figure 115.

As the disc rotates, the light paths are sequentially blocked and unblocked thereby causing the output signal from the light sensors to pulsate. The direction of movement is determined by the *dataTaker* from the pattern of these pulses.

Using the principle described above, but with 360 light/dark transitions on the disk, then the angle of rotation of the disk can be determined as with a precision of ±1 degree.

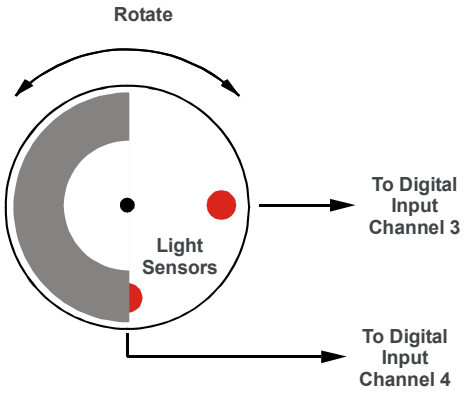


Figure 116 – A Simple Phase Encoder

Figure 116 below illustrates the output signals from the phase encoder and the value of the counter, as the wheel rotates clockwise for 2 turns, and then rotates counter-clockwise for 2 turns.

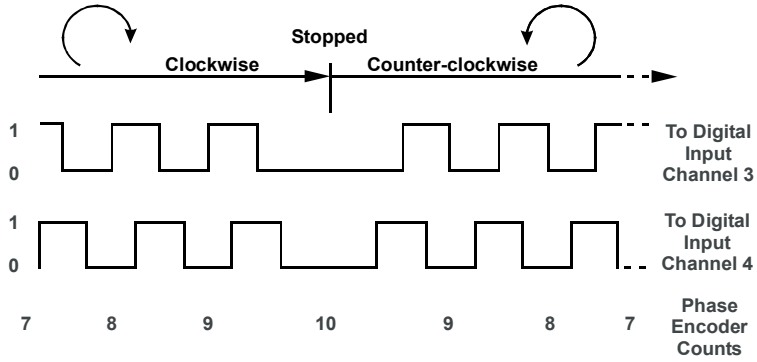


Figure 117 – Operation of a Phase Encoder

When the input to channel 3D is low, then the direction of the transition input to channel 4D determines the direction of counting. The counter is incremented on a transition from 1 to 0 and decremented on a transition from 0 to 1.

The maximum frequency of the phase encoder is determined by the sample period for the digital input channels which is defined by the setting of Parameter13 (See Section II – Low Speed Counters).

Connecting a Phase Encoder

A Phase Encoder can be constructed in various ways, The most common construction however is either

2 microswitches follow individual cam wheels to produce a bi-phasic pattern of contact closures

2 LED/phototransistor pairs which are chopped by a patterned opaque/translucent disc to produce a bi-phasic pattern of logic states

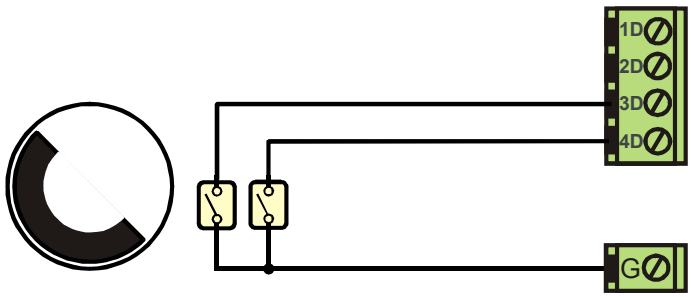


Figure 118 – Connecting a Phase Encoder

Reading as an Accumulating Counter

The phase encoder is operated as an accumulating counter and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
R1M  1PE
```

instructs the *dataTaker* to read the current value for the Phase Encoder counter channel 1.

The PE specifies an accumulating Phase Encoder counter, and the data is returned in units of Counts.

The *dataTaker* will read the Phase Encoder counter every minute, and readings are stopped by the H (Halt) command.

DeLogger does not support the Phase Encoder input type. However the User channel type (DeLogger Version 4.2.15 or later) can be used to enter the DeTransfer commands above.

Reading as a Resetting Counter

The phase encoder is operated as a resetting counter and the data is returned when a Schedule containing the channel is executed.

Using DeTransfer, the command for example

```
R1M  1PE(R)
```

instructs the *dataTaker* to read the current value for the Phase Encoder counter channel 1, and reset the counter to zero.

The PE specifies an accumulating Phase Encoder counter, and the data is returned in units of Counts. The R channel option specifies that the phase encoder is to operate as a resetting counter.

The *dataTaker* will read the Phase Encoder counter every minute, and readings are stopped by the H (Halt) command.

DeLogger does not support the Phase Encoder input type. However the User channel type (DeLogger Version 4.2.15 or later) can be used to enter the DeTransfer commands above.

High Speed Counters

The *dataTaker* data loggers have three high speed counters, which are available as separate input channels from the low speed counters.

The high speed counters are able to count at up to 1 KHz or 1MHz, and may be used as accumulating counters or as resetting counters.

The Channel Expansion Module (CEM-AD) does not have high speed counters.

The high speed counters of the *dataTaker* can be used to count digital pulses in either of two modes

as accumulating counters, which totalise the pulses forever. When accumulating counters are read, the current contents of the counter registers are returned

as resetting counters, which totalise pulses between readings. When resetting counters are read, the current contents of the counter registers are returned, and the register is reset to zero

Individual counters cannot be simultaneously used as accumulating and resetting counters– if you need total and interval counts, then input the same pulse train into two channels, and run one as an accumulating counter and the other as a resetting counter.

The high speed counters are 16 bit up counters, and have a count range of 0 to 65535. The counters begin at zero counts, count up to 65535, 'roll over' to 0 on the next count, and then continue to count upwards.

The high speed counters increment when the state of the digital input switches from high (logic 1) to low (logic 0).

The circuit function schematic for the three high speed counter channels is illustrated below

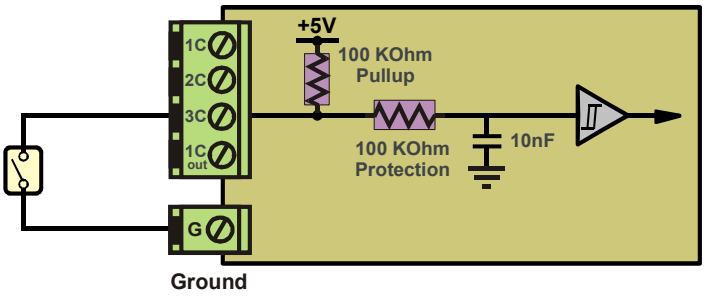


Figure 119 – The High Speed Counters

The high speed counters are normally able to count at up to 1 KHz. However the counters can be modified to increase the frequency to 1MHz by removing filtering capacitors from the counter input. Your *dataTaker* supplier can carry out this modification if required.

Low Power Mode

The high speed counters are hardware counters, in contrast to the low speed counters which are software counters.

The high speed counters continue to register counts while the *dataTaker* is in the low power mode. However the logger can only return high speed counter data while the logger is awake. The maximum count rate decreases while the logger is asleep.

Connecting High Speed Counter Inputs

The high speed count inputs are connected to the high speed counters between the selected counter channel terminal, and any GND or unused Analog Return terminal as follows

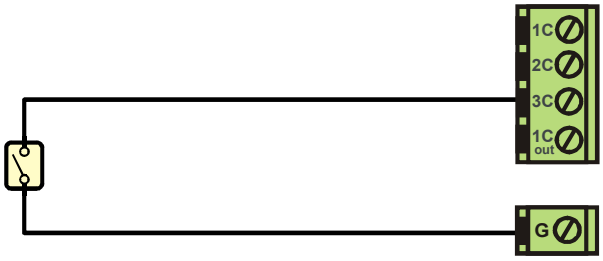


Figure 120 – Connecting a High Speed Count Signal

Reading as Accumulating Counters

The high speed counters are operated as accumulating counters and the count data is returned to the host when a Schedule containing the channel is executed. The counter is not reset.

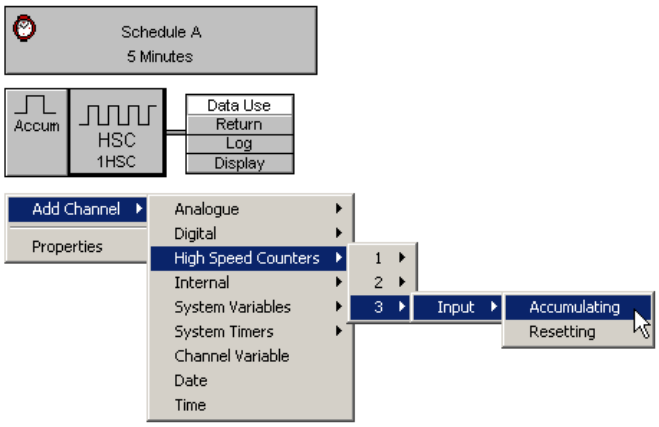
Using DeTransfer, the command for example

```
R5M  1HSC  2HSC
```

instructs the *dataTaker* to read the current values for the high speed counter channels 1 and 2.

The HSC specifies an accumulating high speed counter.

Using DeLogger, the high speed counters can be read as accumulating counters by the following Program Builder program. The accumulating counter mode is selected directly from the channel selection walking menu.



The data is returned in units of Counts.

The *dataTaker* will read the counters every 5 minutes, and readings are stopped by the H (Halt) command.

Reading as Resetting Counters

The high speed counters are operated as resetting counters, and the count data is returned to the host, when a Schedule containing the channel is executed. The counter is reset each time that it is read.

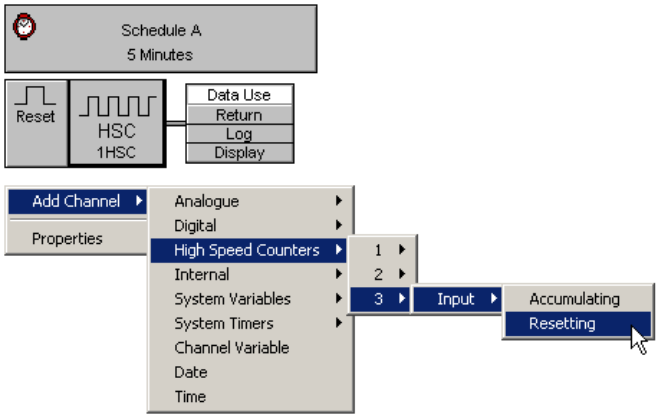
Using DeTransfer, the command for example

```
R5M   1HSC(R)   2HSC(R)
```

instructs the *dataTaker* to read the current values for the high speed counter channels 1 and 2, then reset the counters.

The HSC specifies an accumulating high speed counter. The R channel option specifies that the counter is to be reset to zero when read.

Using DeLogger, the high speed counters can be read as resetting counters by the following Program Builder program. The resetting counter mode is selected directly from the channel selection walking menu.



The data is returned in units of Counts.

The *dataTaker* will read the counters every 5 minutes, and readings are stopped by the H (Halt) command.

Programming of the high speed counters is described in detail in Section III –The Counter Channels.

The Digital Output Channels

The *dataTaker 50* has 5 digital input/output channels, and the *dataTaker 500/600* series loggers have 4 digital input/output channels, which can be used to output digital states. The Channel Expansion Module (CEM-AD) has 10 digital output channels, implemented as 5 open collector outputs and 5 change over relay outputs.

The digital output channels can be addressed either as digital bit outputs, or as digital byte outputs.

The digital output channels are powered down when the *dataTaker* is operating in the low power mode or asleep. Therefore the digital output channels will not hold their output state while the *dataTaker* is asleep. However when the logger wakes, the pre-sleep condition of the digital outputs is restored.

The digital output channels of the loggers are implemented as open collector outputs, able to sink up to 200 mA at 30 VDC. The outputs are buffered, and able to directly drive low voltage relays. Each output has a 15 KOhm pull up resistor to 5 Volts.

The digital output channels 6 - 10 of the Channel Expansion Module (CEM-AD) are also implemented as open collector outputs, able to sink up to 200 mA at 30 VDC. The outputs are also buffered, and able to directly drive low voltage relays. Each output has a 15 KOhm pull up resistor to 5 Volts.

The digital output channels 1 - 5 of the Channel Expansion Module (CEM-AD) are implemented as single pole change over relays, able to switch 240 VAC at up to 5A.

A schematic for the open collector digital output channels is illustrated in Figure 121.

Open Collector Digital Outputs

The open collector digital outputs are TTL/CMOS compatible, and can be used to directly interface to TTL or CMOS circuits. Each channel has a 15 KOhm pull up resistor to 5 Volts.

The open collector outputs are active low. Setting a channel ON results in the voltage at the digital output going to a low state of approximately 0.6 Volts.

The open collector digital outputs are designed to drive relays, solenoids, etc. directly. The actuator device is connected between an external power supply of up to 30 Volts maximum, and a digital output channel. The negative side of the external power supply is connected to the *dataTaker* ground.

The output drivers are not current limited, and so care must be taken to avoid shorting a supply line directly to a digital output.

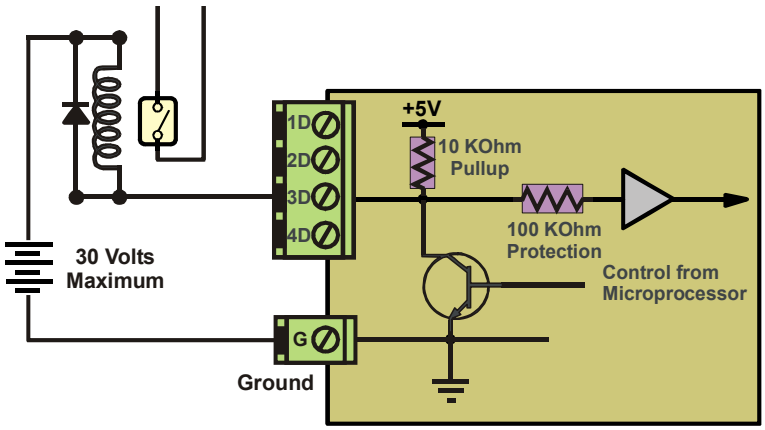


Figure 121 - Digital Input/Output Channel Circuit Schematic

Connecting to Open Collector Digital Outputs

The open collector digital outputs can be connected directly to TTL/CMOS interfaces, and can drive up to 2 TTL loads. The outputs are active low, and are referenced to *dataTaker* ground.

The open collector digital outputs can also be connected directly to low voltage relays, solenoids, etc. The device is connected between the digital output channel and the positive pole of a power supply.

The power supply can be an external power supply of 5 - 30 VDC, or the 5 Volt switched sensor power supply of the *dataTaker*.

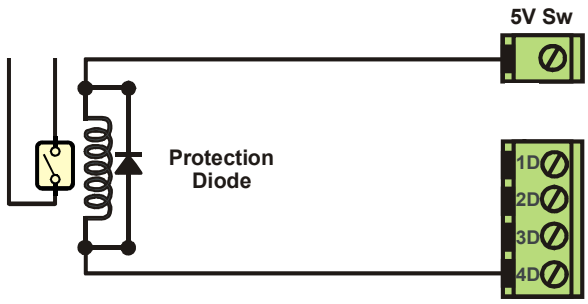


Figure 122 - Connecting Relays to Open Collector Output Channels, Powered from the *dataTaker*

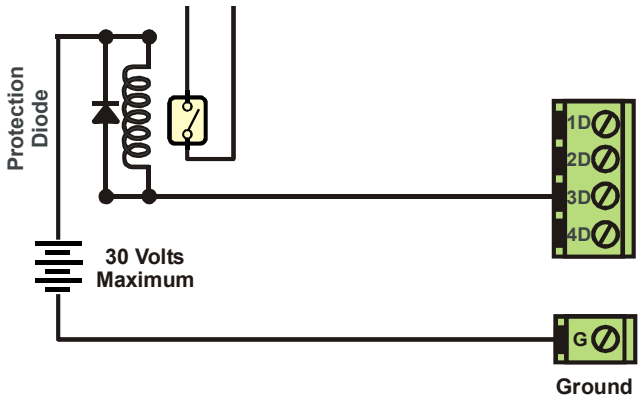


Figure 123 - Connecting Relays to Open Collector Output Channels, Externally Powered

The relay, solenoid, etc. must have a coil voltage/resistance rating which matches the power supply used to drive the device.

While the open collector digital outputs incorporate transient protection for inductive loads, it is a sound practice to place a reversed diode across such loads.

The output drivers are not current limited, and so care must be taken to avoid shorting a supply line directly to a digital output.

Setting Digital Bit Output Channels

The digital output channels are set to required states by assignment commands.

Assigning a 0 to the output channel turns the channel OFF, and the output goes high to approximately 5 Volt (the open collector becomes non-conducting).

Assigning a 1 to the output channel turns the channel ON, and the output goes low to approximately 0.6 Volt (the open collector becomes conducting).

Using DeTransfer, the commands for example

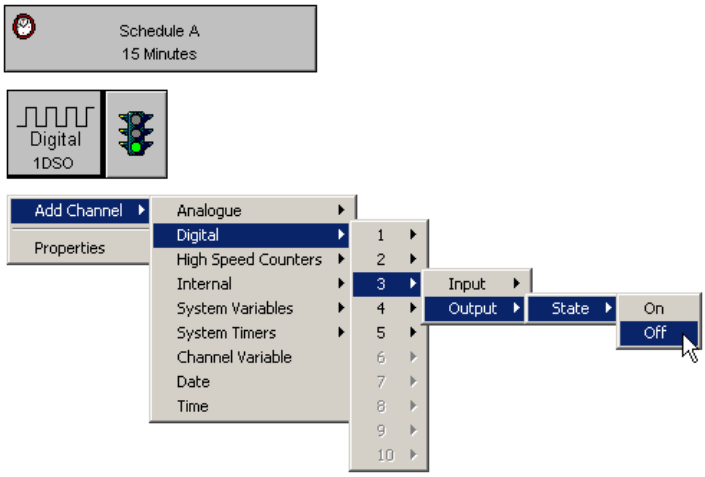
```
1DSO=1
3DSO=0
```

switches the digital output channels as follows

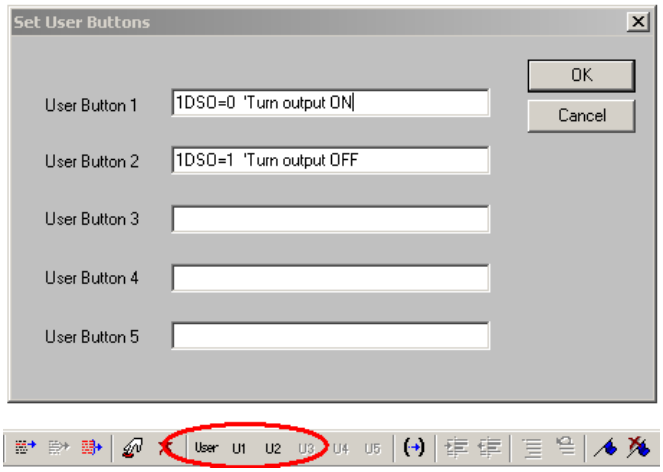
turns the digital output 1 ON, which goes to a low state of approximately 0.6 Volt

turns the digital output 3 OFF, which goes to a high state of approximately 5 Volt

Using DeLogger, the digital output channels can be set from the Program Builder as part of a program as follows. In this case the state of the digital outputs is managed by the Triggered Schedules. The digital output state is selected directly from the channel selection walking menu.



Using DeLogger, the state of digital output channels can also be set directly from the Text View. The appropriate commands are installed behind the User buttons at the bottom of the Entry Screen, and whenever a User button is clicked the corresponding command is sent to the *dataTaker* to change the state of the digital output channel. This switching is independent of the schedules.



Caution : While a digital output channel of the *dataTaker* is turned ON, it cannot be used to monitor digital inputs. However the current state of the output can be read as a digital input.

Use of the digital output channels is described in greater detail in Section III - Commands for Digital Output Channels.

Setting Digital Byte Output Channels

The digital output channels can be set singularly to provide bit outputs, or in groups of channels to provide byte outputs.

The digital byte output channel is set to bit patterns, which are represented by decimal values assigned to the byte channel.

Using DeTransfer, the command for example

```
1DBO=13
```

instructs the *dataTaker* to set the digital output channels to a bit pattern 01101 which corresponds to the digital channels in the order 5 4 3 2 1. This sets the digital output channels as follows

channels 1, 3 and 4 are turned ON

channel 2 (and 5 for *dataTaker 50* and CEM-AD) are turned OFF.

Caution : While the digital output channels are turned ON, they cannot be used to monitor digital inputs. However the current state of the digital outputs can be read as a digital byte input.

Before using the digital channels for monitoring digital states or low speed counting, it is good practice to enter the command

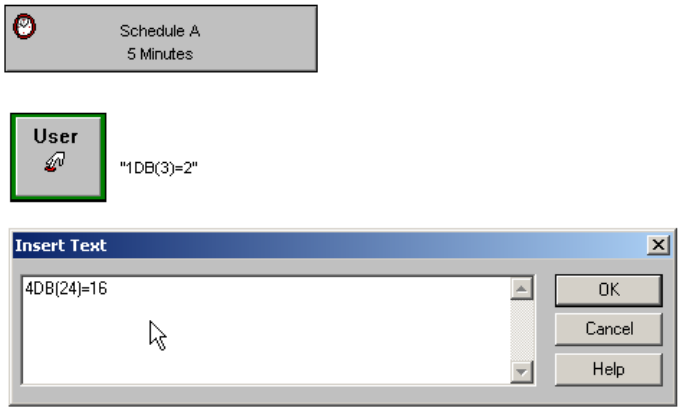
```
1DBO=0
```

to ensure all outputs are OFF.

The digital byte outputs are not directly supported by the Program Builder of DeLogger. However digital byte outputs can be managed

from the User buttons of the Text View as described above

from the User channel type (DeLogger Version 4.2.15 or later) in the Program Builder as follows



Setting of digital output channels is described in greater detail in Section III - Commands for Digital Output Channels.

Channel Expansion Module Digital Output Channels

The Channel Expansion Module (CEM-AD) has 10 digital output channels, configured as 5 open collector outputs and 5 relay contact outputs.

The 5 open collector digital output channels are implemented in the same way as the digital output channels of the *dataTaker* (see above). The open collector digital output channels are channels 6 to 10.

The 5 relay output channels are implemented as contact closures which will switch 240 VAC/DC at 5 Amps. The relay output channels are channels 1 to 5.

The current draw from each digital output relay when switched ON is 35 mA. The relays are automatically switched off when the *dataTaker* enters low power mode (sleep), and the previous relay state is resumed when the logger next wakes.

A circuit schematic for the Channel Expansion Module (CEM-AD) relay digital output channels is illustrated in below

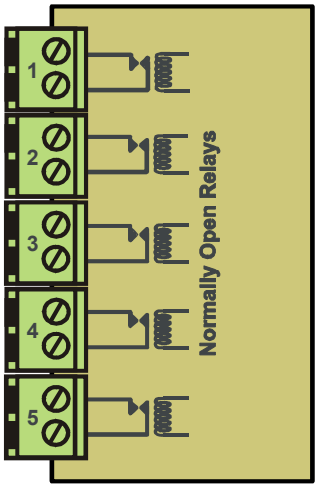


Figure 124 – Relay Output Channel Circuit Schematic for the CEM-AD

Setting Digital Output Channels of the CEM-AD

The digital output commands for the Channel Expansion Module (CEM-AD) are the same as those for the *dataTaker* data loggers. However all digital output commands for the Channel Expansion Module (CEM-AD) are prefixed by the module number.

Using DeTransfer, the commands for example

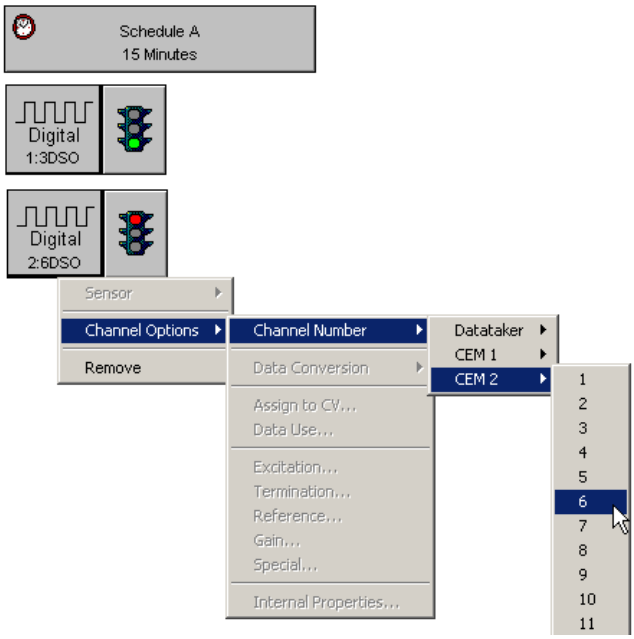
```
1:3DSO=1
2:6DSO=1
```

instruct the *dataTaker* to set the state of the digital outputs of the CEM-AD as follows

CEM-AD module number 1 will switch relay 3 ON

CEM-AD module number 2 will switch open collector output 1 (channel 6) ON

Using DeLogger,



the digital output channels on the CEM-AD can be set from the Program Builder as part of a program. This is done as follows

- firstly set any digital output channel of the logger, say digital output 1, to the required state as described above
- then reassign this channel to the CEM by right clicking on the new 'pseudo' channel icon, and select Channel Options:Channel Number:CEM*n*:*channel number*

Setting of digital output channels for the Channel Expansion Module (CEM-AD) is discussed in greater detail in Section III - Commands for Digital Output Channels, and Section II - Operation of the Channel Expansion Module).

The Channel Expansion Module

The *dataTaker 500/600* series loggers can be expanded by connecting 1 or 2 Channel Expansion Modules (CEM-AD) in a daisy chain configuration.

The Channel Expansion Module (CEM-AD) has a channel capacity as follows

10 analog input channels, implemented as 10 differential or 30 single ended channels which can be used in any mix

20 digital input channels, implemented as TTL/CMOS/Voltage free contact compatible channels

10 digital output channels, implemented as 5 contact closure outputs and 5 open collector outputs

Installing the Channel Expansion Module

One or two Channel Expansion Modules (CEM-AD) can be connected to a *dataTaker 500/600* series logger. The connection between the *dataTaker 500/600* series loggers and the Channel Expansion Modules is via 25 pin connectors located at one end of the logger, and at each end of the Channel Expansion Modules.

The data logger and the modules are interconnected by 25 core cable. Each cable must not exceed 50 cm (20 ins) in length, and the total length of cables including the length of the logger and modules must not exceed 2 metres (6 feet).

Note: The Channel Expansion Modules must not be connected to the dataTaker while the logger is powered.

Disconnect external power and all batteries from the *dataTaker* before connecting the Channel Expansion Module to the logger. The Channel Expansion Module is designed as a permanent fixture for the *dataTaker*, and must not be disconnected until power and all batteries are disconnected.

The Channel Expansion Module is powered by the *dataTaker*, and does not require a separate power supply.

The footprint size of the Channel Expansion Module is identical to that of the *dataTaker*, and is compatible with the range of *dataTaker* enclosures as follows

Large Industrial Enclosure - accommodates one *dataTaker* and one Channel Expansion Module (CEM-AD)

Small Industrial Cabinet - accommodates one *dataTaker* and two Channel Expansion Modules (CEM-AD)

The *dataTaker* and Channel Expansion Module(s) can also be fitted into any suitably sized enclosure, either side by side or end to end depending on the shape and size of the enclosure.

When using thermocouples with the *dataTaker* and the Channel Expansion Modules, it is recommended that the *dataTaker* and Channel Expansion Modules be fitted into a suitable metal enclosure to avoid thermal gradients across the modules. A metal enclosure also protects against magnetic and electrical interference on the modules and interconnecting cables.

Channel Expansion Module Number

The Channel Expansion Modules (CEM-AD) do not have a hardware address. Once the interconnecting cable is fitted and the *dataTaker* is powered up, each Channel Expansion Module available is identified and numbered in software as module 1 and module 2.

The Channel Expansion Module connected to the *dataTaker* becomes module number 1, and the next Channel Expansion Module becomes module number 2.

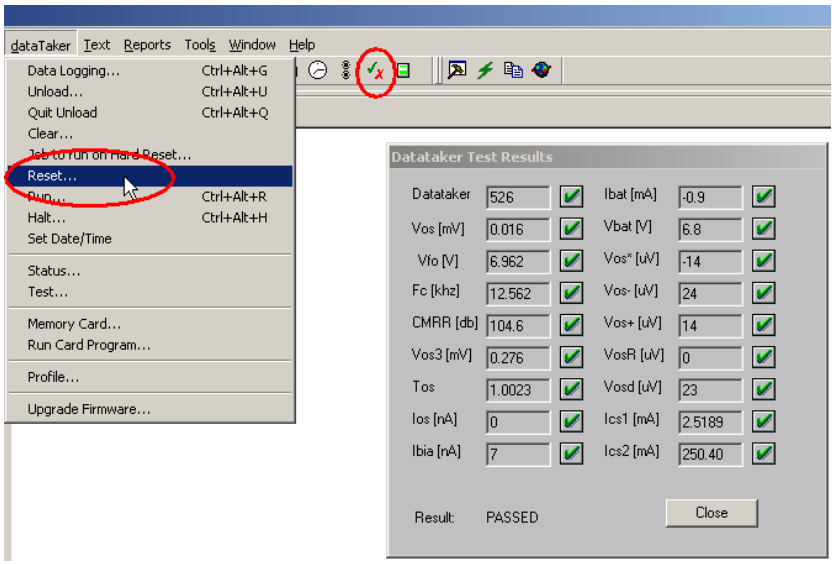
To confirm the connection of a Channel Expansion Module to the *dataTaker*, enter the TEST command from DeTransfer as follows

```
TEST
```

The TEST command will return a report as follows

```
dataTaker 526  Ver 7.00
Vos (mV)           0.117
Vfo  (V)           7.873
Fc (kHz)           17.784
CMRR(db)           101.2
Vos3(mV)           0.322
Tos                1.0022
Ios (nA)           -3
Ibia(nA)           13
Ibat(mA)           -100.0
Vbat (V)           6.3
Vos*(uV)           -95
Vos+(uV)           33
Vos-(uV)           10
VosR(uV)           66
Vosd(uV)           5
Ics1(mA)           2.5011
Ics2(uA)           249.93
PASS
```

Using DeLogger, the TEST command is run by clicking on the TEST button on the main toolbar of DeLogger, or by selecting Reset… under the dataTaker menu..



The first line of the TEST report contains information about the configuration of the *dataTaker* (See Section III – TEST Command) and version of firmware.

The digit 6 is appended to the `dataTaker xxx` for each Channel Expansion Module that is connected to the logger.

Connection of the Channel Expansion Module(s) to the *dataTaker 500/600* series loggers is illustrated below.

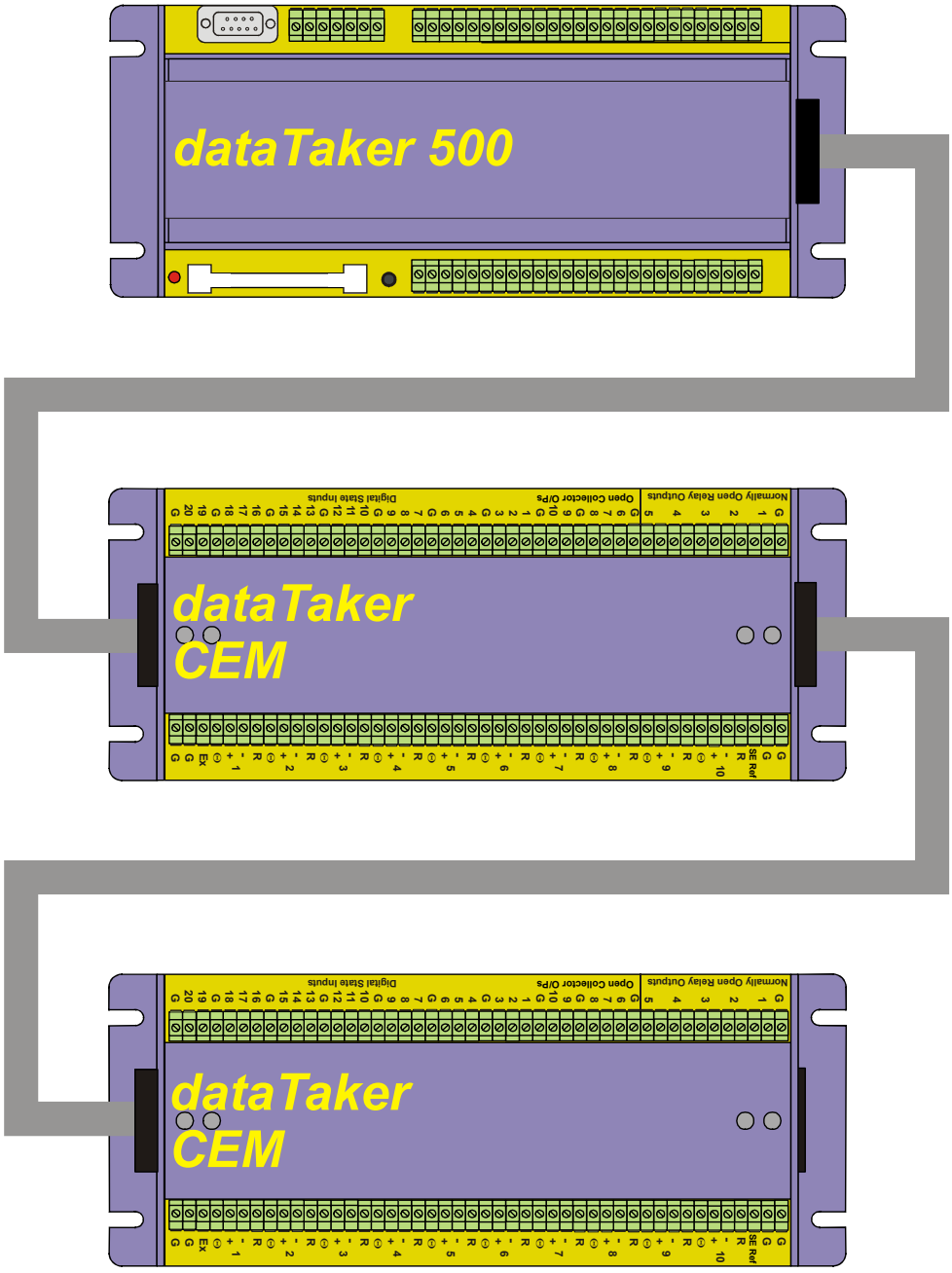


Figure 125 - Channel Expansion Module Connection

Commanding the Channel Expansion Module

The Channel Expansion Module (CEM-AD) provides increased channel capacity for the *dataTaker 500/600* series loggers. In general the channel and signal measurement specifications of the Channel Expansion Module (CEM-AD) are the same as the *dataTaker* model to which it is connected.

These channels and signal types are commanded in the same manner as for the *dataTaker* to which the Channel Expansion Module(s) are connected.

The Channel Expansion Module has a relay multiplexer for the analog input channels, similar to that for the *dataTaker 505,515, 605* and *615*.

If the Channel Expansion Module is connected to a *dataTaker 500* or *dataTaker 600* data logger, then the module has a maximum input voltage of ±2500 mV and a common mode range of ±3.5 Volts.

If the Channel Expansion Module is connected to a *dataTaker 505*, *dataTaker 515*, *dataTaker 605*, *dataTaker 615* data logger, then the module has a maximum input voltage of ±100 Volts and a common mode range of ±100 Volts.

The Channel Expansion Module is capable of withstanding in excess of 500 Volts for 50 mS, or 100 Volts continuously, on any unselected analog input channel.

Module Number

The commands used to specify channels of the Channel Expansion Module (CEM-AD), and channel options for channel configuration, follow the same conventions as those used for the *dataTaker*.

However each command directed to a Channel Expansion Module is prefixed by a module number and a colon character as follows

n:channel type(channel options)

where

n: the CEM number, either 1 or 2.

Using DeTransfer, channels on a Channel Expansion Module are specified as follows

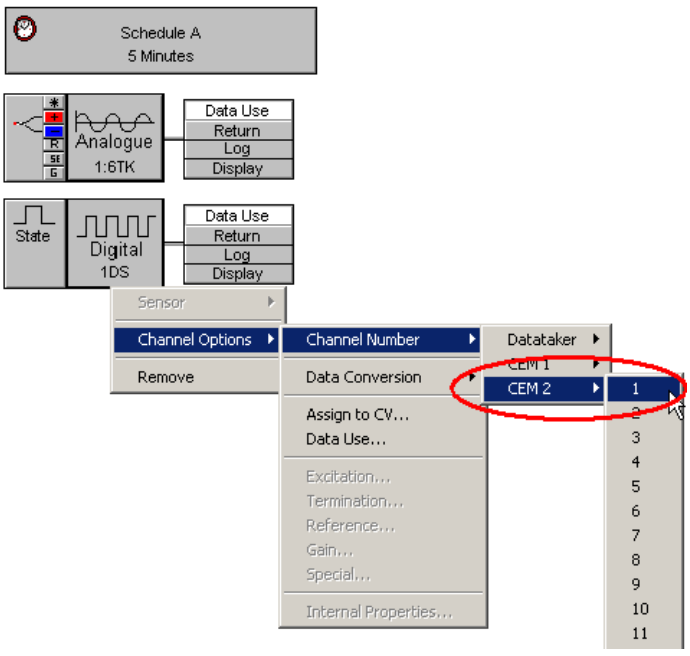
```
1:6TK  2:1..3DS
```

which specifies

a K type thermocouple connected to analog channel 6 of CEM number 1

digital state channels 1 through 3 of CEM number 2

Using DeLogger,



channels on the CEM-AD are selected in the Program Builder as part of a program as follows

Analog Channels:

firstly select any analog channel of the logger, say analog channel 10, and attach the sensor type that is connected to the target channel on the CEM

then reassign this channel to the CEM by right clicking on the new 'pseudo' channel icon, and select Channel Options:Channel Number:CEM*n*:*channel number*

Digital Channels:

firstly select any digital channel of the logger, say digital channel 1, and select the state or counter input type that is connected to the target channel on the CEM

then reassign this channel to the CEM by right clicking on the new 'pseudo' channel icon, and select Channel Options:Channel Number:CEM*n*:*channel number*

The raw data returned for channels of the Channel Expansion Module have the module number prefixed to the channel identification as follows

```
1:6TK  156.65 Deg C
2:1DS  1 State
2:2DS  0 State
2:3DS  1 State
```

Internal Channels

The Channel Expansion Module (CEM-AD) has two internal channels, which are used as reference channels when monitoring thermocouples.

The two internal channels are the module temperature which is sensed by a LM35 monolithic temperature sensor, and an electrical zero channel.

These internal channels are identified in the general format as follows

n:1%LM35	internal temperature channel
n:2%V	electrical zero channel

where *n* is the module number.

Over Voltage Protection

The Channel Expansion Module (CEM-AD) does not include built in energy absorbing lightning protection.

However all analog input channel terminals except for the Analog Return terminals are capable of withstanding 1.5KV for 10µS, 500V for 50mS and 100V indefinitely. The Analog Return terminal can withstand only 5% of these voltages.

The above withstanding voltages apply only to unselected channels. While a channel is selected for reading (a process that typically takes 30mS), these withstanding voltages are reduced to the common mode voltage range of the *dataTaker* as follows

dataTaker 500 and 600	±5 Volts relative to ground
dataTaker 505, 515, 605 and 615	±100 Volts relative to ground

In applications where scanning is infrequent (say not more than every 3 hours), the probability of a scan being co-incident with a lightning strike is very low.

Where lightning is frequent, it is strongly recommended that external energy absorbing lightning protection be wired to each sensor line. Further, it is recommended that the *dataTaker* 505, 605, 515 or 615 loggers which have higher withstanding voltages be used.

Using Thermocouples

The Channel Expansion Module (CEM-AD) fully supports thermocouples for cold junction compensation and zero voltage compensation.

The Channel Expansion Module has a LM35 temperature sensor located so that it senses the temperature of the analog input channel screw terminals. These terminals are the thermocouples reference junction.

The *dataTaker* will automatically measure the Channel Expansion Module temperature and electrical zero when scanning a thermocouple channel on the module. These values are then used for reference junction compensation.

The accuracy of thermocouple measurement is dependent on the isothermal condition of the reference junction. If a temperature gradient develops between the modules temperature sensor and the input terminals, the error will approximately equal the temperature difference.

The module utilises relays for channel selection and digital output. These are a source of heat and each relay dissipates 150mW when switched on. For maximum thermocouple accuracy, ensure that these relays are not left on unnecessarily.

A not so obvious reason that a relay may be left on is that the last channel in a scan remains selected if Parameter15 is set to 1 or 2. This can be resolved if P15 is set to zero, or a dummy channel is placed at the end of the Schedule. A good choice would be 1%V(M18:156,W).

Power Consumption

The Channel Expansion Module (CEM-AD) consumes very little power if the *dataTaker* to which it is attached is allowed to sleep.

While scanning channels on the Channel Expansion Module, the current drawn from the *dataTaker* will increase to approximately 60mA. Additional current is required if the digital output relays are switched on. Each of the five relays will draw 35mA, however these relays are automatically switched off when the *dataTaker* sleeps.

A summary of current draw is tabulated below:

Condition	***Current Draw***
Sleep	100uA
Awake, no scanning or digital I/O	100uA
Digital inputs grounded	4mA
Scanning CEM channels	60mA
All relay outputs ON (1..5DSO=1)	175mA
Worst case condition	240mA

External Excitation

The Channel Expansion Module (CEM-AD) can power or excite sensors from the Excite (⁋) terminal of the analog channels. Excitation can be provided either by the standard current and voltage sources, or by an external power source.

For sensors that require non-standard powering or excitation, an External Excitation input terminal is able to direct power to the Excite (⁋) terminal of selected analog channels.

This option must be enabled by moving a jumper to the outer two pins on a three pin header. For location details of the jumper See Section I - Construction of the Channel Expansion Module.

The normal position of the jumper (between the inner two pins) provides the standard *dataTaker* 250.0 µA, 2.500 mA or 4.5V excitation.

If the external excitation option is selected these standard excitations are not available on any channel of module. Care must be taken in assigning channels. If for example an Expansion Module is wired with a four wire RTD, and several bridges requiring 12 Volt excitation, then the 12 Volt would be applied to the RTD when it is scanned. The RTD would probably be damaged.

Installing the Panel Mount Display

The *dataTaker 50* and *dataTaker 500* series data loggers support an optional Panel Mount Display.

The *dataTaker* 600 series loggers have an integrated display module, and do not support the separate Panel Mount Display.

The Panel Mount Display has a 2 line x 16 character LCD display, which is backlit, and has five keys which control scrolling of the display.

The Panel Mount Display provides a means for local indication of channel data, alarm status and general system information. This information is displayed on the LCD display.

The features of the Panel Mount Display include

- indication of input channel data in numerical or bar graph format
- indication of alarm status, and of alarm channel data in numerical or bar graph format
- indication of various system status, including scanning activity, data memory usage, battery status, etc.
- 6 warning output channels, comprising 3 LEDs, a buzzer and two levels of LCD backlight control
- 5 keys defined as follows
  - Backlight Key to toggle the display backlight
  - List Key to enable/disable selected screens
  - Function Key to access predefined functions
  - Up and Down Scroll Keys to view data and status screens

Installing the Panel Mount Display

The Panel Mount Display can be flush mounted into the door or the top of any standard *dataTaker* enclosure, or flush or surface mounted onto panels, walls, etc.

The Panel Mount Display is supplied with a 1.5 metre ribbon cable connected to the unit. Longer cables are available by special order from your *dataTaker* supplier.

The Panel Mount Display is designed for local use only, and cable length should not exceed 5 metres. This alleviates the likelihood of electrical noise being induced into the cable. Longer cables can be obtained from your *dataTaker* supplier.

Connection of the cable to the Panel Mount Display is via one of two 13 pin plugs located on the back of the module. These two plugs are identical, so choose the most suitable plug for routing of the cable.

The Panel Mount Display should NOT be connected to or disconnected from the *dataTaker* while the logger is powered. Connecting or disconnecting the Panel Mount Display cable while the *dataTaker* is powered may cause the logger to lock up or reset, thereby risking the loss of stored data or program. Remove all sources of power before connecting or disconnecting the cable.

Installation simply involves connecting the Panel Mount Display to your *dataTaker*, and mechanically installing the display module into an enclosure or onto a panel as required.

Connecting the PMD to the dataTaker 50

Use the following procedure when connecting the Panel Mount Display to a *dataTaker* 50

- disconnect all sensors, computer links and power sources from the *dataTaker 50*. Remove any removable screw terminal blocks.
- remove the two screws from each end of the logger which fix the top cover to the chassis, and remove the top cover.
- orientate the logger such that the memory card socket is in the upper right hand corner.

the 13 pin connector for the Panel Mount Display is located near the lower right hand corner of the upper printed circuit board, beside the battery connector.

If your *dataTaker 50* does not have a 13 pin connector, then consult your *dataTaker* supplier.

connect the free end of the Panel Mount Display ribbon cable onto this 13 pin connector. The connector is polarized such that it locates in only one orientation.

carefully fold the ribbon cable over the ribbon cable connector, and down the front side of the logger chassis.

reinstall the top cover of the *dataTaker* 50, ensuring that the ribbon cable exits the logger between the front side of the chassis and the overlapping side of the top cover.

reinstall the two screws in each end of the logger which fix the top cover to the chassis.

power up the logger and test the Panel Mount Display for function.

Connecting the PMD to the dataTaker 500 Series

The following procedure should be followed when connecting the Panel Mount Display to a *dataTaker 500* series logger

disconnect all sensors, computer links and power sources from the *dataTaker* 500 series logger.

remove the screws which fix the top cover of the logger to the chassis, and remove the top cover.

if your *dataTaker* has a battery tray, then remove the screws fixing the battery tray to the end plates, and the screw in the centre of the battery tray. Carefully lift the battery tray upwards to expose the internal battery connector, and unplug the internal battery lead.

remove the end plate with the cut-out for the DB25 pin connector.

orientate the logger such that the memory card socket is in the lower left hand corner.

the 13 pin connector for the Panel Mount Display is located near the lower right hand corner of the upper printed circuit board, behind the screw terminals for analog input channel 5. The connector is labelled J4.

If your *dataTaker 500* series logger does not have a 13 pin connector, then consult your *dataTaker* supplier.

if your *dataTaker 500* series logger has two shorting links on this 13 pin connector, then remove these.

pass the free end of the Panel Mount Display ribbon cable through the cut-out for the DB25 pin connector in the end plate, and plug the ribbon cable connector onto the 13 pin connector. The ribbon cable connector is polarized such that it locates in only one orientation.

carefully lay the ribbon cable back over the body of the DB25 connector, and then fold at 90° so that the ribbon cable passes across the DB25 and the end of the logger.

reinstall the end plate, with the ribbon passing through the cut-out above the DB 25 pin connector.

reconnect the internal battery lead to the battery connector on the upper printed circuit board.

reinstall the battery tray if your *dataTaker* has one

reinstall the top cover of the *dataTaker 500* series logger, and fit the screws which fix the top cover to the chassis.

power up the logger and test the Panel Mount Display for function.

Mechanical Details

The Panel Mount Display can be mounted in the door or top of any of the *dataTaker* enclosures, or other enclosures, or flush or surface mounted onto panels, walls, etc.

The *dataTaker* series data loggers can be purchased in any of the standard enclosures, with the Panel Mount Display fitted into the door or top of the enclosure.

Views of the rear and front of the Panel Mount Display are illustrated below

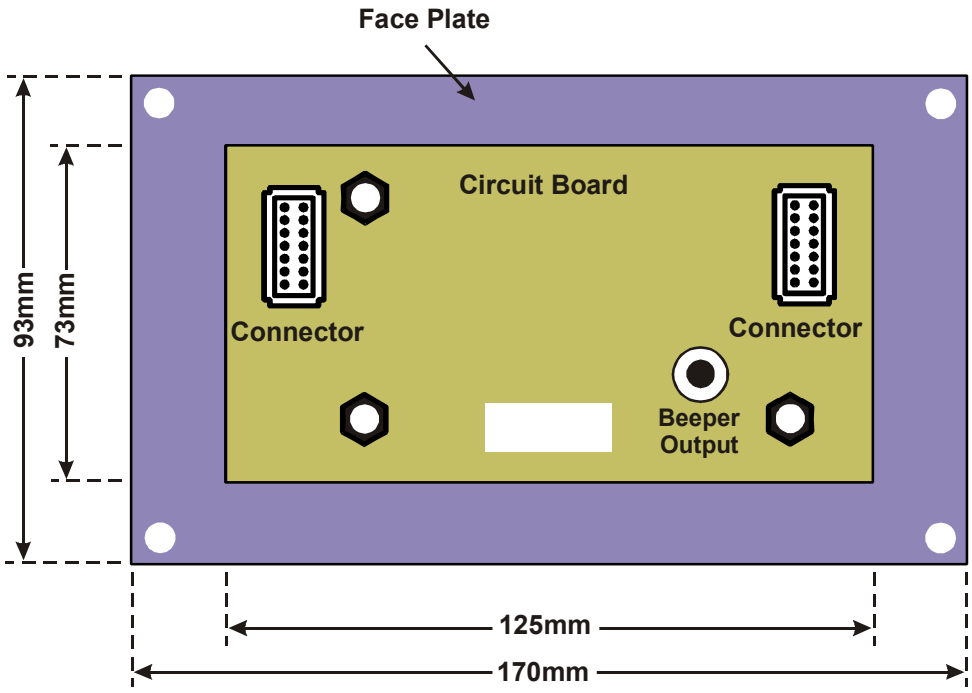


Figure 126 - Rear View of Panel Mount Display

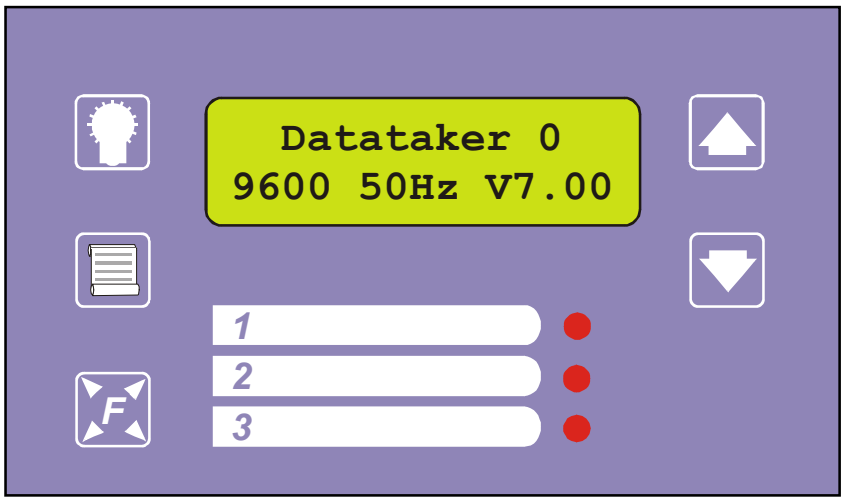


Figure 127 - Front View of Panel Mount Display

Section III

Programming

the

dataTaker 50,500,600 Series 3

Data Loggers

Programming the *dataTaker*

The *dataTaker 50/500/505/515* and *dataTaker 600/605/615* series of data loggers can all be supervised and programmed in either of two ways

using the DeTransfer software to enter simple English-like commands into the logger. Any terminal emulator or communications package can be used to program the *dataTaker* using these commands.

The *dataTaker* commands control the various functions of the logger, and are composed of printable ASCII characters. No special characters or coding techniques are necessary to program the logger.

The DePlot software can be used to chart data in realtime, or to chart stored data when it is later recovered from the logger.

using the DeLogger software, which provides a Graphics User Interface (GUI) from which to supervise and program the *dataTaker*.

The logger is programmed from an icon based programming module, and supervision of the logger is by point and click through various menus and toolbars.

DeLogger also incorporates charting modules, tabulation modules, mimics modules, spreadsheet modules and access to other software packages for visualization and presentation of data.

Both of these software applications will send commands and programs directly to the loggers. Both will also record programs into a memory card, which are read by the *dataTaker* when the memory card is later inserted into the card socket of the logger. In this way remote loggers can be programmed without need for a local computer.

DeTransfer and DeLogger

DeTransfer and DeLogger software run on any desktop or notebook computer running any Microsoft Windows operating system. Communications between DeTransfer or DeLogger and the *dataTaker* is via a serial cable connected between any COM port of the computer, and the RS232 COMMS serial interface of the logger.

The *dataTaker* data loggers are normally supplied with DeTransfer, DePlot and a basic version of DeLogger bundled with the logger.

An advanced DeLogger Pro version can be purchased from your *dataTaker* supplier, which provides database access for storing data, a scripting facility for automating DeLogger, modem support for remote communications, and additional data visualization and presentation modules.

Note : The *dataTaker* data loggers can be purchased in volume without accessories, in which case this software is not bundled.

If You are Using DeTransfer to Program Your dataTaker

If you are going to use DeTransfer to supervise and program your *dataTaker*, then you will need to develop a reasonably good understanding of the *dataTaker* command set that is described through the chapters of Section III of this manual.

Using the *dataTaker* command set from DeTransfer is a faster and more succinct way of using the *dataTaker*, however provides little for analysis, visualization and presentation of resulting data. You will have to turn to other software packages such as Excel, charting tools, etc. for this.

If You are Using DeLogger to Program Your dataTaker

If you are going to use DeLogger or DeLogger Pro to supervise and program your *dataTaker*, then you do not need to know much about the *dataTaker* command set that is described through the chapters of Section III of this manual. In fact you can close the book now if you wish, and go collect some data.

However, we recommend that you at least skip read through the chapters of Section III to develop an overview of the underlying principals of the *dataTaker* command set, and an overview of how the command set works.

This will give you greater understanding of the capabilities of the logger, and of how DeLogger supervises the logger. This will help you achieve sooner and better projects.

The dataTaker Commands

The *dataTaker* commands consist of upper case characters. Lower case characters are ignored unless preceded by /, and so can be used to self-document commands. The exception is the Switch commands (/) where lower case characters have significance in determining the sense in which the particular switch function is to be set.

The *dataTaker* commands fall into a number of general categories

- Scheduling commands
- Alarm commands
- Internal data memory and memory card management commands
- Input/Output channel specification commands
- Output channel setting commands
- Logger addressing commands for networks
- Parameter commands for system configuration
- Switch commands for system configuration
- System commands

Each of these groups of commands is discussed in detail in the following chapters.

Important Concept Number 1 - Schedules and Schedule Lists

General operation of the *dataTaker* data loggers centres around scheduling. Schedules determine when various processes are to occur, and can be triggered by the real time clock, by digital or counter events, by alarm states, and by the host computer. The *dataTaker* supports 7 different types of schedules.

The Schedules manage schedule lists, which contain the various actions that are to be performed when the Schedule is triggered to execute or run. These actions include reading input channels and internal channels, setting output channels, executing calculations, testing alarms, returning data, logging data, etc.

Many of the *dataTaker* commands relate to the Schedules and their schedule lists. Therefore it is important to understand this core concept of Schedules and scheduling.

Important Concept Number 2 - Parameter Commands and Switch Commands

The *dataTaker* is highly configurable, allowing it to be tailored for a wide range of uses and installations. Much of this configuration is global 'look and feel' of the logger, and is achieved through use of the many Parameter and Switch commands

Frequent reference is made to the Parameter and Switch commands that allow

- defining the format of returned data
- defining the format of time and date of the real time clock
- customising the analog to digital converter
- customising various system settings

Each of these commands is discussed in detail throughout the manual in the context of their respective applications, and is also discussed collectively in Section III – Parameter Commands and Section III – Switch Commands.

Communication Protocols and Commands

The *dataTaker 50/500/600* series loggers have several protocols and commands, which control the behaviour of the logger as a communicating device.

These protocols are all obeyed by DeTransfer and DeLogger, and so if you are using either of these applications to supervise and program your *dataTaker* then you can skip this chapter.

However if you are using some other application or tool to supervise and program your *dataTaker*, or are developing a driver for say a SCADA package, then you need to be aware of the *dataTaker* communications protocols.

Communications Baud Rate

The communications baud rate for the RS232 COMMS serial interface can be set by adjusting the DIP switches accessible behind the memory card socket.

Details for setting the baud rate for the RS232 COMMS serial interface are given in Section II – The RS232 COMMS Interface.

The baud rate DIP switches are read and used to initialise the serial interface every 50 millisecs when the digital inputs are strobed (debounce period). This ensures that the *dataTaker* is always sensitive to changes in baud rate.

Data Bits, Stop Bits and Parity

The RS232 COMMS serial interface of the *dataTaker* has the number of data bits permanently set to 8, the number of stop bits permanently set to 1, and the parity checking permanently disabled.

The serial interface of the host computer or terminal should be set to the same settings.

XON/XOFF Protocol

The *dataTaker* data loggers and DeTransfer and DeLogger all supports an XON/XOFF handshake protocol, which controls

the transmission of commands and programs from DeTransfer or DeLogger running on the host computer, to the *dataTaker*

the transmission of data from the *dataTaker* to DeTransfer or DeLogger running on the host computer

The protocol prevents data loss or corruption by the end receiving the transmission, due to communications overrun.

This XON/XOFF handshake protocol is permanently enabled in the *dataTaker* in both directions. Following initial power up, a hardware reset, or a RESET command, the *dataTaker* can receive commands and transmit data immediately (XON true in both directions).

When transmitting data, the *dataTaker* halts transmission within two byte (or character) periods of an XOFF character being received from the host, and resumes transmission when a subsequent XON character is received. This allows the host computer to control the transmission of data from the *dataTaker*.

When receiving long programs comprising many commands, the *dataTaker* issues an XOFF character when the input buffer becomes 50%, 75% and 90% full, and then issues an XON character when the input buffer is empty. This allows the *dataTaker* to control the transmission of programs from the host computer.

The XOFF character is ASCII 19 (DC3), and can be sent directly from the keyboard of a host running DeTransfer or any other communications software as Ctrl S.

The XON character is ASCII 17 (DC1), and can be sent directly from the keyboard of a host running DeTransfer or any other communications software as Ctrl Q.

Note : If the *dataTaker* is in the XOFF state when it drops to the low power mode, then the XOFF state resumes when the *dataTaker* later powers up again.

Caution : If the host computer does not support the XON/XOFF handshake protocol, then care should be taken that the *dataTaker* does not return data too quickly, and overrun the receive buffer of the host computer.

Alternative Commands for XON and XOFF

In some telemetry systems, telemetry equipment such as cellular telephone modems, radio modems, satellite terminals, etc. may intercept XON and XOFF characters and treat these as being for local control. Therefore XON and XOFF characters sent by the host computer software to the *dataTaker* to control the transmission of data from the logger are not received by the logger.

To overcome this potential problem, the *dataTaker* also supports alternative forms of the XON and XOFF commands as follows

```
^ZQXON
```

which is equivalent to XON

```
^ZSXOFF
```

which is equivalent to XOFF

Note : The ^Z indicates the Control Z character which must be sent to the *dataTaker* as ASCII 26, and not literally as circumflex Z.

These commands can be sent to the *dataTaker* from DeTransfer using the backslash command (\nnn) as follows

```
\026QXON
\026SXOFF
```

where \026 is translated by DeTransfer to a Control Z character when the command is transmitted.

When these commands are transmitted by the host, they are passed by the telemetry equipment as text strings to the *dataTaker*, which interprets and responds to them as XON and XOFF respectively.

Resetting the dataTaker *Receive Buffer*

Occasionally telemetry equipment may interfere with XON/XOFF control to the degree that the *dataTaker* input and/or output buffers become blocked.

This condition can be cleared by the communications buffers reset command as follows

```
^ZCMSRST
```

which resets and clears the transmit and receive buffers, and sets the XON/XOFF protocol state to XON.

This command can be sent to the *dataTaker* from DeTransfer using the backslash command as follows

```
\026CMSRST
```

where \026 is translated by DeTransfer to a Control Z character when the command is transmitted.

The *dataTaker* responds to this command with the message

```
RS232 Reset
```

Caution : Use of this command will cause the current contents of the transmit and receive buffers to be lost. However the program in the *dataTaker*, and any data stored in the memory, are unaffected.

Communication Protocols

The *dataTaker* optionally supports a communications protocol to protect against communications errors between the logger and the host. These errors are manifested as data loss or data corruption, and can be caused by electrical noise, temporary loss of carrier in radio or satellite links, interference by telemetry equipment, etc.

The *dataTaker* protocol ensures that command strings received by the logger are the same as those sent by the host computer software, and that the data received by the host computer software is the same as that sent by the *dataTaker*.

Use of the *dataTaker* communications protocol requires that the host software also supports the *dataTaker* protocol. This in turn requires that the host software has been specifically written for the *dataTaker*. The many third party data acquisition packages that are available will not support this protocol, unless a *dataTaker* driver has been specifically written.

Following initial power up, a hardware reset, or a RESET command, the protocol for communications is disabled.

Communications protocol can be used in both the Free Format and the Fixed Format Modes (see below), and is described in Section III – Computer Mode Communications.

The *dataTaker* 500/600 series loggers also implement a protocol in the RS485 network between loggers. However this protocol is always enabled, and is transparent to the user. There are no commands for the network protocol.

Free Format Mode for Data

The *dataTaker* normally returns data and system information to the host computer in the Free Format Mode, which is a conversational style suitable for displaying directly on a computer screen and for printing.

In this mode the user can change the format of the returned data and various system information using the data formatting commands.

Free Format Mode is selected by the Data Format Switch as follows

/H Select Fixed Format Mode

/h Select Free Format Mode (Default)

Following initial power up, a hardware reset, or a RESET command, the Free Format Mode is selected.

The Free Format Mode of returning data and system information is described in detail in Section III – Format of Returned Data.

Fixed Format Mode for Data

The *dataTaker* has a Fixed Format Mode for returning data and system information, that is a fixed data format that cannot be changed.

If the Free Format Mode is used to interface with host software, then provision must be made in the host software to ensure that changes in the data format do not confuse the software.

However the Fixed Format Mode provides the host software with a known *dataTaker* environment in which to operate.

In Fixed Format Mode the *dataTaker* provides

- a fixed format for data and system information, which has special identifiers for each type of message, and date and time details
- a description of the schedules, the channels, calculations, alarms, etc. that have been programmed into the logger

Use of Fixed Format Mode requires that the host software supports the predefined data formats which this mode ensures, and requires that the host software has been written specifically for the *dataTaker*. The third party data acquisition packages available will not support this mode, unless a *dataTaker* driver has been specifically written.

The DeLogger and DeLogger Pro support the Fixed Format Mode, and DePlot supports the Fixed Format Mode. However DeTransfer does not support this mode.

Fixed Format Mode is selected by the Data Format Switch as follows

/H Select Fixed Format Mode

/h Select Free Format Mode (Default)

When Fixed Format Mode is selected, current settings of the Channel Number Switch (/N/n), the Channel Identifier Switch (/C/c), the Units Text Switch (/U/u), the Echo Switch (/E/e), the Return Switch (/R/r), Parameter22, Parameter24 and Parameter38 are saved. These settings are restored when the Fixed Format Mode is exited and the Free Format Mode resumed.

The Fixed Format Mode for returning data and system information is described in Section III – Computer Mode Communication.

Communication Echo

During transmission of commands to the *dataTaker*, transmitted characters may optionally be echoed back to the host computer software.

While character echoing is intended for use when the *dataTaker* is supervised from a terminal device, it may be used by the host computer software to check if characters transmitted to the logger are correctly received.

Echo is enabled by the Echo Switch as follows

/E Enable command echo (Default)

/e Disable command echo

The Echo Switch defaults to /E when the *dataTaker* is initially powered up, is hardware reset or executes a RESET command.

Password

The *dataTaker* has a password protection for the RS232 COMMS serial interface. When the password is enabled, communications through the RS232 COMMS serial interface is only possible after the defined password has been entered.

Password protection is particularly useful when the *dataTaker* is connected to a modem. This eliminates the possibility of line noise being interpreted as commands during call establishment. Unauthorized access also is more difficult.

The password is defined by command in the general formal

PASSWORD="*password*"

where *password* is a text string of up to 10 case sensitive characters. Defining the password as null removes an existing password.

The following command from DeTransfer defines the password for the logger

```
PASSWORD="SecretCode"
```

To establish communications, enter the password followed by a carriage return. This signs the user on.

The RS232 COMMS serial interface remains open while there is communications activity, or until the SIGNOFF command is issued.

If there is no communications activity for 300 seconds (5 minutes), then the RS232 COMMS interface will automatically signoff. The automatic signoff delay can be defined by the Parameter14 command in the range of 1 – 255 seconds.

The *dataTaker* will respond to the DEL character with << CR LF (see below) regardless of the password state, This allows a *dataTaker* to be identified.

The *dataTaker* password can be operated by explicit command from DeTransfer. DeLogger does not support the *dataTaker* password at all.

Entering Commands and Programs

Commands to supervise and program the *dataTaker* are entered into the logger by any of four basic methods

- the most common method is to enter commands into the *dataTaker* directly from the host computer software, via the RS232 COMMS serial interface
- commands can be recorded into a memory card, and automatically read into the *dataTaker* when the memory card is inserted into the card socket of the logger
- commands can be entered into a *dataTaker 500* or *dataTaker 600* series logger via the NETWORK interface. The commands are transmitted from a host computer that is connected to the RS232 COMM interface of another logger in the network, then via the network to the target *dataTaker* on the network.

  Note : The *dataTaker* 50 does not support networking.
- commands can be conditionally initiated internally from alarms, as a result of alarm conditions becoming true

The command buffer within the *dataTaker* will accept up to 250 characters. This limits single command lines to a maximum length of 250 characters.

Entering Commands in the Low Power Mode

The *dataTaker* usually operates in normal power mode, waiting for the next Schedule to be triggered to collect and process data, or for commands to be received into the RS232 COMMS serial interface.

When the *dataTaker* is in the low power mode, various sections of the logger, including the microprocessor and serial interface circuitry, are powered off. When a Schedule triggers, or command characters are received into the RS232 COMMS serial interface, then the *dataTaker* 'wakes up' to the normal power mode and performs the necessary functions.

However if commands are sent to the *dataTaker* while in the low power mode, then a number of the leading characters of the command will be missed as the logger wakes up. The *dataTaker* takes approximately 250 mS to wake up to normal mode, and so any characters received during this period will be missed. The number of characters missed depends on the communications baud rate, and at the higher baud rates entire commands may be missed.

Missed characters will result in command syntax errors due to incomplete commands being received. The *dataTaker* will return error messages for what would appear to be syntactically correct commands.

- if the *dataTaker* is being commanded manually from a terminal host, then character loss is less of a problem because typing speed determines character rate. Generally only the first character or two will be missed. Loss of characters during the wake delay can be overcome by sending several CR/LF characters to wake the logger by tapping the Enter key, before entering command characters.
- if the *dataTaker* is commanded from dedicated host software, the degree of character loss is determined by the baud rate

  DeTransfer and DeLogger both provide for this wake up delay by sending a linefeed character to the *dataTaker*, and waiting for 250 mS before sending the command. A similar practice should be adopted for all host software used to supervise the *dataTaker*.

After the *dataTaker* wakes from low power mode by characters received into the RS232 COMMS serial interface, the logger remains in the normal power mode for 30 seconds after the last character is received, before dropping to the low power mode again.

In applications where low power mode is not required, such as when an adequate power supply is available, then the automatic low power mode can be disabled by the Parameter15=2 command (See Section II - Powering the *dataTaker*). This command can also be used to keep the logger awake during a

programming session, and the logger then returned to the automatic low power mode by the Parameter15=0 command after programming.

The automatic low power mode is disabled whenever the *dataTaker* is powered from external power supplies.

Entering Commands from a Memory Card

The memory card has an area of memory reserved for storage of *dataTaker* commands. This is referred to as the program area, and can contain commands totalling up to 4090 characters.

Whenever a memory card is inserted into the *dataTaker*, the logger firstly checks if any commands stored in the program area of the card. If there are commands stored in the program area of the memory card, then these are read into the command buffer of the *dataTaker* and are executed.

If there are no commands stored in the program area of the memory card, then the *dataTaker* continues the current task.

Any of the *dataTaker* commands can be stored in the program area of the memory card.

The methods for recording commands into the memory card, and use of the memory card for programming the *dataTaker*, are discussed in Section III - Programming the *dataTaker* from the Memory Card.

Entering Commands from the Network - *dataTaker* 500/600 loggers

Commands can be entered into any of the *dataTaker* 500/600 series loggers within a *dataTaker* local area network, via the network.

The host computer is connected to the RS232 COMMS interface of any logger within the network. commands addressed for any other *dataTaker* 500/600 series logger within the network are sent out of the NETWORK interface of host logger, and received into the NETWORK interface of the target logger.

The direction of commands to different *dataTaker* 500/600 series loggers within the network requires that commands be prefixed by an address, which matches the address of the target logger.

The commanding of *dataTaker* 500/600 series loggers in a network is discussed in detail in Section III - Networking the *dataTaker* 500/600 Series Loggers.

General Rules for Entering Commands

There are a number general rules for entering commands into the *dataTaker*, which apply to all methods of command entry. The rules apply when entering commands from DeTransfer, however do not apply when using DeLogger since DeLogger is interacting with the logger rather than the user.

These rules provide for simplicity and flexibility when entering commands, while providing syntax, which is easy to use.

Command Line Editing

The *dataTaker* supports several line-editing functions during entry of commands. These line editing functions particularly apply to the situation where the *dataTaker* is being supervised either from a terminal, or from a computer running either a general communications program or terminal emulation program.

Backspace [BS]

Sending a backspace character (ASCII 8) to the *dataTaker* will cause the last character transmitted by the host to be erased from the internal command buffer of the logger.

If communications echo is enabled by the Echo Switch /E, then the *dataTaker* responds with a 'backspace space backspace' character sequence, which has the effect of deleting the last character from the command line on the terminal screen.

Repeated backspaces will erase successive characters from the command line, and will eventually erase the entire command.

Delete [DEL]

Sending a delete character (ASCII 127) to the *dataTaker* causes a partially or wholly entered command to be aborted. However the delete character must be sent before the carriage return, which terminates the command.

If communications echo is enabled by the Echo Switch /E, then the *dataTaker* responds with the characters << followed by a carriage return and line feed (CR/LF), which indicates on a terminal screen that the command has been aborted.

Carriage Return [CR]

Terminates a command line, and instructs the *dataTaker* to process the command and execute the commanded functions.

If communications echo is enabled by the Echo Switch /E then the *dataTaker* responds with a CR/LF sequence to the host.

Line Feed [LF]

The line feed character (ASCII 10) is ignored.

Tab and Space Characters

The Tab (ASCII 9) and Space (ASCII 32) characters can both be used to insert a space character. The Tab or Space character is the delimiter character between the elements of a complete command.

If communications echo is enabled by the Echo Switch /E, then the Tab and Space characters are echoed as a space character.

Command Separators

Space or tab characters are used within the command lines to separate the various command components, according to the following rules

there must be at least one separator between the command address, command type and individual items in a schedule list as follows

```
#1  R10M  1..10V  1..2DSO=0  3..4DS
  ^     ^       ^          ^

STATUS3  TEST
       ^     ^
```

Omission of separators from between the command components will cause the remainder of the command to be aborted.

separators are not permitted within the command address, the command type, or individual items in a schedule list as follows

```
#  1  R  A  10  M   1  ..  10  V  1 ..2  DS0 = 0
 ^     ^  ^   ^      ^   ^   ^     ^   ^    ^ ^

STATUS  3
      ^
```

Inclusion of separators within the command components will cause the remainder of the command to be aborted.

Self Documentation of Commands

With the exception of the Switch commands (See Section III - Switch Commands), all *dataTaker* command characters and channel type identifier characters are strictly upper case characters (capitals).

Commands may be made self-documenting by completing the name for each command or identifier in lower case as follows

```
Repeat10Min Day Time 1..5Volt /Retn to host
```

When the command is read by the *dataTaker*, all of the lower case characters are ignored and only the upper case characters are retained in the command.

The lower case characters are removed from the command line as the line is entered, and therefore do not consume space in the internal command buffer.

Therefore the above command example would reduce to

```
R10M  D  T  1..5V  /R
```

in the *dataTaker* command buffer.

Self-documentation of commands can be further improved by using the underscore character to separate word combinations

```
Repeat_10_Min Day Time 1..6Thermocouple_T
```

Self-documentation may also be used in the Switch commands (See Section III - Switch Commands). However the case of the first command character following the / character must be maintained appropriate to the setting required (upper case to enable the switch, and lower case to disable the switch) as follows

```
/Return_data  /messages_off  /echo_off

/return_no_data  /Messages_on  /Echo_on
```

Format of Returned Data

The dataTaker normally returns data to the host computer in the Free Format Mode. This is a descriptive or conversational style, which is suitable for displaying data directly on a terminal screen such as DeTransfer, for printing data, and for passing data to purpose written and third party data acquisition software packages.

The Free Format Mode has a default format for returned data that can be changed using a variety of data formatting commands to suit particular applications.

The *dataTaker* also has an alternative mode for returned data, designed for transferring data to purpose written and third party data acquisition packages. This is the Fixed Format Mode, which is a fixed data format that cannot be changed by the user. This mode is detailed in the *dataTaker* Advanced Communications Manual.

The format mode for returned data is selected by the Data Format Switch as follows

/H Select Fixed Format Mode

/h Select Free Format Mode (Default)

Following initial power up, a hardware reset, or a RESET command, the Free Format Mode is selected.

DeTransfer interfaces to the *dataTaker* in either data format mode, however the Free Format Mode is generally preferred.

DeLogger supervises the *dataTaker* in the Fixed Format Mode. DeLogger forces the logger to Fixed Format Mode during connection, and returns it to Free Format Mode during disconnection. The data formatting discussed in this chapter is in the main not appropriate for use with DeLogger.

The Free Format Mode for returned data is described below, including commands for defining the various data formats.

Controlling the Return of Data

The return of data from the *dataTaker* to a host computer or terminal as the input channels are read and calculations performed can be enabled or disabled using the Return Data Switch as follows

/R Enables return of data to the host (Default)

/r Disables return of data to the host

The Return Data Switch defaults to /R whenever the *dataTaker* is initially powered up, hardware reset or executes a RESET command.

Return of data can be disabled when operating the logger stand–alone, and data is logged to data memory or a memory card. This conserves battery power, because the *dataTaker* serial interface is not being used to transmit data to a disconnected host.

DeTransfer and DeLogger automatically enable the return of data during connection, and disable the return of data during disconnection.

During a DeTransfer session, the return of data is managed by the commands

```
/R
/r
```

All channels will return data by default when read, however individual channels can be disabled from returning data by the NR channel option.

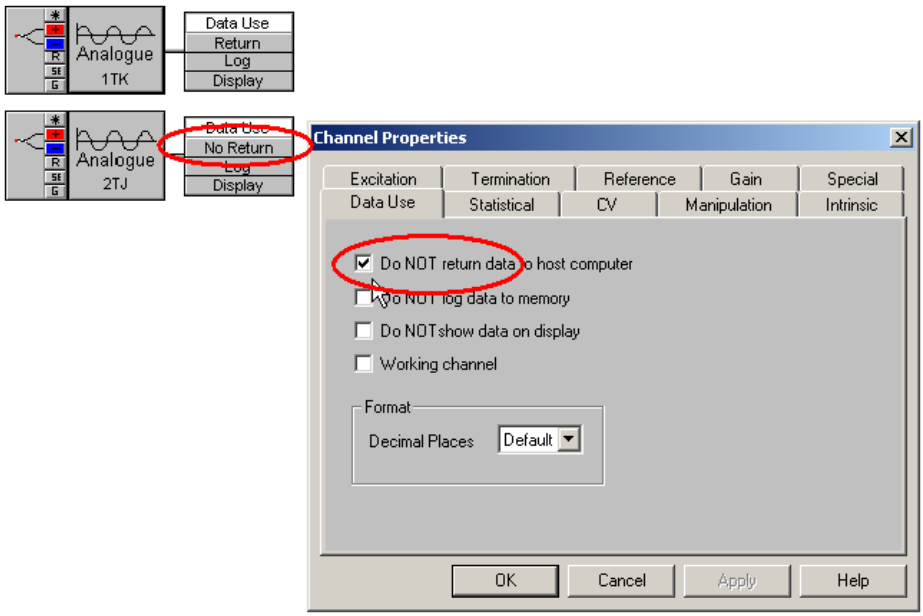
Using DeTransfer, the channel option

```
BEGIN
 RA10S
  1TK   2TJ(NR)
END
```

permanently disables the return of data from the second channel. However data from this channel is still available internally for logging and displaying.

If the Return Data Switch is enabled (/R), then data from the first channel (1TK) will be returned. If the Return Data Switch is disabled, then no data is returned.

Using DeLogger, return of data from individual channels can be disabled in the



Channel Properties dialog in the Program Builder. Firstly define the channel in the normal way, then right click on the Data Use box for the channel, select the Data Use tab, and check Do NOT return data to host computer.

Free Format Mode for Returned Data

In the Free Format Mode, all data is returned from the *dataTaker* in user definable data item and data block formats. The user formats apply to data returned directly to the host as the input channels are read and calculations exeuted, and to stored data returned from the internal data memory or memory card.

The data is returned in ASCII strings as either floating point data or exponential data, with channel identification and units text.

In these formats, each digit is an ASCII character (eg. 123.45), which allows returned data to be displayed directly onto a terminal screen for viewing, to be input into a computer for processing, or to be sent directly to a serial printer to produce hardcopy.

When the returned data is input to a host computer, it can be written directly into disk storage as an ASCII file. This ASCII file format is compatible with a range of data processing packages such as databases, spreadsheets, statistics packages, graphics packages, word processors, etc.

The format of returned data can be changed to suit applications software for particular data processing and presentation tasks.

The Free Format Mode also facilitates data transmission to the host either directly via a cable, via public telecommunications networks (PSTN) using modems, via the cellular phone network, or via radio or satellite telemetry systems.

Data Item Format

In Free Format Mode, each item of data produced by the *dataTaker* is returned to the host computer in a default format. This format is a descriptive or conversational style, and is selected whenever the *dataTaker* is powered up or reset.

However the format of each item of data can be modified to produce a data format suited to particular applications. The numeric format and the description of data items can be changed in a number of ways, and is discussed in the following pages.

Data Numeric Format

Two numeric formats for returned data are available in the Free Format Mode, which can be modified to create a numeric format suited to the host software requirements.

The default numeric format is dependent on the magnitude of the data as follows

data values within the range of $\pm 10^{\pm 6}$, and with fewer than six significant digits, are returned in floating point format. The number of decimal places is set according to the number of significant digits for the channel measurement.

data values outside the range of $\pm 10^{\pm 6}$, or with more than six significant digits, are returned in exponential format. The number of decimal places is set according to the number of significant digits for the channel measurement.

The data from each channel can also be specified individually to be in floating point or exponential format, with a specified number of significant digits and decimal places.

Floating Point Data Format

Data within the range of $\pm 10^{\pm 6}$, and with fewer than six significant digits, is returned in the floating point format as follows

nncc sdddd.ddd uu [delimiter]

where

nn	is the channel number
cc	is the channel type identifier
s	is the sign character, – or space
dddd.ddd	is the floating point decimal data
uu	is the units text
[delimiter]	is the data delimiting character

Some examples of data returned in the floating point format follow

```
3V  -12.277 mV
5TK  367.28 Deg C
1DS  1 State
4C  3451 Counts
```

Whenever the magnitude of data is outside the range of $\pm 10^{\pm 6}$, then the *dataTaker* automatically selects the exponential format (see below).

The floating point format can be forced by the FFn channel option, where n is the number of decimal places (0 – 6).

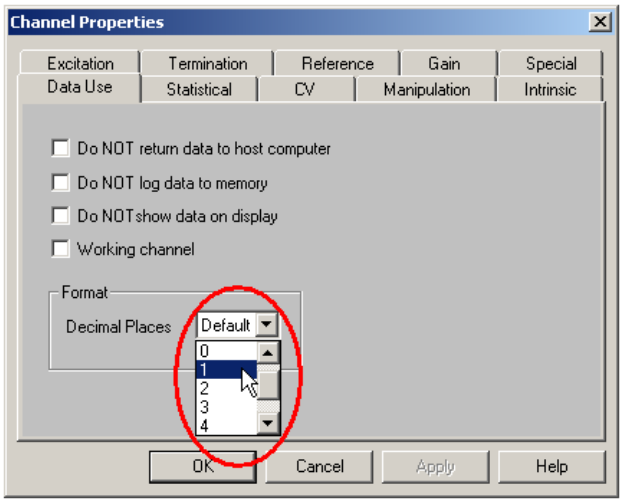
Using DeTransfer, the command for example

```
5BGI(FF2)
```

instructs the *dataTaker* that the strain gauge data is to be returned in Fixed Floating point format with 2 decimal places (FF2).

The FFn channel option is also used to fix the number of decimal places in returned data, which may be appropriate to match the resolution of data to the resolution of a sensor.

Using DeLogger, the number of decimal places for any channel can be set in the Channel Properties dialog box in the Program Builder.



Exponential Data Format

Data values that are outside the range of $\pm 10^{\pm 6}$, or have more than six significant digits, are returned in the exponential format as follows

nncc sd.dddEsdd uu [delimiter]

where

nn	is the channel number
cc	is the channel identifier
s	is the sign character, – or space
d.ddd	is the mantissa of the data
Esdd	is the signed exponent of the data
uu	is the units text
[delimiter]	is the data delimiting character

Some examples of data returned in the exponential format follow

```
2V  1.277E7 Newtons
5V  0.694E-8 Volts
6V  -1.592E3 KPa
```

When the magnitude of data is in the range of $\pm 10^{\pm 6}$, the *dataTaker* automatically selects the floating point format (see above).

The exponential format can be forced by the FEn channel option, where n is the number of significant digits (0 – 6) for the mantissa.

Using DeTransfer, the command for example

```
5BGI(FE2)
```

instructs the *dataTaker* that the strain gauge data is to be returned in Fixed Exponential format with 2 significant digits in the mantissa (FE2).

The FEn channel option is also used to fix the number of significant digits in the mantissa of data returned in exponential for particular channels.

It is not appropriate to force Fixed Exponential format in DeLogger, because DeLogger is using the Fixed Format Mode for data (/H).

The exponential format for data generally applies to data resulting from user defined sensor calibrations (polynomials and spans) and from calculations. The ranges for basic signal types measured by the *dataTaker* rarely require the exponential format for basic data.

Mixed Data Format

The *dataTaker* can be programmed to automatically select between floating point and exponential format for individual channels depending on the magnitude of data.

The FMn channel option forces the data format as follows

floating point format for data in the range of $\pm 10^{-4}$ to $\pm 10^{+n}$

exponential format for data outside the range of $\pm 10^{-4}$ to $\pm 10^{+n}$

where n also specifies the number of decimal places in the floating point data, or the number of significant digits in the mantissa, respectively.

Using DeTransfer, the command for example

```
5BGI(FM2)
```

instructs the *dataTaker* that the strain gauge data is to be returned in Fixed Floating point format if the data is in the range of 0.0001 to 100.00, and Fixed Exponential format if data is outside this range (FM2).

It is not appropriate to force Mixed format in DeLogger, because DeLogger is using the Fixed Format Mode for data (/H).

Global Number of Significant Digits

Data returned from the *dataTaker* normally has 1 to a maximum of 5 significant digits, depending on the magnitude of the data.

The maximum number of significant digits in data returned in real time can be globally defined by the Parameter32 command. This sets the maximum number of significant digits, and data with fewer significant digits is not expanded.

Using DeTransfer, the command for example

```
P32=2
```

globally sets a maximum number of 2 significant digits for all data returned in real time.

Parameter32 can be defined within the range of 1 to 9.

Data stored in the internal memory or memory card is stored to 5 significant digits, irrespective of the setting of the Parameter32 command.

It is not appropriate to set the number of significant digits in DeLogger, because DeLogger is using the Fixed Format Mode for data (/H).

Setting Significant Digits and Decimal Places for Individual Channels

Setting the number of decimal places in floating point data, and the number of significant digits in the exponential data, adjusts the output resolution of the data.

There are a number of practical benefits in being able to define the resolution of returned data, including

- matching the error limits of the sensors used, so that data is not returned to degrees of resolution beyond the capability of the sensors
- improves tabulation of returned data on printers and terminals
- reducing the number of characters which must be transmitted when data is returned. This is particularly relevant where data is transmitted at lower baud rates via modems, radio links, etc.

The *dataTaker* returns all analog data at the maximum resolution possible for each input type, according to the gain (range) used for signal measurement, the number of significant digits possible for such measurements, and the magnitude of the data.

However the maximum resolution is determined purely on the basis of the capabilities of the data logger, and does not take into account the precision of the sensor.

Analog data in the range of $\pm 10^{\pm 6}$ is returned by the *dataTaker* in the floating point format, for which the number of decimal places may be defined.

Digital data such as logic bit state, logic byte state and counts are returned as integer data, and do not have a fractional component. However the average and standard deviation for digital state and counts is returned as floating point data, and the number of decimal places can be defined.

The number of decimal places in returned data can be reduced from the maximum system resolution, but cannot be increased beyond that maximum.

The resolution of data returned in exponential format can also be defined, for which the number of significant digits is defined.

The number of decimal places for data in floating point format can be defined by the FFn and FMn channel options, and the number of significant digits in exponential format can be defined by the FEn and FMn channel options (See Section III – The Analog Input Channels).

Using DeTransfer, the command for example

```
1V(FF1)  2V(FF3)
```

sets the number of decimal places to 1 for the first channel, and sets the number of decimal places to 3 for the second channel, to match sensor capability.

The default setting for the number of decimal places or the number of significant digits is 5, which causes data to be returned at maximum possible resolution as determined by the *dataTaker*.

The maximum number of significant digits can be set by the Parameter32 command in the range of 1 to 9.

The FFn, FEn and FMn channel options sets the maximum number of decimal places or significant digits in returned data, not the total number. Therefore if the internal data has fewer decimal places or significant digits than defined, trailing zeros are not added. This is illustrated in the table below

Internal Data	***FFn***	***FEn***	***Returned Data***
122.324	2	2	122.32 or 1.22e2
23.8	3	3	23.8 or 2.38e1
335.14	0	0	335 or 3e2

When the number of decimal places for returned data is reduced from that of the maximum internal resolution, the last decimal place is rounded upwards according to the magnitude of the next digit. This is illustrated in the table below

Internal Data	*FFn*	*FEn*	*Returned Data*
23.877	2	2	23.88 or 2.39e1
23.872	2	2	23.87 or 2.39e1
125.94	0	0	126 or 1e2

Date and Time Formats

The date is returned from the real time clock of the *dataTaker* in the default formats

Date DD/MM/YYYY (50 Hz operation)

or

Date MM/DD/YYYY (60 Hz operation)

The default format for date is determined from the mains or line frequency setting on the DIP switch.

The time is returned from the real time clock of the *dataTaker* in the default format

Time HH:MM:SS

Examples of the date and time returned in the default formats follow

```
Date  20/6/2003
Time  11:30:15
```

The format for the date and the time can be modified in a number of ways as follows

- the date can be expressed as the date either in DD/MM/YYYY format, MM/DD/YYYY format, or as the number of days since 01/01/1989
- the time can be expressed either in the HH:MM:SS format, as the number of seconds since last midnight, or as decimal hours
- if the default time format of HH:MM:SS is used, the character separating the hours, minutes and seconds can be specified
- a decimal day format is also available which combines the day number and time of day in a single value (See System Variable 12)

The date and time is returned by including these in a Schedule, and by enabling the Date Switch (/D) and the Time Switch (/T) (See Section III – The Real Time Clock).

The commands for formatting the day number, date and time returned from the real time clock of the *dataTaker* are detailed in Section III – The Real Time Clock.

Note : The date and time of the real time clock must be set in the same format as is defined for the return of date and time.

It may be appropriate to change the format for date and time if you are supervising and programming the *dataTaker* from DeTransfer. However changing the format of date and time is not appropriate if you are using DeLogger.

Data Item Description

The default data format returned to by the *dataTaker* is a descriptive or conversational style, which is suitable for displaying data on a host screen such as with DeTransfer, for printing data, and for passing data to data acquisition software, data processing packages and data presentation packages.

The default data format is illustrated by the following examples

```
Date  20/6/2003
Time  11:37:38
3V  12.279 mV
5PT385  -74.29 Deg C
2DS  1 State
8C  2391 Counts
```

Each data item has several descriptors, which identify the data as follows

the channel number indicates the source of the data

the channel identifier indicates the type of signal or sensor

the units text indicates the engineering units for the data

each channel can be given a unique name

This description of data items is definable in a number of ways, and is described in the following sections. These commands are global, and specify the format for all data returned from the *dataTaker*.

It may be appropriate to change the data item descriptions if you are supervising and programming the *dataTaker* from DeTransfer. Changing the data item descriptions other than the channel name is not appropriate if you are using DeLogger.

Channel Number

The return of channel numbers is determined by the Channel Number Switch as follows

/N Enable return of channel numbers (Default)

/n Disable return of channel numbers

The Channel Number Switch defaults to /N when the *dataTaker* is initially powered up, hardware reset or executes a RESET command.

If the channel is located on a Channel Expansion Module, or if a *dataTaker* channel has been addressed as module 0, then the channel number is preceded by the module number. The module number is part of the channel number, and is included if the Channel Number Switch is enabled (/N) and not included if the Channel Number Switch is disabled (/n).

Channel Identifier

Channel identifiers are returned from the *dataTaker* only if the channel numbers are being returned (/N), when the return of channel identifiers with the channel number is determined by the Channel Identifier Switch as follows

/C Enable return of channel identifiers (Default)

/c Disable return of channel identifiers

The Channel Identifier Switch defaults to /C when the *dataTaker* is initially powered up, hardware reset or executes a RESET command. If return of channel numbers is disabled (/n), then return of channel identifiers is also disabled.

The examples below illustrate the interactive effect of the Channel Number Switch and Channel Identifier Switch on returned data

```
5PT385  -74.29 Deg C          /N/C
5  -74.29 Deg C               /N/c
-74.29 Deg C                  /n/C
-74.29 Deg C                  /n/c
```

Channel Name

The default labelling of returned data by the channel number and channel identifier can be replaced by unique channel names. The channel names of up to 16 characters are specified as channel options.

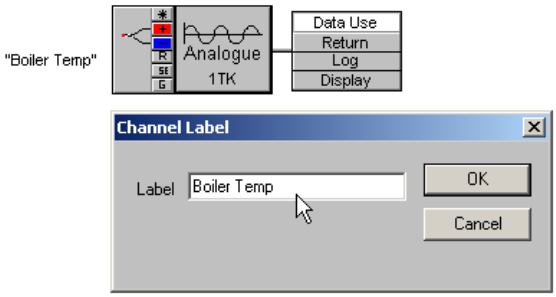
Using DeTransfer, the command for example

```
1TK("Boiler Temp")
```

names the channel in terms of the origin of the data.

If a name has been defined for a channel, then this is returned with the data instead of the channel number and channel type. However the return of a defined channel name is disabled by the /c and /n Switch commands.

DeLogger also supports naming of channels. Channel names can be entered in the Program Builder by clicking on the sensor icon for a defined channel, and selecting Label to open the Channel Label dialog.



The new channel name is then used throughout DeLogger to identify this particular data source.

Units Text

The return of units text is optional, and is determined by the Units Text Switch as follows

/U Enable return of units text (Default)

/u Disable return of units text

The Units Text Switch defaults to /U when the *dataTaker* is initially powered up, hardware reset or executes a RESET command.

Date or Day and Time Labels

The return of the Date or Day label and the Time label is determined by the Channel Number Switch, and return of the units of time is determined by the Units Text Switch, as follows

/N Enable return of Date or Day and Time labels (Default)

/n Disable return of Date or Day and Time labels

/U Enable return of time units (Default)

/u Disable return of time units

The return of time units is only appropriate to the time formats of seconds since midnight and decimal hours (See Section III – The Real Time Clock).

Data Delimiter Character

Each item of the data, the date or day, and the time, can be delimited by a user definable data delimiting character.

If the Units Text Switch is enabled (/U), then each data item will be delimited by the CR/LF pair. If the data is being returned to a terminal screen such as DeTransfer or to a printer, then each data item will be written on a new line.

However if the Units Text Switch is disabled (/u), then data items are delimited by the character defined by the Parameter22 command. The delimiter character is defined as the ASCII decimal code for the required character.

Using DeTransfer, the command for example

```
P22=44
```

defines the data item delimiter to be the comma character (ASCII 44).

The default definition for Parameter22 is CR (ASCII 13) when units text is enabled, and SPACE (ASCII 32) when units text is disabled. Whenever Parameter22 is defined as a CR, then a LF is automatically appended to produce a CR/LF new line sequence.

It is not appropriate to set the data delimiter from DeLogger.

Fixed Field Width

Each item of data returned to the host computer can have a fixed field width capacity, which can be set between 1 and 80 characters.

By default each item of data returned is in variable width, which is dependent on the magnitude of the data. The setting of Parameter33 allows for fixing the number of characters returned with each data item.

The benefits of fixing the field width for data are as follows

- producing tabular data to aid in the clarity of the data returned
- providing a consistent number of characters for each data item, which assists import of data into some third party software packages.

The following example illustrates data returned without fixed field width applied

```
18.8 18.7 3.498 0.000
18.9 16.0 138.08 0.000
18.7 16.0 0.192 0.000
```

If a fixed field width of 8 characters is set by the command from DeTransfer

```
P33=8
```

then the returned data is more tabular as follows

```
18.8     18.7    3.498    0.000
18.9     16.0   138.08    0.000
18.7     16.0    0.192    0.000
```

Each data item is right justified and space padded. If the name or value is larger than the field width specified by the Parameter33 setting, then the name or value is truncated from the right. The $ string

channel, Alarm text messages, Test results and Status reports are not effected by the setting of Parameter33.

It is not appropriate to set the data field width in DeLogger.

Some Examples of Data Item Formats

The default data item format for returned data is defined by the following commands, which could be entered from DeTransfer

```
/U  /N  /C  P22=13
```

Data is returned in a descriptive format, and can be read directly from the DeTransfer screen or printout from a printer.

These default settings are automatically selected when the *dataTaker* is initially powered up, is reset, or executes a RESET command.

The default data item format is illustrated in the following examples

```
Date  12/06/2003
Time  12:30:00
3V  12.279 mV
4PT385  -74.2 Deg C
5I  87.36 mA
2DS  1 State
3C  2391 Counts
```

Many combinations of settings for the data formatting Switches, the data delimiter character, and the data resolution are possible to change the data item format.

Using DeTransfer, the most common alternative is

```
/u  /n  /c  P22=44
```

which returns data in a format suitable for reading directly by host software, as illustrated below

```
12/06/1992,11:30:00,12.279,-74.2,87.36,1,2391
```

The default data item format can require transmission of up to 20 characters for each data item returned to the host.

During real time data acquisition the default data item format may reduce maximum sampling rate, particularly if the communications baud rate is low. This is because the logger will not run a Schedule until all data from the previous running of the Schedule has been sent to the host.

Disabling the channel number, channel identifier and units text, and reducing the number of decimal places, will reduce each item of data to between 2 – 7 characters depending on the data type.

Data Block Format

The data resulting from reading more than one input channel or performing more than one calculation is returned by the *dataTaker* in a block format, which comprises a grouping of the individual channel data.

The format of each data item in the block is user definable, and is described above in Section III – Data Item Format.

The format of the data block is the same for real time data, and data from the internal memory and memory card. Data returned from memory can optionally include an end of stored data character.

The date and/or time of the readings may be included at the start of each data block, depending on the setting of the Date Switch and the Time Switch (See Section III – The Real Time Clock).

Each data block can optionally be preceded by the address of the *dataTaker* returning the data. This allows blocks of data being returned from a network to be identified.

Formatting the data block is often appropriate if you are using DeTransfer to supervise the *dataTaker*, however is not appropriate if you are using DeLogger.

Data Block Format for Real Time Data

The default data block format for data returned directly to the host in real time as input channels are read and calculations performed is illustrated below

```
Date dd/mm/yyyy                Date
Time hh:mm:ss                  Time
nncc  sddddd uu[delimiter]     Data Items
.. ....                        ..
.. ....                        ..
[delimiter]                    Schedule delimiter
character                      (Parameter24)
```

The format of each data item in the block is definable, as described above in Section III – Data Item Format.

Each block of data resulting from successive reading of input channels and evaluation of calculations is delimited by a user definable Schedule or block delimiter character.

If the Units Text Switch is enabled (/U), then successive blocks of data are delimited by CR/LF. If returned data is displayed or printed, then successive blocks of data are separated by a blank line.

If the Units Text Switch is disabled (/u), then successive blocks of data are delimited by the character defined via the Parameter24 command. The delimiter character is defined as the ASCII decimal code for the required character.

Using DeTransfer, the command for example

```
P24=44
```

defines the block delimiter to be the comma character (ASCII 44).

The default definition for Parameter24 is CR (ASCII 13).

Whenever Parameter24 is defined as a CR, then a LF is automatically appended to produce a CR/LF new line sequence.

Data Block Format for Stored Data

The default data block format for data returned to from the data memory or a memory card is the same as that returned in real time, with an additional character after the last data block is returned to signify the end of stored data.

The end of stored data character is defined by the Parameter25 command, as the ASCII decimal code for the required character.

Using DeTransfer, the command for example

```
P25=42
```

defines the end of stored data character as an asterisk (ASCII 42).

The default is Parameter25=0 (NULL), and no character is returned.

Examples of Data Block Formats

The default data block format for data returned in real time, or from the data memory, is defined by the following commands

```
/U  /N  /C  /t  /d  P22=13  P24=13
```

returns data in a format that can be read directly from a terminal screen or printer.

These default settings are automatically selected when the *dataTaker* is powered up, is reset, or executes a RESET command.

The default data block format is illustrated in the following example

```
Date 23/06/2003
Time 15:31:15
3V  -12.27 mV
1C  2391 Counts
5TK  162.2 Deg C
```

Many combinations of settings for the Switches and Parameters are possible to define the data block format. The most common setting is

```
/u  /n  /c  /m  /e  /T  /D  P22=44  P24=13
```

which returns data in a Comma Separated Variable (csv) format suitable for importing directly into analysis software such as Microsoft Excel, as illustrated below

```
23/06/2003,15:31:15,-12.27,2391,162.2
23/06/2003,15:31:30,-12.21,2387,161.2
23/06/2003,15:31:45,-12.15,2267,159.8
```

The length of each block of data is determined by the number of channels read and/or calculations executed. Where two or more groups of channels are read by different Schedules, then whenever two or more of the Schedule times coincide the data is returned in separate blocks for each Schedule.

If the Units Text Switch is enabled (/U) the data blocks are delimited by CR/LF. If the Units Text Switch is disabled (/u) the data blocks are then delimited by the character defined at Parameter24.

The last block of data from the data memory or memory card will be terminated by the end of stored data character if one is defined in Parameter25. This is illustrated in the following example

```
Date 23/06/2003
Time 15:31:15
3V  -12.27 mV
1C  2391 Counts
5TK  162.2 Deg C
*
```

and

```
23/06/2003,15:31:15,-12.27,2391,162.2
23/06/2003,15:31:30,-12.21,2387,161.2
23/06/2003,15:31:45,-12.15,2267,159.8
*
```

where Parameter25 is set to ASCII 42 (asterisk).

dataTaker Address

Each data block can be preceded by the address of the *dataTaker* returning the data. This allows blocks of data being returned from a network to be identified.

The return of the *dataTaker* address is optional, and is determined by the Logger Address Switch as follows

/L Enable return of logger address

/I Disable return of logger address (Default)

The Logger Address Switch defaults to /I when the *dataTaker* is initially powered up, hardware reset or executes a RESET command. If the return of logger address is enabled, then the data block will be similar to the following examples

```
Logger 1
3V  -12.27 mV
1C  2391 Counts
5TK  162.2 Deg C
```

and

```
1,11:45:00,-12.15,2267,159.8
```

Enabling the logger address is not appropriate in DeLogger.

Specifying Channels

Many of the *dataTaker* commands relate to sampling or reading input channels, setting output channels, executing calculations, testing alarms, returning data, logging data, etc. These processes usually involve one or more channels, which must be specified to the process in various ways. This chapter describes this channel specification.

Usually these processes are associated with a Schedule, which determines when the processes are performed. Conversely Schedules can be thought of as having a list of processes to perform, which is called the 'schedule list' or sometimes 'channel list' if it only includes input channels to be read.

The specification of channels includes the module number on which the channel is physically located, the channel number, the type of signal or sensor connected to the channel, and various channel management options.

Channel Module

The input and output channels are located either on the *dataTaker* data logger, or on one or more Channel Expansion Modules connected to the data logger (*dataTaker 500/600* series loggers only).

The *dataTaker* is always module number of 0 (zero). The Channel Expansion Module(s) connected to the logger are allocated module numbers 1 and 2 respectively in the order of position relative to the logger. Module number 1 is nearest the *dataTaker*.

The module number for each Channel Expansion Module connected to the logger is allocated during power up, and whenever the *dataTaker* executes a RESET command.

If the position of Channel Expansion Modules in a daisy chain are changed, then the *dataTaker* must be repowered or reset to reallocate the module numbers.

The module number is specified as part of the channel identification in the format

```
module:channel
```

where

module	is the number of the module on which the channel is located
channel	is the channel number on the module

The module number of the *dataTaker* is optional, and does not need to be specified. If a module number is not specified, then the module number defaults to 0 for the *dataTaker*.

Therefore where a *dataTaker* has no Channel Expansion Modules connected, then the module number is not required as part of channel specifications.

The module number is returned with the data as part of the channel identification.

Channel Number

Channels are specified in the schedule list as individual channels, or as a continuous sequence of channels in the general format

```
first_channel[..last_channel]type(options)
```

where

first_channel	is the first channel number
..	indicates a continuous sequence
last_channel	is the last channel number
type	is the signal type identifier
(options)	are channel management options

The ..last_channel is optional, and if this is omitted in a channel specification then the first_channel is interpreted as an individual channel of the type specified as follows

channeltype(options)

Throughout this manual a single channel is indicated by the general format

ntype(options)

and a continuous sequence or group of channels is indicated by the general format

n..mtype(options)

When channel number sequences are defined, there may be one or more periods between the first and last channel.

If the channel(s) are located on Channel Expansion Modules, then the general format for channel specification is as follows

module:ntype(options)

module:n..mtype(options)

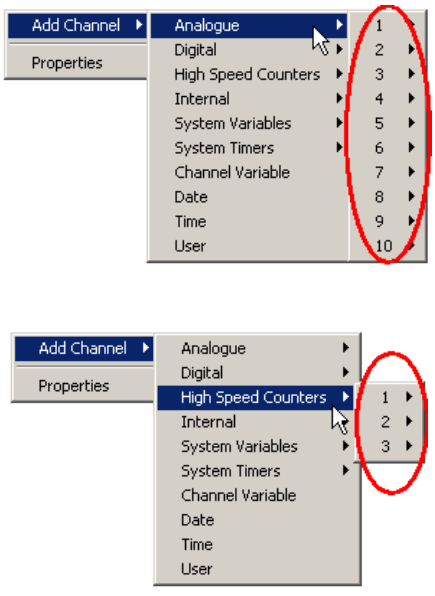
where

module — is the module number, 0 for the *dataTaker* and 1 or 2 for successive Channel Expansion Modules

Channel number sequences can only be defined within modules, and cannot cross module boundaries.

The *dataTaker* will return an error message if module numbers are specified for non-existent modules.

DeLogger also identifies channels to be measured by the channel number, which are displayed by holding down the right mouse button on the work area of the Program Builder



The number of channels displayed varies for the type of input, and for the type of data logger that DeLogger is connected to.

Signal Type Identifier

A signal or sensor type identifier is suffixed to the channel number or channel number sequence to further identify the channel.

The various signal type identifiers for the input and output channels of the *dataTaker* are detailed throughout Section II and Section III of this manual, and are summarised in the following chapters.

Some examples of channel specifications for the *dataTaker* module follow

`3V`	analog channel 3, voltage
`8TK`	analog channel 8, thermocouple type K
`2DS`	digital channel 2, state
`3..5I`	analog channels 1 through 5, current

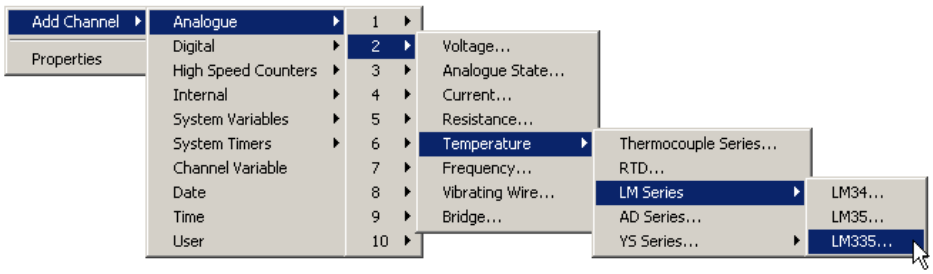
Some examples of channel specifications for Channel Expansion Modules follow

`0:2L`	dataTaker analog channel 2, current loop
`1:6PT385`	CEM 1 analog channel 6, RTD type 385
`2:2DSO`	CEM 2 digital output channel 2

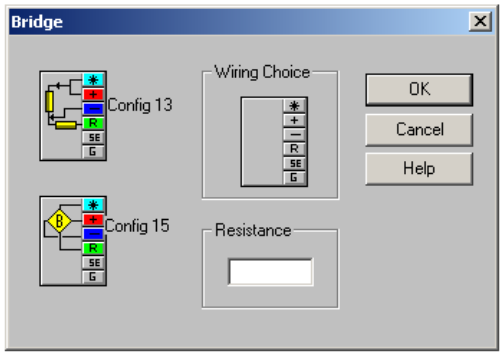
Schedule lists are simple groupings of individual channels into a list for scanning, etc. as the following example illustrates

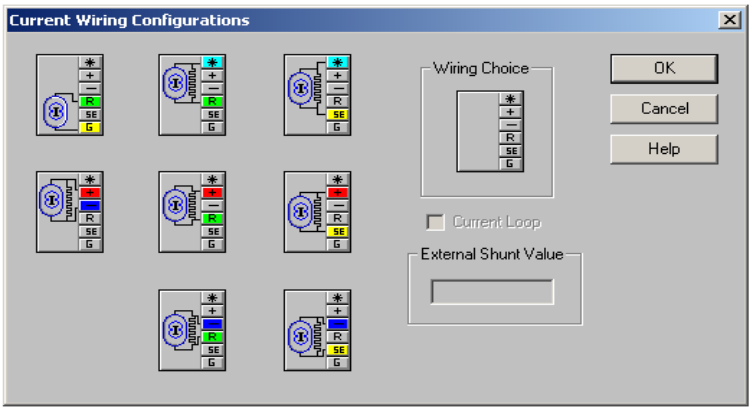
```
1..3V  4PT392  1DS  1:1..6TT  2:1AD590  0:2..4I
```

In DeLogger, the types of signals or sensors that can be input to channels are selected descriptively from the hierarchical walking menus in the Program Builder.



When the required signal or sensor group is selected, a dialog opens which displays the connection options for that group, and other input fields appropriate to the group





Channel Management Options

All signal types can be modified in various ways by channel options, which define the way in which the input channel and attached sensor is managed when read.

These options include types of sensor excitation, termination of the signal, specifying internal or external referencing for analog signals, scaling of measured data, format and destination of channel data, delays for digital output switching, etc.

Each channel type has default channel options, which do not need to be specifically declared. The declaration of channel options is itself optional.

The various channel management options are discussed in detail in Section III – Channel Options, and relevant sections throughout this manual.

The Analog Input Channels

The *dataTaker* analog input channels are used for measuring a large variety of analog signals from sensors, transducers, instruments, etc. The *dataTaker* models have a different number of analog input channels as follows

dataTaker 50	5 differential, 10 single ended analog channels
dataTaker 500/600 series loggers	10 differential, 30 single ended analog channels
Channel Expansion Module (CEM	10 differential, 30 single ended analog channels

The *dataTaker*s and the Channel Expansion Modules also have several internal analog channels, which are used for monitoring calibration references, and case temperature. These channels are not available for measurement of external signals, but can be read in the same manner as the analog input channels.

The allocation of analog input channels for each of the models of the *dataTaker* and the Channel Expansion Module are summarised in the sections below.

Analog Channels of the *dataTaker 50*

The *dataTaker 50* has 7 analog input channels, which are allocated between external user channels and internal reference channels as follows

Channels 1 to 5	– user analog input channels – differential and single ended modes
Channel 1%	– *dataTaker* 50 case temperature – LM35 solid state temperature sensor – cold junction temperature ref for thermocouples – read as channel 1%LM35
Channel 2%	– zero voltage, and 100.0 Ohm resistance reference – 100.0 Ohm resistance, grounded – zero voltage reference for thermocouple compensation – read as channel 2%V for zero voltage – read as channel 2%R for 100.0 Ohm reference

Analog Channels of the *dataTaker 500/600* Series

The *dataTaker 500/600* series loggers have 12 analog input channels, which are allocated between external user channels and internal reference channels as follows

Channels 1 to 10	– user analog input channels – differential and single ended modes
Channel 1%	– *dataTaker 500/600* case temperature – LM35 solid state temperature sensor – cold junction temperature reference for thermocouples – read as channel 1%LM35
Channel 2%	– Internal zero voltage, and 100.0 Ohm resistance reference – 100.0 Ohm resistance, grounded – zero voltage reference for thermocouple compensation – read as channel 2%V for zero voltage – read as channel 2%R for 100.0ª reference

Analog Channels of the Channel Expansion Modules

The Channel Expansion Module (CEM-AD) has 12 analog input channels which are allocated as follows

Channels n:1 to 10	– user analog input channels – differential and single ended modes

Channel n:1%	– CEM case temperature – LM35 solid state temperature sensor – cold junction temperature reference for thermocouples – read as channel n:1%LM35
Channel n:2%	– internal zero voltage – zero voltage reference for thermocouple compensation – read as channel n:2%V for zero voltage

where n is the Channel Expansion Module number. The analog input channels of the Channel Expansion Module are implemented in same way as those of the *dataTaker*.

Differential and Single Ended Analog Inputs

The analog input channels of the *dataTaker* and Channel Expansion Modules can be used for measuring differential inputs and single ended inputs. The differential and single ended input modes can be mixed within an application. Details for connecting analog signals as differential inputs and as single ended inputs are given in Section II.

Differential Analog Inputs

Differential inputs are applied between the positive terminal and the negative terminal of each of the analog input channels. The *dataTaker 50* can accept up to 5 differential inputs, while the *dataTaker 500/600* series can accept up to 10 differential inputs. Details for connecting analog signals as differential inputs are given in Section II.

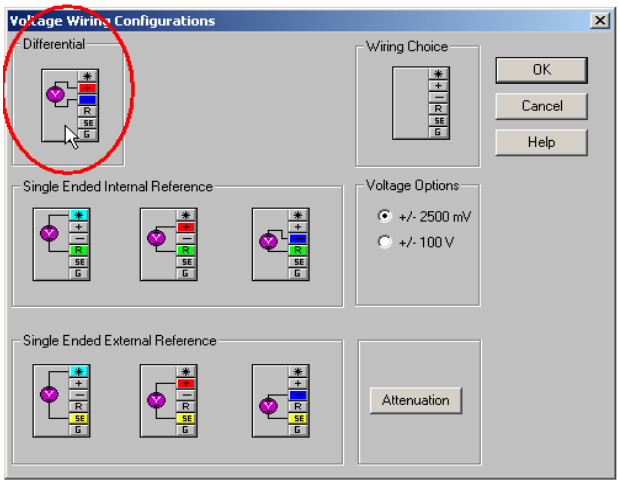
Differential analog inputs are specified by nominating the channel number and signal type as the following examples illustrate

`3V`	voltage between + and - terminals of channel 3
`8TK`	thermocouple K between + and - of channel 8
`1:2I`	current between + and - terminals of channel 2 of CEM 1

Using DeTransfer, differential analog inputs are specified if the channel number has no analog terminal specified, as shown in the examples

```
3V
8TK
1:2L
```

Using DeLogger, differential analog inputs are selected from the dialog which opens for the selected signal or sensor group, for example



Single Ended Analog Inputs

Single ended inputs are applied to the analog input channels in several different configurations

between the positive terminal + and Analog Return terminal R

between the positive terminal + and the SE REF terminal

between the negative terminal – and Analog Return terminal R

between the negative terminal – and the SE REF terminal

between the Excite terminal ✳ and the Analog Return terminal R or the SE REF terminal. This mode is not appropriate for resistance or RTD inputs, and is not available in the *dataTaker 50*

between the Analog Return terminal R and any Ground terminal G for current, current loop and AD590 types only

The single ended inputs between the +ve, –ve or ✳ terminals and the SE REF terminal are actually measured differentially by the *dataTaker*.

However, these input configurations are referred to as single ended input modes, as distinct from the true differential input mode.

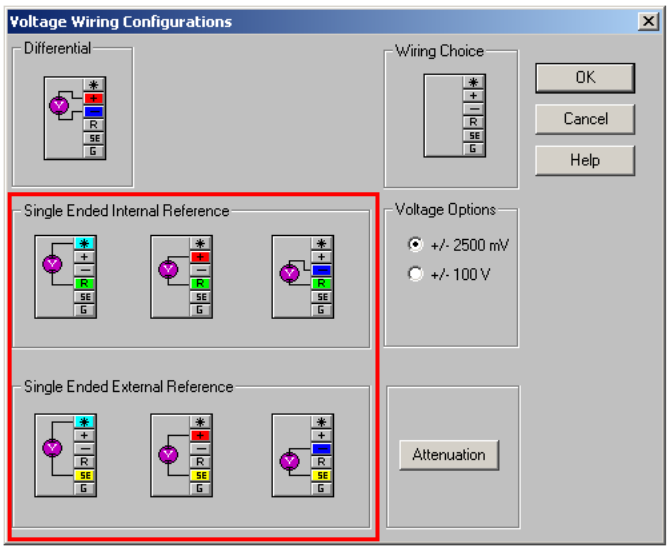
Single ended analog inputs are specified by nominating the channel number and the terminal for the positive lead of the signal, and the signal type as the following examples illustrate

`3+V`	voltage between + and R terminals of channel 3
`8-TK`	thermocouple K between – and R of channel 8
`2:7+I`	current between + and R terminals of channel 7 of CEM 2

Using DeTransfer, single ended analog inputs are specified if the channel number includes an analog terminal which is the nominal positive side of the signal, as shown in the examples

```
3+V
8-TK
2:7+I
```

Using DeLogger, single ended analog inputs are selected from the dialog, which opens for the selected signal or sensor group, for example



Single Ended Inputs for the dataTaker 50

These single ended analog input arrangements provide a for a total number of single ended channels for the *dataTaker 50* as follows

15 single ended input channels for measuring current, 4-20 mA current loop and AD590 temperature sensors

10 single ended input channels for measuring resistance and RTDs

10 single ended input channels for measuring all other analog signal and sensor types.

Details for connecting analog signals to the *dataTaker 50* as single ended inputs are given throughout Section II.

Single Ended Inputs for the dataTaker 500/600 Series

These single ended input arrangements provide a total number of single ended channels for the *dataTaker 500/600* series loggers as follows

40 single ended input channels for measuring current, 4-20 mA current loop and AD590 temperature sensors

20 single ended input channels for measuring resistance and RTDs

30 single ended input channels for measuring all other analog signal and sensor types.

Details for connecting analog signals to *dataTaker 500/600* series loggers as single ended inputs are given throughout Section II.

Analog Input Types

The different types of analog signals and sensors that are supported by the *dataTaker* are described in detail in Section II of this manual.

Each of the signals and sensors supported are specified in schedule lists using input signal type or sensor type identifiers. These type identifiers generally are abbreviations of the actual signal or sensor name.

The type identifiers, channel capacity, connection configurations, default channel options and other details for the various analog signal and sensor types supported by the *dataTaker* are detailed in the table overleaf.

Analog Input Types

Signal Category	Signal or Sensor Details	Channel Type Label	Example	Channel Options	Channel Factor	Resolution	Units
Voltage	Voltage, ±25mV, ±250mV, ±2500mV	V	1V	(1.0,T*)	attenuation factor	1 microV	mV
	High Voltage, ±7V, ±70V, ±100V,	HV	2HV	(1.0,A)	attenuation factor	250 microV	V
Current	Current, internal 100Ohm shunt, or external shunt	I	3#I	(100.0,T*) (100.0,A)	shunt resistance (ohms)	10 nA	mA
	4-20 mA current loop	L	5#L	(100.0,T*)	shunt resistance (ohms)	0.01%	Percent
Resistance	Resistance by 2, 3, 4 wire methods, 7KOhm max	R	4R(II)	(1.0,I)	scale factor	1 mOhm	Ohms
Bridge	Quarter, half, full bridge, 3 & 4 wire, current excitation	BGI	1BGI(60)	(350.0,II)	arm resistance (ohms)	1 ppm	ppm
	Ratiometric, 4 & 6 wire bridges, voltage excitation	BGV	3BGV	(0.0,V,4W)	offset (ppm)	0.1 ppm	ppm
Frequency	Freguency, 0.102Hz to 20KHz	F	2F	(30.0,T*)	longest period (mSec)	0.01 Hz	Hz
	Period, 50uSec to 9.8 sec	P	7+P	(30.0,T*)	longest period (mSec)	1 mSec	uS
	Vibrating wire sensor, frequency	FW	6FW	(ES9,200.0)	delay (mSec)	0.1°C	DegC*
Temperature	Thermocouples B, C, D, E, G, J, K, N, R, S, T	TB, TC, TD, TE, TG, TJ, TK, TN, TR, TS, TT	3TK	(1.0,T*)	attenuation factor	0.1°C	DegC*
	Platinum RTD, a=0.00385, 0.003916	PT385, PT392	1PT385	(100.0,II)	resistance at 0°C	0.1°C	DegC*
	Nickel RTD, a=0.005001	NI	4NI	(1000.0,II)	resistance at 0°C	0.1°C	DegC*
	Copper RTD, a=0.0039	CU	9CU	(100.0,II)	resistance at 0°C	0.1°C	DegC*
	Thermistors (Yellow Springs 400xx series)	YS101..07 16,17	2YS04	(1e10,I)	parallel resistance (ohms)	0.1°C	DegC*
	AD590 and AD592, Analog Devices	AD590	6AD590	100.0,V)	shunt resistance (ohms)	0.1°C	DegC*
	LM335, National Semiconductor Corp	LM335	3LM335	(2.0,V)	attenuation factor	0.1°C	DegC*
	LM34, LM35, National Semiconductor Corp	LM34 LM35	5LM35	(1.0,V)	calibration factor	0.1°C	DegC*
State	Digital state, definable threshold	AS	7-AS	(2500,T*)	threshold (mV)	1	State
				*differential only			* also DegF, DegK, DegR

Reading Differential Analog Channels

The *dataTaker 50* has 5 differential analog input channels, and the *dataTaker 500/600* series loggers have 10 differential analog input channels. These analog input channels are used as differential channels to differentially measure analog signals.

In the differential input mode, the input signals are connected between the positive (+) terminal and the negative (–) terminal of each channel (See Section II – Differential Analog Input Mode).

Specifying Differential Channels

Differential analog input channels are specified by the general format

ntype(options)

and

n..mtype(options)

where

n	is a single differential analog channel
n..m	is a sequence of differential analog channels
type	is the analog input signal type identifier
(options)	are channel management options

If the analog channels are located on a Channel Expansion Module, then the module number precedes the channel number(s).

Some examples of specifying differential analog input channels using DeTransfer are given below

```
BEGIN
 RA5M
  1V(2.5)
  2L(120)
  3..5TK
  10R(4W)
  8F
END
```

which returns data for the channels every 5 minutes as follows

the voltage input to channel 1, multiplied by an attenuation factor of 2.5

the 4 - 20mA current loop input to channel 2 across an external 120 Ohm shunt

the temperature from three type K thermocouples input to channels 3, 4 and 5

the resistance input to channel 10 by a 4 wire configuration

the frequency input to channel 8

In DeLogger, differential channels are specified for reading in the Program Builder by selecting the channel number and channel types descriptively from the walking menus and dialog boxes. For example

Reading Single Ended Analog Channels

The analog input channels of the *dataTaker* can also be used for single ended inputs, to measure analog signals that are referenced to a common point.

The *dataTaker 50* supports 10 single ended analog inputs (two per physical analog channel), while the *dataTaker 500/600* series loggers support 30 single ended analog inputs (three per physical analog channel).

In the single ended input mode, the input signals are applied to the analog input channels in a number of ways (See Section II - Single Ended Analog Input Mode).

The single ended measurements can be referenced to the *dataTaker* ground, to an external ground different to *dataTaker* ground, or to a 2.5 Volt reference within the *dataTaker*.

Single Ended Inputs Referenced to dataTaker Ground

Single ended analog inputs which are to be measured with reference to *dataTaker* ground are connected to analog channels as follows

between the positive terminal + and Analog Return terminal R

between the negative terminal – and Analog Return terminal R

between the Excite terminal ✳ and Analog Return terminal R. This mode is not suitable for resistance or RTD inputs, and is only available in the *dataTaker 500/600* series loggers

between the Analog Return terminal R and any Ground terminal G for current, current loop and AD590 temperature sensors only

Single Ended Inputs Referenced to an External Ground

Single ended analog inputs that are to be measured with reference to an external ground are connected to analog channels as follows

between the positive terminal + and the SE REF terminal

between the negative terminal – and the SE REF terminal

between the Excite terminal ✳ and the SE REF terminal. This mode is not suitable for resistance, RTD or thermistor inputs, and is only available in the *dataTaker 500/600* series loggers.

The Number of Single Ended Inputs

The single ended configurations provide for the following number of single ended channels in the *dataTaker 50*

15 single ended analog input channels for current, current loop and AD590 measurements

10 single ended analog input channels for all other analog signal measurements.

The single ended configurations provide for the following number of single ended channels in the *dataTaker 500/600* series loggers

40 single ended analog input channels for current, current loop and AD590 temperature sensor measurements

30 single ended analog input channels for all other analog signal measurements.

Single ended input channels cannot be used for some analog input types such as 4 wire resistances, 4 wire RTDs, current excited bridges, etc. as detailed in the table of Analog Input Types above.

Specifying Single Ended Channels

When single ended inputs are measured with reference to the Analog Return terminal, the input mode is specified by suffixing the following special characters to the channel number

✳ for Excite terminal single ended inputs

\+ for positive terminal single ended inputs

– for negative terminal single ended inputs

\# for Return terminal single ended inputs

The general formats of commands for specifying single ended analog inputs referenced to Analog Return follow

n✳type

n+type

n–type

n#type

The *dataTaker 50* does not support the excite terminal as an input terminal.

The positive side of the signal is connected to these terminals, and the negative side is connected to the reference (Analog Return or SE REF). The commands detailed below configure the channel inputs internally for the correct polarity.

Specifying Sequences of Single Ended Channels

The general format for specifying sequences of single ended analog inputs referenced to Analog Return follows

n✳..m✳ type	n✳..m+type	n✳..m–type
n+..m✳ type	n+..m+type	n+..m–type
n–..m✳type	n–..m+type	n–..m–type
	n#..m#type	

where

n	is a single ended analog channel
n..m	is a sequence of single ended analog channels
✳	single ended input between excite ✳ and Analog Return
+	single ended input between positive (+) and Analog Return
–	single ended input between negative (–) and Analog Return
#	single ended input between the Analog Return (R) and Ground
type	is the input signal type identifier

If the analog channels are located on a Channel Expansion Module, then the module number precedes the channel number(s).

When sequences of single ended inputs are specified in these formats, then the actual sequence of successive single ended input channels is interpreted as follows

if the single ended channel sequence is specified in terms of the +ve and –ve terminals only, the sequence is interpreted as successive +ve and –ve terminals. For example

`1+..3–` interprets as 1+ 1– 2+ 2– 3+ 3–

if the single ended channel sequence is specified in terms of the ✳ terminal and either the +ve or the –ve terminal, then the sequence is interpreted as successive ✳, +ve and –ve terminals. For example

`3✳..4–` interprets as 3✳ 3+ 3– 4✳ 4+ 4–

if the single ended channel sequence is specified in terms of the Analog Return # terminal, then the sequence is interpreted as successive Analog Return # terminals. For example

`1#..4#` interprets as 1# 2# 3# 4#

Some examples of specifying single ended analog input channels referenced to the Analog Return (R) using DeTransfer are given below

```
RA1H  3+V  5-V
```

returns data every hour for the single ended voltage input connected between channel 3+ and R, and the single ended voltage connected between channel 5– and R.

```
R5S  1*..2-TK
```

returns data every 5 seconds for the six single ended thermocouples inputs connected between channels 1✳ and R, 1+ and R, 1– and R, 2✳ and R, 2+ and R, 2– and R.

```
R1M  3+..4+F
```

returns data every minute for the three single ended frequency inputs connected between channels 3+ and R, 3– and R, 4+ and R.

In DeLogger, single ended channels referenced to Analog Return (R) are specified for reading in the Program Builder by selecting the channel number and channel types descriptively from the walking menus and dialog boxes.

Single Ended Inputs Referenced to an External Common

When single ended inputs have a common external ground point which is different to *dataTaker* ground, then these can be measured with respect to that external ground by connecting the ground to the SE REF terminal. Measurements made with respect to this external ground are actually made differentially between the input terminal and SE REF. This reference to the external common rather than to *dataTaker* ground is specified by the X (eXternal ground) channel option

channel(X)

The general format for specifying externally referenced single ended analog inputs is the same as that described above for single ended analog inputs referenced to Analog Return.

Some examples of specifying single ended analog input channels referenced to an external ground connected to SE REF using DeTransfer are given below

```
R2H  5+V(X)  6-LM35(X)
```

returns data every 2 hours for a single ended voltage input between channel 5+ and SE REF, and a single ended LM35 input between channel 6– and SE REF.

```
R1M  4+..5*V(X)
```

returns data every minute for the three single ended voltage inputs between channels 4+, 4–, 5✳, and SE REF

```
R10S  3*..4-TK(X)
```

returns data every 10 seconds for the six single ended thermocouples connected between channels 3✳, 3+, 3–, 4✳, 4+, 4– and SE REF.

In DeLogger, single ended channels referenced to an external grounds are specified for reading in the Program Builder by selecting the channel number and channel types descriptively from the walking menus and dialog boxes.

Combining Input Modes

Schedule lists can include a mix of differential and single ended channels, and a mix of single ended references. The list can also include a mix of single channels and channel sequences.

An example of a channel list which includes both differential and single ended analog channels is given below

```
R1D  1+..6-V(X)  7R  8TN  9+F  10*..10-LM34
```

Reading the Same Channel for a Number of Inputs

Each analog input channel of the *dataTaker* data loggers can be read for any number of different analog input types if required.

For example the following readings for a type J thermocouple connected between the +ve and –ve terminals of analog input channel 3 are allowable

```
3V  3+V  3-V  3TJ
```

which will return

the differential voltage output by the thermocouple in response to temperature

the offset voltage or common mode voltage for each side of the thermocouple, with respect to *dataTaker* ground

the temperature sensed by the thermocouple

The Digital Input Channels

The *dataTaker 50* has 5 digital channels, and the *dataTaker 500/600* series loggers have 4 digital channels. The channels are bi-directional TTL/CMOS compatible, and can be used for digital inputs, for digital outputs and for low speed (10 Hz) counting.

The Channel Expansion Module has 20 TTL/CMOS compatible digital input channels, and 10 digital output channels of which 5 are implemented as open collector outputs and 5 are implemented as relays. The digital channels cannot be used for counting.

The digital input channels can be used in a variety of ways as follows

monitoring the logic state of bit inputs

monitoring the logic state pattern of byte inputs

low speed counting to 10 Hz, 16 bit range, presettable (See Section III – The Counter Channels)

detecting digital events to trigger Schedules (See Section III – Schedules Triggered by Digital Events)

detecting count events to trigger Schedules (See Section III – Schedules Triggered by Counter Events)

detecting digital state for conditional trigger (See Section III – Schedules Triggered While Condition)

Note 1 : The digital channels are powered off when the *dataTaker* is in the low power mode. Therefore the digital input channels cannot be used for counting, or detecting events to trigger Schedules, during low power mode operation.

Note 2 : The digital channels of the *dataTaker* are bi-directional, and cannot be used to monitor inputs if they have previously been used as outputs and left switched ON. The channels can be used to monitor digital inputs only if their output state is OFF.

Analog input channels can also be used to monitor digital state. Analog State compares input voltage with a definable threshold to determine logic state. Analog State inputs can be differential or single ended, and cannot be used for counting.

Digital State Inputs

There are several types digital inputs, depending on the type of channel and whether the channels are used as bit or byte inputs. These are detailed in the table below

Digital Input Types

Input Signal	Input Type Identifier	Number of Channels DT50	DT500 DT600	CEM	Units Text
Digital Inputs					
Digital bit	DS	5	4	20	State
Digital byte	DB(mask)		See below		Byte
Analog Inputs					
Differential	AS(threshold)	5	10	10	State
Single Ended to Analog Return	AS(threshold)	10	30	30	State
Single Ended to SE REF	AS(threshold,X)	10	30	30	State

Reading Digital Bit Inputs

Digital bit input channels are specified in a schedule list in the format

nDS

and

n..mDS

where

n	is a single digital channel
n..m	is a sequence of digital channels
DS	is the digital bit input identifier

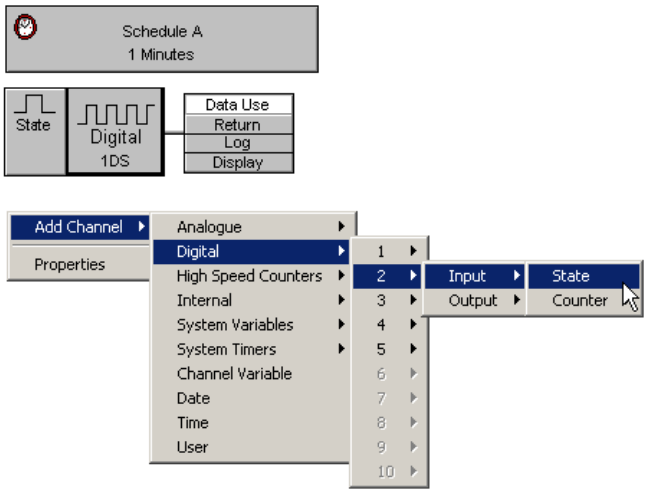
If the digital input channels are located on a Channel Expansion Module, then the module number precedes the channel number(s).

Using DeTransfer, digital bit input channels are read using the commands for example

```
BEGIN
 RA1M
  1DS  2DS  1:3..4DS
END
```

which returns data for the digital bit inputs every minute.

Using DeLogger, digital bit input channels are selected in the Program Builder as follows



The data returned by reading digital bit inputs is either 1 or 0, indicating whether the input is high or low respectively.

When digital bit inputs are averaged (See Section III – Statistical Sub Schedule), data is returned in the range of 0.00 - 1.00. This proportionally represents the number of times that the input was high when sampled during the averaging period. Digital bit averaging can be used as a measure of the duty cycle of the digital inputs.

Reading Digital Byte Inputs

The *dataTaker* is also able to read the digital input channels as byte inputs. Digital byte inputs are implemented as groups of successive digital bit input channels.

The digital byte command can be used to read groups of digital input channels simultaneously, returning a decimal value representing the bit pattern of digital inputs.

Digital bytes are read beginning at a specified digital input channel number, for the next 8 successive digital inputs. If there are fewer than 8 digital inputs beyond the first digital channel (as will be the case for the *dataTaker)* then the byte is returned only for those digital channels present.

The digital byte input is specified in a Schedule list by the general format

nDB(mask)

where

n	is the first channel to begin reading the byte
mask	is an optional bit mask of which bit channels are to be included in the byte (see below)
DB	is the digital byte input identifier

The digital byte input channel number n is defined as the first digital bit input channel to be read. If the digital input channels are located on a Channel Expansion Module, then the module number precedes the channel number(s).

Using DeTransfer, digital byte inputs are read using the commands for example

```
BEGIN
 RA1M
  1DB
END
```

returns the byte for all digital input channels of the *dataTaker* starting from digital input channel 1, and

```
BEGIN
 RA1M
  3DB
END
```

returns the byte for digital input channels 3, 4 and 5 of the *dataTaker 50*, or digital input channels 3 and 4 of *dataTaker 500/600* series loggers.

Digital Byte Mask

The optional digital byte mask allows individual digital input channels within the group to be selected as follows

channels associated with set bits in the mask are included in the byte input

channels associated with clear bits in the mask are not included in the byte input

The relationship between the bit number in the mask and the digital channels is below

```
Mask bit          8  7  6  5  4  3  2  1
Channel 1-4       x  x  x  x  4  3  2  1    DT500
Channel 1-8       8  7  6  5  4  3  2  1    CEM
Channel 5-13     12 11 10  9  8  7  6  5    CEM
                     where x indicates ignored bits
```

The mask is specified as a decimal value equivelent to the required bit pattern. The mask value is the sum of the decimal weighting for each bit, which are as follows

```
Mask bit           8   7   6   5   4   3   2  1
Decimal weight   128  64  32  16   8   4   2  1
```

Some examples of mask values and associated bit patterns follow

```
3       0 0 0 0 0 0 1 1
15      0 0 0 0 1 1 1 1
129     1 0 0 0 0 0 0 1
255     1 1 1 1 1 1 1 1
```

If a mask is not specified, then the mask defaults to 255 and all channels from the nominated first channel are included in the byte.

The range of the data returned by reading the full byte is as follows

dataTaker 50	0 – 31
dataTaker 500/600 series loggers	0 – 15
Channel Expansion Module	0 – 255

When digital byte inputs are averaged, data is returned in the range

dataTaker 50	0.00 – 31.00
dataTaker 500/600 series loggers	0.00 – 15.00
Channel Expansion Module	0.00 – 255.00

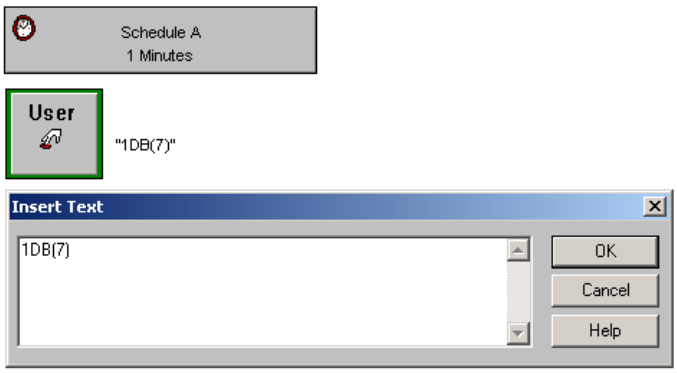
This average represents a weighted proportion the number of times that the individual digital inputs were true or high when sampled during the averaging period.

Using DeTransfer, masked digital byte inputs are read using the commands for example

```
BEGIN
 RA1M
  1DB(7)
END
```

will return the byte beginning at digital input channel 1, and masks to include the first three channels 1, 2 and 3.

DeLogger does not directly support digital bytes, however these can be read using the User channel in the Program Builder as follows



Analog State Inputs

The analog input channels of the *dataTaker* can be used to monitor logic state using the Analog State function.

Differential or single ended input voltage levels are measured and compared to a user definable threshold to determine the logic state.

In contrast to TTL/CMOS compatible logic state inputs where the logic thresholds are <1.5 Volt for 0, and >3.5 Volts for 1, threshold for the Analog State input type is variable and is defined as a voltage in the range of -2500 mV to +2500 mV.

Analog State does not support byte inputs, or counters, and cannot be used to trigger Schedules.

If the input voltage is greater than the ±3.5 Volt common mode range of the *dataTaker*, then it is treated as a 5 Volt level. For threshold comparison the input is treated as logic 1, and provides compatibility with TTL and CMOS inputs.

Analog State of higher voltages, such as 24VDC industrial logic, can also be measured by externally attenuating the signals before connection to the logger.

Reading Differential Analog State Inputs

When Analog State is to be measured for differential inputs, the voltage is applied between the +ve and –ve terminals of the analog input channel (See Section II - Measuring Analog State).

The measured differential voltage is compared with the threshold voltage which is defined as a channel option for the channel, and the Analog State is evaluated as follows

if the measured voltage is less than the threshold voltage,
then logic 0 is returned

if the measured voltage is greater than or equal to the threshold voltage,
then logic 1 is returned

Differential Analog State input channels are specified in a schedule list by the general formats

nAS(threshold)

and

n..mAS(threshold)

where

n	is a single differential AS channel
n..m	is a sequence of differential AS channels
AS	is the input signal type identifier
threshold	is the threshold voltage for logic comparison

If the analog input channels are located on a Channel Expansion Module, then the module number precedes the channel number(s).

The threshold voltage must be defined in the range of -2500 mV to +2500 mV. If a threshold voltage is not defined, then the threshold voltage defaults to 2500 mV.

Using DeTransfer, differential Analog States are measured by the commands for example

```
BEGIN
 RA2M
  1AS  2AS(1000)
END
```

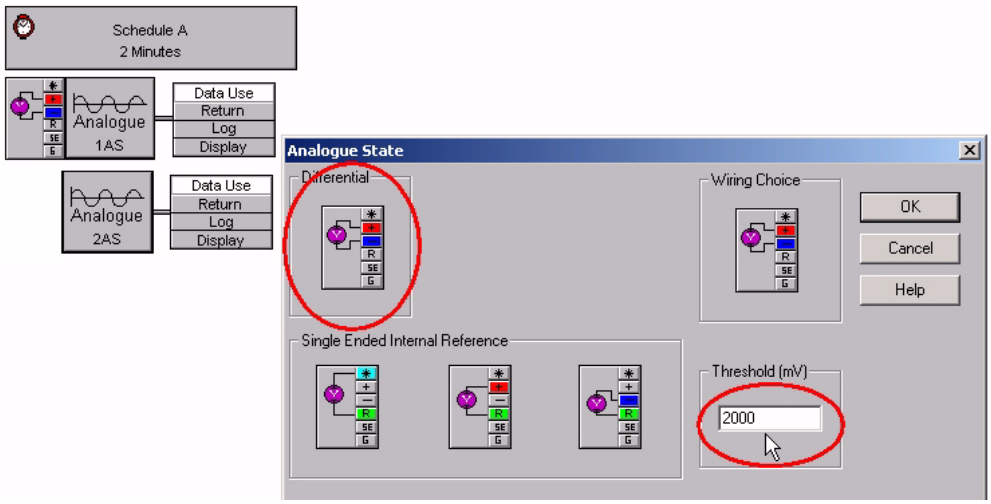
which instructs the *dataTaker* to measure analog states every 2 minutes as follows

compare the differential voltage on channel 1 with the default 2500 mV threshold

compare the differential voltage on channel 7 with a threshold of 1000 mV

The *dataTaker* returns logic 0 if the measured voltage is less than the threshold, and returns 1 if the measured voltage is greater than the threshold. Note that different channels can have different thresholds.

Using DeLogger, differential Analog States can be measured by the following Program Builder program. The analog connection and threshold are defined in the Analog State dialog which opens when you have selected the channel and input type



Reading Single Ended Analog State Inputs

When Analog State is to be measured for single ended inputs, the voltage is applied between the +ve, –ve or ✳ terminal of the analog input channel, and the Analog Return (R) or SE REF terminal. (See Section II - Measuring Analog State).

The measured single ended voltage is compared with the threshold voltage, which is defined as a channel option to the channel type, and the Analog State is evaluated as follows

if measured voltage is less than threshold voltage, then logic 0 is returned

if measured voltage is greater than threshold voltage, then logic 1 is returned

Single ended Analog State input channels are specified in a schedule list in the general formats

n ✳ AS(threshold)	n+AS(threshold)	n–AS(threshold)

and

n✳..m✳ AS(threshold)	n✳..m+AS(threshold)	n✳..m–AS(threshold)
n+..m✳ AS(threshold)	n+..m+AS(threshold)	n+..m–AS(threshold)
n–..m✳ AS(threshold)	n–..m+AS(threshold)	n–..m–AS(threshold)

where

n	is a single ended AS channel
n..m	is a sequence of single ended AS channels
✳	single ended input between excite and R or SE REF terminals
+	single ended input between positive and R or SE REF terminals
–	single ended input between negative and R or SE REF terminals
AS	is the input signal type identifier
threshold	is the threshold voltage for logic comparison

Note that the *dataTaker 50* does not support the excite terminal as an input terminal.

If the analog input channels are located on a Channel Expansion Module, then the module number precedes the channel number(s).

Threshold voltage is defined in the range of -2500 mV to +2500 mV. If a threshold is not defined, then the threshold voltage defaults to 2500 mV.

If the single ended Analog State input is referenced to SE REF, then this must be specified as a X (eXternal reference) channel option as follows

n+AS(threshold,X)

If this option is not specified, then the single ended Analog State is measured with respect to Analog Return.

Using DeTransfer, single ended Analog States are measured by commands for example

```
BEGIN
 RA10M
  1+AS   3+AS(1000)   9-AS(-2500,X)
END
```

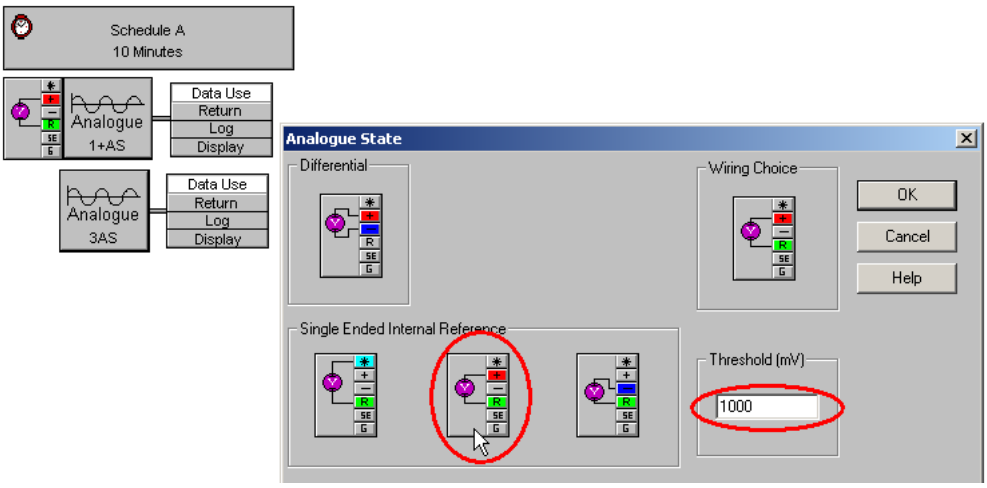
which instructs the *dataTaker* to measure analog states every 10 minutes as follows

compare the single ended voltage on channel 1+ and Analog Return, with the default 2500 mV threshold

compare the single ended voltage on channel 3+ and Analog Return, with a threshold of 1000 mV

compare the single ended voltage input to channels 9– and SE REF with a threshold of –2500 mV

Using DeLogger, Analog States can be measured by the following Program Builder program. The analog connection and threshold are defined in the Analog State dialog which opens when you have selected the channel and input type



Gray Code Conversion

The digital input channels of the *dataTaker* data loggers also support the Gray code binary system. Gray Code is often used for wind direction indicators and shaft encoders which have multi bit digital outputs.

The principal of Gray code converters is that only one bit changes for each increment. This eliminates false intermediate codes that could occur in natural binary conversion.

The *dataTaker* implements Gray code conversion as Intrinsic Function F7, which is attached to a channel variable as a channel option. (See Section III - Scaling Data – Polynomials, Spans and Functions)

The following example illustrates programming from DeTransfer for a 6 bit wind direction sensor. Using a *dataTaker* 50 the 5 digital input channels are used as the first 5 bits of the encoder output, and an analog state is used for 6th bit and is the highest order bit.

```
BEGIN
S1=0,360,0,63"Deg."    'Span signal from 0 to 63
                       'to 0 to 360 degrees
RA1S
 1DB(=1CV,W)            'Assign to Chan Variable
 1AS(=2CV,W)            'Assign to Chan Variable
 3CV(W)=1CV+(2CV*32)    'Combine 1DB & 1AS
 4CV(W)=3CV(F7)         'Convert Gray code
 4CV(S1,Wind Dir.")     'Apply Span
END
```

Channel Expansion Modules have 2 full 8 bit byte channels available. Therefore a 6 or 8 bit Gray code can be read directly. However if a 6 bit Gray code is used, then the 7th & 8th bits must be grounded or the first 6 bits masked by 63 (00111111).

Digital Event Inputs

The digital input channels of the *dataTaker* can also be used to detect external digital events, which can then be used to trigger Schedules.

A digital event may be defined to be a negative transition, a positive transition, or both. The use of digital events for triggering Schedules is discussed in Section III - Schedules Triggered by Digital Events.

The digital input channels of the Channel Expansion Modules cannot be used to trigger Schedules.

Digital While Inputs

The digital input channels of the *dataTaker* can also be used to monitor external digital conditions, and use these conditions to control the triggering of Schedules only while the particular digital input is true or high.

Use of digital input channels for conditional triggering of Schedules is discussed in Section III - Schedules Triggered While Condition.

The digital input channels of the Channel Expansion Modules cannot be used for conditional scheduling.

The Counter Channels

The *dataTaker* supports three different types of counters as follows

Low speed counters

High speed counters

Phase encoder counter

The counters are all 16 bit counters, and count upwards over the range of 0 to 65535. When the maximum count is reached, the next transition on the input causes the counters to 'roll over' to 0.

The counters begin counting at zero by default, however an initial count other than zero can be defined for individual counters. The counter begins counting from the initial count and 'rolls over' to zero when the maximum count is reached.

The count range for individual counters can be redefined within the range of 1 to 65535, such that the counter 'rolls over' at maximums of less than 65535.

The phase encoder counter is a special purpose counter, which has two inputs. The counter increments when the two pulse trains are in one phase, and decrements when the pulse trains are in the reverse phase. By default the phase encoder counter counts over the range of 0 to 65535 beginning at zero, however the start count and range can also be changed.

The *dataTaker* counters run automatically, and do not need to be specifically enabled. The counters can be included in Schedules and Alarms, and can be singularly read by the Immediate Schedule.

When counters are included in a Schedule, or as inputs in Alarms, they are not automatically cleared to zero. The counters begin at whatever count they contain, and so if a zero initial count is required then the counters must firstly be set to zero before scanning begins.

All counters are cleared to zero by the RESET command.

The low speed counters are software counters and have a maximum frequency of 10Hz. The low speed counters are associated with the digital input channels of the *dataTaker*.

The *dataTaker 50* has 5 low speed counters. The *dataTaker 500/600* series loggers have 4 low speed counters.

The high speed counters are hardware counters, with a maximum frequency of 1KHz or 1MHz. All of the *dataTaker*s have 3 high speed counters.

The Channel Expansion Module has no counters.

All counter inputs are compatible with TTL and CMOS logic inputs, and are compatible with voltage free contact closure inputs.

Counter Modes

The *dataTaker* counters can be used in either of two modes

accumulating or totalising counters, which increment counts and when read return the current accumulation of counts. Successive reads return the progressive accumulation of counts.

resetting or interval counters, which increment counts and when read return the current accumulation of counts, and are reset to zero again. Successive reads return the number of counts recorded during the period since the last scan.

Individual counters can be used either as accumulating counters or as resetting counters, but cannot be used both as accumulating and resetting counters in the same application.

Counter Types

The counter type identifiers for the low speed counters, high speed counters and the phase encoder counter are detailed in the table below

Counter Types

Counter	Input Type Identifier	Number of Counters DT50	DT500 DT600	CEM	Units Text
Low Speed Counters					
- accumulating	C	5	4	0	Counts
- resetting	C(R)	5	4	0	Counts
High Speed Counters					
- accumulating	HSC	3	3	0	Counts
- resetting	HSC(R)	3	3	0	Counts
Phase Encoder Counter					
- accumulating	PE	1	1	0	Counts
- resetting	PE(R)	1	1	0	Counts

Low Speed Counters

The low speed counters are software counters, which share the digital input channels of the *dataTaker*. The counters increment whenever a negative transition occurs on the digital input channel.

The low speed counters count upwards, with a maximum frequency of 10Hz default. The maximum frequency is determined by the digital input sample period, which defaults to 50mS. The sample period can be changed by the Parameter13 command, however if this is set too fast then the *dataTaker* will become slow in other functions.

These low speed counters are 16 bit counters, with a count range of 0 to 65535. They can be used as accumulating counters, or as resetting counters.

The *dataTaker 50* has 5 low speed counters, *dataTaker 500/600* loggers have 4 low speed counters.

Note 1: The low speed counters are powered down whenever the *dataTaker* drops to the low power mode, and cannot be used for counting during these periods.

Note 2: The digital channels used for the low speed counters are bi-directional, and cannot be used as counters if they have previously been used as digital outputs and left switched ON. The digital channels can be used as counters only if their output state is OFF.

Setting the Low Speed Counter Initial Count

The low speed counters normally begin counting from zero, and increment to the maximum count. However the counters can be set to an initial count as follows

nC=count

n..mC=count

where

n	is a single counter
n..m	is a sequence of counters
C	is the low speed counter identifier
count	is the initial count

The initial count can be specified as a constant, a channel variable or an expression. If the value of the initial count is a decimal value, then the value is rounded to the nearest integer.

If the value of the initial count is greater than 65535, then an “E15-assignment error” is returned and the counter is initialised to 99999.9 which is the universal *dataTaker* error value.

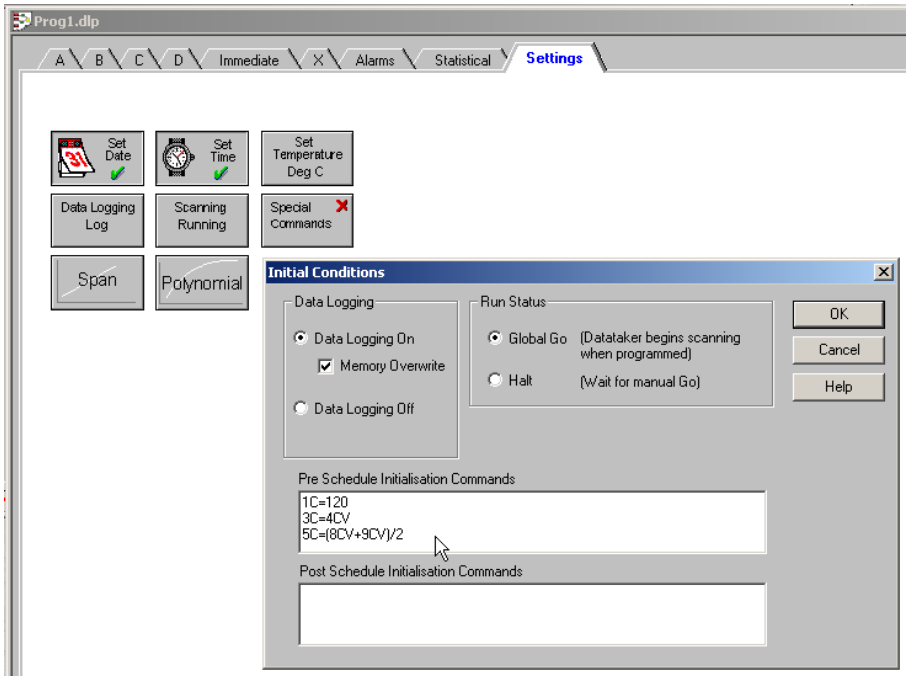
The counter begins counting from the initial count and 'rolls over' to zero when the maximum count is reached.

Using DeTransfer, the initial count of low speed counters is set by the commands for example

```
1C=120
3C=4CV
5C=(8CV+9CV)/2
```

Whenever the initial count of the low speed counters is set, the counter is read and the new value is returned.

Using DeLogger, the initial count for low speed counters is set in the Pre Schedule Initialization Commands under the Settings tab of the Program Builder. Commands in this position are executed once when the logger is initially programmed.



The return of the new count can be disabled by including the W (Working)or NR (No Return) channel options in the counter specification as follows

```
1C(W)=120
3C(NR)=4CV
5C(W)=(8CV+9CV)/2
```

Commands to set the initial count of low speed counters should not be included in Schedules, since this will cause the counters to be set to the initial count each time the Schedule runs.

Setting the Low Speed Counter Range

The low speed counters normally have a count range of 0 to 65535. However a different count range can be specified by including the required range as a channel option to the counter as follows

nC(range)

where

n	is the counter number
C	is the low speed counter identifier
range	is a value in the range of 0 to 65535

If the value of the range specified is greater than 65535, then the range is set to 65535. If the value for the range is a decimal value, then this is truncated to an integer.

The counter counts from the initial value, and then 'rolls over' to zero when the range is reached.

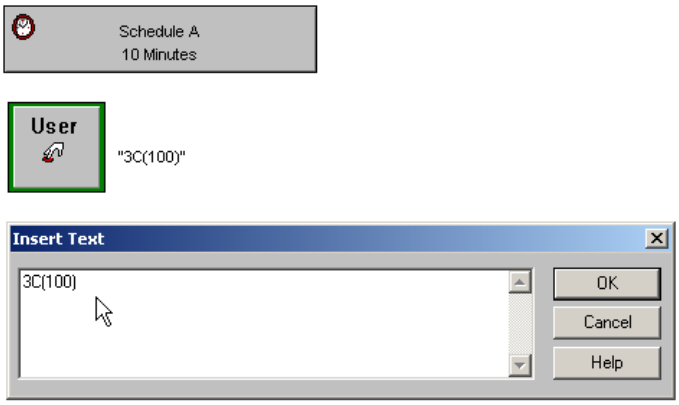
Using DeTransfer, the count range of low speed counters is set by the commands for example

```
1C(50)

BEGIN
 RA1M
  3C(100)
END
```

The command to set the range of low speed counters can either be used directly as an initialization, or within Schedules.

Using DeLogger, the count range for low speed counters can be set in a Schedule using the User channel in the Program Builder (requires DeLogger Ver 4.2.16 or later), or can be initialised in the Pre Schedule Initialization Commands under the Settings tab of the Program Builder (see above for example of this).



Setting the Low Speed Counter Range and Initial Count

The range and initial count can be set in the same command if required. Using DeTransfer, this can be done by the command for example

```
2C(200)=50
```

which starts the counter at 50 counts, and sets the rollover at 200 counts.

Reading the Low Speed Counters

The low speed counters are specified in a schedule list in the formats

nC

nC(R)

n..mC

n..mC(R)

where

n	is a single counter
n..m	is a sequence of counters
C	is the low speed counter type identifier
R	is the channel option for a resetting counter

The low speed counters function as accumulating counters unless the R channel option is included specifying a resetting counter.

The low speed counters can be used either as accumulating counters or as resetting counters. They cannot be used both as accumulating and resetting counters in the same application.

Using DeTransfer, the low speed counters is read by the command for example

```
BEGIN
 RA1M
  1C  2C(100)  3..4C(R)
END
```

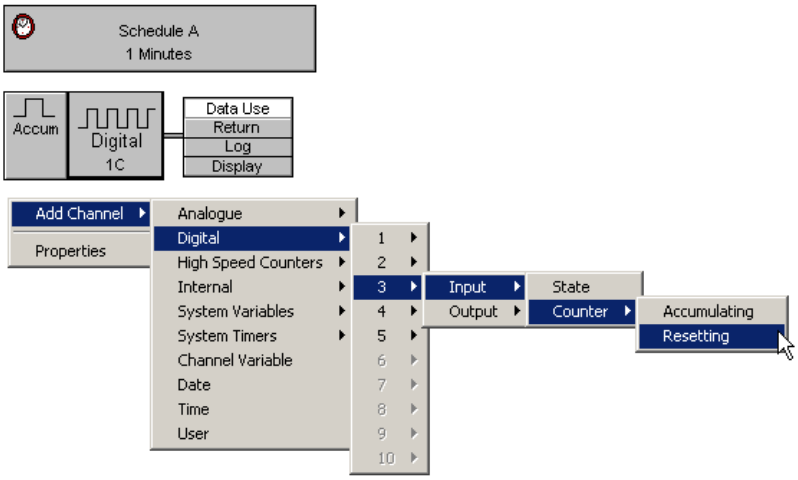
which instructs the *dataTaker* to read the low speed counters every minute as follows

counter 1 is read as an accumulating counter, range is 65535

counter 2 is read as an accumulating counter, range 100

counters 3 and 4 are read as resetting counters, that are reset to zero each time they are read.

Using DeLogger, the low speed counters is read by the program for example in the Program Builder



Data from the low speed counters is returned as integers with units of Counts. When the counters are averaged, the data is returned as floating point data to two decimal places.

The low speed counters can also be used to produce counter events, for triggering Schedules (See Section III –Schedules Triggered by Counter Events).

High Speed Counters

The three high speed counters of the *dataTaker* have a maximum frequency of 1 KHz by default, which can be increased to 1 MHz by removing a capacitor from the counter input circuits (contact your *dataTaker* supplier for this modification).

The high speed counters count upwards, and increment whenever a negative transition occurs on the input terminal.

These counters remain powered at all times, including whenever the *dataTaker* is operating in the low power mode. Therefore they are suitable for counting pulses when the *dataTaker* is battery powered and frequently sleeps down to conserve power.

These high speed counters are 16 bit counters, with a count range of 0 to 65535. They can be used either as accumulating counters, or as resetting counters.

High speed counter number 1 also has an output function, which can be used to generate a user definable frequency in the range of 1 to 19200Hz.

High Speed Counter 1 Input

In Series 3 *dataTakers,* the input to high speed counter 1 can be configured in a number of ways depending on the setting of Parameter27 as follows:

P27=0	external frequency input via C1 terminal, gate is permanently enabled
P27=1	an internal 32767 Hz frequency is input to high speed counter 1, and the gate is set by the state on C2, the input for high speed counter 2
P27=2	external frequency input via C1 terminal, , and the gate is set by the state on C2, the input for high speed counter 2
P27=3	an internal 1024 Hz frequency is input to high speed counter 1, and the gate is permanently enabled

The gate for high speed counter 1 is active low, and can be enabled by a contact closure to *dataTaker* ground.

Setting the High Speed Counter Initial Count

The high speed counters normally begin counting from zero, and increment to the maximum count. However the counters can be set to an initial count as follows

nHSC=count

n..mHSC=count

where

n	is a single counter
n..m	is a sequence of counters
HSC	is the high speed counter identifier
count	is the initial count

The initial count can be specified as a constant, a channel variable or an expression. If the value is a decimal value, then it is rounded to the nearest integer.

If the value of the initial count is greater than 65535, then an “E15-assignment error” is returned and the counter is initialised to 99999.9 which is the universal *dataTaker* error value.

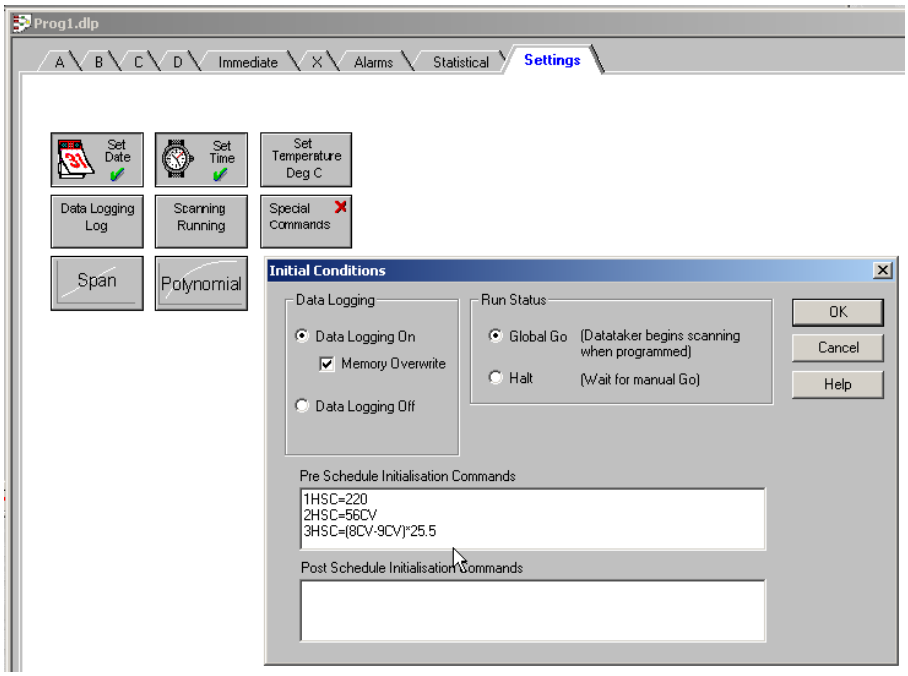
The counter begins counting from the initial count and 'rolls over' to zero when the maximum count is reached.

Using DeTransfer, the initial count of high speed counters is set by the commands for example

```
1HSC=220
2HSC=56CV
3HSC=(8CV-9CV)*25.5
```

When the initial count of the high speed counters is set, the counter is read and the new value is returned.

Using DeLogger, the initial count for high speed counters is set in the Pre Schedule Initialization Commands under the Settings tab of the Program Builder. Commands in this position are executed once when the logger is initially programmed.



The return of the new count can be disabled by including the W (Working)or NR (No Return) channel options in the counter specification as follows

```
1C(W)=120
3C(NR)=4CV
5C(W)=(8CV+9CV)/2
```

Commands to set the initial count of high speed counters should not be included in Schedules, since this will cause the counters to be set to the initial count each time the Schedule runs.

Setting the High Speed Counter Range

High speed counters normally have a count range of 0 to 65535. However a different count range can be specified by including the required range as a channel option to the counter as follows

nHSC(range)

where

n	is the counter number
HSC	is the high speed counter identifier
range	is a value in the range of 0 to 65535

If the value of the range specified is greater than 65535, then the range is set to 65535. If the value for the range is a decimal value, then this is truncated to an integer.

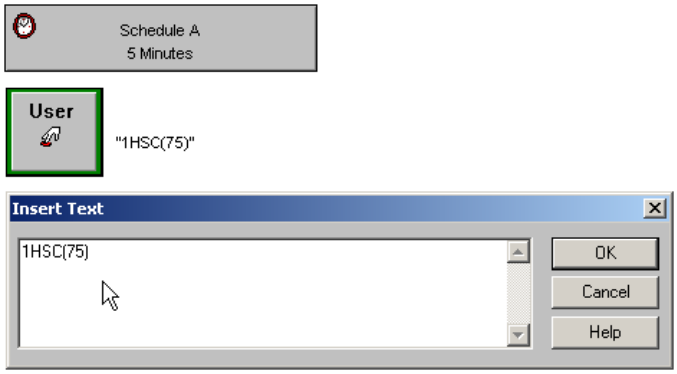
The counter counts from the initial value, and then 'rolls over' to zero when the range is reached.

Using DeTransfer, the count range of high speed counters is set by the command for example

```
1HSC(75)
BEGIN
 RA5M
  3HSC(500)
END
```

The command to set the range of high speed counters can either be used directly as an initialization, or within Schedules.

Using DeLogger, count range for high speed counters can be set in a Schedule using the User channel in the Program Builder (requires DeLogger Ver 4.2.16 or later), or can be initialised in the Pre Schedule Initialization Commands under the Settings tab of the Program Builder (see above for example of this).



Setting the High Speed Counter Range and Initial Count

The range and initial count can be set in the same command if required. Using DeTransfer, this can be done by the command for example

```
3HSC(5000)=155
```

which starts the counter at 155 counts, and sets the rollover at 5000 counts.

Reading the High Speed Counters

The high speed counters are specified in a schedule list in the formats

nHSC

nHSC(R)

n..mHSC

n..mHSC(R)

where

n	is a single counter
n..m	is a sequence of counters
HSC	is the high speed counter type identifier
R	is the channel option for a resetting counter

The high speed counters function as accumulating counters unless the R channel option is included specifying a resetting counter.

The high speed counters can be used as accumulating counters or as resetting counters. They cannot be used both as accumulating and resetting counters in the same application.

Using DeTransfer, the low speed counters is read by the command for example

```
BEGIN
 RA15M
  1HSC  2HSC(500)  3HSC(R)
END
```

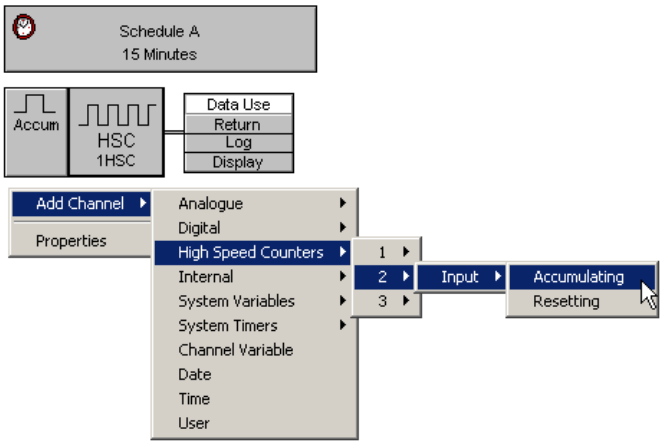
which instructs *dataTaker* to read the high speed counters every 15 minutes as follows

counter 1 is read as an accumulating counter, range is 65535

counter 2 is read as an accumulating counter, range 500

counter 3 is read as a resetting counter, which is reset to zero each time it is read.

Using DeLogger, the high speed counters is read by the program for example in the Program Builder



Data from the high speed counters is returned as integers with units of Counts. When the counters are averaged, the data is returned as floating point data to two decimal places.

The high speed counters can also be used to produce counter events, for triggering Schedules (See Section III – Schedules Triggered by Counter Events).

High Speed Counter 1 Output

High speed counter 1 has an output function, which can be used as a programmable prescaler or frequency divider to produce a variable frequency on the output terminal $C1_{out}$ of high speed counter 1. The frequency can be used directly, or fed to frequency to voltage or frequency to 4-20mA converters to produce an analog output.

The output function requires that a reference frequency is applied to the input terminal of high speed counter 1, and the *dataTaker* program assigns a divisor to the counter to provide the desired output frequency. The output frequency is a sub-frequency of the input frequency, and cannot be greater than the input frequency.

The reference frequency can either be provided in various ways as follows:

All Series of dataTaker – Connect a suitable frequency from an external source to C1, the input terminal of 1HSC

Series 1 and Series 2 of dataTaker - internally connect the 19200 Hz frequency from the baud rate generator to the input of 1HSC (this modification can be performed by your *dataTaker* supplier)

Series 3 of dataTaker – input to 1HSC is determined by the setting of Parameter27**.** See High Speed Counter 1 Input above.

If the input frequency is greater than 1KHz, then a capacitor must be removed from the counter input circuits to increase the sensitivity of the counter. This modification can be performed by your *dataTaker* supplier.

The output function of high speed counter 1 is defined by the general command format

```
1HSCO(mode)=N
```

where

1HSCO	is the high speed counter output type identifier
mode	defines the output waveform
N	counter range as a constant, channel variable or expression

The mode defines the frequency waveform output on the $C1_{out}$ terminal, for which the settings are as follows

mode = 0	output switches high after N pulses are received on the input
mode = 2	the output is held high for N-1 pulse widths on the input, and low for 1 pulse width on the input
mode = 3	the output is high for N / 2 pulse widths on the input, then low for N / 2 pulse widths on the input
mode = 4	the output is held high for N pulse widths on the input, and low for 1 pulse width on the input

The range value N can be specified as a constant, a channel variable or an expression.

The most common use of the output is mode 3, which is effectively a frequency divider, where N is the divisor

$$F_{out} = F_{in} / N$$

If the divisor is a channel variable or an expression, then the divisor can be changed dynamically during the course of an application thereby producing a variable output.

The divisor must not be greater in value than the input frequency. If the divisor is a decimal value, then this is rounded to the nearest integer.

Using DeTransfer, the high speed counter 1 can be used as a frequency divider by the commands for example

```
'Input frequency 19.2KHz
1HSCO(3)=48
```

which will produce a square wave output at a frequency of 400Hz on $C1_{out}$.

```
'Input frequency 19.2KHz
BEGIN
  R1S
   5TT(=1CV,W)                    'Temperature 1
   6TT(=2CV,W)                    'Temperature 2
   1HSCO(3)=ABS(1CV-2CV)*100
END
```

will produce a square wave output at a variable frequency which is proportional to the difference in the two temperatures.

The output from high speed counter 1 can be connected to the input of another high speed counter, and used to measure the output frequency if necessary.

DeLogger does not directly support the high speed counter 1 output, however the commands shown above for DeTransfer can be produced in the Program Builder using the User channel (requires DeLogger Ver 4.2.16 or later).

High speed counter 1 functions as a normal counter when not being used for output. However if the input frequency has been internally connected from the 19.2KHz baud rate generator, then high speed counter 1 will not be available for use as a normal counter.

Phase Encoder

The phase encoder counter is a special purpose counter, which has two inputs. The counter increments when the two pulse trains are in one phase, and decrements when the pulse trains are in the reverse phase.

Phase encoders are used either to measure rotary motion directly, or to measure linear motion which is translated to rotary motion by a wheel or pulley.

Applications for phase encoders include measurement of rotation, distances, and levels such as river height, tank level, etc.

By default the phase encoder counter counts over the range of 0 to 65535 beginning at zero, however the initial count can be changed. Maximum input frequency is 10Hz.

The phase encoder is implemented via digital input channels 3 and 4 of the *dataTaker* (See Section II - The Phase Encoder).

Note 1: The phase encoder is powered down when the *dataTaker* drops to the low power mode, and cannot be used for counting during these periods.

Note 2 : The digital channels used for the phase encoder are bi-directional, and cannot be used for phase decoding if they have previously been used as digital outputs and left switched ON. The digital channels can be used for phase encoding only if their output state is OFF.

Setting the Phase Encoder Initial Count

The phase encoder counter normally begins counting from zero, and increments or decrements from this value. However the counter can be set to an initial count as follows

1PE=count

where

1PE	is the phase encoder counter identifier
count	is the initial count

The initial count can be specified as a constant, a channel variable or an expression. If the value of the initial count is a decimal value, then the value is rounded to the nearest integer.

If the value of the initial count is greater than 65535, then an “E15-assignment error” is returned and the counter is initialised to 99999.9 which is the universal *dataTaker* error value.

Generally the phase encoder counter should be set to the mid-range (32,767), and the rotation geared such that the counter will not 'roll over' in either direction.

Using DeTransfer, the phase encoder can be set up using the following commands for example

```
1PE=32767
1PE=8CV/2
```

Whenever the initial count of the phase encoder is set, the counter is read and the new value is returned. The return of the new count can be disabled by including the W or NR channel options in the counter specification as follows

```
1PE(W)=32767
1PE(NR)=8CV/2
```

DeLogger does not directly support the phase encoder, however the commands shown above for DeTransfer can be produced in the Program Builder using the User channel (requires DeLogger Ver 4.2.16 or later).

Commands to set the initial count of phase encoder should not be included in Schedules, since this will cause the phase encoder to be set to the initial count each scan.

Setting the Phase Encoder Counter Range

It is generally not appropriate to set the range for the phase encoder counter.

However if setting the range is required, then this can be done in the same manner as for the low and high speed counters.

Reading the Phase Encoder

The phase encoder is specified in a schedule list by the general format

1PE

where PE is the phase encoder type identifier

The phase encoder is an up-down counter, and using it as a resetting counter 1PE(R) is generally not appropriate to the application.

Using DeTransfer, the phase encoder is initialised and read by the commands for example

```
1PE=32767
BEGIN
 RA1H
  1PE
END
```

which sets the initial count to the mid-range, and reads the phase encoder every hour.

DeLogger does not directly support the phase encoder, however the commands shown above for DeTransfer can be produced in the Program Builder using the User channel (requires DeLogger Ver 4.2.16 or later).

Counter Event Inputs

The low speed and high speed counters of the *dataTaker* can also be used to generate counter events to trigger Schedules (See Section III – Schedules Triggered by Counter Events).

Low speed counter events cannot be used while the *dataTaker* is operating in the low power mode, since during this period the digital input channels are powered down and incoming pulses are ignored.

The high speed counter events can be used while the *dataTaker* is operating in the low power mode. The incoming event is detected and counted, however the incoming pulse train must also be diode connected to the Wake terminal to wake the logger to respond to the event.

The Digital Output Channels

The *dataTaker 50* has 5 bi-directional TTL/CMOS compatible digital channels, which can be used for digital inputs or digital outputs.

The *dataTaker 500/600* series of data loggers have 4 bi-directional TTL/CMOS compatible digital channels, which can be used for digital inputs or digital outputs.

As digital output channels, these are implemented as open collector CMOS/TTL compatible logic outputs.

The Channel Expansion Module has 10 digital output channels, of which 5 are implemented as open collector CMOS/TTL compatible logic outputs, and 5 are implemented as single pole relays.

These open collector output channels are active low, in that they switch low to ground when turned ON. This allows the connection of low voltage relays directly between a supply voltage and the output channel terminal.

An optional delay period can be invoked after switching the digital outputs, which pauses the *dataTaker* for a period before proceeding with execution of the *dataTaker* program. During the delay period only time keeping, counting and servicing of the serial interfaces is maintained.

An optional reset can be specified when switching digital outputs, to reset the digital outputs to the opposite state to that selected thereby producing a pulse output.

Digital Output Types

There are several digital output type identifiers, depending on the type of channel and whether the channels are used for bit or byte outputs. These are detailed in the table below

Digital Output Types

Output Signal	Output Type Identifier	Number of Output Channels		
		DT50	DT500 DT600	CEM
Digital Bit Output	DSO	5	4	10
Digital Byte Output	DBO(mask)		see below	
Warning Outputs	WARN	5 if have a display	5 if have a display	0

Switching the Digital Bit Outputs

The digital bit output channels are switched ON by assigning any positive or negative value to the channel, and are switched OFF by assigning a 0 to the channel.

An optional delay period can be invoked after switching the digital outputs, which pauses the *dataTaker* to allow external conditions to settle before proceeding with execution of the *dataTaker* program. The delay can be from 1 to 65535 mS.

An optional reset can be specified when switching digital outputs, to reset the digital output to the opposite state to that selected thereby producing a pulse output. If a delay period is specified, then the output is reset after the delay.

The digital bit output channels are switched by commands in the general formats

nDSO(delay,R)=state

and

n..mDSO(delay,R)=state

where

n	is a single digital output channel
n..m	is a sequence of digital output channels
DSO	is the digital bit output channel identifier
delay	is an optional delay period in millisecs after switching
R	optional reset to opposite state after switching
state	indicates the state to which the output is set

If the digital output channels are located on a Channel Expansion Module, then the module number precedes the channel number(s).

The optional delay period is specified as an integer within the range of 1 to 65535 mS (a little more than 1 minute). If there is no delay period specified, the digital output is switched and the *dataTaker* continues without delay.

If the reset option is not specified, then the digital output remains in the state to which it is switched.

The state can be specified as a constant, a channel variable or an expression. If the value for state is a decimal value, then the value is rounded to the nearest integer.

A state value of 0 will turn the output OFF, while any value other than 0 will turn the output ON.

Using DeTransfer, digital output channels can be switched by commands for example

```
1DSO=1
2DSO=(5CV/3)
1..4DSO=0
1:2..5DSO=1
```

which

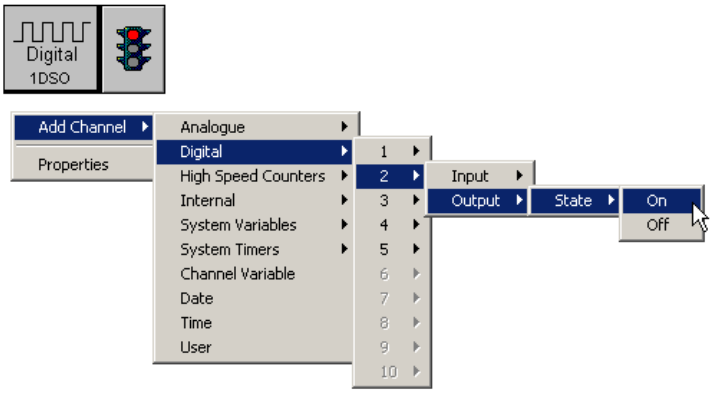
turns digital output channel 1 of the *dataTaker* ON

turns digital output channel 4 of the *dataTaker* ON (assuming the expression does not evaluate to zero)

turns digital output channels 1 through 4 of the *dataTaker* OFF

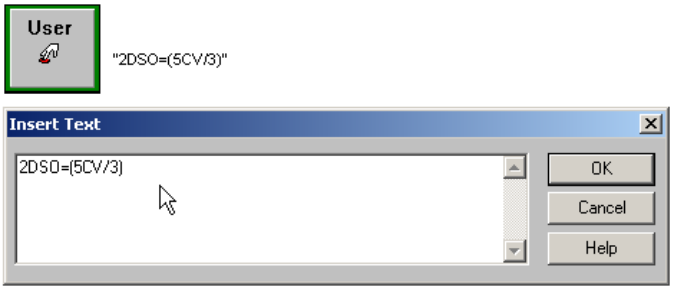
turns digital output channels 2 through 5 of Channel Expansion Module ON

Using DeLogger, digital output channels can be directly switched in the program created in the Program Builder as follows



The ON state is indicated by a green traffic light, and the OFF state is indicated by a red traffic light.

However this method does not support assigning expressions to the digital output channels. Expressions can only be assigned by using the User channel in the Program Builder (DeLogger ver 4.2.16 or later) as follows



Delay Period and Reset

The optional delay period pauses the *dataTaker* before proceeding with execution of the *dataTaker* program. During the delay period only time keeping, counting and servicing of the serial interfaces is maintained.

However if the delay period is used when *dataTaker* networks are implemented, then delay periods of greater than 1000mS will slow communications over the network. Similarly long delay periods will slow response by the *dataTaker* when being polled for data by the host.

The reset channel option R resets the digital outputs to the opposite state to that selected, thereby producing a pulse output.

If a delay period is not specified, then the digital output is toggled quickly, producing a pulse width of 10 [a]– 100mS. If however a delay period is specified, then the output is reset after the delay period producing a longer pulse.

The delay period and reset option have limited application when switching the digital outputs directly from commands, but have considerable application when switching digital outputs from within Schedules (See Section III – Switching Digital Outputs in Schedules).

Using DeTransfer, the digital output channels can be switched with delays and reset by commands for example

```
4DSO(1000)=1
```

which turns digital output channel 4 of the *dataTaker* ON, and then pauses the program for 1000mS (1 sec) before continuing

```
2DSO(7500,R)=1
```

which turns digital output channel 2 of the *dataTaker* ON, pauses the program for 7.5 seconds, and then resets the output OFF

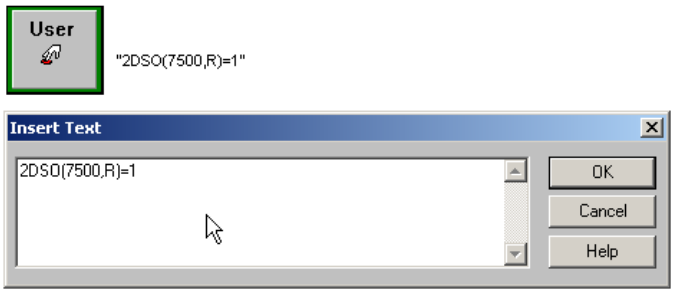
```
1:8DSO(5000,R)=0
```

which turns digital output channel 8 of Channel Expansion Module 1 OFF, pauses the program for 5 seconds, and then resets the output ON

```
3DSO(0,R)=1
3DSO(R)=1
```

which both turn digital output channel 3 of the *dataTaker* ON, and then immediately turns it OFF again.

The delay period and reset are not directly supported by DeLogger, however these can be entered into programs by using the User channel in the Program Builder (DeLogger ver 4.2.16 or later) as follows



Switching the Digital Byte Output

The *dataTaker* is also able to switch the digital output channels as byte outputs. Digital byte outputs are implemented as groups of successive digital bit output channels.

The digital byte output command can be used to switch groups of digital output channels simultaneously, producing a bit pattern.

Digital bytes are switched beginning at a specified digital output channel number, for the next 8 successive digital outputs. If there are fewer than 8 digital outputs beyond the first digital output channel (as is the case for the *dataTaker*, but not for the Channel Expansion Module), then the byte is switched only for those output channels that are present.

The digital byte output is specified in a *channel list* by the identifier

nDBO(mask)=pattern

where

n	is the first channel to begin switching the byte
DBO	is the digital byte output identifier
mask	is an optional bit mask of which bit channels are to be included in the byte, beginning from n
pattern	is the decimal value of the bit pattern required

The digital byte output channel number is defined as the first digital bit output channel to be switched. If the digital output channels are located on a Channel Expansion Module, then the module number precedes the channel number(s).

The pattern can be specified as a constant, a channel variable or an expression. If the pattern is a decimal value, then the value is rounded to the nearest integer.

The optional mask allows individual digital output channels within the group to be enabled, as specified by the bits, which are set in the mask. Channels associated with bits, which are not set in the mask, are not enabled and so are not switched.

If a byte mask is not specified, then the mask defaults to decimal 255 (11111111) and all channels from the nominated first channel are switched.

The delay and reset options for digital bit outputs are not supported for digital byte outputs.

Using DeTransfer, the digital byte output can be switched by commands for example

```
1DBO=13
```

which turns digital output channels 1, 3 and 4 of the *dataTaker* ON, and turns digital output channel 2 OFF. Decimal 13 has the bit pattern of 1011.

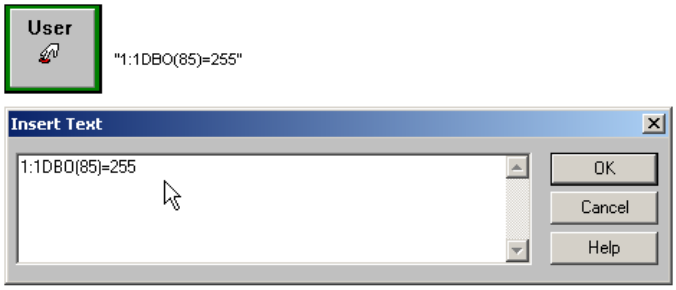
```
5CV=11
2DBO=5CV
```

which turns digital output channels 2, 3 and 5 of the *dataTaker* ON, and turns digital output channel 4 OFF. Digital output channel 1 is not addressed.

```
1:1DBO(85)=255
```

which turns the digital output channels 1, 3, 5 and 7 of the Channel Expansion Module ON. Digital output channels 2, 4, 6 and 8 are not enabled by the mask, and so are not switched.

The digital byte output channels are not directly supported by DeLogger, however can be entered into programs by using the User channel in the Program Builder (DeLogger ver 4.2.16 or later) as follows



Switching Digital Outputs in Schedules

The digital bit and byte outputs can be made to change state at the start of a program, and at intervals during execution of a program, by placing the commands in Immediate and Triggered Schedules.

The delay period and reset operate in the same manner as for direct digital output commands. During the delay period all analog to digital conversion is suspended.

Switching of digital outputs in this manner allows them to be used to generate pulse trains, control relays which switch power to sensors only when the sensors are to be read, etc.

Digital bit or byte output channels are programmed to be switched in Schedules simply by including the digital output commands in the schedule lists.

Using DeTransfer, digital output channels can be pulsed by the commands for example

```
BEGIN
 RA10M
  1DSO(5000)=1  1DSO=0
END
```

which will turn digital output channel 1 ON for 5 seconds, then turns it OFF, every 10 minutes.

The switching can be completed either by turning the digital output channel OFF again with no delay, or by specifying the reset option as follows

```
BEGIN
 RA10M
  1DSO(5000,R)=1
END
```

The Schedule Triggered by Poll can be used to switch digital output channels from the host computer. For example the command from DeTransfer

```
BEGIN
 RX
  3DSO(5500,R)=1
END
```

will turn digital output channel 3 ON for 5.5 seconds then OFF again whenever a Poll Trigger ***X*** is received from the host.

Applications

Switching digital output channels in Schedules has a variety of uses, as the following examples illustrate

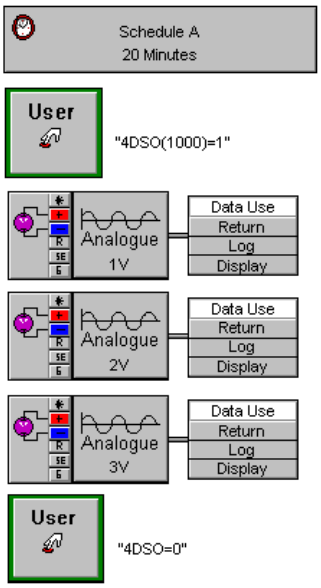
```
BEGIN
 RA2S
  1DS0(1000,R)=1
END
```

produces a pulse train from digital output channel 1, that is ON for 1 second, and OFF for 1 second.

```
BEGIN
 RA20M
  4DSO(1000)=1
  1..3V
  4DSO=0
END
```

digital output channel 4 is controlling a relay, which is switching power to a group of sensors. Every 20 minutes the sensors are powered up, the system waits 1 second while the sensors settle, the sensors are read, and the sensor power supply is turned off again.

These actions can also be programmed from DeLogger as follows



Counting Digital Output Switching

Changes of state of digital output channels in response to switching commands can be counted by the low speed counters associated with each of the digital channels. Each low speed counter is incremented each time the respective output is switched ON.

WARN Channels

The *dataTaker 600* series loggers and the Panel Mount Display have a special form of digital output channels called the warning channels. These warning channels control the 3 LEDs on the display, the display backlight, and the beeper in the display.

There are 6 WARN channels, which are allocated as follows

1WARN	Display panel LED 1
2WARN	Display panel LED 2
3WARN	Display panel LED 3
4WARN	Beeper
5WARN	Flash display backlight at approx 1Hz
6WARN	Turn display backlight on and off

Switching the WARN Channels

The WARN channels are switched ON by assigning any positive or negative value to the channel, and are switched OFF by assigning a 0 to the channel.

An optional delay period can be invoked after switching the WARN channels, which pauses program execution. The delay can be from 1 to 65535 mS.

An optional reset can be specified when switching WARN channels, to reset the WARN channel to the opposite state to that selected. If a delay period is specified, then the WARN channel is reset after the delay.

The WARN channels are switched by commands in the general formats

nWARN(delay,R)=state

and

n..mWARN(delay,R)=state

where

n	is a single WARN channel
n..m	is a sequence of WARN channels
WARN	is the WARN channel identifier
delay	is an optional delay period in millisecs after switching,
R	optional reset to opposite state after switching
state	indicates the state to which the WARN channel is set

The optional delay period is specified as an integer within the range of 1 to 65535 mS (a little more than 1 minute). If no delay period is specified, the WARN channel is switched and program execution continues immediately.

If the reset option is not specified, then the WARN channel remains in the state to which it is switched.

The state can be specified as a constant, a channel variable or an expression. If the value for state is a decimal value, then the value is rounded to the nearest integer.

A state value of 0 will turn the WARN channel OFF, while any value other than 0 will turn the WARN channel ON.

Using DeTransfer, the WARN channels can be switched by commands for example

```
1WARN=1
2..3WARN=5CV/3
4WARN(2000,R)=1
5WARN=1
```

which

turns LED 1 of the display panel ON

turns LED 2 and LED 3 of the display panel ON (assuming that the expression does not evaluate to zero)

turns the beeper ON for 2 seconds, then turns it OFF

flashes the display backlight

Applications

The WARN channels are used to visibly or audibly annunciate alarm commands, internal system states, program states, etc.

The following examples using DeTransfer illustrate some uses for the WARN channels

```
ALARM3(2TK>110)1WARN
```

turns LED 1 of the display panel ON when the temperature is greater than 110°C, otherwise holds the LED OFF (See Section III – Alarms).

```
BEGIN
 RA1S
  3WARN(500,R)=1
END
```

flashes LED 3 of the display panel at approximately 1Hz

```
BEGIN
  R5M
    1V(=1CV)
    4WARN=(1CV<1000)
END
```

turns the beeper on if the measured voltage signal falls below 1 Volt.

The WARN channels are not directly supported in the Program Builder of DeLogger, however these channels can be included in programs by using the User channel (requires DeLogger Ver 4.2.16 or later).

The Real Time Clock

The *dataTaker* real time clock keeps the Day Number or Date in three formats, and the Time of day in three formats. The real time clock is used for triggering Schedules, for date and time stamping data acquisition, Alarms, System Timers, and for other system timing functions.

The formats for date and time described below only apply if the Free Format data mode selected (/h).

Day Formats

The day can be returned as the Day Number or Date in one of three formats. The format is specified by the Parameter31 command as follows

P31=0 selects the Day Number format, for example	`Day 5316`
P31=1 selects the DD/MM/YYYY format, for example	`Date 23/07/2003`

This is the default date format for 50 Hz line frequency operation, as determined by the line frequency setting of the DIP switch.

P31=2 selects the MM/DD/YYYY format, for example	`Date 07/23/2003`

This is the default date format for 60 Hz line frequency operation, as determined by the line frequency setting of the DIP switch.

Time Formats

The Time of day can be returned in one of three formats, which is specified by the Parameter39 command as follows

P39=0 (default) selects the hh:mm:ss format, for example	`Time 15:30:45`

The character used to separate the hours, minutes and seconds can be defined as any ASCII character by the Parameter40 command. The default is ASCII 58 (:)

P39=1 selects the seconds from midnight format, for example	`Time 23465 Secs`
P39=2 selects the decimal hours format, for example	`Time 22.38522 Hours`

Decimal Day

The *dataTaker* also has a Decimal Day format for time, which is a decimal value for which the whole number is the day number, and the fraction is the time of day expressed as a fraction of a day.

The Decimal Day is returned by reading the System Variable 12 by the command

```
12SV
```

which returns the decimal day as follows

```
12SV  1269.9
```

If the global maximum number of significant digits is increased to say 7 using the Parameter32=7 command, then greater resolution for the fraction of the day can be obtained as follows

```
12SV  1269.953
```

Reading 12SV(FF0) returns the day number only.

Setting the Day or Date

The Day Number or Date is set in the format which is defined by the current setting of the Parameter31 command as follows

If P31=0 then the Day Number is set by the command

D=ddddd

If P31=1 then the Date is set by the command

D=dd/mm/yyyy

If P31=2 then the Date is set by the command

D=mm/dd/yyyy

where

ddddd	is the day number in the range 0 – 49,700
dd	is the day of the month in the range 1 – 31
mm	is the month in the range 1 – 12
yyyy	is the year in the range 1989 – 2060

In the *dataTaker* the Date is referenced from 1/1/1989, which is equivalent to Day Number 0.

Using DeTransfer, the day number or date are set in the different formats by the commands for example

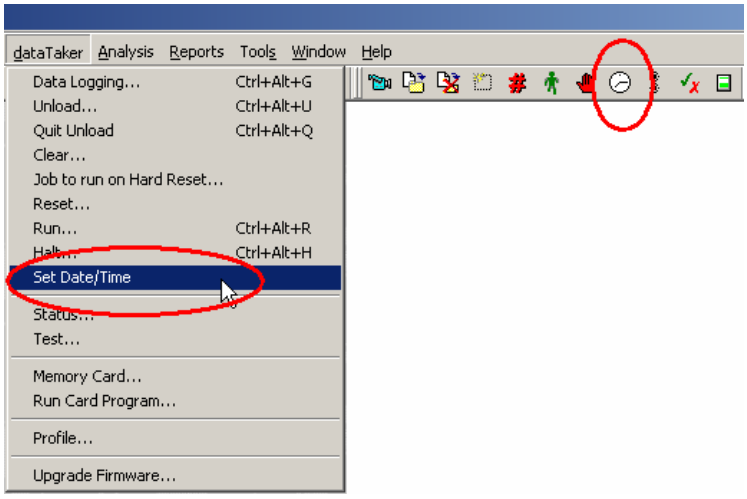
```
D=5316
D=23/7/2003
D=7/23/2003
```

DeTransfer also has a special command for setting the date of the *dataTaker*

```
D=\d
```

which sets the date according to the clock of the host computer. Note : The Date format for the *dataTaker* must be the same as the Date format set in the Regional Options applet of the Windows Control Panel.

There is no requirement to set the date format in DeLogger – DeLogger handles this automatically. The *dataTaker* date is set by selecting Set Date/Time on the *dataTaker* menu, or by clicking the Set Date and Time toolbar button.



Setting the Time

The Time is set in the format which is defined by the current setting of the Parameter39 and Parameter40 commands as follows

If Parameter39=0 then Time is set in the hh mm ss format, using the separator defined by Parameter40

T=hh:mm:ss

There are a number of variations to this time set command as follows

T=hh	sets clock to hh:00:00
T=:mm	sets clock to 00:mm:00
T=::ss	sets clock to 00:00:ss
T=hh::ss	sets clock to hh:00:ss

where

hh	is the hours
mm	is the minutes
ss	is the seconds
:	is the separator, declared by Parameter40

Using DeTransfer, the Time is set in hh:mm:ss format by the commands for example

```
T=15:45:30
T=10
T=::30
```

DeTransfer also has a special command for setting the time of the *dataTaker*

```
T=\t
```

which sets the time according to the clock of the host computer. Note : The Time format for the *dataTaker* must be set to HH:MM:SS to match the Windows time format.

If Parameter39=1 then Time must be set in seconds from midnight by the command

T=sssss

where

sssss	is the number of seconds, 0 – 86,400

Using DeTransfer, the Time is set in seconds format by the commands for example

```
T=0
T=46200
```

If Parameter39=2 then Time must be set in decimal hours by the command

T=hh.hhhhh

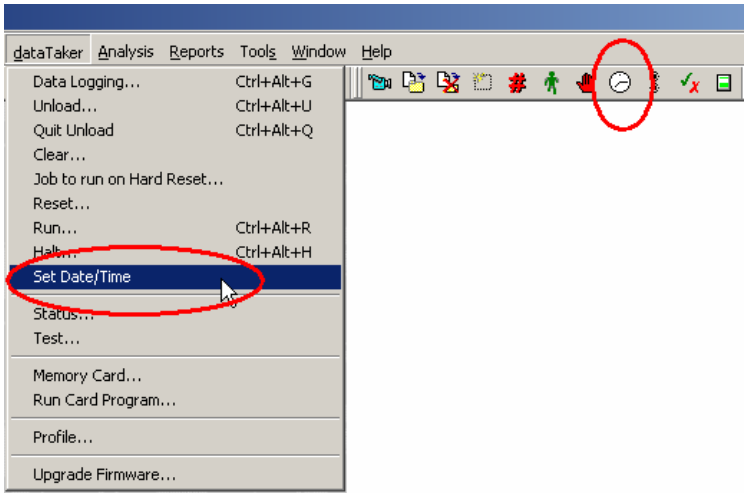
where

hh.hhhhh	is the time in decimal hours, 0.000277 hours = 1 second

Using DeTransfer, the Time is set in decimal hours format by the commands for example

```
T=0.0
T=12.50833
```

There is no requirement to set the time format in DeLogger – DeLogger handles this automatically. The *dataTaker* time is set by selecting Set Date/Time on the *dataTaker* menu, or by clicking the Set Date and Time toolbar button.



Reading The Real Time Clock

The Day Number or Date and the Time 'channels' of the real time clock are treated in the same manner as other input channels. The date and the time of scanning input channels can be returned with the scan data by including the Day or Date and the Time channels in the schedule lists.

The Day or Date and the Time are specified by the identifiers

Day Number or Date

D

Time

T

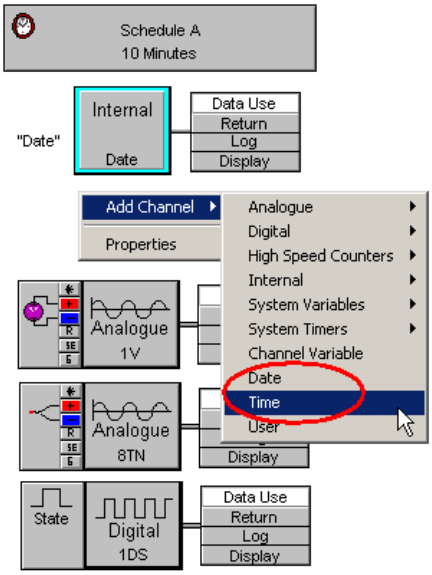
Using DeTransfer, the date and time of scanning input channels are included with the input channel data by the commands for example

```
BEGIN
 RA10M
  D T
  1..6V
  8..10TN
  1..2DS
  3HSC(R)
END
```

which returns the Day Number or Date and Time at which each scan with the input channel data.

The date and time information is actually formatted by the *dataTaker* as these are transmitted to the host computer. Therefore if you are using DeTransfer, the format of date and time in logged data can be changed after data collection, before the logged data is returned to the computer.

Using DeLogger, the date and time of scanning input channels is included in the returned data by including the Date and Time channels in the program in the Program Builder as follows



Automatic Date and Time Recording

The *dataTaker* automatically records the date and time that Schedules are run to scan input channels, perform calculations, etc, even if the original Schedule does not include the D and T channels.

The return of this automatically recorded date with the logged or real time data is determined by the Date Switch command as follows

/D Enable return of automatically recorded date

/d Disable return of automatically recorded date (Default)

The Date Switch defaults to /d when the *dataTaker* is powered up, hardware reset or executes a RESET command.

The return of this automatically recorded time with the logged or real time data is determined by the Time Switch command as follows

/T Enable return of automatically recorded time

/t Disable return of automatically recorded time (Default)

The Time Switch defaults to /t when the *dataTaker* is powered up, hardware reset or executes a RESET command.

When the Date Switch and the Time Switch are enabled, then the date and time of data acquisition is returned at the start of the data block for each Schedule in the program. If date and time channels are included explicitly in the schedule lists, and the Date and Time Switches disabled (/d /t), then the date and time are returned in the position that they were placed in the schedule list relative to other channels.

DeLogger does not use the Date and Time Switches – DeLogger uses the Fixed Format mode for all data, which includes date and time stamps.

Assigning Date and Time to Channel Variables

The date and time can be assigned to Channel Variables whenever the real time clock is read in Schedules, and used in calculations, as test inputs and setpoints for alarms, etc.

The Day Number, Date and Time are assigned to Channel Variables by specifying the Channel Variable as a channel option, in the same manner as assigning data from input channels to Channel Variables (see Section III – Channel Variables and Calculations).

Using DeTransfer, assigning the date and time to Channel Variables is illustrated by the following examples

```
D(=1CV)
T(=2CV)
```

Irrespective of the current formats defined for date and time, the assignment of these to channel variables is fixed as follows

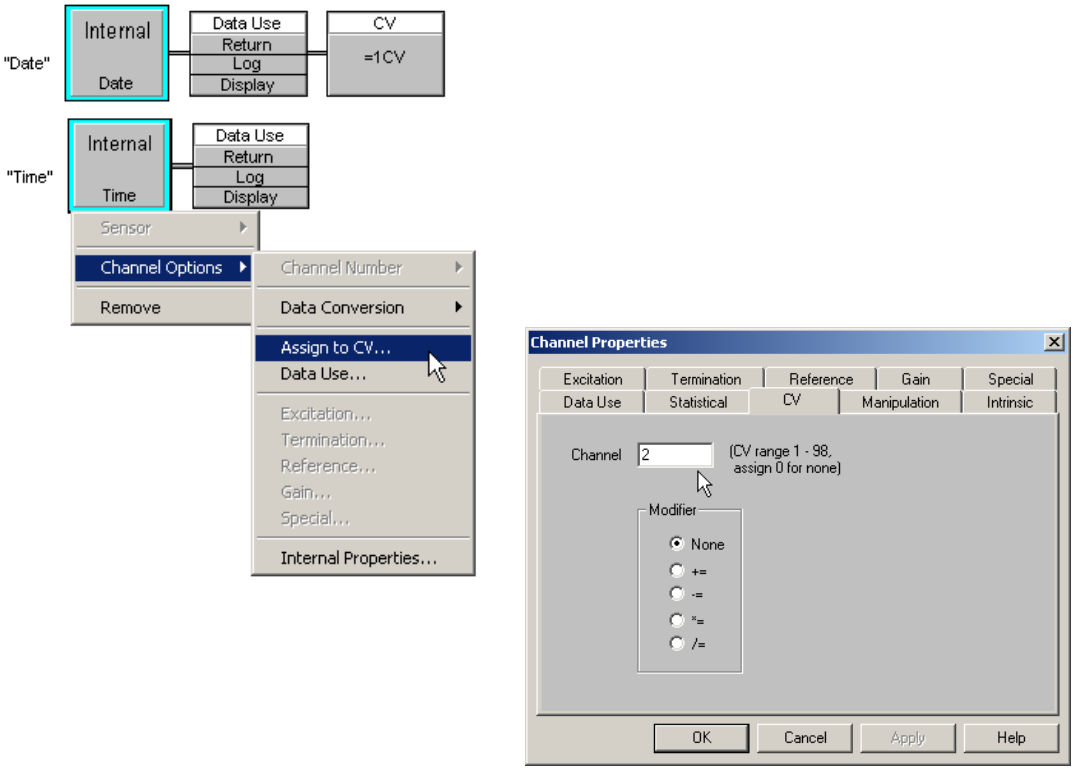
date is assigned as the number of days since 01/01/1989, which is day number 0

time is assigned as the number of seconds since last midnight

When these are read and assigned to a Channel Variable, the date and/or time is also returned to the host in the current format. This can be disabled by including the W or NR channel options in the command as follows

```
D(=1CV,W)
T(=2CV,NR)
```

Using DeLogger, the date and time can be assigned to Channel Variables in the Program Builder as follows



Date and Time in Alarms

The date and time can also be used in Alarms (see Section III – Alarms) to test for the arrival of a particular date or particular time. The date or time 'channel' are used as an input channel for an alarm, and are tested against setpoints.

The date setpoints must be entered in the format currently defined by Parameter31, and the time setpoints must be entered in the format currently defined by Parameter39 and Parameter40.

Using DeTransfer, alarms which test for a date or time are entered by the commands for example

```
ALARM1(D>28/07/2003)"Tomorrow becomes today"

ALARM2(T><10:30:00,16:30:00)"Time window open"
```

In the first example the alarm will become true at midnight, and remain true for the duration of the date.

In the second example the alarm will become true at 10:30:00, and remain true until16:30:00, each day.

The Internal Channels

The *dataTaker* has a number of internal channels that are used to extend function of the logger. The internal channels include System Timers, System Variables, a Text channel, and calibration channels.

System Timers

The *dataTaker* has four System Timers that can be used for a wide variety of applications. The System Timers are linked to the real time clock as follows

System Timer 1 increments every second, and has a default range of 60 seconds. This system timer is identified as channel 1ST, and provides the second of the minute.

System Timer 2 increments every minute, and has a default range of 60 minutes. This system timer is identified as channel 2ST, and provides the minute of the hour.

System Timer 3 increments every hour, and has a default range of 24 hours. This system timer is identified as channel 3ST, and provides the hour of the day.

System Timer 4 increments every day (at midnight), and has a default range of 7 days. This system timer is identified as channel 4ST, and provides the day of the week as follows

0 = Sunday
1 = Monday
2 = Tuesday
3 = Wednesday
4 = Thursday
5 = Friday
6 = Saturday

The System Timers each count up through their respective ranges, and automatically reset to zero when their respective terminal count is reached. The System Timers increment as follows

System Timer 1 increments on the second of the real time clock

System Timer 2 increments on the minute of the real time clock

System Timer 3 increments on the hour of the real time clock

System Timer 4 increments at midnight of the real time clock

The System Timers are synchronised to the real time clock and will re-synchronise each time the real time clock is changed.

Using the System Timers

The System Timers can be included in schedule lists to return elapsed time, and can be used in Alarms to measure off periods of time.

The System Timers are specified in schedule lists and Alarms by the general format

nST(R)

where

n	is the System Timer number
ST	is the System Timer type identifier
R	is an optional timer reset when read

The reset channel option resets the System Timer to zero when read. This allows the elapsed time between successive readings to be returned, and can be used for example to measure the elapsed time between digital events, etc.

Using DeTransfer, the System Timers are used by the commands for example

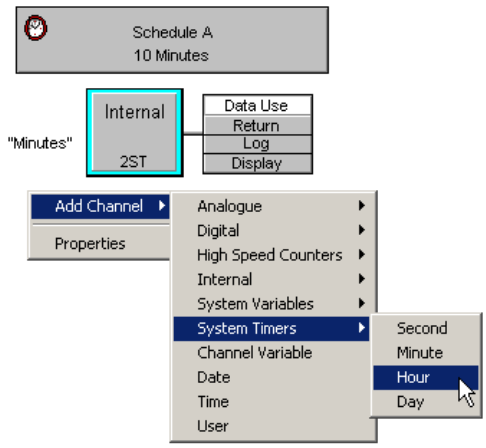
```
BEGIN
 RA10M
  2ST
END
```

which returns the value of System Timer 2 every 10 minutes.

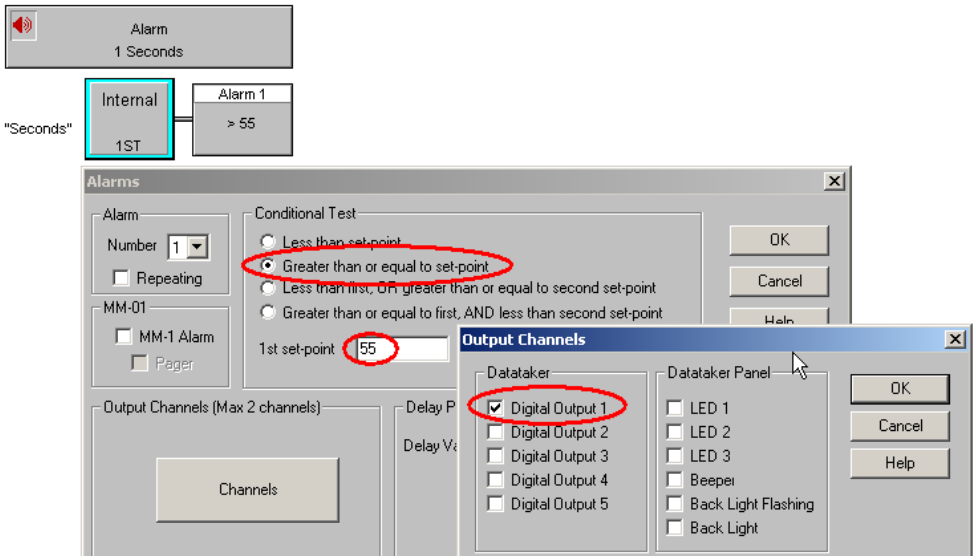
```
BEGIN
 RZ1S
  ALARM1(1ST>55)1DSO
END
```

will turn digital output channel 1 ON for the last 5 seconds of every minute (See Section III – Alarms for details of how the Alarm command works).

Using DeLogger, the System Timers can be included in programs in the Program Builder as follows



and can be included in Alarms as follows



Setting the Range of System Timers

The range of the system timers can be changed from the defaults to suit the particular needs of applications.

The system timer ranges are specified by including the new range as a channel option as follows

nST(range)

where

n	is the System Timer number
ST	is the System Timer type identifier
range	is a value in the range of 0 to 65535

If the value of the range specified is greater than 65535, then the range is set to 65535. If the value for the range is a decimal value, then this is truncated to an integer.

Using DeTransfer, the command for example

```
2ST(120)
```

programs System Timer 2 is to count through the range of 0 to 119 minutes, then reset to zero.

DeLogger does not directly support setting of the System Timer ranges, however this can be done via the general purpose User channel in the Program Builder (requires DeLogger ver 4.2.16 or later).

Initialising the System Timers

The System Timers begin counting from zero when the *dataTaker* is initially powered up or RESET. However the System Timers can be set to an initial count as follows

nST=count

where

n	is the System Timer number
ST	is the System Timer type identifier
count	is the initial count

The initial count can a constant, a channel variable or an expression. If the value of the initial count is a decimal value, then the value is rounded to the nearest integer.

However if the value of the initial count is greater than 65535, then an “E15-assignment error” is returned and the System Timer is initialised to 99999.9 which is the universal *dataTaker* error value.

The System Timer begins counting from the initial count and 'rolls over' to zero when the range count is reached.

Using DeTransfer, System Timer initial count can be set by the commands for example

```
1ST=180
2ST(1000)=125
3ST=10CV-12CV
```

Whenever the initial count of the System Timer is set, the System Timer is read and the new count is returned. The return of the new count can be disabled by including the W or NR channel options in the counter as follows

```
1ST(W)=180
2ST(1000,NR)=125
3ST(W)=10CV-12CV
```

Commands to set the initial count of System Timers should not be included in Triggered Schedules, since this will cause the System Timer to be set to the initial count each time the Schedule runs.

Note : After setting the initial count, the first increment of System Timers 2, 3 and 4, will occur at the next increment for the respective unit of time of the real time clock .

Therefore the first increment of the System Timer may occur after a period which is less than the full period for the respective unit of time.

System Timers are synchronised to the real time clock and will re-synchronise each time the real time clock is changed.

DeLogger does not directly support setting of the System Timer initial values, however this can be done via the general purpose User channel in the Program Builder (requires DeLogger ver 4.2.16 or later).

Applications for System Timers

The System Timers can be used for a number of ways, including to return the elapsed time between asynchronous events, can be used in Alarms to measure off periods of time, etc.

Using DeTransfer, the program for example

```
BEGIN
 RA1+E
  2ST(65535,R)
END
```

will return the elapsed time in minutes between events received on digital input channel 1. A large range is chosen, and the System Timer is reset to zero when it is read.

Also

```
T(=1CV,W)
1ST(60,W)=1CV%60
BEGIN
 RZ1S
  ALARM1(1ST>50)1DSO
END
```

System Timer 1 is synchronised with the real time clock, and counts over the range of 0 – 59 seconds. The timer is used as the input to an alarm command, which toggles a relay connected to digital output 1 ON when the count exceeds 50 seconds. When the count reaches 59 seconds the system timer automatically resets to zero, and the alarm toggles the relay OFF.

This scenario could be used for instance to turn a pump on for 10 seconds before the end of every minute.

Text Channel

The *dataTaker* has a user definable Text Channel into which a text string can be stored, and later returned with real time or logged data.

This text string could be the project name, the operators name, site details, column headings for tabulated data, initialisation strings for printers and modems, etc. The Text Channel is implemented and used in the same manner as any other input channel, and is identified as channel $.

Using DeTransfer, a text string is entered into the Text Channel by direct assignment, for example

```
$="A text string of 1 to 80 characters"
```

The text string can contain any of the ASCII character set, including control characters. The ASCII control characters are entered in the format

^H	for CTRL H	(Backspace)
^I	for CTRL I	(Tab)
^G	for CTRL G	(Bell)
^M	for CTRL M	(Carriage Return)
^J	for CTRL J	(Line Feed)
etc.		

and occupy one character position although two ASCII characters are required for their definition.

An example of control characters in a text string follows

```
$="Day^ITime^ITemp1^ITemp2^IPressure^M^J
```

This string defines a heading for tabulating data, and uses tab characters to align the columns.

All characters received between the opening and closing quotes are entered into the text string. The Text Channel may be re-defined at any time, however any previous definition is lost.

The Text Channel is treated as any other input channel, and may be included in schedule lists for return to the computer, and flagged in the data storage memory for return with logged data.

Once defined, the Text Channel is returned with channel data at regular intervals by including the text channel in a Triggered Schedule as follows

```
$="Boiler Temperatures at^M^J"
BEGIN
 RA5M
  $
  T
  1..10TK
END
```

In this case the Text Channel is returned as the first line of data resulting from each scan.

Whenever data from a Schedule that has the Text Channel in it is stored in the data memory, then redefining the text string has important implications

- whenever the text string is logged, a text flag is written into the data memory rather than the text string *per se*
- when the data memory is later unloaded to the computer, the text string currently defined in the Text Channel is returned whenever the text flag occurs in the store
- if a new text string is assigned to the Text Channel during the course of a data logging application, then the previous text string is lost and the new text string is returned for all occurrences of the text flag in the data memory

The current definition of the Text Channel is also stored into the memory card if used. The text string is available for inclusion in data which is subsequently read from the memory card, either by another *dataTaker* or the *dataTaker* MCI-04 Memory Card Interface.

If however the Text Channel is changed during the course of a data logging application, then the new string is also written to the memory card and the previous string is lost.

System Variables

The *dataTaker* has a number of special channels which provide access to various system variables. The System Variables are used in the same manner as normal input channels, except that the data returned is from internal sources rather than from input channels.

The System Variables can be included in Schedules to return system information at intervals. The System Variables can also be used in Alarms to provide warnings of internal conditions such as data memory capacity, etc.

The allocation of internal information to the System Variables is as follows

1SV	Data points free in internal data memory
2SV	Data points stored in internal data memory
3SV	Data points free in the memory card
4SV	Data points stored in the memory card
5SV	Number of times that the Statistical Sub Schedule has run
6SV	Number of remote network errors, equivalent to Parameter9 (*dataTaker 500/600* series)
7SV	ADC settling time, equivalent to Parameter10
8SV	ADC sampling period (mains frequency), equivalent to Parameter11
9SV	Flag indicating if a memory card is inserted 1 = a memory card is inserted 0 = no memory card is inserted
10SV	Returns the Schedule identity when included in the schedule list, where 0 = RX – Schedule Triggered by Poll 1 = RA – Triggered Schedule A 2 = RB – Triggered Schedule B 3 = RC – Triggered Schedule C 4 = RD – Triggered Schedule D 5 = Immediate Schedule 6 = RZ - Alarms Schedule
11SV	Always returns zero (useful as a reference channel)
12SV	Decimal day (eg. 1276.5 is midday of day 1276)
13SV	*dataTaker* address
14SV	System Attenuation Factor (default value 214.61)

The system variables are returned by the command format

nSV

where

n	is the System Variable number 1 – 14
SV	is the System Variable type identifier

Using System Variables in Schedules

When System Variables are included in Schedules, the current contents of the System Variables are returned each time that the Schedule is run.

Using DeTransfer, the programs for example

```
BEGIN
 RA10M
  1V  2R  3F
  1..2SV
END
```

returns the number of data points free in the internal memory, and the number of data points stored in the internal memory, along with other channel data every 10 minutes.

```
BEGIN
 RA15M
  D  T
  5SV
  1..5V(AVE)
END
```

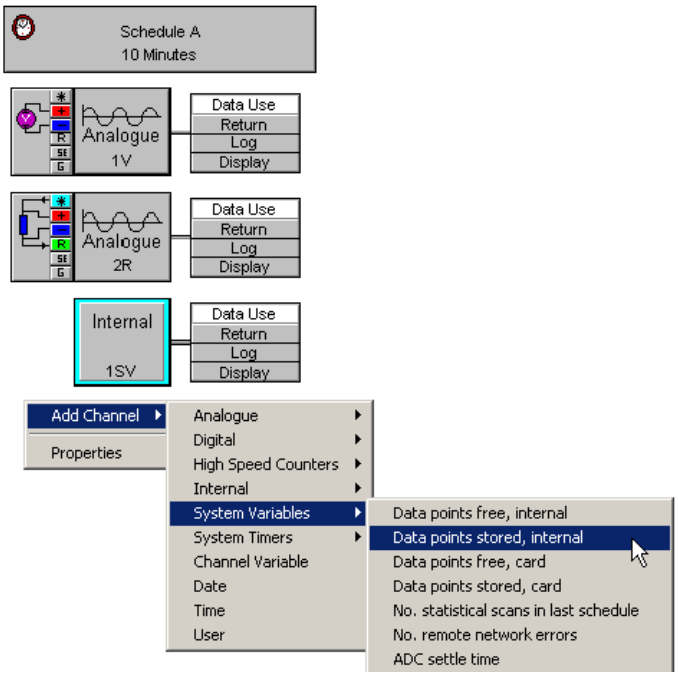
returns the number of times that the Statistical Sub Schedule has run, along with the average data every 15 minutes.

```
BEGIN
 RA5M
  10SV
  1..4V

 RB10M
  10SV
  5..8V
END
```

returns the identity of the Schedule which produced the data as the first item in each data block.

Similar programs can be created in the Program Builder of DeLogger, using the Add Channels:System Variables path to select the System Variables required.



Using System Variables in Alarms

System Variables can also be used as the input channel to Alarms. One application for this function is to indicate if the internal data memory or a memory card is becoming full, or the memory card has been removed.

Using DeTransfer, the alarm for example

```
ALARM3(4SV>340000)1WARN"[Halt]"
```

will monitor the number of data points stored in the memory card, and will Halt scanning and turn the warning LED 1 of the display panel on when more than 340,000 data points have been stored.

Changing System Variables

System Variables 6, 7 and 8 can also be assigned values, which has the effect of setting the appropriate Parameter (see table above). The legal assignments for each Parameter are detailed in Section III – Parameter Commands.

The assignment can be either as a constant, Channel Variable or an expression.

Using DeTransfer, assignments can be made for example

```
8SV=100
```

which sets the mains frequency to 100 Hz (Parameter11), thereby reducing the analog to digital conversion period to 10 mS.

Unlike the Parameter commands, assignments to System Variables 6, 7 and 8 can be included in schedule lists and so can be changed whenever Schedules run.

For example the program

```
BEGIN
 RA1H
  7SV=20
  1..4V
  7SV=5
 5..8V
END
```

changes the ADC settling time during each scan as follows

the first group of voltages are read following a 20mS delay after selecting each channel

the second group of voltages are read following a 5mS delay after selecting each channel

Internal Analog Channels

The *dataTaker* and the Channel Expansion Module have a number of internal analog channels that are used for calibration of the analog to digital converter, and for sensor support.

The internal channels in the *dataTaker* are as follows

1%LM35	– Case temperature – LM35 solid state temperature sensor – Default reference junction temperature for thermocouples connected to the *dataTaker*
2%V	– Internal zero voltage – Default zero voltage for compensation of thermocouples connected to the *dataTaker*
2%R	– Internal precision resistance – 100.0 [a], ±0.1%, grounded
0%V(M18:156,101)	– Battery voltage
0%I(M18:220,-0.22)	– Battery current

The internal channels in the Channel Expansion Modules are as follows

n:1%LM35	– Case temperature for CEM n – LM35 solid state temperature sensor – Default reference junction temperature for thermocouples connected to the CEM
n:2%V	– Internal zero voltage for CEM n – Default zero voltage for compensation of thermocouples connected to the CEM

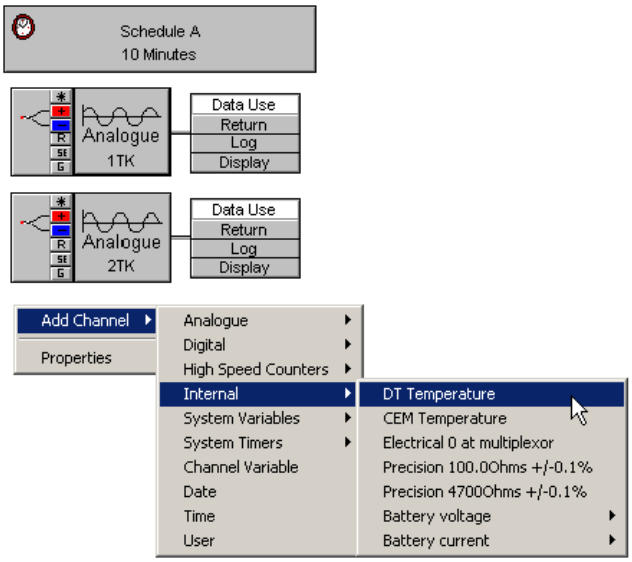
These internal analog channels can be scanned at any time by including the appropriate channels in a schedule list.

Using DeTransfer, the case temperature of the logger can be read along with other channels by the program for example

```
BEGIN
 RA5M
  1..5TK
  1%LM35
END
```

In this instance, monitoring the case temperature of the logger also monitors the reference temperature for the thermocouples being measured, which may be useful as a check.

DeLogger also supports the internal channels, so that these can be included in programs in the Program Builder.



Channel Options

Channel options allow the function of the *dataTaker* channels to be tailored in various ways including

input configuration

sensor excitation

ADC gain

statistical reporting

special calculations

variable assignment

destination of data

data format

The various channel options available are listed in the Channel Options Table overleaf, and are described in the following pages.

Channel options relate both to individual channels and to sequences of channels, and are specified in brackets following the channel type label in the general formats

ntype(options)

n..mtype(options)

module:ntype(options)

module:n..mtype(options)

where

n	is a single channel number
n..m	is a sequence of channels
module	is the module number, 0 for the *dataTaker* and 1 or 2 for successive Channel Expansion Modules
type	is the signal type identifier
options	is a list of one or more channel options

The option list may include one or more channel options. If more than one channel option is listed, then the individual channel options are separated by a comma (no spaces allowed), and may in any order.

When a sequence of channels is specified, then the channel options are applied to all channels in the sequence.

The channel options apply only to the channel specification in which they are found. When the same channel is included more than once in a schedule list or channel list, then each inclusion is treated as a separate entity and each can have different channel options.

The channel options are applied in a particular order during execution of the channel, irrespective of the order of specification in the options list.

Channel Options Table

Category	Option	Function	Mutual Exclusion	Range of Option	Applicn Order
Input termination	T	Terminates +, -- inputs with 1 MOhm to ground	]		1
	U	Un-terminates +, -- inputs	]		1
Resistance	4W	Configures channel for 4 wire measurement			1
Single ended input	X	Use SE Ref terminal as common	]		1
	2V	Use internal 2.500V Vref as common	]		1
Gain	GLn	Gain lock		1, 10, 100	1
Attenuation	A	Attenuation	]		1
	NA	No attenuation	]		1
Excitation	G	Guard signal	]		1
	V	Voltage excitation, approx 4.5V via 1KOhm	]		1
	I	Current excitation, 250 uA	]		1
	II	Current excitation, 2.500mA	]		1
	N	No excitation	]		1
Special	Esn	Extra ADC samples when reading channel		0 to 15	1
	Mx:y	Special input signal routing		0 to 255	1
Resetting	R	Reset counter, timer, channel variable, after read			2
Scaling	f.f	Channel factor		±1e18	2
	Yn	Polynomial	]	1 to 20	3
	Sn	Span	]	1 to 20	3
	Fn	Intrinsic function	]	1 to 6	3
Data manipulation	DF	Difference dX	]		4
(cannot use in alarms)	RC	Rate of change dX/dT	]		4
	RS	Rate of change X/dT	]		4
	IB	Integrate (X-dX/2)*dT	]		4
Reference channels	TR	Thermocouple reference junction temp channel	]		5
(not logged or displayed)	TZ	Thermocouple reference zero channel	]		5
	BR	Bridge excitation measurement channel	]		5
Statistical	AV	Average	]		6
(cannot use in alarms)	SD	Standard deviation	]		6
	MX	Maximum	]		6
	MN	Minimum	]		6
	TMX	Time of maximum	]		6
	TMN	Time of minimum	]		6
	DMX	Date of maximum	]		6
	DMN	Date of minimum	]		6
	INT	Integral	]		6
	Hx:y:n..mCV	Histogram, x=lower limit, y=upper limit	]	x, y ±1e18	6
Variables	=nCV	Assign data to variable	]	1 to 100	7
	+=nCV	Add data to variable	]	1 to 100	7
	-=nCV	Subtract data from variable	]	1 to 100	7
	*=nCV	Multiply variable by data	]	1 to 100	7
	/=nCV	Divide variable by data	]	1 to 100	7
Data formatting	FFn	Fixed point with n decimal places	]	0 to 6	8
	FEn	Exponential with n significant digits	]	0 to 6	8
	FMn	Mixed FF and FE with n decimal places	]	0 to 6	8
	“text”	Channel label		ascii text	8
Bar graph	BGx:y	Bar graph		±1e18	8
Data destination	NR	No return			8
	NL	No log			8
	ND	No display			8
	W	Working channel			8

Using DeTransfer, channel options are specified in a bracketed list following the channel(s) these apply to as follows

```
1V(V)
```

where a single channel option which specifies that a 4.5 VDC excitation is to be output from the excite terminal of channel 1 before reading the input signal.

And

```
5PT385(4W,200.0,"Steam Temp.",FF0)
```

where a number of channel options for an RTD temperature sensor that is connected to the *dataTaker* for measurement by the 4 wire resistance measurement mode (4W). The temperature sensor has a 200 Ohm resistance at 0^{o} C. The channel data is to be labelled "Steam Temp." for output, and the data is returned with a resolution of 1^{o}C (FF0).

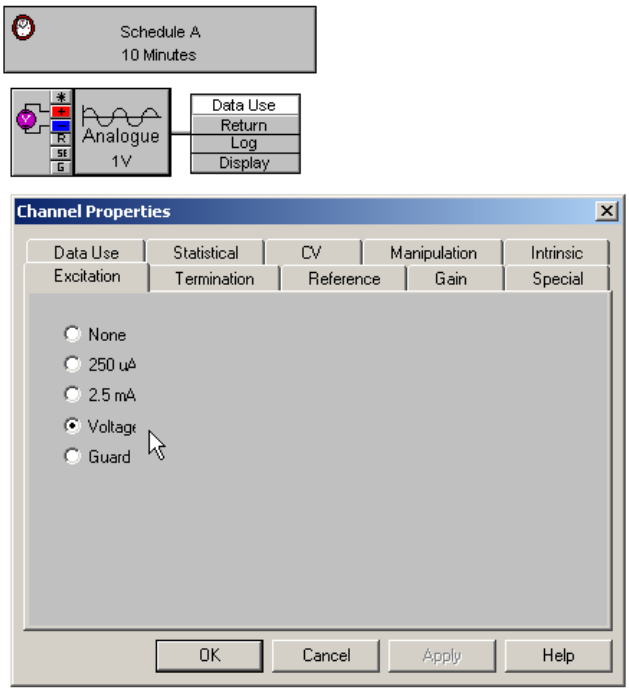
The *dataTaker* will return data as

```
Steam Temp.  266 Deg C
```

instead of the default

```
5PT385  265.7 Deg C
```

DeLogger supports many of the channel options from the Channel Properties dialog, where the options are grouped under various tabs. The Channel Properties dialog is opened by right clicking on a selected channel, and selecting Channel Options... which then displays a list of the key option groups. Select the required option group to open the Channel Properties dialog.



Mutually Exclusive Options

Most of the channel options fall into separate function categories, and the individual channel options within each of these categories are mutually exclusive. These categories and the ranges of mutual exclusivity are indicated in the Channel Options Table.

Only one of the channel options in each category can be specified in a channel options list. If more than one mutually exclusive channel option is defined, then the last defined option is applied.

Order of Application

There is a particular order of application for channel options, and is indicated in the Channel Options Table. The *dataTaker* applies channel options in this order, irrespective of the order in which they are included in the options list.

The order ensures for example that the sensor is excited before a polynomial is applied, and that the polynomial is applied before the data is averaged, etc.

The Channel Options Grouping

The different channel options fall into a number of categories of related functions. The channel options and their function are described in the following sections.

Options for Input Signal Configuration and Mode of Operation

Several options are available for the input configuration, and the mode of operation of the analog input channels. The input configurations and modes of operation options are as follows

Input Terminated

T – the analog input is terminated to *dataTaker* ground via a 1 Mª resistance during measurement, providing an input bias current path.

Differential inputs of voltage, current, frequency, thermocouples and analog state are terminated by default. Single ended inputs for these signals are not terminated by default.

Input Unterminated

U – the analog input is disconnected or unterminated from the 1 Mª terminating resistor during measurement. The input impedance is > 100Mª when inputs are unterminated. The signal source must provide an input bias current of approx 5nA.

Use of this option is appropriate to un-terminating input types for which termination is the default configuration, such as for differential voltage, current, frequency, thermocouples, analog state, etc.

4 Wire Measurement

4W – selects the four wire method for measurement of resistance, RTD, bridge, and thermistor inputs. This configuration provides full compensation for resistance of cable wires between the sensor and the *dataTaker*.

The default resistance measurement is three wire.

External Common

X – the SE REF terminal is to be used as the 'common' terminal for measurement of externally referenced single ended inputs. The active side of the single ended signals are connected to the +ve, –ve or ✴ terminals of the analog input channels, and the non active sides are commoned and connected to SE REF.

The SE REF terminal is a high impedance input (>100 MOhm), similar to the analog input channels.

The default is to measure single ended inputs with reference to *dataTaker* ground.

2.5 Volt Common

2V – an internal 2.5 Volt reference is to be used as the 'common' for various signal types, rather than an applied common or *dataTaker* ground.

This can extend the single ended input voltage range of the *dataTaker* to the common mode voltage limits (0 - 4.5VDC). However the measurement is sensitive to ground loop noise, and should not be used when precision is required.

Examples

Using DeTransfer, the following channel options for example

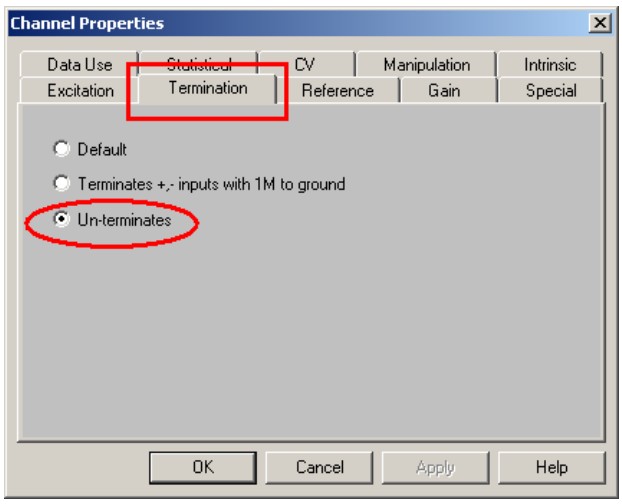
```
2+V(X,U)
```

commands the *dataTaker* to measure voltage between the positive terminal of analog channel 2 and the SE REF terminal, with the internal 1 Mª terminating resistor unterminated.

```
2R(4W)
```

commands the *dataTaker* to measure the resistance connected to analog input channel 2 by the 4 wire measurement method.

Using DeLogger, the Unterminate channel option for example is set from the Channel Properties dialog in the Program Builder as follows



Options for ADC Gain

The Analog to Digital Converter of the *dataTaker* automatically selects gain to suit the level of the measured signal. However this can be disabled and a constant gain set for individual channels. This is often used to provide constant resolution of data as the signal moves between two or three ranges of the logger.

The *dataTaker 505,605,515* and *615* also have an internal attenuator which can optionally be selected. This attenuator is common to all analog input channels, and has an attenuation ratio of 214.6 : 1.

Gain Lock

GLn – Inhibits auto-ranging and presets or locks the amplifier gain to n, where n is 1, 10 or 100.

Internal Attenuation

A – Selects the internal attenuators in the *dataTaker 505,605,515* and *615*. The high voltage input type has the attenuators selected by default.

NA – Deselect the internal attenuators in the *dataTaker 505,605,515* and *615* where their selection is a default channel option.

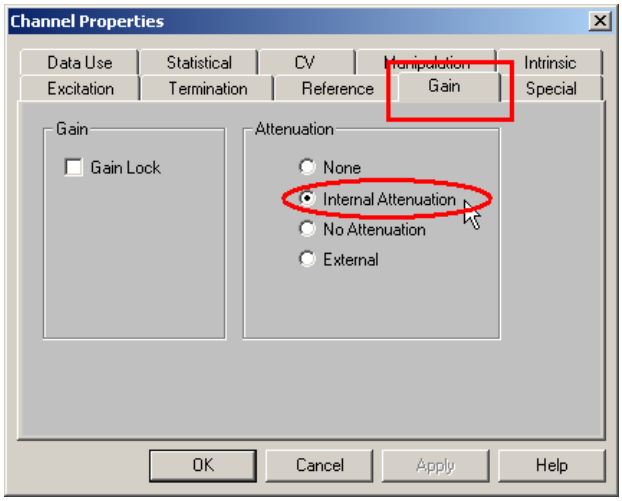
Examples

Using DeTransfer, the following channel options for example

```
1:1V(A)
1I(100,A)
```

selects the internal attenuators for measurement of the voltage on the Channel Expansion Module 1 channel 1.

Using DeLogger, the Internal Attenuation channel option for example is set from the Channel Properties dialog in the Program Builder as follows



Options for Excite Terminal Function

The Excite terminal of the analog input channels can provide sensor excitation voltages, currents, and guard voltage outputs.

The excitation conditions are established 10mS before the channel is sampled. This time is the channel settling time, can be varied by 7SV and Parameter10.

The Excite terminal options are mutually exclusive, and if more than one is specified then the last option is implemented.

Guard Output

G – Provide a guard output voltage from the Excite terminal when the channel is selected for reading. The guard voltage is equal to the input common mode voltage via approximately 3 KªOhm, and is used when the signal source has high output impedance where cable leakage is a problem.

Voltage Output

V – Output approximately 4.5 Volt via 1 KOhmª from the Excite terminal when the channel is selected for reading. This can be used to power sensors, and is applied by default for the monolithic temperature sensors and ratiometric bridge circuits.

Note: This voltage is not regulated and may drift with temperature. The 5 Volt sw output terminal can be used as a regulated 5 volt supply.

250.00 µA Current Output

I – Output a precision 250.00 µA excitation current from the Excite terminal when the channel is selected for reading. This can be used to excite resistive type sensors, and is applied by default for resistance, nickel RTDs, and thermistors.

The current source is very stable over a wide temperature range.

2.500 mA Current Output

II – Output a precision 2.500 mA excitation current from the Excite terminal when the channel is selected for reading. This can be used to excite resistive type sensors, and is applied by default for platinum and copper RTDs, strain gauges and bridge inputs.

The current source is very stable over a wide temperature range.

No Output

N - Switches the Excite terminal off if the default channel option for the channel type includes excitation.

When the Excite terminal is not used for sensor excitation, it can be used as a single ended input which can be referenced to the Analog Return or SE REF input. This single ended input option is not supported in the *dataTaker* 50.

Examples

Using DeTransfer, the channel options for example

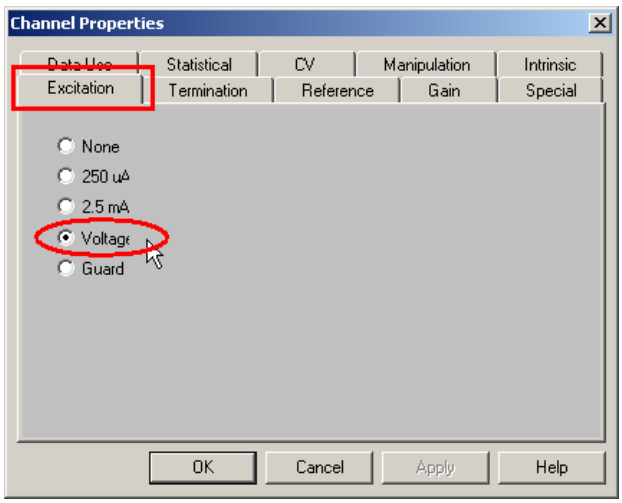
```
5V(V)
```

commands the *dataTaker* to measure a differential voltage on analog input channel 5, from a sensor which is powered by 4.5 Volt output from the Excite terminal during sampling.

```
2V(II)
```

commands the *dataTaker* to measure the voltage drop across a resistive element, excited by a 2.500 mA current output from the Excite terminal during sampling.

Using DeLogger, the Excite terminal channel options for example are set from the Channel Properties dialog in the Program Builder as follows



Special Channel Options

Several special channel options are available to facilitate sampling of particular analog signals.

Extra Samples

ESn – The Extra Sample channel option allows an analog channel to be sampled multiple times, and the result averaged and returned. The n specifies the number of extra samples, in the range of 1 – 15. The extra samples are taken successively, and do not involve any additional setup time.

This channel option is useful for reducing the effects of noise in the signal, and for increasing resolution.

Vibrating wire frequency measurement by the *dataTaker 515* and *615* is the result of 10 successive samples by default (ES defaults to 9).

Signal Routing

Mx:y – This channel option allows for special input signal routing, and is primarily used during development for special signals, but can also be used for measuring the internal battery voltage and battery current as follows

`0V(M18:156,101.0)`	returns the battery voltage
`0I(M18:220,-0.22)`	returns the battery current

A positive battery current indicates charging, and a negative battery current indicates discharging.

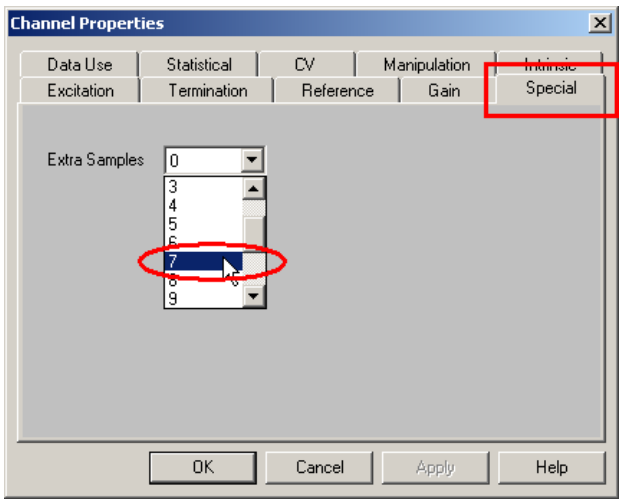
Examples

Using DeTransfer, the channel option for example

```
5FW(ES7)
```

commands the *dataTaker* to sample a frequency of a vibrating wire strain gauge on analog input channel 5 seven times and return the average. The default is to sample nine times.

Using DeLogger, the Extra Samples channel option for example is set from the Channel Properties dialog in the Program Builder as follows



Options for Resetting to Zero

R – The reset channel option is used for resetting counters, digital outputs, warning outputs, System Timers and Channel Variables after execution or scanning. The reset can also be used to pulse digital and warning outputs.

Examples

Using DeTransfer, the following channel option for example

```
1C(R) 2DSO(1000,R)=1
```

commands the *dataTaker* to reset counter 1 to zero after reading, and to set digital output channel 2 high for 1000 mS then reset the output to low.

Options for Scaling of Channel Data

The channel option list can also specifying scaling and linearizing of the raw data sampled from the input and counter channels. Scaling and linearizing of the raw data may be specified as follows

Channel Factor or Scale Factor

f.f – a single channel factor may be declared, which is used to scale the sampled data. The channel factor is a decimal value, and can be declared in floating point or exponential format.

The channel factor has a particular purpose for different signal types, such as an attenuation factor for voltage signals, magnitude of the current shunt for current signals, the resistance at 0.0 Deg C for RTDs, the threshold for analog state, etc.

For channel input types where the channel factor has no defined purpose, then the channel factor can be used for direct scaling.

The polynomial, span and intrinsic function scaling options are all mutually exclusive, and if more than one is specified then the last option specified is implemented. The channel factor however is not mutually exclusive to these, and can be used in conjunction with a polynomial, span or intrinsic function, to linearise data.

Polynomials

Yn – a polynomial can be specified, which is used to convert the sampled data according to linear or curvilinear calibrations. The polynomial must be declared prior to use (See Section III – Scaling Data - Polynomials, Spans and Functions)

Spans

Sn – a span can be specified, which converts the sampled data according to linear calibrations. The span must be declared prior to use (See Section III – Scaling Data - Polynomials, Spans and Functions)

Intrinsic Functions

Fn – a predefined function can be specified, which scales the sampled data. (See Section III – Scaling Data - Polynomials, Spans and Functions)

Examples

Using DeTransfer, the following channel options for example

```
1V(5.5)
```

commands the *dataTaker* to measure the differential voltage signal connected to analog channel 1, and multiply this by 5.5. This is appropriate to scaling an externally attenuated voltage.

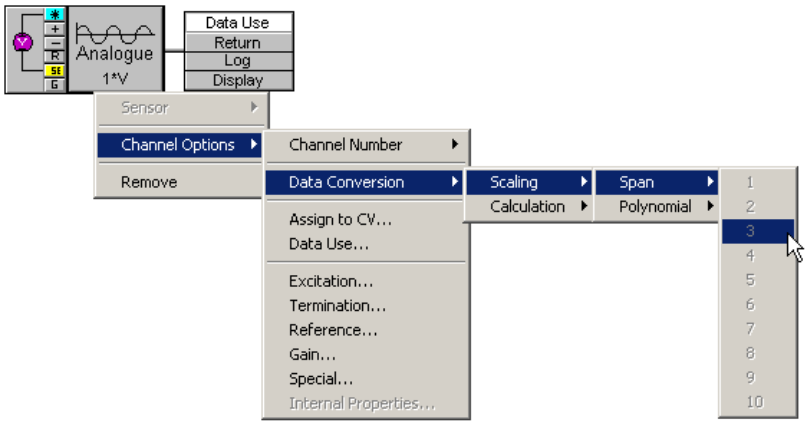
```
3V(Y2)
```

commands the *dataTaker* to measure the differential voltage signal connected to analog channel 3, and to linearise this by the defined polynomial Y2.

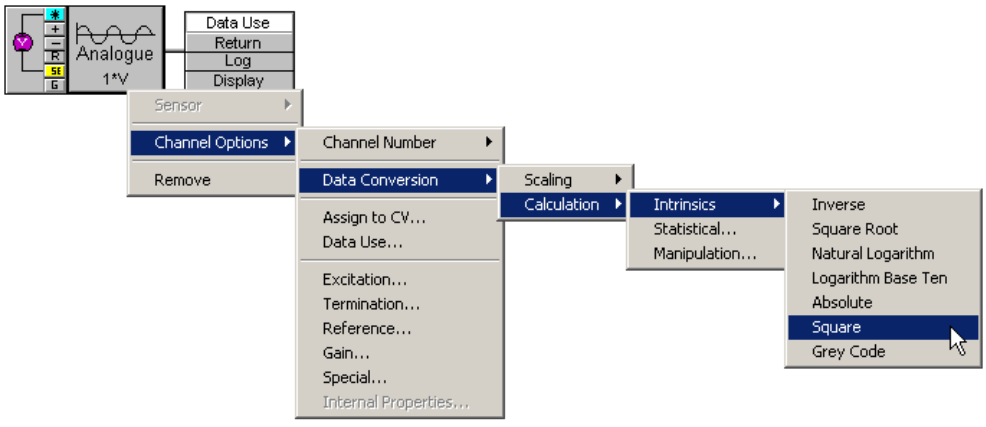
```
8V(10.5,Y9)
```

commands the *dataTaker* to measure the attenuated differential voltage signal connected to analog channel 8, to scale the data by the attenuation factor of 10.5, and to linearise this by polynomial Y9.

Using DeLogger, data scaling channel options via spans and polynomials are selected in the Program Builder from the walking menus after a channel is defined as follows



and Intrinsic Functions are also selected in the Program Builder from the walking menus after a channel is defined as follows



Options for Data Manipulation

Channels can be read, and special data returned which relates to the previous readings for that channel. These options include difference, rate of change and integration.

Data manipulation options cannot be used for the input channels of Alarms. Instead, the channel must be included in a Triggered Schedule where the channel is read, the data is manipulated, and the result is assigned to a Channel Variable (nCV). The Channel Variable can then be tested in the Alarm statement.

Difference

DF – Returns the difference between the latest reading and the previous reading for the channel.

Rate of Change

RC – Returns the rate of change between the last reading and the previous reading, with respect to the sample times (x/dt). The time base is in seconds, and data is returned in units/sec for the signal type.

Rate of Change

RS – Returns the rate of change between the latest reading and the previous reading, with respect to the sample times (dx/dt), where the original data is already a difference (for example resetting counters). The time base for rate of change is in seconds, and the data is returned in units/sec for the signal type.

Integrate

IB – Returns the integral between the latest reading and the previous reading, with respect to the sample times. The time base for the integral is in seconds, and the data is returned in units.secs for the signal type.

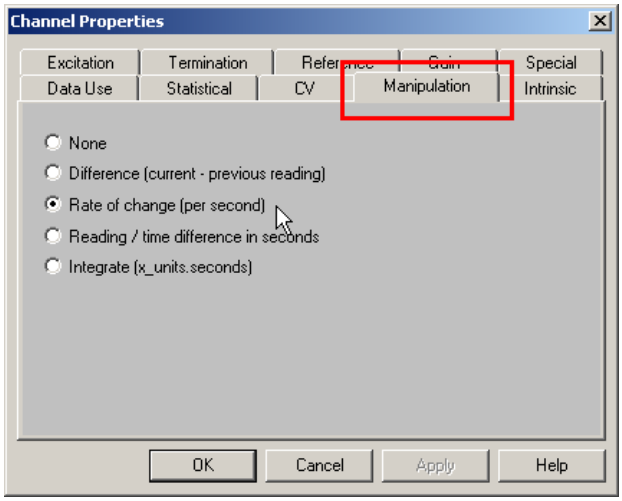
Examples

Using DeTransfer, the following channel option for example

```
4PT385(200.0,DF)
```

commands the *dataTaker* to return the difference in temperature between successive readings of a PT385 with 200ª zero resistance, connected to analog channel 4.

Using DeLogger, the Data Manipulation channel options are available on the Manipulation tab of the Channel Properties dialog in the Program Builder.



Options for External References

Thermocouple and bridge inputs require external references under some circumstances, which are used in the calculation of the related measurement inputs.

The reference channels are scanned, and the data retained internally for use as references or for compensation of other channel types. The reference channels do not return, log or display any data.

The reference channel inputs are identified by channel options as follows

Thermocouple Reference Temperature

TR – Indicates that the input is an external thermocouple reference junction temperature sensor. It is used as an option for the RTD, thermistor and monolithic temperature sensors, which can all be used as thermocouple reference junction temperature sensors.

If a thermocouple is junction compensated by an external conditioning device, then use 11SV(TR) as the reference channel. The System Variable 11SV returns mathematical zero.

Thermocouple Zero Reference Channel

TZ – Indicates that the input is an external thermocouple zero voltage reference. It is used as an option for the voltage input type.

Bridge Excitation Voltage Channel

BR – Indicates that the input is an external supply voltage for a bridge input, and is to be used as the bridge reference for ratiometric bridge measurement.

Options for Statistical Evaluation

Input channels can be read at intervals, and produce a statistical summary of the data at longer intervals (see Section III – Statistical Sub Schedule).

Channels which have statistical channel options are sampled at the interval defined by the Statistical Sub Schedule, and the statistical data is calculated and returned at the associated Repeating Schedule time.

Statistical channel options cannot be used in Alarms. The statistical data must first be generated in a Triggered Schedule/ Statistical Sub Schedule, and the data assigned to a Channel Variable (nCV) which then be tested in an Alarm.

Average

AV – Returns the average reading for the period, which is the sum of all the channel readings divided by the number of readings.

Standard Deviation

SD – Returns the standard deviation of the readings for the period, which is a measure of the variability of the data about the average. The population size used in the standard deviation calculation is (n-1).

Minimum & Maximum

MN & MX – Returns the minimum and the maximum readings for the period, which are a measure of the range of the data about the average.

Date & Time of Minimum & Maximum

DMN, DMX, TMN, TMX – These channel options return the date and the time at which the minimum and maximum values occurred. Any combination of these channel options can be used to return data as required.

Integration

INT – Returns the integral of the data for the period. The calculation uses a trapezoidal approximation.

The integral is calculated with respect to the time period in seconds. Other time bases can be used by applying appropriately defined polynomials.

Histogram

Hx:y:n..mCV – Returns a histogram (statistical frequency distribution) of the channel data for a period.

The range of the histogram classes is set by x and y, and the number of classes is set by the number of channel variables between n..m. The class intervals are calculated by the *dataTaker* for these ranges.

The histogram is produced by counting the number of readings that lie within each class. The class frequency results are placed in the sequentially numbered channel variables.

The histogram is described in detail in Section III - Statistical Sub Schedule.

Examples

Using DeTransfer, the following channel options for example

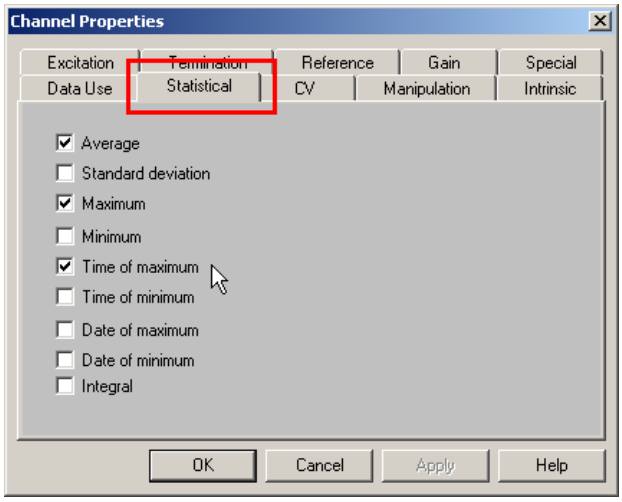
```
3TK(AV)(MN)(TMN)(MX)(TMX)
```

commands the *dataTaker* to return the average temperature, the minimum temperature and time of the minimum temperature , and the maximum temperature and the time of maximum temperature, for a Type K thermocouple connected to channel 3.

```
3PT385(INT)
```

commands the *dataTaker* to return the integral temperature (heat sum) in degree**.**seconds measured by an RTD (PT385) connected to channel 3.

Using DeLogger, these channel options are available on the Statistical tab of the Channel Properties dialog in the Program Builder.



Options for Channel Variables

Channel Variables are floating point memory registers, that are used to temporarily store data for use elsewhere in the application (See Section III - Channel Variables and Calculations).

The current reading for any channel can be assigned to a Channel Variable by including the Channel Variable as a channel option. The data can be assigned to Channel Variables in a number of ways.

Assign

=nCV – Assign the present channel reading to the Channel Variable, overwriting any previous value.

Add

+=nCV – Add the present channel reading to the current contents of the Channel Variable.

Subtract

–=nCV – Subtract the present channel reading from the current contents of the Channel Variable.

Multiply

*=nCV – Multiply the current contents of the Channel Variable by the channel reading.

Divide

/=nCV – Divide the current contents of the Channel Variable by the channel reading.

Examples

Using DeTransfer, the following channel options for example

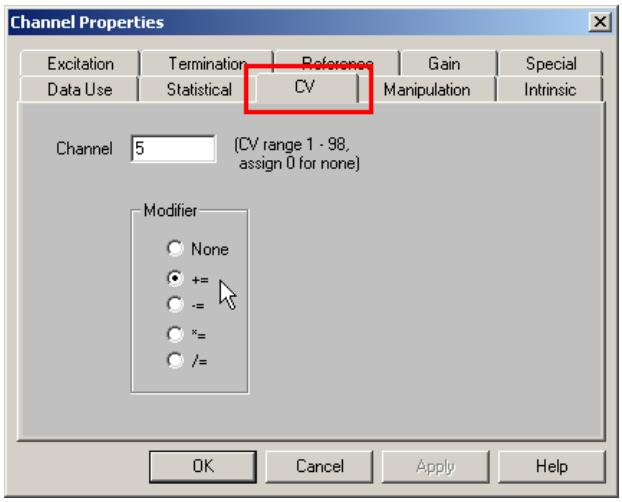
```
1TK(=3CV)
```

commands the *dataTaker* to assign the reading for a thermocouple type K connected to channel 1 to Channel Variable 3.

```
2C(R,+=9CV)
```

commands the *dataTaker* to read counter channel 2, reset the counter when it is read, and add the value to Channel Variable 9.

Using DeLogger, the channel variable options are available on the CV tab of the Channel Properties dialog in the Program Builder.



Options for Output Data Format

The *dataTaker* can format the data returned to the computer, and the data displayed on the LCD display, in a number of ways.

The following channel options allow each channel to be tailored to a custom data format for its destination. These channel options are described in detail in Section III - Format of Returned Data.

Fixed Floating Point

FFn – Data is returned and displayed for the channel in floating point format with n decimal places.

Fixed Exponential digits

FEn – Data is returned and displayed for the channel in exponential format with n significant digits in the mantissa.

Fixed Mixed output

FMn – Data is returned and displayed for the channel in floating point format with n decimal places if between 10^{-4} and 10^{+n} , otherwise in exponential format with n significant digits in the mantissa.

Channel identification Text

"*text*" – A unique channel name of up to 16 characters which replaces the default channel number and type which is normally returned and displayed.

No Return

NR – Data from the channel is not to be returned to the host, but can be logged and displayed.

No Log

NL – Data from the channel is not to be logged, but can be returned to the host and displayed.

No Display

ND – Data from the channel is not to be displayed, but can be returned to the host and logged.

Intermediate or Working Channel

W – Data from the channel is not to be returned to the host, logged, or displayed. This channel option is used when performing intermediate or working calculations, for which the data is only used internally.

Bar Graph

BGx:y – When a display is fitted to the *dataTaker*, this channel option allows for data to be displayed in a bar graph format on the LCD. The bar graph channel option is described in Section III - Keypad and Display.

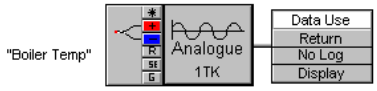
Examples

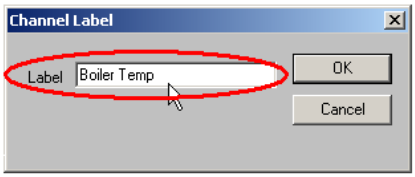
Using DeTransfer, the following channel options for example

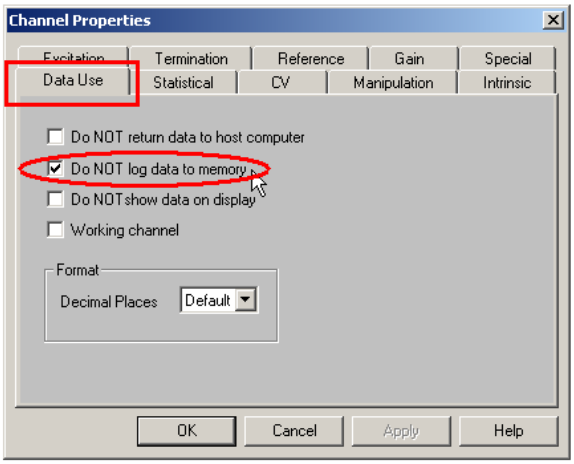
```
1TK("Boiler Temp.",FF1,NL)
```

commands the *dataTaker* to read a thermocouple type K connected to analog channel 1, and to return the data named as 'Boiler Temp' in the Floating Point format to 1 decimal place. This data is not logged into memory.

Using DeLogger, the channel label is defined in the Channel Label dialog, and the data format and data use options are defined in the Channel Properties dialog


"Boiler Temp"
Analogue
1TK
Data Use
Return
No Log
Display


Channel Label
Label
Boiler Temp
OK
Cancel


Channel Properties
Excitation
Termination
Reference
Gain
Special
Data Use
Statistical
CV
Manipulation
Intrinsic
Do NOT return data to host computer
Do NOT log data to memory
Do NOT show data on display
Working channel
Format
Decimal Places
Default
OK
Cancel
Apply
Help

Schedules

General operation of the *dataTaker* data loggers centres around scheduling. Schedules determine when various processes are to occur, and can be triggered by the real time clock, by digital or counter events, by alarm states, and by the host computer. The *dataTaker* supports 7 different types of schedules.

The Schedules manage schedule lists, which contain the various actions that are to be performed when the Schedule is triggered or runs. These actions include reading input and internal channels, setting output channels, evaluating expressions, testing alarms, returning data, logging data, etc.

Many *dataTaker* commands relate to the Schedules and their schedule lists. Therefore it is important to understand this core concept of Schedules and scheduling.

There are three basic types of Schedules

Immediate Schedule – the schedule list is processed or scanned once immediately, and any data produced is returned to the host computer. The schedule list can be rescanned at any time by poll triggers from the host computer. Data from the Immediate Schedule cannot be logged or displayed.

Triggered Schedules – the schedule list is processed or scanned at regular or irregular intervals of time. Any data that is produced can be returned to the host and logged to the internal data memory or a memory card.

Statistical Sub Schedule – the schedule list is processed or scanned repeatedly and the data is used to calculate a periodic average, standard deviation, minimum and/or maximum or integral data for the input channels.

This Sub Schedule is used in conjunction with the Triggered Schedules, which determine the interval at which the statistical data is calculated and returned to the host, logged into the data memory or memory card, and displayed.

The Triggered Schedules and the Statistical Sub Schedule can be triggered by

real time, using the *dataTaker* real time clock – clock events

a change of state of one or more digital input channels – digital events

one or more counters incrementing to a terminal value – counter events

while one or more digital input channels is high – while conditional

There are six Schedules which can operate simultaneously to read input channels, evaluate expressions, etc. Each Schedule can trigger at different intervals in response to different triggers, and each with a different schedule list.

The six different types of Schedules are

one Immediate Schedule

five Triggered Schedules

one Statistical Sub Schedule, which can be associated with any or all of the Triggered Schedules.

These could be considered as the Data Schedules. There is also a single Alarm Schedule, which contains all of the alarms to be processed (See Section III – Alarms).

Number of Channels for Schedules

Whenever Schedules with schedule lists are entered into the *dataTaker*, the channels that are to be scanned and calculations that are to be performed are all entered into an internal channel table.

The channel table contains all of the details of the individual channels to be scanned, including the type of input signal, the scan interval, how resultant data is to be processed, where resultant data is to be sent, etc. The channel table also contains a RPN (Reverse Polish Notation) form of each expression.

The Data Schedules and the Alarm Schedule all share the internal channel table. A maximum of 110 channels, calculations and alarms can be entered into the channel table.

After power up, a hardware reset or a RESET command, the internal channel table is allocated between the Data Schedules and the Alarm Schedule as follows

90 entries in the channel table are allocated for the Data Schedules

20 entries in the channel table are allocated for the Alarm Schedule

Reallocating the Internal Channel Table

The allocation of the internal channel table for the Data Schedules and the Alarm Schedule can be changed using the Parameter30 command (See Section III – Parameter Commands).

Parameter30 defines the number of entries in the internal channel table which are to be allocated to the Alarm Schedules. The balance is allocated to the Data Schedules.

For example, using DeTransfer the number of entries is changed by the commands

```
P30=0
```

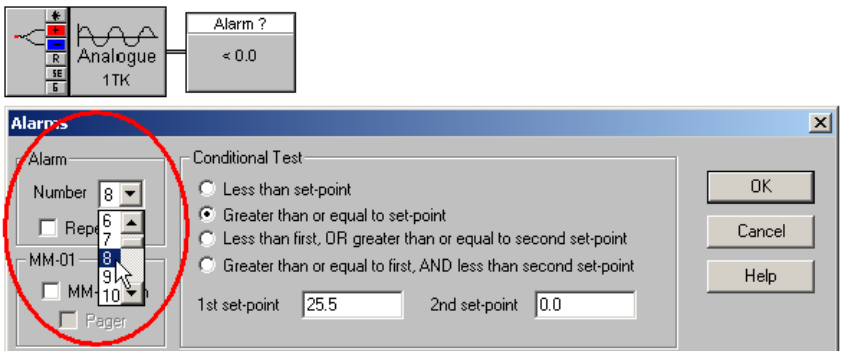
which allocates no space in the channel table for the Alarm Schedule, and sets the maximum space for the Data Schedules to 110 entries, and

```
P30=50
```

which allocates 50 entries in the channel table to the Alarm Schedule, and allocates 60 entries in the channel table for the Data Schedules.

The internal channel table must be reallocated if necessary before any Data or Alarm Schedules are entered. Attempts to change the channel table allocation after any Schedules have been entered will result in the error message 'E8- Parameter read/set error'.

DeLogger allows up to 60 alarms to be defined in a program, and automatically sets the Parameter30 command to the highest alarm number used in a program.



Always define alarms beginning at alarm number 1, rather than randomly using alarm numbers which will waste space in the channel table.

The internal channel table can be cleared using the CSCANS and CALARMS commands.

Protecting the Internal Channel Table

When Data Schedules are entered into the internal channel table, entry of successive Data Schedules will overwrite and destroy any existing Data Schedules.

The exception to this general rule is when the original Schedules have logged data into data memory, when attempts to enter new Schedules are rejected.

The risk of accidentally overwriting the internal channel table, or of accidentally altering Schedules, can be overcome by setting the Fix Channel Table Switch as follows

/F Fix or secure the channel table, not allowing entry of new Data Schedules or re-defining triggering for existing Data Schedules

/f Free the channel table, allowing entry of new Data Schedules or re-defining triggering of existing Data Schedules

The Fix Channel Table Switch defaults to disabled (/f) after a power up, a hardware reset or a RESET command. Attempts to enter new Schedules, or to re-define the triggering for existing Schedules, while the Fix Channel Table Switch is enabled will result in the error message 'E48-schedule list fixed'.

Immediate Schedule

A single channel or expression, or a group of channels and/or expressions, can be processed at any time by the *dataTaker* using the Immediate Schedule.

The Immediate Schedule can be used for a number of purposes, including checking, testing or calibrating channels, initialising variables, changing the state of outputs, etc. The Immediate Schedule does not disrupt any other Schedules which may be in progress, and so can be used when programs are running.

The Immediate Schedule runs only once, immediately after the schedule list is received from the host computer. Any data that is produced is returned immediately to the host computer, and cannot be displayed on the logger display panel or logged into the data memory.

The Immediate Schedule does not repeat automatically, however the Immediate Schedule can be run again at any time by entering the schedule list again, or by entering the Immediate Trigger.

Immediate Schedule Command

An Immediate Schedule is specified by entering a schedule list containing one or more input channels and/or expressions, in the general format

schedule list

There is no separate trigger for this Schedule – the entering of the schedule list is the trigger.

Using DeTransfer, Immediate Schedules are entered by the following commands for example

```
T
1V  1DS
1..5TT
```

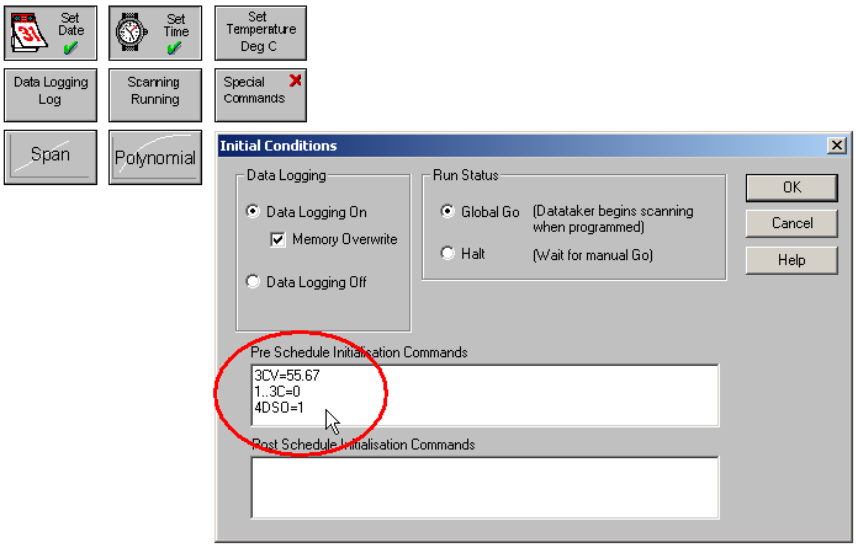
These 3 Immediate Schedules will respectively

return the current time

return the voltage measured on analog input channel 1, and the logic state on digital input channel 1

return the temperature from five T type thermocouples connected to analog channels 1 through 5

When a new Immediate Schedule is entered, it replaces any previous Immediate Schedule list.



DeLogger does not directly support the Immediate Schedule, however when channel specifications or expressions are placed in the Pre Schedule or Post Schedule fields of the Initial Conditions dialog, these are assembled into the final program as Immediate Schedules.

Immediate Schedules can also be sent from the Text View, either directly from the lower Entry Screen, or from the User buttons.

Immediate Trigger Command

After an Immediate Schedule has been entered into the *dataTaker*, the Schedule can be run again by re-entering the same schedule list, or by entering the Immediate Trigger command as follows

* (the Immediate Trigger is the asterisk character)

The schedule list associated with the last entered Immediate Schedule can be processed as many times as required, simply by entering the Immediate Trigger for each execution required. If there is no Immediate Schedule schedule list entered when the Immediate Trigger is received, then the command is ignored.

The Immediate Schedule operates similarly to the Schedule Triggered by Poll (see below), except that

the Immediate Schedule first runs immediately after it is entered into the logger

Schedule Triggered by Poll first runs after the first Poll Trigger command (X) is received (see below).

DeLogger does not support the Immediate Trigger– it is generally not appropriate in the DeLogger environment. However if necessary, a User button in the Text View could be set up to send the Immediate Trigger command (*) whenever clicked with the mouse.

Using Immediate Schedules in Programs

The positioning of the Immediate Schedule within *dataTaker* programs using the BEGIN . . . END block construct is important.

The Immediate Schedule list must be positioned either

outside of the BEGIN . . . END block construct, before BEGIN or after END

following BEGIN, but before any Triggered Schedules are entered

The Schedule will execute immediately after entry in both cases.

Using DeTransfer, the Immediate Schedule can be included in programs as follows

```
1V(=10CV,W)
BEGIN
      'Triggered Schedules, etc
END
```

where the Immediate Schedule list is entered before the BEGIN, and is immediately executed.

Also

```
BEGIN
      1V(=10CV,W)
      'Triggered Schedules, etc
END
```

where the Immediate Schedule list is entered after the BEGIN but before any Triggered Schedules, and is immediately executed on entry.

However, if the Immediate Schedule list is entered after the BEGIN and after a Triggered Schedule as follows

```
BEGIN
     'Triggered Schedules, etc
     1V(=10CV,W)
END
```

then it will be attached to the Triggered Schedule and will not behave as an Immediate Schedule.

Triggered Schedules

The Triggered Schedules provide the most useful form of data acquisition and data processing by the *dataTaker*. These Schedules allow input channels to be repeatedly scanned, expressions to be repeatedly evaluated, etc. often without the need for intervention or control by the host.

The Triggered Schedules can also be regarded as the repeating schedules or reporting schedules, in that these schedules repeatedly report data at intervals.

A maximum of five Triggered Schedules can be entered into the *dataTaker*, and each having a different schedule list, each running at different intervals in response to different triggers.

The data from Triggered Schedules can be returned to the host computer in real time, can be logged into the internal data memory or a memory card, and can be displayed.

The Triggered Schedules repeatedly scan a group of input channels, and/or repeatedly evaluate a group of expressions, all of which are defined in the schedule lists. The repetition is based on various triggers and conditions as follows

Schedule Triggered by Poll – runs whenever a poll command is received by the logger

Schedule Triggered by Time – runs at regular intervals of time from seconds to months, and is triggered by the logger's real time clock

Schedule Triggered by Digital Events – runs at regular or irregular intervals, and is triggered by state changes or 'events' on the digital inputs

Schedule Triggered by Counter Events – runs at regular or irregular intervals, and is triggered by total count 'events' from the counters

General Format for the Triggered Schedules

The Triggered Schedules are specified by the prefix R, that is extended by parameters which define the trigger and optional conditions for triggering.

The general format of the Triggered Schedules is

R(*trigger*) [*schedule list*]

where

R	is the Trigger Schedule identifier
(trigger)	is the trigger for when the Schedule runs
[schedule list]	list of the channels to be read, expressions to be evaluated, etc

The trigger for the Schedules can be defined in terms of any of the following

a poll trigger, received by the logger as the Poll Trigger command (X)

a time trigger, specified as a time interval between triggers from the real time clock

a digital event trigger, specified by one or more digital input channels that receive events

a counter event trigger, specified by one or more counters which will generate counter total events

The specifications of the different types of triggers for initiating the Triggered Schedules are described in the following chapters.

Normally, when Triggered Schedules are entered, these will replace or overwrite any existing Triggered Schedules. However a number of Triggered Schedules can be entered using a special BEGIN…END block construct, which causes the logger to append successive Triggered Schedules into one channel table list (See Section III - General Schedule Commands).

Identifying Triggered Schedules

When only one Triggered Schedule is entered into the *dataTaker*, then the general command format described above is sufficient. However the *dataTaker* supports up to five Triggered Schedules, which have individual identification.

If two or more Triggered Schedules are entered in the *dataTaker*, then individual Schedules are identified as Triggered Schedules RX, RA, RB, RC and RD.

The general formats for defining the five Triggered Schedules are listed below

RX[*schedule list*]

RA(*trigger*)[*schedule list*]

RB(*trigger*)[*schedule list*]

RC(*trigger*)[*schedule lis*]

RD(*trigger*)[*schedule lis*]

Triggered Schedules A, B, C and D all have a (*trigger*) which is defined in terms of time, digital events or counter events. The Repeat Schedules run in response to their defined trigger. The Schedule Triggered by Poll (RX) does not have a variable trigger. The trigger is the Poll Trigger command (X) that is sent from a host computer or from elsewhere in the program, such as from alarms.

If the Triggered Schedules are not given an identity at the time of entry, then the *dataTaker* assigns an identity in the order A, B, C then D.

If one Triggered Schedule without identity is entered, this assumed to be Triggered Schedule A. If two or more Triggered Schedules without identity are entered in one command line, then the first is assigned as Triggered Schedule A, the second as Triggered Schedule B, etc.

The Triggered Schedules only run in response to their defined trigger. When the Schedules are first entered, the Schedule does not run until the first trigger occurs. This is in contrast to the Immediate Schedule, in which case the Schedule runs as soon as it is entered.

Halting and Resuming Triggered Schedules

The Triggered Schedules can be globally and individually halted, and globally and individually resumed, by the Halt and Go commands as follows

H	Halt all Triggered Schedules
G	Resume all Triggered Schedules
HA	Halt Triggered Schedule A
GA	Resume Triggered Schedule A
HB	Halt Triggered Schedule B
GB	Resume Triggered Schedule B
HC	Halt Triggered Schedule C
GC	Resume Triggered Schedule C
HD	Halt Triggered Schedule D
GD	Resume Triggered Schedule D

When the *dataTaker* is initially powered up or executes the RESET command, all Triggered Schedules are in the halted state. This allows the *dataTaker* to sleep.

When Triggered Schedules are entered, they are automatically enabled at the time of entry which allows the Schedules to run at the next trigger.

Note that the Schedule Triggered by Poll cannot be halted, since by its very nature it is halted until the next Poll Trigger command is received.

Order of Data Returned by Triggered Schedules

Whenever two or more Triggered Schedules are entered into the *dataTaker*, and all are running, then the data that is produced is returned to the host computer in the order

– Immediate Schedule data

– Schedule Triggered by Poll data

– Triggered Schedule A data

– Triggered Schedule B data

– Triggered Schedule C data

– Triggered Schedule D data

This ordered return of data occurs irrespective of the order in which the Triggered Schedules were originally entered into the *dataTaker*.

If the Units Text Switch is enabled (/U), then a CR and LF character sequence is returned at the end of the block of data from each Triggered Schedule.

If the Units Text Switch is disabled (/u), then the block delimiter character defined by the Parameter24 command is returned at the end of the block of data from each Triggered Schedule.

Extending Triggered Schedules

The use of more than one Triggered Schedule in a program is discussed in detail in Section III – Multiple Triggered Schedules.

Operation of the Triggered Schedules can also be conditional on external conditions, that are monitored as the logic state on one or more digital input channels. This conditional triggering is described in Section III - Schedules Triggered While Condition.

Schedule Triggered by Poll

The simplest of the Triggered Schedules for repeatedly executing a schedule list is the Schedule Triggered by Poll. This Schedule executes its schedule list each time that a Poll Trigger command is received by the logger's command processor.

The Poll Trigger command can come from

a host computer, where software running on the computer determines when the Schedule must run, and sends a Poll Trigger command to the logger to run the Schedule

other parts of the running program such as alarms, which can generate the Poll Trigger command when an alarm state occurs. This is useful for performing tasks such as returning, logging or displaying data, changing output channels, etc. in response to alarm conditions.

The *dataTaker* has only one Schedule Triggered by Poll.

When a Schedule Triggered by Poll is been entered into the logger, the schedule list is not executed until a Poll Trigger is received. When a Poll Trigger is received the schedule list is executed. The logger does not execute this list again until the next Poll Trigger command is received.

Resulting data can be returned to the host computer, logged to the internal data memory or a memory card, and displayed.

Whenever successive Schedule Triggered by Polls are entered, these will replace or overwrite existing Schedules.

Schedule Triggered by Poll Command

The Schedule Triggered by Poll is specified by the general command format

RX [*schedule list*]

where

RX	is the Schedule Triggered by Poll command
[schedule list]	list of the channels to be read, expressions to be evaluated, etc

The Schedule Triggered by Poll has a command trigger. The trigger is an X character sent from the host computer or from an alarm.

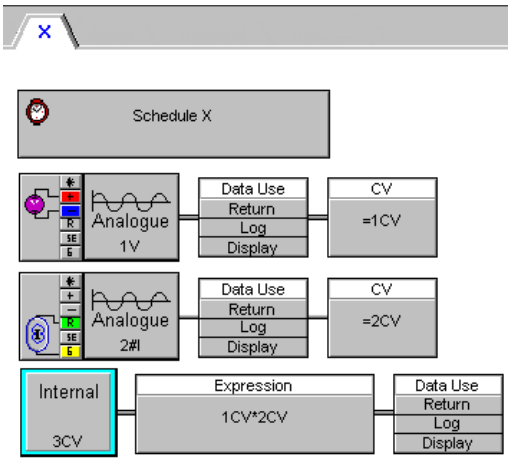
Using DeTransfer, a Schedule Triggered by Poll is entered as follows

```
BEGIN
 RX
  1V(=1CV)   2I(=2CV)   3CV("Power")=1CV*2CV
END
```

The schedule list is executed each time a Poll Trigger is received from the host or an alarm.

Using DeLogger, a Schedule Triggered by Poll can be entered into a program in the Program Builder as follows. Click on the 'X' tab to open the Schedule Triggered by Poll work area, and then enter the input channels and calculations to be executed into the work area as for other schedules.

Note that the Schedule button cannot be opened – the Schedule trigger is permanently set to the X command.



Triggering the Schedule from the Host Computer

When a Schedule Triggered by Poll has been entered into the *dataTaker*, the schedule list is not executed until the appropriate trigger is received. Whenever a Poll Trigger command is received by the logger's command processor, this triggers execution of the schedule list.

The Poll Trigger command is the X character.

Using DeTransfer, a Schedule Triggered by Poll is triggered to execute by sending the X character to the logger, terminated by CR or CR/LF, for example

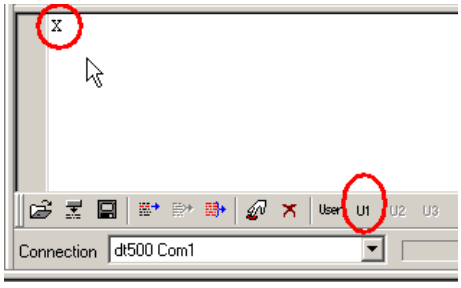
```
X
```

The schedule list executed each time this trigger is received.

This form of input channel scanning is appropriate to polling the *dataTaker* from a dedicated host computer to obtain data at intervals determined by the host. A number of the third party data acquisition software packages manage the *dataTaker* by this method.

Using DeLogger, a Schedule Triggered by Poll can be executed by either of two methods

- in the Text View, type the 'X' command into the lower or Entry Screen, and send it to the logger by hitting the Return key
- in the Text View, define a User button on the lower toolbar as an 'X' command, and then click this button each time the Schedule is required to execute.



Make sure that the Text View is logically connected to the logger – select the connection from the drop down list at the bottom of the window.

Triggering the Schedule from an Alarm

The Alarms can also issue the Poll Trigger command (X), thereby triggering execution of the schedule list.

The following example with DeTransfer illustrates triggering a Schedule Triggered by Poll from an alarm

```
BEGIN
 RX
  1TK
  9CV(W)=9CV+1

 RS1S
  ALARM5(1TK>150)"[X]"
end
```

The schedule list is executed each time the temperature of the thermocouple exceeds 150°C. ALARM5 detects the over range condition, and triggers the Schedule to record the temperature under alarm conditions

Schedules Triggered by Time

The most widely used of the Triggered Schedule for executing schedule lists is the Schedule Triggered by Time. This Schedule is used to execute schedule lists at regular intervals of time ranging form 1 second to months.

The schedule list is executed automatically at each interval of time, and data can be returned to the host computer, logged into the internal data memory or a memory card, and displayed.

The Schedule Triggered by Time is suited to the situation where the *dataTaker* is used independently of the host, recording data at regular intervals of time into the internal data memory or a memory card.

Schedule Triggered by Time Command

Schedules Triggered by Time are triggered by the real time clock, and the trigger is specified in terms of the time interval between successive triggers. This therefore allows specification of the interval between successive readings of input channels, etc.

The real time scan interval can be from one second to months, and is specified in terms of seconds, minutes, hours or days.

Up to four Schedules Triggered by Time can be entered into the *dataTaker*, with each executing a different schedule list at different intervals of time.

The general formats for Schedules Triggered by Time are as follows

R(interval)S[*schedule list*]

R(interval)M[*schedule list*]

R(interval)H[*schedule list*]

R(interval)D[*schedule list*]

where

R	is the Triggered Schedule identifier
(interval)	is the trigger interval in the range 1 – 65535
S	the time interval is in seconds
M	the time interval is in minutes
H	the time interval is in hours
D	the time interval is in days
[schedule list]	list of the channels to be read, expressions to be evaluated, etc

The trigger *interval* must be an integer in the range of 1 to 65535. This value is interpreted in terms of the S, M, H or D unit of time specified to determine the real time trigger interval.

If the trigger *interval* is zero, then the Schedules Triggered by Time run at the maximum rate irrespective of whether S, M, H or D is specified. If no time interval is specified, then the Schedule will also run at the maximum rate possible.

When a schedule list is to be executed at an interval which is a fraction or part of a minute, hour or day, then the interval must be specified in the next lowest unit of time. If for example the Schedule is to be triggered every 1.5 hours (1.5H), then the trigger interval must be specified as 90 minutes (90M).

Using DeTransfer, Schedules Triggered by Time to run at regular intervals of time are for examples

```
BEGIN
 R15M
  1..5V  8TK  1..2DS
END
```

where the Schedule will run every 15 minutes. This Schedule is not given an identity, and so is assigned as Schedule A by the logger

```
BEGIN
 RA20S
  1..5V  8TK  1..2DS
END
```

where the Schedule A will run every 20 seconds.

```
BEGIN
 RB12H
  1..5V  8TK  1..2DS
END
```

where the Schedule B will run every 12 hours.

The same trigger interval can be defined in several ways, which allows complete flexibility in the definition of the time interval. For example the Schedules

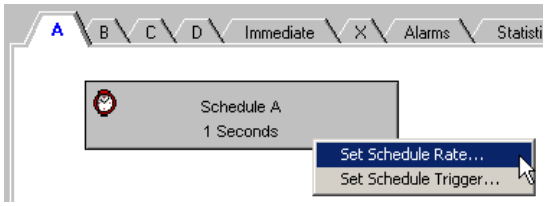
```
RA3600S 1..5V 8TK 1..2DS

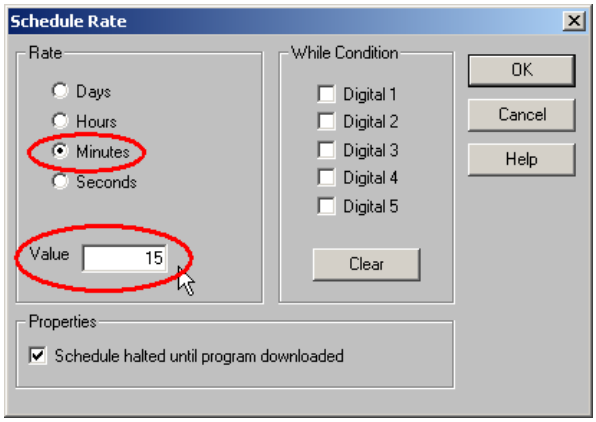
RA60M 1..5V 8TK 1..2DS

RA1H 1..5V 8TK 1..2DS
```

will all run every hour.

Using DeLogger, the time interval for Schedules Triggered by Time is defined in the Program Builder for the chosen Schedule. Click on the A, B, C or D tab to select the schedule, and right click on the Schedule button at the top of the work area and select Set Schedule Rate…





The Schedule Rate dialog opens, click on the required time units radio button, and enter the number of time units in the Value field. The example above sets the trigger interval for Schedule A to 15 minutes.

Running Schedules at the Maximum Speed

The *dataTaker* cannot be programmed to run Schedules at intervals of less than 1 second. However, where intervals of less than 1 second are required, then the 'free run' or maximum rate can be used.

If no time interval trigger is defined, or an interval trigger of 0 (zero) time units is defined, then the Schedule will execute continuously at the maximum rate possible for the program.

The analog to digital converter of the *dataTaker* has a sampling rate of 16.67 conversions/second (60Hz) or 20 conversions/second (50Hz). In the case of digital state and counter channels, the sampling rate is faster. Calculations only take 2 – 5 millisecs.

The maximum Schedule execution rate is also influenced by other factors such as

the configuration of the analog to digital converter

the number and complexity of other Schedules which are also in progress

whether input channels sampled are simple analog inputs such as voltage, current, frequency, etc. or are complex inputs such as thermocouples, RTD's, etc. requiring additional processing

the linearization of channel data by polynomials and span limits

whether data is being returned to the host as it is acquired

whether data is being logged as it is acquired

whether alarms are being monitored

whether the internal data memory or memory card is being simultaneously unloaded to the host

Despite these factors, the maximum Schedule execution rate for the *dataTaker* is generally 8 – 10 channels/second if the data is being returned to the host computer in real time (9600 baud), and 25 channels/second if the acquired data is being logged to the internal data memory or memory card.

The maximum analog sampling rate of the *dataTaker* is discussed in detail in Appendix E.

Command for Scanning at the Maximum Rate

Continuous execution of Schedules Triggered by Time at the maximum conversion rate is programmed by entering a Schedule with no time interval trigger, in the general format

R [*schedule list*]

where

R	is the Triggered Schedules identifier
[schedule list]	list of the channels to be read, expressions to be evaluated, etc

Continuous execution at the maximum conversion rate can also be programmed by entering a Schedule with a 0 (zero) time interval trigger.

Using DeTransfer, a Schedule Triggered by Time is programmed to execute continuously at the maximum rate as follows

```
BEGIN
 RA
  1V 2R 3F 4DS
END
```

Note: The regularity of the repetitive executions may vary slightly due to automatic recalibration of the analog to digital converter, if this is enabled. However such delays are rarely a problem.

Halting and Resuming Schedules Triggered by Time

Schedules Triggered by Time can be individually and globally halted and resumed by the Halt and Go commands

H	Halt all Triggered Schedules
G	Resume all Triggered Schedules
HA	Halt Triggered Schedule A
GA	Resume Triggered Schedule A
HB	Halt Triggered Schedule B
GB	Resume Triggered Schedule B
HC	Halt Triggered Schedule C
GC	Resume Triggered Schedule C
HD	Halt Triggered Schedule D
GD	Resume Triggered Schedule D

When the *dataTaker* is initially powered up or executes the RESET command, all Triggered Schedules are in the halted state. This allows the *dataTaker* to sleep.

When Triggered Schedules are entered, they are automatically enabled at the time of entry which allows the Schedules to run at the next trigger.

Schedules Triggered by Digital Events

Schedules Triggered by Digital Events are used to run Schedules and execute schedule lists whenever a change of state occurs on one or more of the digital input channels. The resulting data can be returned to the host computer, logged to the internal data memory or a memory card, and displayed.

The digital event used to trigger the Schedule can be one of three different types of event as follows

any change of state that occurs on the nominated digital input channel(s)

a low to high (negative to positive) change of state occurs on the nominated digital input channel(s)

a high to low (positive to negative) change of state occurs on the nominated digital input channel(s)

One or more digital input channels of the *dataTaker* can be used to trigger Schedules. When more than one digital event input is used, then triggering operates in a logical OR manner such that the Schedule runs when any of the digital inputs changes state.

Schedules Triggered by Digital Events can be used where the *dataTaker* is operating independently of a host computer, to record data at intervals determined by external asynchronous events that often come from the machine or system under test. These events are usually a logic state change or contact closures.

Schedules Triggered by Digital Events could for example be used as the basis of an event recorder, or for a process 'flight recorder', or for gathering data in response to intermittent faults within a machine, process or system, etc.

Note 1 : The digital channels of the *dataTaker* are powered off while the logger is in the low power or sleep mode, and so cannot detect changes of state. Therefore the digital input channels cannot be used to detect external events for triggering Schedules during low power operation. Digital event triggering can only be performed if the automatic low power selection of the *dataTaker* is disabled by setting Parameter15=2, or if the logger is externally powered.

Note 2 : The digital channels used to detect external events for triggering scans are bi-directional. These channels cannot be used to detect external events if they have previously been used for digital outputs, and left switched ON. The digital channels can only be used to detect events if their output state is OFF.

Note 3 : The digital channels of the Channel Expansion Modules do not support triggering of Schedules.

Schedule Triggered by Digital Events Command

The general formats for Schedules Triggered by Digital Events command are as follows

RnE[schedule list]

Rn..mE[schedule list]

Rn+E[schedule list]

Rn..m+E[schedule list]

Rn–E[schedule list]

Rn..m-E[schedule list]

where

R	is the Triggered Schedule identifier
n	is any digital input channel
n..m	is a sequence of digital input channels
E	is the digital event identifier for triggering on any transition
+E	is the digital event identifier for triggering on positive transitions
–E	is the digital event identifier for triggering on negative transitions
[schedule list]	list of the channels to be read, expressions to be evaluated, etc

Using DeTransfer, Schedules Triggered by Digital Events are programmed for example

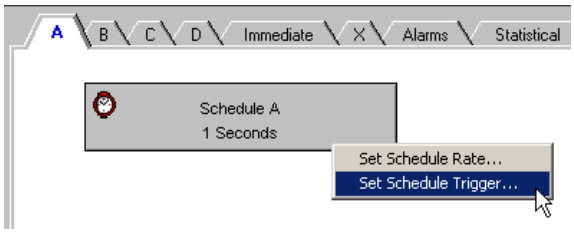
```
BEGIN
 RA1+E
  1..5V 8TK 1..2DS
END
```

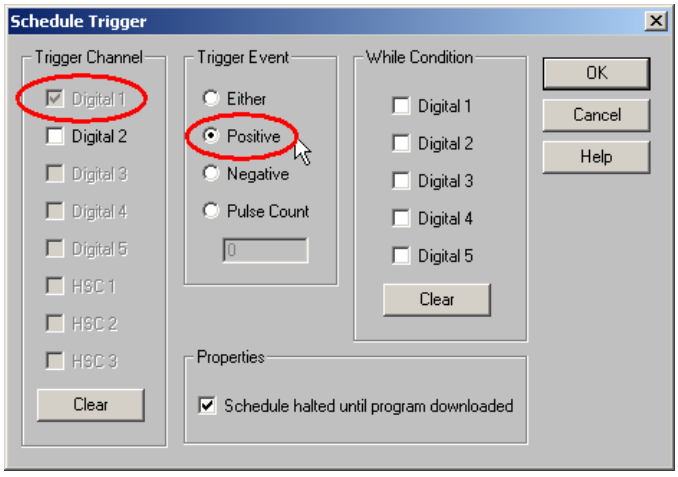
where the Schedule will run each time a positive transition occurs on digital input channel 1

```
BEGIN
 RA3..4E
  1..5V 8TK 1..2DS
END
```

where the Schedule will run whenever a positive or negative transition occurs on either of digital input channels 3 OR 4.

Using DeLogger, the digital event(s) for Schedules Triggered by Digital Event is defined in the Program Builder for the chosen Schedule. Click on the A, B, C or D tab to select the schedule, and right click on the Schedule button at the top of the work area and select Set Schedule Trigger…





The Schedule Trigger dialog opens, check the Trigger channel(s), and click the appropriate Trigger Event radio button.

The example above sets the trigger event for Schedule A as a positive going (low to high) state change on digital input channel 1.

If more than 1 digital input channel is used as a trigger, then these must be consecutive channels.

Halting and Resuming Schedules Triggered by Digital Events

Schedules Triggered by Digital Events can be individually and globally halted and resumed by the Halt and Go commands

H	Halt all Triggered Schedules
G	Resume all Triggered Schedules
HA	Halt Triggered Schedule A
GA	Resume Triggered Schedule A
HB	Halt Triggered Schedule B
GB	Resume Triggered Schedule B
HC	Halt Triggered Schedule C
GC	Resume Triggered Schedule C
HD	Halt Triggered Schedule D
GD	Resume Triggered Schedule D

Halting of the Schedules merely disables the sensitivity of the trigger to changes of state on the digital input channel(s).

When the *dataTaker* is initially powered up or executes the RESET command, all Triggered Schedules are in the halted state. This allows the *dataTaker* to sleep.

When Triggered Schedules are entered, they are automatically enabled at the time of entry which allows the Schedules to run at the next trigger.

Some Applications Ideas

When a number of digital events are used to trigger a Schedule, the digital event which initiated each execution can also be recorded by including the appropriate digital input channels in the schedule list as logic state inputs, as the following example illustrates

```
BEGIN
 RA2..4E
  2..4DS 1..10V
END
```

Alternatively the logic state byte input (1DB) could be included in the schedule list.

The extension of this technique is a digital event recorder, simply implemented by either of programs

```
BEGIN
 RA1..4E
  1..4DS
END

BEGIN
 RA1..4E
  1DB
END
```

which reads the digital input channels whenever an event occurs on any digital channel.

Schedules Triggered by Counter Events

Schedules Triggered by Counter Events are used to run Schedules and execute schedule lists whenever one or more of the low speed or high speed counters generates a counter event. The resulting data can be returned to the host, logged to the internal data memory or a memory card, and displayed.

Low Speed Counter Events

Low speed counter events are generated when one or more nominated low speed counters reaches a specified terminal count. When more than one low speed counter is nominated, then triggering operates in a logical OR manner such that the schedule list is scanned when any counter generates an event.

Note 1 : The digital input channels of the *dataTaker* are powered off while the logger is in the low power mode, and so cannot count input pulses or generate triggers. Therefore the low speed counters cannot be used to generate counter events during low power operation. Counter event triggering can only be performed if the automatic low power of the *dataTaker* is disabled by setting Parameter15=2, or if the logger is externally powered.

Note 2 : The digital channels used to generate the low speed counter events are bi-directional. These channels will not generate counter events if they have been used for digital outputs, and left ON. Digital channels will only generate counter events if the output state is OFF.

High Speed Counter Events

High speed counter events are produced when one or more nominated high speed counters receives a pulse. The *dataTaker* checks the high speed counters every 50mS, and if a counter nominated for counter event scheduling has received one or more pulses in the previous 50mS, then a trigger is generated.

When more than one high speed counter is nominated, the triggering operates in a logical OR manner such that the schedule list is scanned when any counter generates an event.

Note : The High Speed Counters are always powered, and can sense pulses while the logger is in the low power mode. However the logger must be awake to find the count(s) and generate triggers.

The sleeping logger can be waked by the input pulse, by diode connecting the high speed counter(s) to the Wake terminal. The input pulse wakes the *dataTaker* and is sensed by the high speed counter, which then generates a trigger. The diode is necessary to prevent a 10Hz frequency which is output by the Wake terminal when the *dataTaker* is awake, from reaching the high speed counter input.

Schedules Triggered by Counter Events Command

The general formats for Schedules Triggered by Counter Events command are as follows

RnC(*count*) [*schedule list*]

Rn..mC(*count*) [*schedule list*]

RnHSC [*schedule list*]

Rn..mHSC [*schedule list*]

where

R	is the Triggered Schedule identifier
n	is any counter channel
n..m	is a sequence of counter channels
C	is the Low speed counter event identifier
HSC	is the High speed counter event identifier
(count)	is the terminal count, in the range of 0 to 65535 (low speed counters only)
[schedule list]	list of the channels to be read, expressions to be evaluated, etc

The *count* parameter for the low speed counters sets the terminal count at which the counter generates an event to trigger the associated Schedule on Counter Events. The *count* parameter must be an integer, in the range of 0 to 65535.

The counters used for Schedules Triggered by Counter Events are set to zero when the original Schedule is entered into the *dataTaker*. Incoming pulses are counted until the terminal count is reached, and an event is generated and the counter is reset to zero.

Using DeTransfer, Schedules Triggered by Counter Events are programmed for example

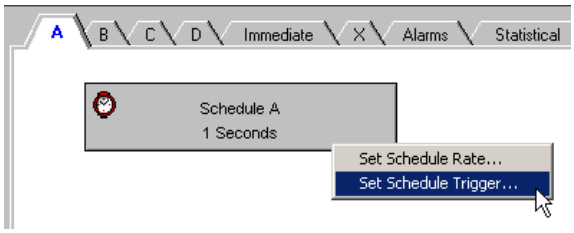
```
BEGIN
 RA1C(150)
  1..5V 8TK 1..2DS
END
```

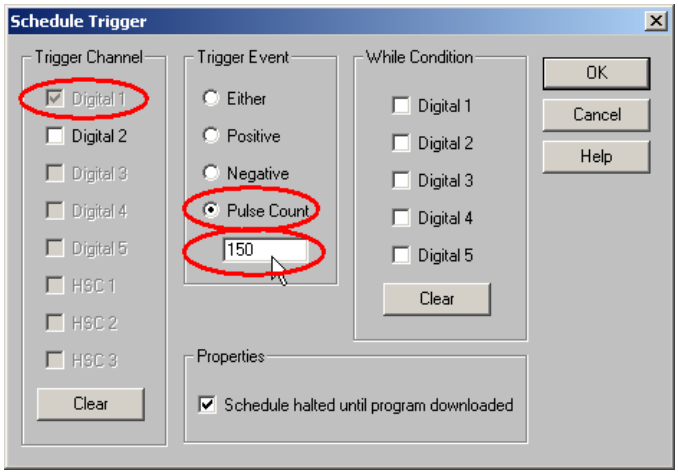
where the Schedule will run following each 150th count of low speed counter 1.

```
BEGIN
 RA1HSC
  1..5V 8TK 1..2DS
END
```

where the Schedule will run following each pulse sensed by high speed counter 1.

Using DeLogger, the digital event(s) for Schedules Triggered by Counter Event is defined in the Program Builder for the chosen Schedule. Click on the A, B, C or D tab to select the schedule, and right click on the Schedule button at the top of the work area and select Set Schedule Trigger…





The Schedule Trigger dialog opens, check the Trigger channel(s), click the Pulse Count radio button, and enter the terminal count value in the input field. The example above sets the trigger for Schedule A as a count of 150 on low speed counter 1 (digital input channel 1).

If more than 1 counter channel is used as a trigger, then these must be consecutive channels.

DeLogger does not support the High Speed Counters being used as events for triggering Schedules.

Halting and Resuming Schedules Triggered by Counter Events

Schedules Triggered by Counter Events can be individually and globally halted and resumed by the Halt and Go commands

H	Halt all Triggered Schedules
G	Resume all Triggered Schedules
HA	Halt Triggered Schedule A
GA	Resume Triggered Schedule A
HB	Halt Triggered Schedule B
GB	Resume Triggered Schedule B
HC	Halt Triggered Schedule C
GC	Resume Triggered Schedule C
HD	Halt Triggered Schedule D
GD	Resume Triggered Schedule D

When the *dataTaker* is initially powered up or executes the RESET command, all Triggered Schedules are in the halted state. This allows the *dataTaker* to sleep.

When Triggered Schedules are entered, they are automatically enabled at the time of entry which allows the Schedules to run at the next trigger.

Some Application Ideas

When a number of counter events are used to trigger a Schedule, the counter which initiated each scan can be recorded by including the counter channels in the schedule list as is illustrated in the following program

```
BEGIN
 RA1..4C(30)
  1..4C(R)
  1..5TK 6V 10R
END
```

The trigger counters that can be used to trigger Schedules share the same digital inputs as the normal data counters, however the two counter types are not related internally. Therefore clearing of a trigger counter when the terminal count is reached does not clear the normal data counter that could be being used for the same digital input channel.

Statistical Sub Schedule

The *dataTaker* supports statistical sampling, which allows for the periodic average, standard deviation, minimum, maximum, and integral data to be generated for the channels and channel variables in a schedule list.

The date and time at which the minimum and maximum data occurs can also be generated.

The statistical data can be returned to the host computer, logged into internal data memory or a memory card, and displayed at the next trigger of an associated Triggered Schedule.

The Statistical Sub Schedule does not report data to any destination. Further, the Immediate Schedule cannot be used to return data from the Statistical Sub Schedule.

The Statistical Sub Schedule runs and executes the schedule list at specified intervals, and the data is retained internally for calculation of average, standard deviation, minimum, maximum and integral.

Each time the Statistical Schedule runs, the channels and channel variables are read, and the data is managed internally as follows

the data is summed for later calculation of the average and standard deviation

the square of the data is summed for later calculation of the standard deviation

a scan count is maintained for later calculation of the average and standard deviation

the progressive minimum data, and the date and time of minimum is updated

the progressive maximum data, and the date and the time of maximum is updated

the progressive integral is maintained, using trapezoidal approximation

Note : Only those procedures are carried out for which statistical functions are required (see below).

At the next associated Triggered Schedule interval, the average and standard deviation of the data for each channel are computed. The average, standard deviation, minimum, maximum, and integral data are then returned, logged and displayed.

The statistical data for linearized channel data (RTDs, monolithic, thermocouples, spans, polynomials and functions) is calculated after the linearization.

Applications for average, standard deviation, minimum, maximum and integral include

averaging of analog input channels reduces data, smooths signal noise, smooths inherently fluctuating signals, etc.

averaging of digital input channels returns the proportion of the total number of scans for which the digital input channel was high or true. This is in essence a measure of duty cycle.

averaging of resetting counters produces the arithmetic mean count

averaging of date and time produces the midpoint time for the Statistical Schedule interval.

standard deviation data for all channel types provides indication of the variability of the inputs

minimum and maximum data for all channel types provides an indication of the range of the inputs

minimum and maximum of date and time produces the start and end date and time for the Statistical Scheduling period

Statistical Sub Schedule Command

The Statistical Sub Schedule specifies the trigger at which channels requiring statistical functions are sampled. Only one Statistical Sub Schedule can be specified, and it applies to all Triggered Schedules which contain channels and channel variables which require statistical functions.

A schedule list is not associated with the Statistical Sub-Schedule as in the case for Triggered Schedules. Instead the channels sampled by the Statistical Sub Schedule are defined in other Triggered Schedules, and have statistical channel options declared.

The Statistical Sub-Schedule can be triggered in the same way as the Triggered Schedules as follows

RS	continuous statistical sampling (default)
RS*n*E	triggered by *n* digital event
RS*n*C	triggered by *n* low speed counter event
RS*n*HSC	triggered by *n* high speed counter event
RS*n*S	triggered by *n* seconds
RS*n*M	triggered by *n* minutes
RS*n*H	triggered by *n* hours
RS*n*D	triggered by *n* days

Refer to the preceding sections where the various trigger mechanisms for Triggered Schedules are described, for details of use of these triggers here.

The Statistical Sub Schedule trigger defaults to continuous sampling, and so a trigger does not need to be declared unless a different trigger is required. This allows statistical scanning to commence when statistical channel options are included for channels in the Triggered Schedule lists.

The Statistical Sub Schedule trigger can be changed at any time.

The Statistical Sub Schedule trigger interval must be shorter than the associated Triggered Schedule's trigger interval. If this is not the case then statistical data of 99999.9 will result, and the error message 'E53 - no statistical samples' will be returned to the host.

This can occur inadvertently when the Statistical Sub Schedule trigger is an event, or the Statistical Sub Schedule has been halted.

Note : The Statistical Sub Schedule trigger defaults to continuous scanning, and so the *dataTaker* will not power down to the low power mode. If low power operation is required, then a longer Statistical Sub Schedule interval must be specified.

Specifying Channels for Statistical Functions

Statistical processing of the channel data is specified by including appropriate channel options for the channels in schedule list*s* associated with Triggered Schedules. The channel options for each of the statistical functions are listed in the table overleaf (See also Section III – Channel Options).

Average

AV – Returns the average reading for the period, which is the sum of all the channel readings divided by the number of readings. Averaging is useful for reducing noise in channel data, and reducing the volume of data collected.

The average is returned in the units of the original data, suffixed with (Ave).

Standard Deviation

SD – Returns the standard deviation of the readings for the period, which is a measure of the variability of the data about the average.

The standard deviation is returned in the units of the original data, suffixed with (SD).

Statistical Function	*Channel Option*	*Units Text*
Average	AV	(Ave)
Standard Deviation	SD	(SD)
Minimum	MN	(Min)
Maximum	MX	(Max)
Date of Minimum	DMN	(Dmn)
Date of Maximum	DMX	(Dmx)
Time of Minimum	TMN	(Tmn)
Time of Maximum	TMX	(Tmx)
Integral	INT	(Int)

Minimum & Maximum

MN & MX – Returns the minimum and the maximum readings for the period, which are a measure of the range of data about the average.

The minimum and the maximum are returned in the units of the original data, suffixed with (Min) or (Max).

Date & Time of Minimum & Maximum

DMN, DMX, TMN, TMX – Returns the date and the time at which the minimum and maximum values occurred. Any combination of these channel options can be used to return data as required.

The date of minimum and maximum data can be returned as the day number or date. This is determined by the setting of Parameter31 (See Section III – The Real Time Clock).

The time of minimum and maximum data can be returned as time in HH:MM:SS, decimal hours or seconds format. This is determined by the setting of Parameter39 (See Section III – The Real Time Clock)

The date and time returned is suffixed with (Dmn), (Dmx), (Tmn) or (Tmx) respectively.

Integral

INT – Returns the integral of the readings for the period. The integral calculation uses a trapezoidal approximation. Integration provides a means of converting rate measurements to quantity measurements (eg. flow rate to volume, velocity to distance, etc.). The integral is calculated with respect to the time period in seconds. Other time bases can be used by applying appropriately defined polynomials.

The integral is returned in the units of the original data, suffixed with (Int). Integral units text are not derived by the logger.

For example a flow meter measuring flow in litres/sec is integrated, the units text for the integral will be litres/sec (Int), and no simply litres.

Examples of Programming the Statistical Functions

Only one statistical sampling rate can be defined in an application, Therefore the Statistical Sub Schedule trigger must be chosen to provide adequate sampling for channels in those Schedules that are running most frequently.

Some programming examples follow which illustrate the use of the Statistical Sub Schedule and statistical functions of the *dataTaker.* These example programs can all entered into the logger using DeTransfer

```
BEGIN
 RA10S
  3TK(AV,"Temperature Ave")
END
```

instructs the *dataTaker* to calculate the average temperature every 10 seconds for a Type K thermocouple connected to analog channel 3. Because there is no Statistical Sub-Schedule trigger specified in the program, the Statistical Sub-Schedule trigger defaults to the maximum sampling rate.

```
BEGIN
 RA5M
  3PT385(AV)(MN)(MX)(TMN)(TMX)
END
```

instructs the *dataTaker* to calculate every 5 minutes the average, minimum, maximum, time of minimum and time of maximum of temperature for a platinum RTD connected to channel 3. Since a Statistical Sub Schedule trigger is not specified, the channel is sampled at the default maximum rate.

Note where more than one Statistical function is required, then these are entered with each in a separate pair of braces. Any other channel options appropriate to the channel management are only entered in the first pair of braces.

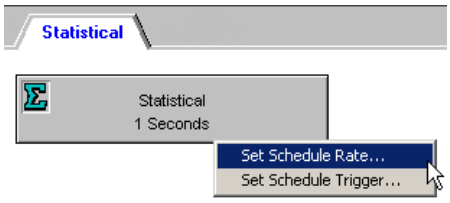
```
BEGIN
 RA5M
 RS1S
  5V(AV)(MN)   6..8TK(AV)(MX)(MN)
END
```

instructs the *dataTaker* to read the input channels every 1 second (RS1S), and return the statistical data every 5 minutes (RA5M).

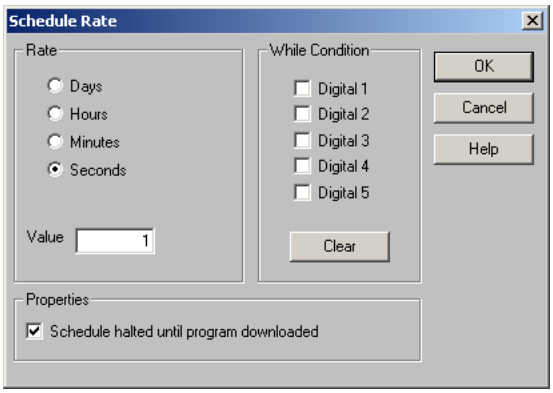
```
BEGIN
 RA5M
  5V(AV)   6..10TK(MX)   1HSC(AV)
 RS1+E
END
```

instructs the *dataTaker* to read the input channels every time a positive external event occurs on digital input channel 1, and return the statistical data every 5 minutes. Note that the RS schedule command can be anywhere between BEGIN and END.

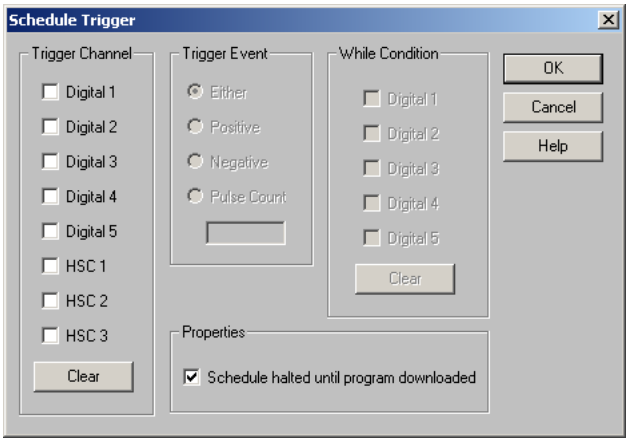
DeLogger fully supports the Statistical Sub Schedule and statistical functions of the *dataTaker*. To set the trigger for the Statistical Sub Schedule click on the Statistical tab in the Program Builder, right click on the Statistical button, then select the trigger type. This opens the appropriate dialog where the trigger is set in terms of a time interval, a digital event or a counter event.



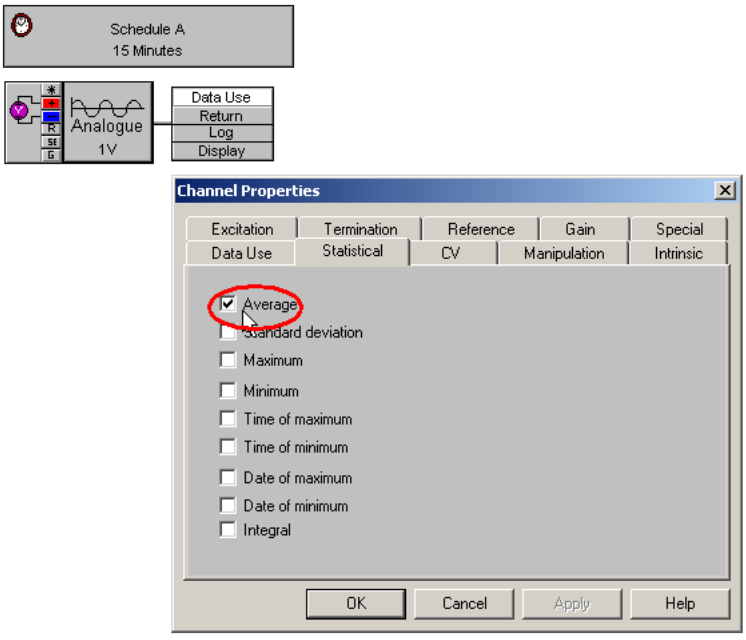
In the Schedule Rate dialog, set the time trigger by selecting a time unit (seconds, minutes, hours or days), and entering the number of time units.



In the Schedule Trigger dialog, select the digital or counter channel for the trigger, and select the type of trigger event.



The channels for which the statistics are required are tagged with the appropriate channel options in the Data Schedules A through D.



Define the channel in the normal way, then right click on the Data Use icon, and select Properties... to open the Channel Properties dialog. Click on the Statistical tab, and click in the check box(s) for the statistical functions required.

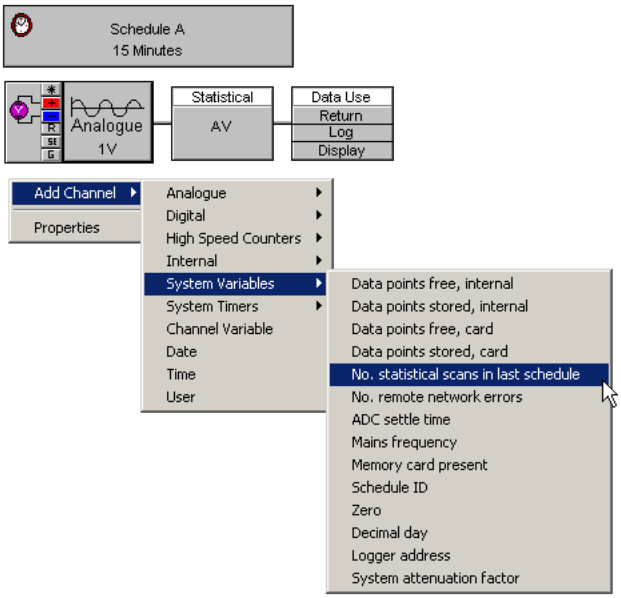
Number of Statistical Samples

The number of samples taken to generate Statistical data can be returned as system variable 5SV (See Section III - System Variables), which is included in the schedule list as follows

```
BEGIN
 RA10M
  5SV
  1..10V(AV)(MN)(MX)
 RS1M
END
```

In this example 5SV will return 10, because the Schedule A is triggered every 10 minutes (RA10M) and a Statistical Sub Schedule is triggered every 1 minute (RS1M).

In DeLogger, the number of statistical samples can be selected in the Program Builder. Right click in the work area to Add Channel, select System Variables, and select No.of statistical scans in last schedule.



Returning the number of samples is more appropriate to situations where the Triggered Schedule and/or the Statistical Sub Schedule are triggered by external events.

Limit to the Number of Statistical Scans

The Statistical Sub Schedule must not run more than 65535 times during the interval of the embracing Schedule which returns the statistical results. This limitation only applies when average and standard deviation data is required.

For example in the program

```
BEGIN
 RA1D
  1..10V(AV)(SD)(MN)(MX)
 RS1S
END
```

the Statistical Sub Schedule will run 86,400 times between triggers for Schedule A. This will produce erroneous average and standard deviation data, but will not affect the minimum or maximum data.

Statistical Sub Schedules within Multiple Triggered Schedules

Only one Statistical Sub-Schedule can be specified in a program. When two or more Schedules that contain channels for statistical functions are entered, all channels in all Schedules will be read at the same Statistical Sub Schedule interval.

Format of Returned Statistical Data

The format of statistical data returned to the host is the same as for normal data, and is fully user configurable (See Section III - Format of Returned Data). Statistical data has the text for the statistical type suffixed to the units text.

If the Units Text Switch (/U) is enabled , then returned statistical data is labelled with the statistical type.

The following program with statistical functions

```
BEGIN
 RA10M
  1TK(AV)(SD)(MN)(TMN)(MX)(TMX)
RS15S
```

will return data similar to

```
1TK  25.6 Deg C (Ave)
1TK  0.160 Deg C (SD)
1TK  22.1 Deg C (Min)
Time  11:12:15 (Tmn)
1TK  27.8 Deg C (Max)
Time  11:14:30 (Tmx)
```

Format of Displayed Statistical Data

Statistical data is displayed in the same format as for returned data. However channel identification text can be added to each statistical function to improve readability of the returned data in the display.

For example

```
BEGIN
 RA10M
  1TK(AV,"Ave Temp")(MX,"Max Temp")(MN,"Min Temp")
 RS15S
END
```

will display statistical data as follows

```
Ave Temp
25.6 Deg C (Ave)

Max Temp
27.8 Deg C (Max)

Min Temp
22.1 Deg C (Min)
```

The display of statistical data is usually updated when the main Schedule executes. However the display of minimum and maximum data can be updated each time that the Statistical Sub Schedule executes.

The displaying of minimum and maximum data is controlled by the Min Max Display Switch as follows

/x	Update minimum and maximum data when Schedule executes (Default)
/X	Update minimum and maximum data when Statistical Sub Schedule executes

Following initial power up, a hardware reset, or a RESET command, minimum and maximum updates when the Schedule executes.

Halting and Resuming the Statistical Sub Schedule

The Statistical Sub Schedule can be globally and individually halted and resumed by the Halt and Go commands as follows

H	Halt Data and Statistical Sub Schedules
G	Resume Data and Statistical Sub-Schedules
HS	Halt Statistical Sub-Schedule
GS	Resume Statistical Sub-Schedule

When *dataTaker* is initially powered up or executes the RESET command, the Statistical Sub Schedule defaults to halted. This allows the *dataTaker* to sleep.

When Schedule containing channels for statistical functions are entered, then Statistical Sub Schedule is automatically enabled at the time of entry. This allows the scanning to proceed at the next scan trigger.

The Schedule associated with a Statistical Sub-Schedule can be halted, without halting the Statistical Sub-Schedule.

Halting the Statistical Sub-Schedule does not halt the associated Repeating Schedules. However channels specified for statistical functions will only be read at the Schedule interval, and raw data will be returned.

Schedules Triggered While Condition

The execution of Data Schedules, the Statistical Sub Schedule and the Alarm Schedule can be made conditional on the status of one or more digital input channels.

The conditional execution of Schedules is implemented by the Schedules Triggered While command which is used in conjunction with the Triggered Schedules, Statistical Sub Schedule and Alarm Schedule.

The Schedules Triggered While Condition enables the Schedule while nominated digital input channels are high or true, and disables the Schedule while the digital input channels are low or false.

The digital input channels of the Channel Expansion Module cannot be used for this purpose.

Schedules Triggered While Condition

The general format for the Schedules Triggered While Condition are extensions to the triggers for the Schedules, Statistical Sub Schedule, and Alarm Schedule as follows

R(trigger):nW [schedule list]

R(trigger):n..mW [schedule list]

RS(trigger):nW

RS(trigger):n..mW

RZ(trigger):nW

RZ(trigger):n..mW

where

R	is the Triggered Schedule
RS	is the Statistical Sub Schedule
RZ	is the Alarm Schedule
:W	is the Trigger While
n	is any digital input channel
n..m	is a sequence of digital input channels
(*trigger*)	is the Schedule trigger or interval
[*schedule list*]	list of the channels to be read, expressions to be evaluated, etc

When a sequence of conditional channels is specified, then the While is true and scanning will occur if any of the digital input channels is high or true. The While channel sequence therefore has a logical OR characteristic.

Using DeTransfer, Schedules Triggered While Condition are programmed as follows for example

```
BEGIN
 RA5M:1W
  1..5V 7P 2C(R)
END
```

where the schedule list runs every 5 minutes while digital input channel 1 is high or true.

```
BEGIN
 RAS1M:2W
 R10M:1W
  1..5V(AV)
END
```

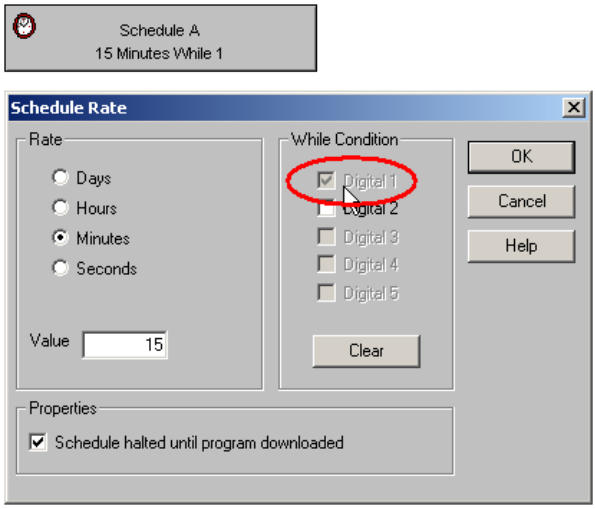
where the schedule list is executed for averaging every minute while digital input channel 2 is high, and the averaged data returned every 10 minutes while digital input channel 1 is high.

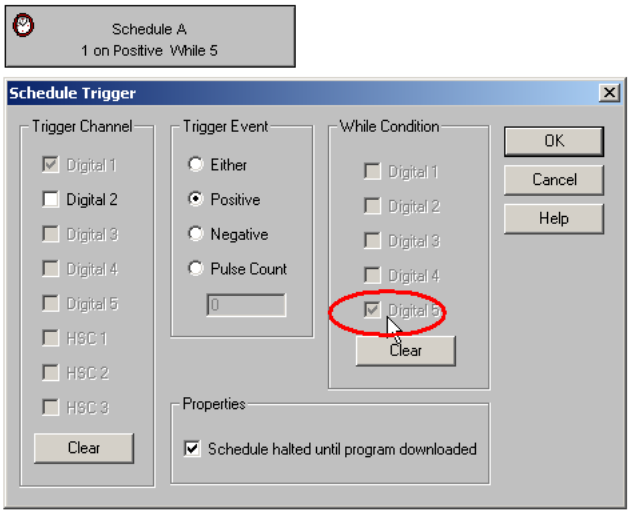
```
BEGIN
 RA1H:1..3W
  1..5V 7P 2C(R)
END
```

where the schedule list is executed every hour while any of the digital input channels 1, 2 OR 3 are true or high.

Note : The digital channels used for conditional scanning are bi-directional. These channels cannot be used for conditional scanning if they have been used as digital outputs, and left ON. The digital channels can only be used for conditional scanning if their output state is OFF.

Using DeLogger, Schedules Triggered While Condition are programmed in the Program Builder. Firstly define the schedule trigger, then in the While Condition field click the digital input channel(s) which are the trigger conditions.





General Schedule Commands

There are a number of general characteristics of the Schedules such as

- Entering multiple Schedules
- Changing triggers for an existing Schedule
- Halting and resuming Schedules
- Combining Schedules and Alarms for specific effects
- Altering the way in which the Schedules operate

Multiple Scan Schedules

The *dataTaker* can support up to five simultaneous Data Schedules, each scanning different schedule list*s* in response to different triggers of time, digital event or counter event.

The multiple Schedules can be entered in a single line format, or in a block format.

Single Line Format (DeTransfer)

The simple method of entering more than one Schedule is to enter all schedules in one command line. This is illustrated in the following command

```
RX 1V R10M 1..3V R1H 4..5TN R12H 7R R1E 1C
```

This example has five Schedules

a voltage on each poll from the host (RX 1V)

five voltages every 10 minutes (R10M 1..3V)

two thermocouples every hour (R1H 4..5TN)

a resistance every 12 hours (R12H 7R)

a resetting counter every time an event occurs (R1E 1C).

The five Schedules will function simultaneously, executing the respective schedule lists at respective intervals.

Entering multiple Schedules in a single command line however is limited by the maximum allowable command length of 250 characters. In the above example, the Schedules with triggers are not separately identified. However the *dataTaker* identifies the schedules as Schedules RA, RB, RC and RD from left to right.

The Schedule Triggered by Poll (RX) is permanently identified, where the X indicates a host request scan trigger.

The example above could have been entered from DeTransfer as

```
RX 1V RA10M 1..3V RB1H 4..5TN RC12H 7R RD1E 1C
```

The Schedule identifiers RA, RB, RC and RD can be used for subsequent re-scheduling, Halting and Going of the individual schedules.

Block Format BEGIN ... END (DeTransfer)

In practice most multiple schedule programs will extend over several command lines. However if these multiple Schedules are entered in single line format in successive command lines, each overwrites or replaces the last.

A multiple line command block or program format is used to enter multiple Schedules, which also improves readability of the application program.

DeTransfer provides for sending both single line commands (Send Line) and multiple line commands (Send All). Refer to the relevant documentation with these programs for details.

All the commands required to be entered which are distributed over more than one line must be placed between the keywords BEGIN and END.

Features and rules of the BEGIN ... END construct are listed below

- each line between can be up to 250 characters long, and must be terminated by a carriage return
- ' text preceded by an apostrophe can be entered as comments
- blank lines can be used to improve layout
- commands which are executed on entry can be included
- Schedules can be placed on lines without schedule lists
- channels can be placed on successive lines without a schedule header, are included in the previous schedule. A carriage return must be between channel definitions or schedule headers.
- when a BEGIN command is received, all schedules (including the alarm schedule RZ) are Halted, and existing RA, RB, RC, RD , RS and RX schedules are deleted.

  The exception is when data logging is enabled, the schedules are fixed (/F), or there is data logged into memory. In these cases the BEGIN command is rejected, and an error message is returned.
- when the END command is received , the original Halt - Go state of each Schedule is restored.
- channels cannot be appended to schedules after the END command is issued. The full set of schedules must be re-entered.
- Alarms, data format commands, system commands, etc. which are not a part of Schedules can also be placed between BEGIN ... END, although this is not mandatory

An example of a program using the BEGIN ... END construct in DeTransfer is shown below

```
BEGIN
      RA10S
            1TK("Oven Temp",FF1)
            2TK("Flue Temp",FF1)
            3..6TK("Product Temps",FF0)
      RB1S
            1C(S1,"Gas Flow")
            2C(S2,"Electricity")

END
```

where

- on receipt of the BEGIN command all schedules are Halted and existing schedules deleted
- the Repeat Schedule A comprises the six thermocouple channels
- the Repeat Schedule B comprises the counter channels
- on receipt of the END command the original settings for Halt - Go are restored
- note that tabs, spaces and blank lines can be used to improve layout and readability

Documenting Programs

Comments can be used in DeTransfer programs to describe the application, the intention of particular steps, etc.

The rules for comments in *dataTaker* programs are as follows

a comment is prefixed by an apostrophe (') character, and all text up to the next carriage return is treated as the comment

comments can be placed on separate lines

comments can be placed after commands on the same line

comments cannot be placed before commands on the same line

Examples of the use of comments in a program is illustrated below

```
'Boiler monitoring program for the dataTaker
'Author: Henry Higgins
'Created                  23/4/2003
'Last Modified     25/4/2003

CSCANS
CALARMS
CDATA

'Switch units text and channel numbers off
'Switch return of Time and Date on
     /u/n/T/D

'Define spans for Gas & Electricity
S1=0,50,0,100"M^3"
S2=0,20,0,1000"KW"

BEGIN              'Begin program
'Repeat Schedule A records temperatures
     RA10S
            4TT("Oven Temp",FF1)
            5TK("Flue Temp",FF1)
            3..6TK("Product Temps",FF0)

'Repeat Schedule B records Gas & Electricity
     RB1S
            1C(S1,"Gas Flow")
            2C(S2,"Electricity")

LOGON       'Log all channel to memory

END
```

Commands other than Schedule commands can be included between the BEGIN … END commands, or can be placed outside of the BEGIN … END commands. These include

Switches commands

Parameter commands

Span and Polynomial declarations

Alarm commands

System commands

Data Logging commands

Date and Time setting commands

Order of Data Returned from Multiple Scan Schedules

Whenever two or more Schedules are entered into the *dataTaker*, and these are simultaneously executing their respective, schedule lists, the data is returned to the host computer in the order

Immediate Scan Schedule	
Repeating on Host Request	RX
Repeating Schedule A	RA
Repeating Schedule B	RB
Repeating Schedule C	RC
Repeating Schedule D	RD

This ordered return of data occurs irrespective of the order in which the Schedules were originally entered into the *dataTaker*.

If the Units Text Switch is enabled (/U) then a CR/LF character sequence is returned at the end of the block of data from each Scan Schedule.

If the Units Text Switch is disabled (/u) the block delimiter character defined by the Parameter24 Schedule is returned at the end of the block of data from each Scan Schedule.

Halting and Resuming Schedules

The Schedules can either be globally or individually halted and resumed by the Halt and Go commands

H	Halt all Data Schedules
G	Resume all Data Schedules
HA	Halt Triggered Schedule A
GA	Resume Triggered Schedule A
HB	Halt Triggered Schedule B
GB	Resume Triggered Schedule B
HC	Halt Triggered Schedule C
GC	Resume Triggered Schedule C
HD	Halt Triggered Schedule D
GD	Resume Triggered Schedule D
HS	Halt Statistical Sub-Schedule
GS	Resume Statistical Sub-Schedule

When the *dataTaker* is initially powered up or executes the RESET command, all Schedules default to halted. This allows the *dataTaker* to sleep.

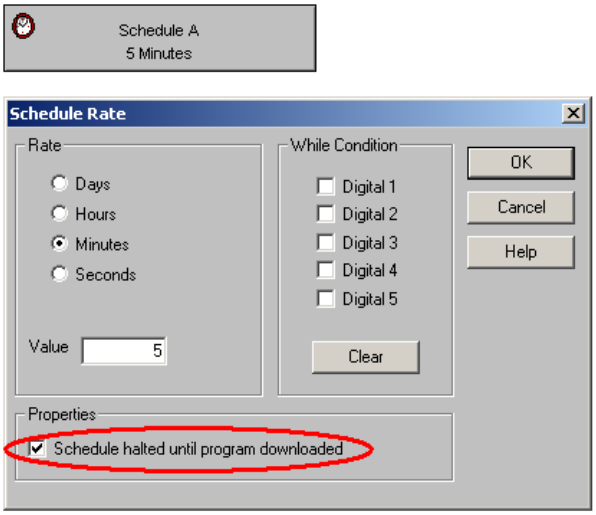
When Triggered Schedules are entered, they are automatically enabled at the time of entry which allows the Schedules to run at the next trigger.

Using DeTransfer, if a newly entered Schedule should not be enabled immediately, then a Halt command can be included as follows to hold the Schedule

```
BEGIN
 RAB10M  HB
  1..5V 8TK 1..2DS
END
```

A GA command is entered later when the Schedule is to be enabled.

Using DeLogger, Schedules can be initially held by checking the 'Schedule halted until program downloaded' in the Schedule dialog of the Program Builder.



Combining Schedules

Schedules can be combined in various ways to provide greater flexibility to data acquisition tasks.

For example the data acquisition program

```
BEGIN
 RX              'Inspection
  1..6PT385
 RA1H            'Data collection
  1..6PT385
END
```

provides the flexibility of channel inspection where

the temperature channels will be read every hour according to the Schedule RA1H

the current temperature readings can be inspected at any time by entering a Poll Trigger command X

Re-Triggering Schedules

The triggers for the Schedules and Statistical Sub Schedule can be changed at any time by entering new Schedule commands without a schedule list.

Schedules with time triggers can be rescheduled to different time intervals, or can be rescheduled to digital or counter event triggering.

This feature is useful to test or verify a data acquisition system and sensor setup, preparatory to commencing any long interval or event initiated data collection.

For example, during a data acquisition system installation and test phase, the following Schedules could be entered from DeTransfer

```
BEGIN
 RS5S
 RA1M
  1..5TK(AV)(MN)(MX)
 RB1M
  6..8V(AV)(MN)(MX)
END
```

where a short Schedule trigger of 1 minute is used.

When all adjustments are complete, the Schedules are rescheduled by the following commands to set the intervals appropriate to the requirements of the task.

```
RS5M  RA1H  RB3+E  LOGON
```

Rescheduling of running Schedules is not supported in DeLogger.

Synchronizing Schedule Time Triggers to Last Midnight

Time triggers for Schedules can be set to one of two real time modes

trigger intervals are relative to the time of entry of the Schedule

trigger intervals are synchronized to last midnight (Default)

The default mode is that Schedule time triggers are synchronized to midnight, and the schedule list is executed on every multiple of the interval since last midnight. For example a Schedule entered as follows

```
BEGIN
 RA1H
  1V 2R 3F 4TT
END
```

first executes the schedule list on the next hour since last midnight, and every hour thereafter.

Synchronization of Schedule time triggers is defined by the Synchronization Switch as follows

/S	Synchronize scans to midnight (Default)
/s	Disable synchronization of scan intervals

The Synchronization Switch defaults to /S after initial power up, or execution of a RESET command.

An unsynchronised Schedule is entered as follows

```
BEGIN
 /s
 RA10M
  1V 2R 3F 4TT
END
```

In this case the Schedule is first executed 10 minutes after entry, and every 10th minute thereafter.

Synchronized time triggers have various effects depending on the magnitude of the interval

when the synchronized interval is a multiple of 1 hour (eg. 30S, 10M, 20M but not 35M), then the intervals will also synchronize with each hour of the day

if a synchronized interval is not a multiple of 24 hours (eg. 50M, 7H, 13H), then the last interval before midnight will be less than the specified interval.
This enables the next interval to synchronize to 00:00:00 time at midnight.

Checking Entered Schedules

Schedules can be checked at any time by the STATUS command, which returns a summary of system status including a summary of the Schedules.

The system status is returned by the command

STATUS

which produces the response (when /U/N are enabled)

```
dataTaker 0 Version 7.00
A,none Scan Schedules Active, Halted
3,0 Alarms Active,Halted
3 Polynomial/Spans Defined
Logging is OFF
166530,0 Internal Data Points Free,Stored
81900,0 Card Data Points Free,Stored
4090, 0 Program Characters Free,Stored
/a/C/d/E/f/h/J/K/l/M/N/o/Q/R/S/t/U/v/w/x/y/Z
```

Specific details of the current Schedules is returned by the STATUS2 command

STATUS2

which returns a listing of the current Scan Schedules as follows

```
A B,D Scan Schedules Active,Halted
RA5M 1..4V(AV)(MN)(MX) 6..10TT
RB1H 1..4DS 1..4C
RD15M 5L(S2)
```

In this example there are three Schedules which have been entered into the *dataTaker*.

Two of these Schedules are currently enabled to execute (Active), and one Schedule has been paused by a Halt command (Halted).

Clearing Schedules from the Internal Channel Table

Whenever Schedules and schedule lists are entered into the *dataTaker*, the channels to be scanned are entered into an internal channel table. The internal channel table is limited to 110 entries, and is shared with the Alarms (See Section III – Alarms).

When the *dataTaker* is initially powered, hardware reset or executes a RESET command, the internal channel table initialises to 20 entries for Alarms and 90 for Data Schedules.

The allocation of the internal channel table between the Data Schedules and Alarms can be changed to suit particular needs (See Section III · Number of Channels for Schedules).

If any channels are already entered in the alarm or data acquisition sections of the channel table, then the channel table allocation cannot be changed. The channel table must firstly be cleared.

The section of the internal channel table that is allocated to the Data Schedules can be cleared by the command from DeTransfer

```
CSCANS
```

This command will cause any current Schedules and execution to be aborted, and can also be used to stop all data acquisition tasks.

The CSCANS command is ignored if the Fix Channel Table Switch is enabled /F or if data is stored in the internal data memory.

The section of the internal channel table that allocated to the Alarms is cleared by the

```
CALARMS
```

command (See Section III – Clearing Alarms from the Internal Channel Table).

These commands are not supported by DeLogger.

Priorities for Returning and Logging Data

When data is simultaneously being logged to the internal data memory or a memory card, and is being returned to the host computer, the logging of data has priority over returning of data.

Following each scan of input channels, data is logged immediately, and then returned to the host if time permits. However if the next Schedule trigger occurs then returning of data is aborted and the new scan commences. If scanning is rapid, then there may be no time available to return real time data to the host computer.

Alternatively if the data is not being logged to the internal data memory or a memory card, then all data will be returned to the host computer. If the next Schedule trigger occurs before all data from the previous scan is returned, then the new scan trigger is ignored.

However this priority can be reversed if required, using the Data Priority Switch as follows

/Y	Return all data to host when data logging is enabled, ignoring new triggers
/y	Return data to the host when data logging is enabled only if time permits

The Data Priority Switch defaults to disabled (/y) following an initial power up, or execution of a RESET command.

Preventing Schedules from Waking the *dataTaker*

When the channels are being read or alarms are being processed at intervals of less than 2 seconds, then the *dataTaker* will not enter the low power mode or sleep mode.

Conversely whenever channels are being scanned or alarms are being processed at an interval of more than 2 seconds, then the logger will sleep between successive scans if other functions are not keeping it awake (if logger is battery powered, or Parameter15 is set to 1).

The *dataTaker* can be programmed to ignore Schedule triggers, and to sleep and wake according to specified schedules only.

The Schedules which can wake the *dataTaker* are defined by setting a bit mask using the Parameter20 command as follows

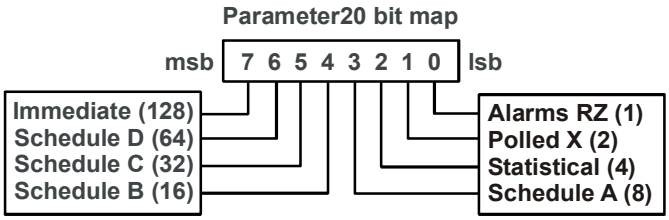


Figure 127 – Parameter20 Bit Map

Whenever the *dataTaker* is initially powered up, is hardware reset or executes a RESET command, Parameter20 defaults to P20=0 which allows all schedules will wake the logger when they become due.

The use of the Parameter20 command is illustrated by the following example

```
P20=8
BEGIN
 RA30M
  7..10TT
 RB30M
  1..5V
END
```

With Parameter20=8, the *dataTaker* sleeps and wakes according to the Schedule A, and only executes Schedule B while awake to executes Schedule A.

Any combination of Data Schedules and the Alarm Schedule can be disabled from waking the *dataTaker*.

Alarms

The Alarms of the *dataTaker* allow the data from various sources to be compared with range setpoints, and to perform actions if the data exceeds these setpoints.

The different types of data which can be tested by Alarms includes

Analog input, digital input and counter channel data

Internal channel data

Day, Date and Time of the real time clock

System Timers

Channel Variables

System Variables

The data from these different sources can be compared against one or two setpoints, which define the alarm conditions. The four different types of comparison which can be performed are

data less than a setpoint (low alarm)

data greater than or equal to a setpoint (high alarm)

data within the range of two setpoints (in range alarm)

data outside the range of two setpoints (out of range alarm)

Note : There is no test condition for equality of data to setpoint, since in a data acquisition system such as the *dataTaker* equality is only a brief state if it occurs at all.

The action taken by the *dataTaker* in response to alarm states be to

return a message text string to the host, and display the message on the display panel

switch digital output channels, or switch the warning LEDs, beeper and backlight of the display panel

execute a command or program. This allows the *dataTaker* to reprogram itself to operate differently when an alarm occurs

Whenever an alarm state occurs the alarm action can be performed either immediately, or after a defined period of time has elapsed for which the alarm state remains true. This allows short term nuisance alarm states to be ignored.

Alarms can be singular, or can be linked together using AND, OR and XOR logical operators to monitor complex conditions involving more than one input channel or source of data.

The Alarms are normally used for true alarms which will annunciate out of range conditions.

However the Alarms can also be used as a programming device for controlling the operation of the *dataTaker*. For this reason the Alarms are also interchangeably referred to as the IFs.

Some examples of applications of the alarms include

generating alarms if data from an input channel exceeds a single setpoint or a setpoint range

controlling an external process to a setpoint (simple control)

generating alarms if internal data for System Variables, Channel Variables, etc. exceeds a single setpoint or a setpoint range

generating alarms before or after specified dates and/or times, or when System Timers exceed specified elapsed times

defining windows of time during which *dataTaker* operations such as channel scanning for data acquisition can operate

changing the operation of the *dataTaker* in response to particular conditions or events

When alarm input channels or system data is being monitored for range, this test data can be returned to the host computer for inspection. This overcomes the need for separate Data Schedules to inspect this data. However the test data returned from the alarms cannot be logged into the internal data memory or a memory card.

The alarm input channels and system channels can be considered to be scanned as a background function, in that other data acquisition functions can also be carried out whenever alarms are in operation.

Scanning of the alarm input channels is continuous by default to provide the maximum sensitivity. However scanning of the alarm input channels can be scheduled on the basis of time intervals or external events if required, in the same manner as the Triggered Schedules for data acquisition.

The *dataTaker* can support up to 110 simultaneous alarms, with the default being 20.

Number of Channels for Alarm Commands

When Alarms are entered into the *dataTaker*, the source of data to be tested is entered into an internal channel table.

The *dataTaker* treats all sources of data for Alarms in the same manner, irrespective of whether the source is a physical input channel or system data such as a Channel Variable, Day, Date, Time, System Timers, System Variable, etc.

If the source of data is a physical input channel, then the internal channel table contains details of the input channel to be scanned including the channel number, the type of input signal, the sensor calibration, how the resultant data is to be processed, etc.

If the source of data is system data, then the internal channel table contains all details of the type of data, location of the data, how the data is to be processed, etc.

The Alarms share this internal channel table with the Data Schedules, which also keep input channel details in the table for data acquisition (See Section III – Schedules).

A maximum of 110 entries can be made into this table by the Alarms and the Data Schedules.

Following initial power up, a hardware reset or execution of a RESET command, the default allocation of the internal channel table between the Alarms and the Data Schedules is as follows

20 entries are allocated for the Alarms

90 entries are allocated for the Data Schedules

Therefore a maximum of 20 Alarms can be entered into the *dataTaker* by default.

However allocation of the internal channel table between the Alarms and the Data Schedules can be changed to suit particular applications.

Setting the Number of Alarms

The allocation of entries in the internal channel table between the Alarms and the Data Schedules can be specified by the ***Parameter30*** command (See Section III – Parameter Commands).

This command defines the number of entries in the internal channel table which are allocated to Alarms. The balance of the entries are allocated to the Data Schedules.

Using DeTransfer, the command for example

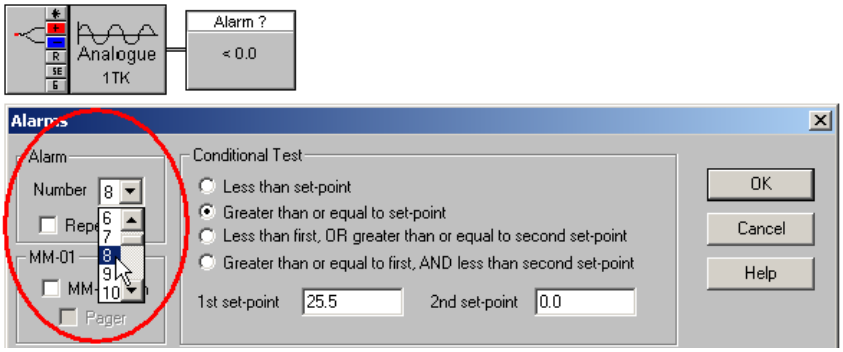
```
P30=50
```

allocates space for 50 entries in the internal channel table for Alarms, and reduces the number of entries in the table for Data Schedules to 60.

The number of the Alarm entries in the internal channel table can be set between 0 to 110, resulting in an allocation of 110 to 0 entries to the Data Schedules.

Note : The channel table must be re-allocated by the Parameter30 command before any Alarms or Data Schedules are entered. Attempts to re-allocate the internal channel table after either Alarms or Data Schedules have been entered will result in the error message 'E8–Parameter read/set error'.

If you are using DeLogger, then you do not need to set the number of alarms. DeLogger allows up to 60 alarms to be defined in a program, and automatically sets the Parameter30 command to the highest alarm number used in a program.



Always define alarms beginning at alarm number 1, rather than randomly using alarm numbers which will waste space in the channel table.

The internal channel table can be cleared using the CSCANS and CALARMS commands.

Alarm Keywords

The Alarms have two basic command keywords which can be used interchangeably.

The ALARM keyword can be used for those Alarms which are used in the traditional manner to test input channels or system data for out of range conditions, and perform actions to annunciate and alarm.

The IF keyword can be used for those Alarms which are used as a device to control the operation of the *dataTaker*, or used to control the flow of applications programs (See Section III – Writing Programs).

The Alarms can act in either of two modes when an alarm state occurs.

In the first mode, the Alarm performs the alarm action when the alarm state actually becomes true. The alarm state must then become false, and true again before the alarm action is performed again. This mode of the Alarms is specified by the two keywords ALARM and IF.

In the second mode, the Alarm performs the alarm action repeatedly while the alarm state is true. The repetition of the alarm action is determined by the Alarm Schedule (see below). This mode of the Alarms is specified by the two keywords ALARMR and IFR.

Entering Alarms

There are four general command formats for Alarms, allowing input channels or system data to be tested in four different ways as follows

input channel or system data greater than or equal to a setpoint

ALARMn(testdata>setpoint1/delay)action

ALARMRn(testdata>setpoint1/delay)action

or

IFn(testdata>setpoint1/delay)action

IFRn(testdata>setpoint1/delay)action

input channel or system data less than a setpoint

ALARMn(testdata<setpoint1 / delay)action

ALARMRn(testdata**<**setpoint1 / delay)action

or

IFn(testdata<setpoint1 / delay)action

IFRn(testdata<setpoint1 / delay)action

input channel or system data outside the range of two setpoints

ALARMn(testdata<>setpoint1,setpoint2 / delay)action

ALARMRn(testdata<>setpoint1,setpoint2 / delay)action

or

IFn(testdata<>setpoint1,setpoint2 / delay)action

IFRn(testdata**<>**setpoint1,setpoint2 / delay)action

input channel or system data between two setpoints

ALARMn(testdata><setpoint1,setpoint2 / delay)action

ALARMRn(testdata><setpoint1,setpoint2 / delay)action

or

IFn(testdata><setpoint1,setpoint2 / delay)action

IFRn(testdata><setpoint1,setpoint2 / delay)action

where

ALARM)	
ALARMR)	[aa]are the Alarm
IF)	[aa]identifiers
IFR)[a]	
n	is the alarm number
testdata	is the input channel or system data that is tested for alarm state
>	operator for testdata greater or equal to setpoint
<	operator for testdata less than setpoint
<>	operator for testdata outside setpoints
><	operator for testdata between setpoints
setpoint1	alarm setpoint1
setpoint2	alarm setpoint2 (for <> and >< only)
/ delay	is optional time period the alarm must be true before alarm state is changed
action	specifies action(s) if alarm is true

Each of these elements of the Alarms are discussed in detail in the following sections.

The Alarm Number

Each Alarm has an alarm number in the range of 1 to the number of alarms defined by Parameter30. The alarm number has several functions

identifies the alarms when the input channel or system data being tested is returned to the host

defines the sequencing of alarms when alarms are combined by logical operators AND, OR, XOR

identifies individual alarms to be halted and resumed

identifies individual alarms to be cleared from the internal channel table

The number of alarms which can be entered in the *dataTaker* can be increased or decreased by the Parameter30 command (see Setting the Number of Alarms above). The default is for 20 alarms which are numbered 1 to 20.

The Alarm Test Data

All of the types of data produced by the *dataTaker* can be tested in the Alarms. The different data types fall into two main categories - input channel data, and internal or system data.

The input channels that can be monitored by Alarms include all of the *dataTaker* physical input channels

differential and single ended analog input channels

digital logic state and byte input channels

counter input channels

The internal system data sources that can be monitored by Alarms includes

internal reference channels and the battery status channels

the Date, Day and Time of the real time clock

the System Timers

the System Variables

the Channel Variables

The only internal data source that cannot be tested by the Alarm is the Text String channel.

The input channels to be tested are specified in the same manner as input channels for the Data Schedules. However only a single input channel can be specified, and channel sequences are not permitted.

Most of the channel options can be used for input channels in Alarms, including

channel configuration, gain and excitation

scale factors, Polynomials, Spans and Intrinsic Functions

assignment to Channel Variables

data formatting and output options

bar graph display

Channel options that cannot be used for input channels in Alarms include

the data manipulation functions DF, RC, RS, and IB

the statistical AVE, SD, MX, DMX, TMX, MN, DMN, TMN and INT

the histogram function

NL option to not log data

Alarms for several types of input are shown below, and can be entered using DeTransfer

```
ALARM1(1V>1000)"Alarm 1"

ALARMR2(1+V(X)<200.0)"Low Power"

IF4(T>12:00:00)"Lunch Time"

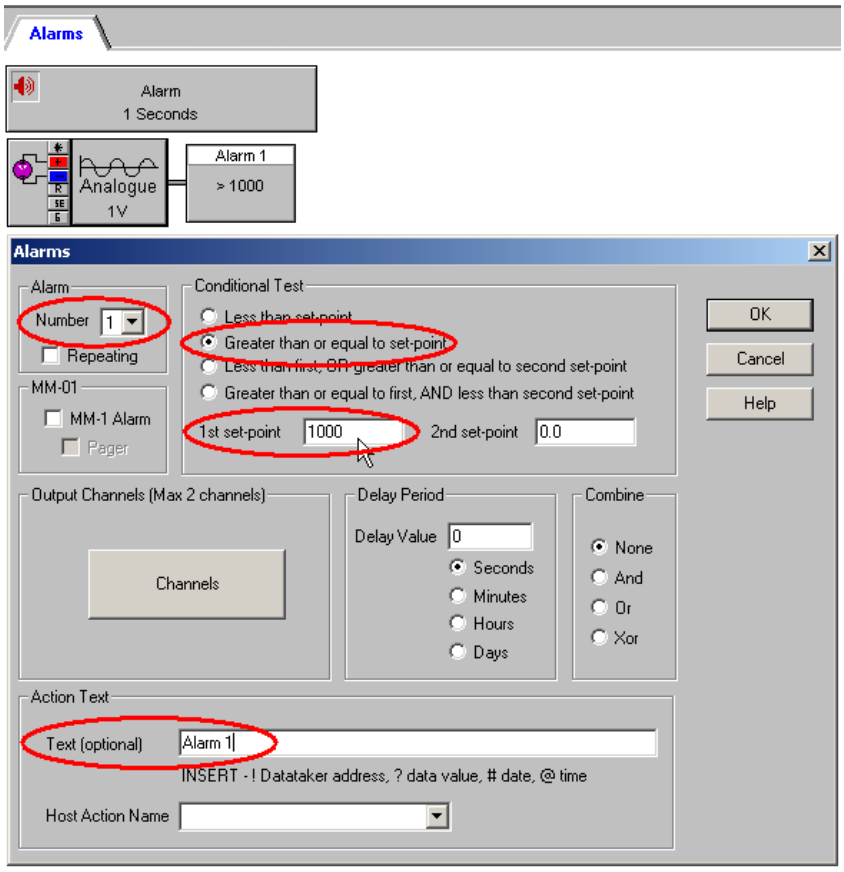
ALARM10(5L(120,S5)<>100,500)"Temp Alarm"

IFR15(3CV(S3,NR)><47.5,82.7)"Out of Range"
```

These Alarms respectively have the following actions

ALARM1	- returns the message to the host each time the voltage measured on analog input channel 1 exceeds 1000mV
ALARMR2	- returns the alarm message to the host continuously while the single ended voltage measured on analog input channel 1 remains less than 200mV
IF4	- returns the alarm message to the host once at midday
ALARM10	- returns the alarm message to the host each time that the value from the current loop sensor falls below 100 units, or exceeds 500 units
IFR15	- returns the alarm message to the host continuously while the value of Channel Variable number 3 is outside of the range of 47.5 to 82.7

Using DeLogger, Alarms are programmed in the Alarms tab of the Program Builder. The alarm input channel is selected in the same manner as selecting a data acquisition channel, however the channel icon has an Alarm icon where the alarm tests and actions are defined. Right click on the Alarm icon and select Properties… Enter the details for the alarm test and actions, as shown in the example below.



Reference Channels

When input channels tested by Alarms require reference channels, such as for voltage excited bridge inputs, or thermocouple inputs from an external isothermal block, these reference channels are declared in dummy Alarms.

These dummy Alarms for the reference channels must precede the actual Alarm whose input channel uses the references.

Using DeTransfer, a reference channel is declared in a dummy Alarm as follows

```
ALARM1(4V(BR))

ALARM2(2BGV(4W,N)>250)1DSO
```

which defines the bridge reference channel for a voltage powered bridge input (See also Section II – Measuring Bridges).

The reference channels for thermocouple inputs from an external isothermal block are declared in two dummy Alarms as follows

```
ALARM3(8LM35(TR))
ALARM4(9V(TZ))

ALARM5(1TT>100)"High Temperature"
```

which defines the reference junction temperature channel and the zero voltage channel (See Section II – Measuring Temperature with Thermocouples).

Using DeLogger, the reference channel is first declared in a similar manner by specifying the input channel and setting the reference option via the Channel Options : Reference… dialog. Then set the Alarm number in the Alarms dialog, and set a test that will never be true.

If reference channels are not declared, then the appropriate default internal references will be used.

Difference and Statistical Data in Alarms

If alarms are required for input channels which have difference data manipulation (DF, RC. RS, IB channel options) or statistical data manipulation (AV, SD, MN, MX, INT channel options), then the appropriate data must first be obtained in a Data Schedule and assigned to a Channel Variable. The Channel Variable can then be tested in an Alarm.

The Data Schedule must execute at the same intervals as the Alarm Schedule. This is achieved by setting the Data Schedule and the Alarm Schedule (see below) for the same trigger.

Alarms are executed after the Data Schedules in each scan, and so data passed forward in this manner is appropriate to the current scan of the alarm.

Using DeTransfer, the program for example

```
BEGIN
 RA5S
  1TT(RC,=5CV,W)
 RZ5S
  ALARM1(5CV>1.5)"Heating Too Fast"
END
```

measures rate of temperature rise sensed by a type T thermocouple in Schedule A, and assigns this to the Channel Variable. The Channel Variable is then tested in the Alarm. The Data Schedule (RA5S) and the Alarm Schedule (RZ5S) are triggering at the same intervals.

The same approach can be used to program this type of Alarm in DeLogger.

The Alarm Comparison Operators

The comparison operator defines the way in which the Alarm compares the test data with the setpoint(s).

The alarm comparison is specified by one of the relational operators

- `>` - greater than or equal to setpoint (high alarm)
- `<` - less than setpoint (low alarm)
- `<>` - less than setpoint1 OR greater than or equal to setpoint2 (high low alarm)
- `><` - greater than or equal to setpoint1 AND less than setpoint2 (in range alarm)

Only one alarm comparison operator can be specified in each Alarm.

The number of setpoints required depends on the comparison, with one setpoint required for simple alarms, and two setpoints required for range alarms.

Using DeTransfer, Alarm comparison operators are programmed as shown in the examples

```
ALARM1(2V>1000)"Alarm 1"

ALARMR2(1+V(X)<200.0)"Low Power"

ALARM10(5L(120,S5)<>500,1500)"Temp Alarm"

IFR15(3CV(S3,NR)><47.5,82.7)"Out of Range"
```

The true alarm states for the various relational operators occur as follows

> - the test data increases to equal to or greater than the setpoint (high alarm)

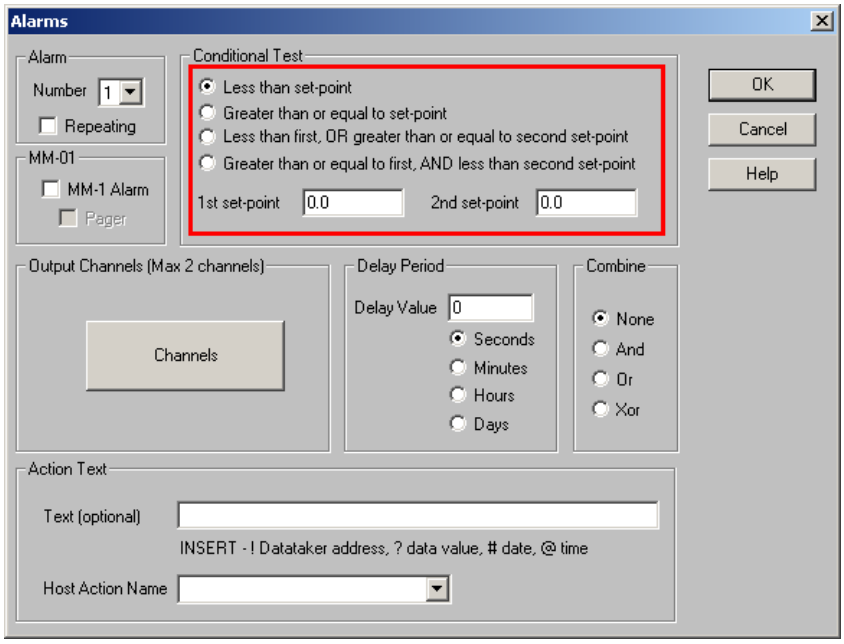
< - the test data decreases to less than the setpoint (low alarm)

<> - the test data decreases to less than the first setpoint, or increases to equal to or greater than the second setpoint (high low alarm)

>< - the test data increases from a low value to equal to or greater than the first setpoint, or decreases from a high value to less than the second setpoint (in range alarm)

The alarm actions are performed when the alarm state first becomes true (ALARM and IF commands) or remains true (ALARMR and IFR commands).

Using DeLogger, the alarm comparison operator and setpoint(s) are set in the Alarms dialog.



The Alarm Setpoints

The Alarm setpoints define the limits against which the test data is compared to determine the alarm state. The alarm setpoints can be specified as numeric constants, Channel Variables, Date or Time as follows

as numeric constants in floating point format, for example 127.64, -0.2245, 1000, etc.

as numeric constants in exponential format, for example 1.276e3, -2.245e-1, 1e4, etc.

as Channel Variables in the format *n*CV

as Date in the date format defined by the Parameter31 command

as Time in the time format defined by the Parameter39 and Parameter40 commands

The numeric constant and Channel Variable setpoints are used for all comparisons except Date and Time. The specification of date and time setpoints are discussed below.

Using DeTransfer, Alarm setpoints are programmed as shown in the examples

```
ALARM1(2V>100.0)"Alarm 1 High"

ALARM2(1+V(X)<-2.0e-2)"Low Power"

ALARM10(5L(120,S5)<>5e2,1.5e3)"Pressure Alarm"
```

Setpoint1 must always be defined, while setpoint2 is only defined where test data is monitored for in range (><) or out of range (<>) conditions.

DeLogger only allows the setpoints to be programmed as floating point numeric constants.

Using Channel Variables as Setpoints

The use of Channel Variables as setpoints allows the setpoints to be dynamically altered during the course of the application, as Channel Variables take on different values. This feature eliminates the need to re-enter the whole Alarm if varying setpoints are required.

The Channel Variables are assigned new values within Data Schedules, as a result of calculations or special manipulation of the input channel data. The Data Schedule must execute at the same intervals as the Alarm Schedule.

Alarms are executed after the Data Schedules in each scan, and so data passed forward in this manner is appropriate to the current scan of the alarm.

Using DeTransfer, Alarm setpoints are programmed as Channel Variables as shown in the examples

```
ALARM11(1TK>22CV)"Temperature High"

ALARM12(1:3TK><15CV,16CV)"Temp in Range"
```

Note that Channel Variables can also be the Alarm test data. For example the following Alarm is legitimate

```
ALARM17(1CV<>2CV,3CV)"Out of Range"
```

DeLogger does not support setpoints being programmed as Channel Variables.

Date and Time Setpoints

When the Date is tested in an alarm, setpoints must be specified in the current date format as defined by the Parameter31 command (See Section III – The Real Time Clock). Date setpoints are appropriate for the dd/mm/yy and mm/dd/yy date formats only. If the Date format is Day Number, then the setpoints are specified as a numeric constant.

Similarly when the Time is tested in an alarm, setpoints must be specified in the current time format as defined by the Parameter39 and Parameter40 commands (See Section III – The Real Time Clock). Time setpoints are appropriate for the hh:mm:ss time format only. If the Time format is decimal hours or seconds, then setpoints are specified as a numeric constant.

Using DeTransfer, date and time setpoints are programmed as shown in the examples

```
P39=0  P40=58
ALARM4(T>12:30:00)"Lunch Time"

P39=2
ALARM4(T>12.5)"Lunch Time"

P31=1
ALARM10(D>25/12/92)"Christmas Day"

P31=2
ALARM10(D>12/25/92)"Christmas Day"

ALARM12(T><8:00:00,16:30:00)"Pump is On"
```

Setpoint1 must always be defined, while setpoint2 must only be defined where input channels are monitored for in range (><) or out of range (<>) conditions.

DeLogger does not support setpoints being programmed as date or time.

Setpoints for Digital Input Channels

When digital input channels are tested in Alarms, the setpoints could reasonably be expected to be specified as 0 and 1.

However because of the possibility of occasional very small errors in the floating point comparisons (a common shortcoming of floating point routines by 8 bit microprocessors), it is recommended that all digital setpoints be specified as 0.5 as follows

```
ALARM3(1DS>0.5)"Relay Open"

ALARM7(3DS<0.5)"Relay Closed"
```

The Alarm Delay Period

The alarm delay period is optional, and decreases the sensitivity of alarms in both transitions as follows

- the alarm comparison must be true for the delay period before the alarm state becomes true and the alarm action performed.
- the alarm comparison must be false for the delay period before the alarm state becomes false.

The alarm delay period therefore allows any short term alarms or nuisance alarms to be ignored.

If a delay period is not specified for the alarm, then the alarm state will change immediately the alarm comparison becomes true or becomes false. If during the delay period the alarm state changes again, then the delay timer resets and no action is taken.

The delay period is specified by a leading / character followed by a duration in the range of 1 to 255 units of time. The units of time are specified as S for seconds, M for minutes, H for hours or D for days.

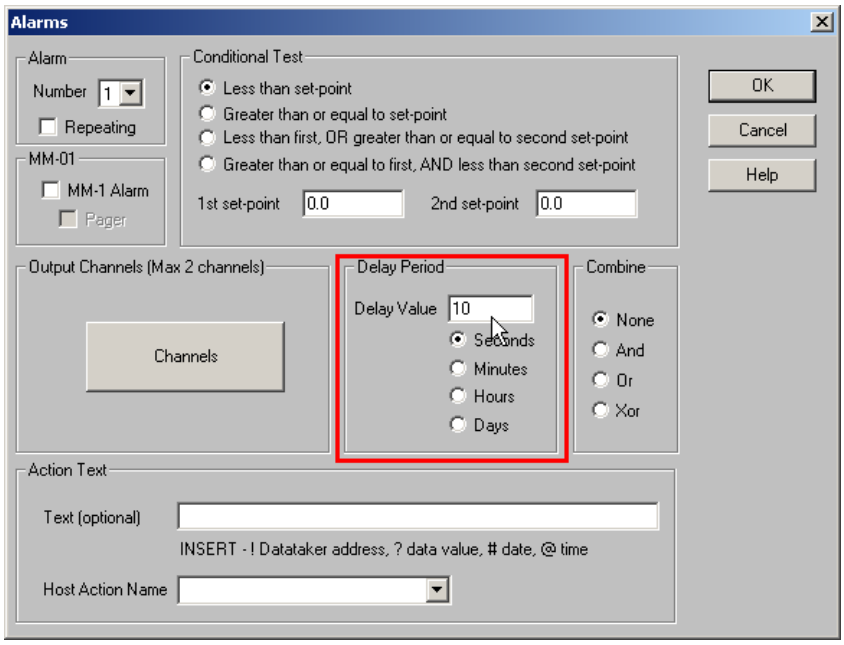
Using DeTransfer, Alarm delays are programmed as shown in the examples

```
ALARM8(10PT385>50.00/10S)"Temp Alarm"

ALARM14(6V<>1e3,2e3/5M)"Voltage Alarm"
```

The delay period cannot be specified as combinations of the time units. For example a delay period of one hour and thirty minutes cannot be specified as 1H30M, but must be specified as 90M.

Using DeLogger, the alarm delay is defined in the Delay Period box of the Alarms dialog. Select the units of time, and enter the Delay Value in the input field.



The Alarm Actions

The Alarm action defines the action to be taken when the alarm comparison is true. There are several different actions which can be performed

return an alarm message to the host computer or a printer

display an alarm message on a display panel (if fitted)

switch one or two digital output channels

switch a warning LED, beeper or LCD backlight on the display panel (if fitted)

execute *dataTaker* commands

The action is optional, but if not defined then no action can be taken when an alarm occurs. The different alarm actions which can be performed are described in the following sections.

Returning a Message to the Host Computer or Printer

When Alarms become true, they can return an alarm message to the host computer or printer, and display a message on the logger's Display Panel if fitted.

Many of the examples used in the preceding sections have alarm messages defined as part of the Alarm declarations.

The length of the Alarm message is limited to 250 characters for each Alarm, and to a total of 4000 characters for all Alarm messages in the program.

When the Alarm message storage area is full, attempts to enter additional messages will result in the error message 'E52-alarm text memory full'.

The logger's Display Panel will only display the first 16 characters of an Alarm message.

The Alarm messages are returned differently for the two operating modes of the Alarms

ALARM and IF – the message is returned or displayed when the Alarm first becomes true. If the Alarm subsequently clears and becomes true again, then the message is returned or displayed again.

ALARMR and IFR – the message is repeatedly returned and displayed while the alarm remains true.

An Alarm message is defined simply as a text string enclosed by quotes, which follows the alarm condition as follows

ALARMn(condition)"Message string"

ALARMRn(condition)" Message string "

IFn(condition)" Message string "

IFRn(condition)" Message string "

where

ALARM	a
ALARMR	)aa are the Alarm
IF	) identifiers
IFR	)
n	is the alarm number
(condition)	is the alarm comparison
"Message string"	is the alarm message

The Alarm message text string can contain control characters for ringing terminal bells, printer formatting, etc. Control characters are entered in the format ^J (LF), ^M (CR), ^G (Bell), ^[(ESC), etc. While two characters are required to enter the control characters, only one character is stored internally.

When an Alarm message is sent to the host, a CR/LF is not appended to the message as might be expected. If a CR/LF sequence is required to terminate the messages, then these characters must be included in the Alarm message as the control characters (^M^J) .

Using DeTransfer, Alarm messages are programmed as shown in the examples

```
ALARM1(2V>1000)"Alarm 1^M^J"

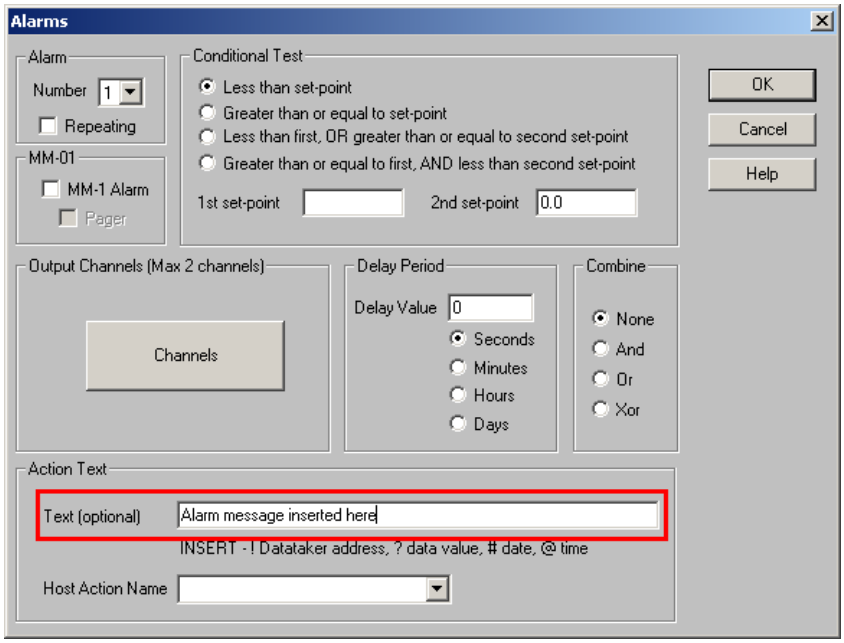
ALARMR2(1+V(X)<200.0)"Low Power^M^J"

IF4(T>12:00:00)"Lunch Time^G"

ALARM10(5L(120,S5)<>100,500)"Temp Alarm^M^J"

IFR15(3CV(S3,NR)><47.5,82.7)"Out of Range"
```

Using DeLogger, the alarm message is defined in the Action text box of the Alarms dialog. The alarms message is entered without " " in the input field.



Alarm Message Substitution Characters

The Alarm message can include a number of substitution characters to pass information about the alarm to the host computer and Display Panel (if fitted). These substitution characters and their function are

! – substitute with the *dataTaker* address and alarm number in the alarm message. In the case of combined alarms the logger address–alarm number for the last alarm is given.

? – substitute with the current value for the alarm test data in the alarm message returned to the host. In the case of combined alarms the current value for the last alarm is given.

– substitute with the current day number or date in the alarm message. The day or date format defined by the Parameter31 command is used.

@ – substitute with the current time in the alarm message. The time format defined by the Parameter39 and Parameter40 commands is used.

The following examples illustrate use of substitution characters in the Alarm messages to pass variable information about the alarm to the host. These substitution characters can be used for defining Alarm messages in both DeTransfer and DeLogger.

```
ALARM4(10PT392>150.0)"Alarm ! High Temp @ ^M^J"
```

In this case the *dataTaker* 1 will return the message on alarm

```
Alarm 1-4 High Temp 11:33:00
```

indicating that alarm number 4 is true on *dataTaker* 1 at 11:33:00.

```
ALARM6(5V(Y1)<>90,110/30S)"Low Press ? psi^M^J"
```

In this case the *dataTaker* will return the message

```
Low Press 85.67 psi
```

indicating the present pressure level which has been low for 30 seconds.

Disabling the Return of Alarm Messages

By default the Alarm messages are returned to the host computer when the Alarm changes from false to true. However the return of the Alarm messages can be globally controlled by the Alarm Messages Switch as follows

/Z Enable return of alarm message (default)

/z Disable return of alarm message

The Alarm Messages Switch defaults to /Z whenever the *dataTaker* is initially powered up, hardware reset or executes a RESET command.

The Alarm Messages Switch has a global effect, and when used it will enable or disable returns of all Alarm messages.

The return of individual Alarm messages to the host can also be disabled by including a NR (No Return) channel option for the input channel or internal channel being tested. However the message can still be displayed on the Display Panel if fitted.

Using DeTransfer, the following alarm for example

```
ALARM3(1V(NR)>110.5)"High Voltage"
```

will not return the alarm message to the host when the alarm state becomes true. However the alarm message can still be displayed. This channel option can also be used if you are programming from DeLogger.

Displaying Alarms

*dataTaker*s fitted with a Display Panel can display Alarm states and Alarm messages (See Section III – Keypad and Display).

A maximum of 16 characters can be displayed on the top line of the LCD display of the Display Panel.

When Alarms without an Alarm message are entered, then default messages are displayed. For example the Alarm

```
ALARM3(1TK>100)1DSO
```

will be displayed as follows when the temperature is below 100°C and therefore not in alarm

```
ALARM 3 OFF
95.8 Deg C
```

and will be displayed as follows when the temperature is above 100°C and therefore in alarm

```
ALARM 3 ON
105.2 Deg C
```

When Alarms which have an Alarm message are entered, then the message is displayed when the Alarm is true. For example the Alarm

```
ALARM3(1TK>100)"High Temp"
```

will be displayed as follows when the temperature is below 100°C and therefore not in alarm

```
ALARM 3 OFF
95.8 Deg C
```

and will be displayed as follows when the temperature is above 100°C and therefore in alarm

```
High Temp
105.2 Deg C
```

The optional channel name for the input channel or internal channel being tested can be used to display a message when the Alarm is not in alarm. When the Alarm is false the optional channel name is displayed, and when Alarm is true the Alarm message is displayed.

For example the Alarm

```
ALARM3(1TK("Low Temp")>100)"High Temp"
```

will be displayed as follows when the temperature is below 100°C and therefore not ion alarm

```
Low Temp
95.8 Deg C
```

and will be displayed as follows when the temperature is above 100°C and therefore in alarm

```
High Temp
105.2 Deg C
```

The display of Alarm messages on the Display Panel can be globally controlled by the Alarm Display Switch as follows

/A Enable display of alarm message

/a Disable display of alarm message (default)

The Alarm Display Switch defaults to /a whenever the *dataTaker* is initially powered up, hardware reset or executes a RESET command.

The Alarm Display Switch must be enabled (/A) before the Alarms are entered into the logger, for the display of Alarm messages to be enabled. The Alarm messages will then be in the scroll list of the display (See Section III – Keypad and Display).

However if Alarms are entered into the logger while the Alarm Display Switch is disabled (/a), then the individual Alarms can be enabled into the display scroll list using the List Enable key of the keypad (See Section III – Keypad and Display).

The display of individual Alarm messages can also be disabled by the ND (No Display) channel option for the input channel or internal channel being tested.

For example the Alarm

```
ALARM3(1V(ND)>110.5)"High Voltage"
```

will not display the Alarm message onto the Display Panel when the alarm becomes true. However the Alarm message will still be returned to the host. This channel option can also be used if you are programming from DeLogger.

Executing dataTaker Commands from an Alarm

The Alarms can execute *dataTaker* commands or a *dataTaker* program by loading the command(s) into the internal command buffer whenever an Alarm becomes true. The command(s) is executed as if entered directly by a host computer, and any of the *dataTaker* commands can be issued by this function.

This function can be used to command the *dataTaker* in any way on a true Alarm, including changing a scanning rate, enabling the logging of data, assigning new constants to Channel Variables, switching digital output channels, or completely reprogramming the logger.

The commands are executed differently for the two modes of the Alarms as follows

ALARM and IF – the commands are executed once when the Alarm becomes true. If the Alarm clears and becomes true again, the commands are executed again.

ALARMR and IFR – the commands are repeatedly executed while the Alarm remains true.

The *dataTaker* commands for a true Alarm are declared in a text string similar to the Alarm message.

The commands are enclosed in [square brackets] as follows

ALARMn(condition)"[commands]"

ALARMRn(condition)"[commands]"

IFn(condition)"[commands]"

IFRn(condition)"[commands]"

where

ALARM	)
ALARMR	) are the Alarm
IF	) identifiers
IFR	)
n	is the alarm number
(condition)	is the alarm comparison
"[commands]"	are the dataTaker commands

Only one set of [commands] is permitted in the text string, and so all commands associated with the Alarm action must be declared together.

When commands are entered into the command buffer by the Alarms, a CR/LF is appended to the commands to ensure that the command string is terminated correctly and will be executed. Therefore these line control characters do not need to be included in the command string.

The following examples can all be used in DeTransfer and DeLogger, and illustrate the use of *dataTaker* commands as actions to Alarms

```
ALARM1(10TT>100)"[1DSO=1]"
ALARM2(10TT<100)"[1DSO=0]"
```

The first Alarm turns digital output channel 1 ON when the temperature exceeds 100°C. When the temperature falls below 100°C, the digital output channel is turned OFF by the second Alarm.

```
ALARM3(T>09:00:00)"[RA5S 1..10V LOGON]"
```

The *dataTaker* is programmed to log the voltages read for analog input channels 1 through 10 after 09:00:00 am.

```
BEGIN
 RX
  D T 1V 2R 3F 5TK

 RZ15S
  ALARMR12(1CV>125.5)"[X]"
END
```

The Alarm will repeatedly Poll trigger the Poll Schedule (RX) at the Alarm Schedule rate while the Channel Variable (assigned within another Data Schedule) is greater than 125.5 units.

The Alarm text string may contain both a message and *dataTaker* commands, as the following example illustrates

```
ALARM4(2L(Y1)>15)"Alarm [R1M 2L(Y1) LOGON]"
```

Pump pressure is being monitored on analog channel 2. If this exceeds 15 PSI a warning is sent to the host, and the logger is commanded to begin to acquire and log data on the high pressure.

This could also be done by using the Alarm to start a separate Data Schedule when in alarm as follows

```
BEGIN
 RA1M  HA
  2L(Y1)

 ALARM4(2L(Y1)>15)"Alarm [GA LOGON]"
 ALARM5(2L(Y1)<15)"Alarm [HA LOGOFF]"
END
```

The commands in Alarm actions are stored in the same 4000 character block of memory as the Alarm messages. When this storage area is full, attempts to enter additional Alarm text will result in the error message 'E52-alarm text memory full'.

Cautions When Commanding the dataTaker *from an Alarm*

There is no limit to the type or number of commands that can be specified in the action part of an Alarm. However there are several precautions which should be considered when designing commands for Alarm actions, as follows

new Alarm should not be added, or existing Alarms changed, in Alarm actions

where the commands in an Alarm action change the status of the *dataTaker*, the original status of the logger is not resumed when the Alarm recovers. Instead a reversing Alarm is needed to return the logger to the original status.

when both an Alarm message and commands are included in an Alarm action, the message is returned to the host before the commands are executed. This applies irrespective of the order in which the two are declared in the text string.

Switching Outputs from an Alarm

The Alarms can switch one or two of the output channels when in alarm. An output channel can be a digital output channel or a warning channel on the Display Panel (if fitted). The switched digital outputs can be used to control external devices such as a bell, light, solenoid, motor, etc. or to interface the logger to Programmable Logic Controllers, TTL/CMOS logic circuits, etc.

The warning channels on the Display Panel include the LEDs, the LCD display backlight, and the beeper.

The Alarm manages the outputs in a toggling manner, in that the output is switched ON when the Alarm is true and is switched OFF when the Alarm is false.

The output channels are set according to the Alarm comparison each time the alarm is tested. This holds the output ON while the Alarm remains true, and holds the output OFF while the Alarm remains false.

This action is the same for both the ALARM / IF and the ALARMR / IFR modes.

The output channels to be switched are declared following the alarm condition in the general formats

ALARMn(condition)FirstOutput,SecondOutput

ALARMRn(condition)FirstOutput,SecondOutput

IFn(condition)FirstOutput,SecondOutput

IFRn(condition)FirstOutput,SecondOutput

where

ALARM ALARMR IF IFR	) [a]) are the Alarm) identifiers)	
n	is the alarm number	
(condition)	is the alarm comparison	
FirstOutput SecondOutput	first output channel to be switched second output channel to be switched	) Any combination of a DSO) or WARN output channels

Using DeTransfer, the output channels for Alarms actions are defined in the command for example

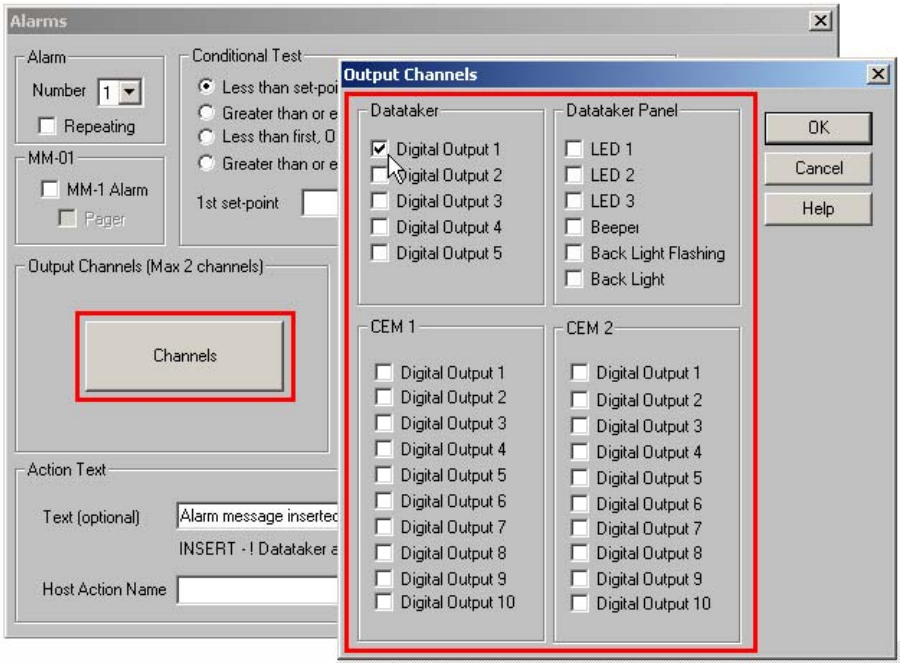
```
ALARM1(5TK>100.00)1DSO,2DSO
```

where the Alarm will turn digital output channels 1 and 2 ON when the temperature exceeds 100°C, and turns the digital output channels OFF when the temperature is below 100°C.

```
ALARM5(2V<>100.0,500.0)1DSO,2WARN
```

where the Alarm will turn digital output channel 1 ON and warn channel 2 ON when the voltage is less than 100 mV or greater than 500 mV, and turns the output channels OFF when the voltage is between 100 mV and 500 mV.

Using DeLogger, the output channels for Alarm actions are defined in the Alarms dialog. Click on the Channels button in the Output Channels panel, then select the output channel(s) in the Output Channels dialog as follows



Cautions When Switching Outputs from an Alarm

The Alarms can be scanned up to 20 times a second at the maximum scan rate. The digital output channel is switched according to the result of the Alarm condition after each scan.

This procedure holds the output channel ON while the Alarm state remains true, and holds the output channel OFF while the Alarm state remains false.

As a general rule two or more Alarms should not manage the same output channel directly, unless the strategy is carefully planned.

Managing the same output directly from two Alarms is illustrated in the following example

```
ALARM1(2V>1000)4DSO
ALARM2(3R<5000)4DSO
```

where the Alarms are processed sequentially, and will set the output ON and OFF on each scanning pass depending on the state of the individual Alarms. One Alarm may be true and attempting to hold the output ON, while the other Alarm may be false and attempting to hold the output OFF.

However where the same output must be managed by two of more Alarms, then the following approach which manages the output by setting it by Alarm action commands, is recommended

```
ALARM1(2V>1000)"[4DSO=1]"
ALARM2(2V<1000)"[4DSO=0]"
ALARM3(3R<5000)"[4DSO=1]"
ALARM4(3R>5000)"[4DSO=0]"
```

Although this approach requires two additional Alarms, switching of the output channel only occurs when the alarm actually becomes true or false, and not while the Alarm remains true or false. This therefore produces a more desirable action of the output channel.

Combining Alarm Actions

These various Alarm actions can be combined to provide greater function to Alarms and more power to Alarm processing. Direct management of output channels, Alarm message strings to be returned to the host and commands for the *dataTaker* can all be declared in the same Alarm.

Using DeTransfer, this can be done as follows

```
ALARM1(5F>85)4DSO"Hi Freq[R1S 5F LOGON]"
```

The Alarm produces the following actions when true

ring a buzzer connected to digital output channel 4 to locally indicate the error condition

send the warning message 'Hi Freq' to the host to indicate that there is an error condition

command the *dataTaker* to begin logging data of the over range frequency.

The management of the digital output channel could also have been carried out using *dataTaker* commands in the Alarm action as follows

```
ALARM4(5F>85)"Hi Freq[4DSO=1 R1S 5F LOGON]"
ALARM5(5F<85)"[4DSO=0]"
```

where the digital output is switched by direct commands. Note that two Alarms are now required, one to switch the output ON when a high alarm occurs, and the other to switch the alarm OFF when there is no alarm.

Various combinations of Alarm actions can also be created using DeLogger, in the Alarms dialog as illustrated for the various examples above.

Whenever a directly managed output channel, an alarm message for the host, and commands for the *dataTaker* are all included in an Alarm action, the actions are carried out in the following sequence

firstly the output channels are switched

secondly the message for the host computer is transmitted

thirdly the *dataTaker* commands are executed

This sequence applies irrespective of the order in which different Alarm actions are declared in the original Alarms.

The Alarm Logical Operators

Alarm logical operators allow a number of Alarms to be combined together into a single Alarm, to provide greater flexibility to Alarms and more powerful Alarm processing.

The logical operators supported are AND, OR and XOR, and are declared following the Alarm condition in the place of other Alarm actions.

The general format for using logical operators is

ALARMx(condition)logic-operator

ALARMy(condition)logic-operator

ALARMz(condition)actions

where

ALARM	is the Alarm identifier
x, y, z	are sequential alarm numbers
(condition)	are the alarm comparisons
logic-operator	is AND, OR or XOR logical operator
actions	is the combined Alarm action

The Alarm actions can only be declared for the final Alarm in the sequential group, and are carried out only if the combined group of Alarms becomes logically true.

Note : The logical operators bind the Alarm to the next sequential Alarm number defined, which may not necessarily be the next entered Alarm.

The sequential group of Alarms is evaluated logically from the lowest Alarm number to the highest Alarm number. If there are missing Alarm numbers in the sequence, then this will not be a problem unless these Alarms are defined elsewhere. If these Alarms are defined elsewhere, then they will be included in the combined alarm.

The following group of alarms

ALARM1 AND ALARM2 OR ALARM3 XOR ALARM4

are evaluated logically in sequential order

(((ALARM1 AND ALARM2) OR ALARM3) XOR ALARM4)

Using DeTransfer, logical operators to combine Alarms are used as follows

```
ALARM1(4TT>55)AND
ALARM2(5TT>55)"High Temps^M^J"
```

In this case the message 'High Temps' is returned to the host if the temperatures monitored on channel 4 AND channel 5 both exceed 55°C.

These alarms could also be entered in a single line for greater clarity. However a space separator must be included between successive alarms as follows

```
ALARM1(4TT>55)AND  ALARM2(5TT>55)"High Temp"
```

Alternatively for a similar group of alarms

```
ALARM1(4TT>55)OR
ALARM2(5TT>55)OR
ALARM3(6TT>55)"High Temperature^M^J"
```

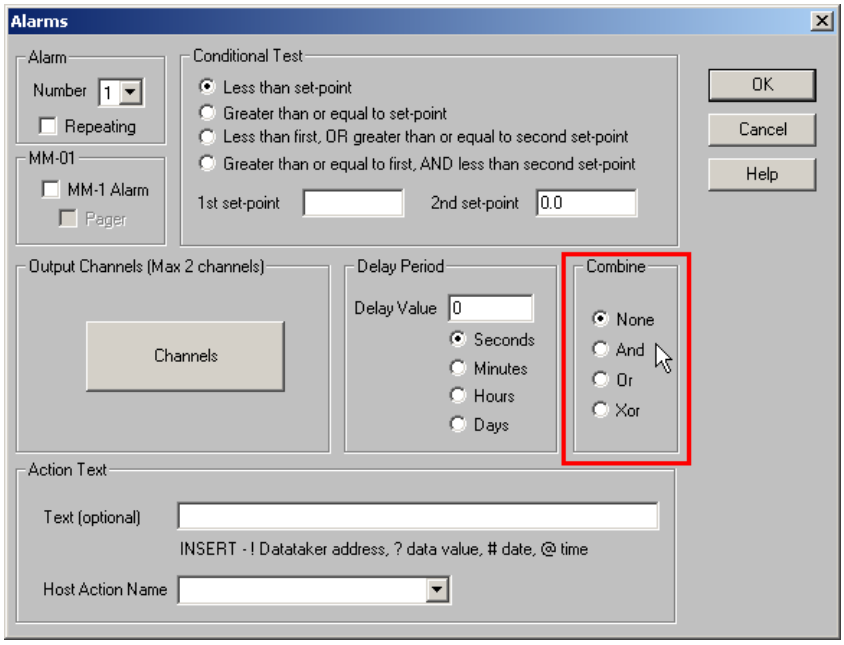
the combined alarm is true if any of the three temperature channels is greater than or equal to 55°C.

Where a delay is required for a combined Alarm, then the delay must be specified within the last Alarm of the group as follows

```
ALARM1(4TT>55)AND
ALARM2(5TT>55)OR
ALARM3(6TT>55/30S)"High Temperatures^M^J"
```

Delays specified for any other alarms in the group are ignored.

Combined Alarms can also be programmed using DeLogger. The logical operator is selected from the list in the Combine panel of the Alarms dialog. DeLogger does not specifically prevent you defining other actions with logical operators, so be careful not to do this.



Caution

The logical operator binds the Alarm for which it is declared to the next sequential Alarm number defined. The next sequential Alarm does not have to be the next entered alarm.

If the following Alarms are entered in the order

```
ALARM5(5V>5000)1DSO"Over Voltage^M^J"
ALARM3(9V>1000)AND
ALARM1(2R<1500)"Low Resistance^M^J"
```

the combined alarm would be processed as

```
ALARM1(2R<1500)"Low Resistance^M^J"

ALARM3(9V>1000)AND
ALARM5(5V>5000)1DSO"Over Voltage^M^J"
```

The Alarm Schedule

The Alarms are processed at intervals of time determined by the Alarm Schedule. Processing of Alarms involves scanning an input channel or internal channel, or reading a Channel Variable, System Variable, System Timer, or the real time clock.

By default the Alarms Schedule is running continuously, and Alarms are processed at the maximum rate possible. This provides fastest response time for Alarms.

However the Alarms Schedule can also be triggered at defined intervals, in the same way that the Data Schedules for data acquisition can be triggered at defined intervals. Running the Alarms Schedule slower can in some cases be an advantage.

The Alarm Schedule trigger can be specified in terms of time, digital event or counter event.

The general format for the Alarm schedule trigger as time is

RZ	– continuous	(Default)
RZnS	– time in seconds	
RZnM	– time in minutes	
RZnH	– time in hours	
RZnD	– time in days	

where n is any integer in the range of 1 to 65535

The general format for the Alarm Schedule trigger as digital events is

RZnE	– trigger on both transitions
RZn+E	– trigger on positive transitions
RZn–E	– trigger on negative transitions

where n is any digital input channel, or a sequence of digital input channels

The general format for the Alarm Schedule trigger as counter events is

RZnC(*count*)	– trigger on *count* from a low speed counter
RZnHSC	– trigger on a high speed counter pulse

where n is any counter channel, and *count* is the terminal count on which to trigger

The Alarm Schedule can also include the Trigger While Condition for conditional triggering (See Section III – Schedules Triggered While Condition).

The Alarm Schedule can be entered into the *dataTaker* at any time, and does not have to be entered in association with actual Alarm commands.

Only one Alarm Schedule interval can be defined at any time, and when an Alarm Schedule is entered any existing Alarm Schedule is replaced.

The Alarm Schedule can also be changed at any time from time based to event based, etc.

Using DeTransfer, the Alarm Schedule is entered for example

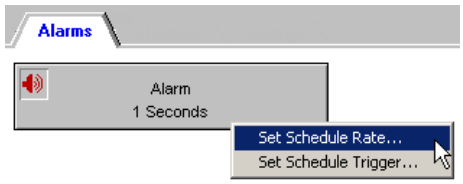
```
RZ5S

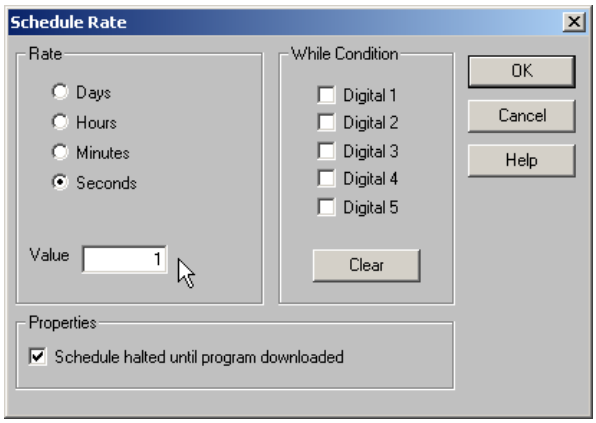
RZ12H

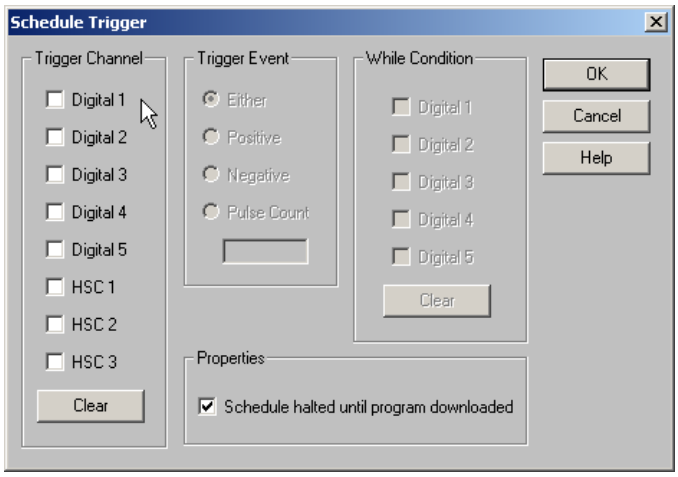
RZ3-E
```

Using DeLogger, the Alarm Schedule is programmed in the Program Builder under the Alarms tab. Right click the Alarm button at the top of the work area, and select Set Schedule Rate… or Set Schedule Trigger… as required.

Enter the trigger details in the following dialog in the same way as described for the Data Schedules.







If an Alarm Schedule is not defined, then the Alarms are processed continuously at the maximum rate.

Halting and Resuming Alarm Schedule

The processing of all Alarms can be globally halted and resumed, and the processing of individual Alarms can also be halted and resumed at any time.

Halting and Resuming All Alarms

The sampling and processing of all Alarms can be halted and resumed by the global Halt and Go commands, and the Alarm Halt and Go commands

H	Halt scanning of all data acquisition channels and Alarms
G	Resume scanning of all data acquisition channels and Alarms
HZ	Halt processing of all Alarms
GZ	Resume processing of all Alarms

When the *dataTaker* is initially powered up, is hardware reset or executes a RESET command, the Alarm Schedule defaults to HZ. When a new Alarm is entered, the Alarm Schedule is enabled (GZ) at the time of entry. This allows the Alarms to be processed at the next Alarm Schedule interval.

Note : Alarm processing is also controlled by the global Halt and Go commands for data acquisition scanning. If the Halt or Go command is entered to halt or resume data acquisition, this will also halt or resume all alarms processing.

Halting and Resuming Individual Alarms

The sampling and processing of individual Alarms can be halted and resumed by the commands

HZn	Halt processing of an individual alarm
GZn	Resume processing of an individual alarm

where n is the alarm number

If an undefined alarm number is suspended by the command HZn, or resumed by the command GZn, then the command has no effect.

When a CALARMS command is executed (See Section III – Clearing Alarms from the Channel Table) all Alarm information is erased from the alarms section of the internal channel table.

Preventing Alarm Schedules from Waking the *dataTaker*

When Alarms are being processed or data acquisition channels are being scanned at intervals of less than 2 seconds, then the *dataTaker* will not automatically enter the low power mode or sleep mode.

Conversely whenever Alarms are being processed or data acquisition channels are being scanned at an interval of greater than 2 seconds, then the logger will sleep between scans if other functions are not keeping it awake.

The *dataTaker* can be programmed to ignore Alarm Schedule triggers or Data Schedule triggers, and to sleep and wake according to specified schedules only.

Where used for Alarms, the effect of this is that the Alarms are only processed whenever the *dataTaker* is awake to read data acquisition channels, even though the Alarm Schedule interval may be more frequent than the Data Schedule intervals.

The schedules which can wake then *dataTaker* are defined by setting a bit mask using the Parameter20 command as follows

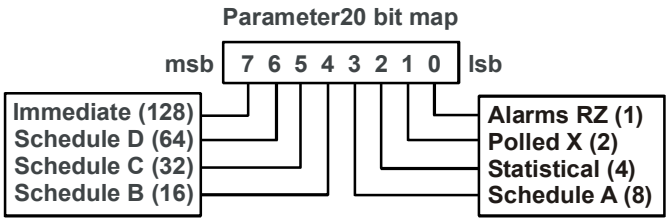


Figure 128 – Parameter20 Bit Map

Whenever the *dataTaker* is initially powered up, is hardware reset or executes a RESET command, Parameter20 defaults to 0 which allows all schedules will wake the logger when they become due.

The use of the Parameter20 command is illustrated by the example

```
P20=1
BEGIN
 RZ1S
  ALARM(1V>1000)"Voltage Alarm"

 RA30M
  1..5V  10TT
END
```

where the Alarm Schedule is triggered every second, and the data acquisition schedule is triggered every 30 minutes.

With Parameter20=1, the *dataTaker* sleeps and wakes according to the data acquisition interval of Schedule A, and only processes the Alarm on a 1 second interval while awake to read the data acquisition channels.

Returning Data From Alarm Input Channels

The most recently read value of the Alarm test data can be returned to the host computer by the Alarm Query commands

?n	Returns the current data for a single alarm
?ALL	Returns the current data for all alarms

where

?	is the Alarm Query identifier
n	is an alarm number
ALL	specifies all alarms

The format of the data returned to the host computer by the Alarm Query command is similar to the format of the data returned by the Data Schedules.

The only difference in data format for data is returned by the Alarm Query command is that the alarm number of the alarm testing the input data is returned with each item of data rather than the channel number of the input channel or internal data source.

The format of the data returned by the Alarm Query command is illustrated in the following example

```
ALARM3(10V>1800.0)3DSO

?3
```

will return data in the format

```
3  1550.8 mV
```

The returned data can be formatted by the Switch commands /U, /u, /N, /n, /C and /c, and the Parameter22 and Parameter24 commands, in the same manner as data returned from the Data Schedules (See Section III – Format of Returned Data).

If an undefined Alarm or a halted Alarm are queried by the Alarm Query command, then the command is ignored and no data is returned.

Data returned by the Alarm Query commands cannot be logged to internal memory or memory cards.

Assuming that the following Alarms have been entered in the *dataTaker*, and are currently being processed

```
ALARM1(T>10:00:00)"[GA]"
ALARM2(T>12:00:00)"[HA]"
ALARM5(10TT<100.5)2DSO"Low Temperature^M^J"
```

then the Alarm Query commands for these alarms will produce the following responses

```
?5
```

will return the current value for the thermocouple as follows

```
A5  115.35 Deg C
```

```
?ALL
```

will return the current values for all input channels as follows

```
A1  10:20:33
A2  10:20:33
A5  115.35 Deg C
```

The Alarm Query command returns the most recent value of the Alarm test data. The command does not invoke a new reading of the Alarm input.

The Alarm Query command is not directly supported by DeLogger, however can be sent from the lower Entry Screen of the Text View if needed.

Checking the Alarms

The Alarms can be checked at any time by the STATUS command.

The general STATUS command lists the number of Alarms currently defined as part of the general report, and the STATUS3 command lists the actual definitions (except Channel Options) of the Alarms.

For example, using DeTransfer the command

```
STATUS
```

produces a general status report to the host

```
dataTaker 0 Version 7.00
A B C,none Scan Schedules Active, Halted
3,0 Alarms Active,Halted
3 Polynomial/Spans Defined
Logging is OFF
166530 Internal Data Points Free,Stored
0,0 Card Data Points Free,Stored
4090,0 Program Characters Free,Stored
/a/C/d/E/f/h/J/K/l/M/N/o/Q/R/S/t/U/v/w/x/y/Z
```

and the command

```
STATUS3
```

produces a detailed report of the Alarms similar to

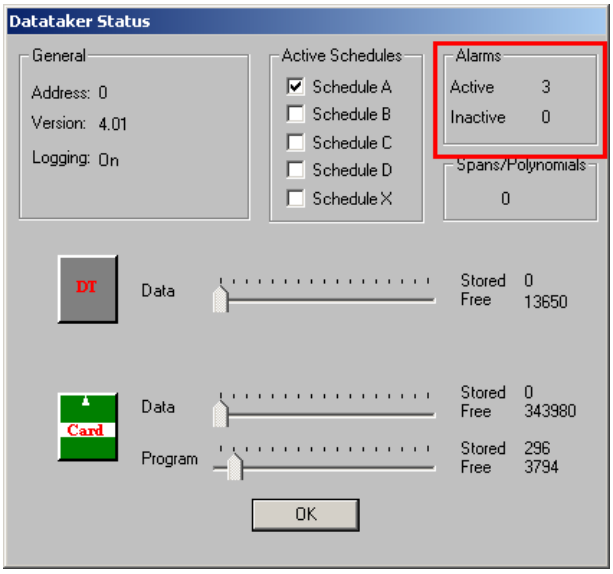
```
3,0 Alarms Active,Halted
RZ2S
ALARM3(1V>1000.00)"Volt Over Range^M^J"
ALARM4(5TT<>100,105/5S)3DSO"Boiler^M^J"
ALARM22(1C><256,512)"[RA1S 1C]"
```

Individual Alarms which have been halted by the HZn command are reported with the Alarm keyword in lower case characters as follows

```
alarm4(5TT<>100,105/5S)3DSO"Boiler^M^J"
```

and are reported in the normal manner when resumed by the GZn command.

Using DeLogger, the number of Alarms active and halted can be checked by the dataTaker : Status option from the main menu bar, or the Status button on the main toolbar. Neither of these however will list the actual Alarms.



Clearing Alarms from the Internal Channel Table

When Alarms are entered into the *dataTaker*, the Alarm input channels are entered into an internal channel table. The internal channel table is shared with the Data Schedules for data acquisition (See Section III – Schedules).

Whenever the *dataTaker* is initially powered up or executes a RESET command, the internal channel table is initialised to 20 entries for Alarms.

The allocation of entries in the channel table for Alarms can be either increased or decreased by the Parameter30 command. However if there are any entries in the alarm or data acquisition sections of the channel table, the Parameter30 command cannot be used until the internal channel table is cleared.

The section of the internal channel table which is allocated to the Alarms is cleared from DeTransfer by the command

```
CALARMS
```

The section of the internal channel table allocated to Data Schedules is cleared by the CSCANS command, which must also be used before channel table allocation can be altered by Parameter30.

Individual Alarms can be cleared from the internal channel table by the command

```
CALARMn
```

where *n* is the number of the individual alarm to clear. The alarm number must be in the range of 1 to the setting of Parameter30.

If an Alarm is entered with the same Alarm number as that of an already entered Alarm, then the new Alarm will replace the original Alarm. This can be used to interactively redefine Alarm function as an application progresses.

Using Alarms for Controlling

The Alarms can be used for simple 'bang bang' control functions, which provide switching for an actuator at the alarm setpoint. The *dataTaker* does not support any PID control functions.

The following examples illustrate the use of Alarms as simple controllers

```
ALARM1(1TT<50.00)1DSO
```

where the thermocouple is monitoring the temperature of a water bath, and digital output channel 1 is controlling a water heater. If the temperature falls below 50.0°C the heater is switched on, and when the water is heated to 50.0°C the heater is switched off.

This simple controller has no hysteresis characteristic and so will tend to 'hunt or chatter' around the setpoint, particularly if there is little inertia in the controlled system.

A bang bang system will always hunt around the setpoint. If hunting is too rapid, the Alarm can be slowed by a delay for example

```
ALARM1(1TT<50.00/30S)1DSO
```

which will delay switching of the heater when the temperature falls below the setpoint. However this will cause greater excursions from the setpoint.

Hysteresis can be obtained using two Alarms as follows

```
ALARM1(1TT<50.00)"[1DSO=1]"
ALARM2(1TT>51.00)"[1DSO=0]"
```

which provides a 1°C deadband.

Other Applications of Alarms

The Alarms can be used for a variety of purposes other than for monitoring and acting on specific alarm conditions. Some more common applications are illustrated below, but there are many other applications which will become self evident when need dictates.

1. Read input channels and log the data a window of time each day. Consider the program

```
BEGIN
 RA5M
  1..10TT  HA

 ALARM1(T>06:00:00)"[GA]"
 ALARM2(T>15:00:00)"[HA]"
END
LOGON
```

In this case a Data Schedule is to log the data for ten thermocouple inputs, and the Schedule is immediately halted. Two Alarms are defined where Alarm 1 starts the data logging at 6:00:00, and Alarm 2 halts the data logging at 15:00:00.

2. Log data from the input channels only if the data is outside an acceptable range. Consider the program

```
BEGIN
 RA10M
  1V  LOGOFF

 ALARM1(1V<2000)"[LOGON]"
 ALARM2(1V>2000)"[LOGOFF]"
END
```

In this case Schedule A is reading a voltage input, and data logging is initially disabled. Two Alarms are defined where Alarm 1 enables data logging if the measured voltage is less than 2 volts, and Alarm 2 disables data logging if the measured voltage is greater than 2 volts.

3. Read a group of channels for a period of time, then read a second group for a period of time. Consider the program

```
BEGIN
 RA1M
  1..5L(S1)  HA
 RB1M
  6..10V     HB

 ALARM1(T>10:00:00)"[GA]"
 ALARM2(T>12:00:00)"[HA]"
 ALARM3(T>16:00:00)"[GB]"
 ALARM4(T>18:00:00)"[HB]"
END
```

Two Data Schedules scan different groups of inputs, and both are halted at program entry. Four Alarms are defined which have the following effects

Alarm 1 commands Schedule A to run (GA) at 10:00:00

Alarm 2 commands Schedule A to halt (HA) at 12:00:00

Alarm 3 commands Schedule B to run (GB) at 16:00:00

Alarm 4 commands Schedule B to halt (HB) at 18:00:00

4. Read a group of channels and log the data for short periods of time during longer periods of time. Consider the program

```
BEGIN
 RA5S
  1..5V  HA

 HZ
 ALARM1(2ST(60)>55)"[GA]"
 ALARM2(2ST(60)>5)"[HA]"
 GZ
END
LOGON
```

This application uses the System Timer 2, which increments every minute (See Section III – System Timers). Schedule A is entered to scan a group of voltage inputs, and is immediately halted. The Alarm Schedule is temporarily halted (to prevent the loss of initial synchrony) and the two Alarms are entered. Data logging is enabled.

Alarm1 starts Schedule a scanning when the System Timer exceeds 55 counts (55 minutes), and Alarm2 stops Schedule A scanning when the System Timer exceeds 5 counts (5 minutes). The System Timer 2 is synchronized to the real time clock, and resets to a count of zero on a count of 60 (minutes).

The program is started by enabling the Alarms Schedule. The effect of the program is to scan the group of channels and log the data from 5 minutes before each hour until 5 minutes after each hour

Scaling Data - Polynomials, Spans & Functions

The *dataTaker* has many methods for scaling and manipulating data collected from input channels. Often a combination of the different methods can be used to linearise or calibrate the data from sensors.

Scaling data is principally used for converting or linearizing the electrical signals measured on the input channels, into units of the physical quantity that is being measured.

Other reasons for scaling data include adjusting for attenuation of inputs, specifying the resistance of current shunts, changing the magnitude of data, etc.

The methods available for scaling data with the *dataTaker* include

Automatic Scaling

Channel Factors

Spans

Polynominals

Intrinsic Functions

Calculations

Automatic Scaling

The input channels of the *dataTaker* accept many different signal types. The *dataTaker* automatically scales the standard signal types to their respective engineering units.

The fundamental signal types supported by the *dataTaker* include voltage, current, resistance, frequency and digital state, and this data is returned in predefined engineering units of millivolts, milliamps, Ohms, Hertz, State, etc.

Sensors with standardised characteristics such as thermocouples, RTD's, IC temperature sensors, vibrating wire transducers, and thermistors are automatically linearized to physical units such as temperature by internal polynomials, prior to being returned to the host or logged.

Automatic scaling of input channels is illustrated in the following program

```
BEGIN
 RA1M
  1V  2TK  3R
END
```

In this example the *dataTaker* automatically scales the data from raw output of the analog to digital converter to electrical units, and in the case of the type K thermocouple the *dataTaker* also performs cold junction compensation and linearization of the measured voltage to units of temperature. Data from tis program will be returned as

```
1V  1283.67 mV
2TK  86.2 Deg C
3R  2876 Ohms
```

The *dataTaker* has linearization functions for all of the commonly used types of thermocouples, RTDs, IC temperature sensors and YSI thermistors.

Channel Factor

The simplest method of scaling raw data from an input channel or internal channel is by applying a Channel Factor directly to the channel data. The raw data is simply multiplied by the Channel Factor.

The Channel Factor has a predefined function for most of the input signal types supported by the *dataTaker*, as follows

Voltage, Thermocouples, LM335 – the Channel Factor specifies an attenuation factor for these signal types if inputs have been externally attenuated. External attenuation mainly applies to voltage signals which exceed the input voltage range of the *dataTaker*, and must be externally reduced by an attenuator. The measured voltage is scaled up by the attenuation factor to the magnitude range of the original signal. The default Channel Factor for these types is 1.0.

Current, Current Loop, AD590, AD592 – the Channel Factor specifies the resistance of the current shunt used for measuring the current signals. The default current shunt resistance is 100.0 Ohms[a], however larger and smaller shunt resistances can be used to suit different current magnitudes to be measured. The shunt resistance is used by the *dataTaker* to calculate current flow from the measured voltage produced across the shunt by the unknown current.

Bridge – the Channel Factor specifies the arm resistance for current excited resistance bridges. The default Channel Factor is 350.0 Ohms.

RTDs – the Channel Factor specifies the resistance of the RTD element at 0°C. However RTDs with a different resistance at 0°C can be supported by the *dataTaker* by specifying this as the Channel Factor. The default Channel Factor is 100.0 Ohms.

System Timers, Counters – the Channel Factor specifies the maximum or terminal count. The range of the counters can be customised by specifying different maximum or terminal counts as a Channel Factor. The default Channel Factor varies for System Timers, and is 65535 Counts for counters.

Frequency and Period –the Channel Factor sets the minimum measurable frequency, by defining the longest period. The default Channel Factor is 30 mSec, for 33Hz minimum frequency.

Vibrating Wire –the Channel Factor sets the delay between plucking the wire and reading the wire frequency. The default Channel Factor is 200 mSec.

Resistance, Channel Variables, System Variables – the Channel Factor is undefined, and so can be used to scale the data by a linear factor.

Digital – the Channel Factor is for various functions, depending on the signal types.

The Channel Factor is entered as a Channel Option (See Section III – Channel Options) for the input channel as follows

```
RA5S 1V(10.1) 4PT385(250.0) 7F(1000) 3CV(2.2)
```

1V(10.1) – an externally attenuated voltage is applied to analog channel 1. The *dataTaker* measures the input voltage, and multiplies this by the attenuation factor and returns the data in the magnitude range of the original signal.

4PT385(200.0) – a platinum RTD connected to analog channel 4 has a resistance of 250[a] Ohms at 0°C. The *dataTaker* uses this resistance rather than the default of 100 Ohms[a] when calculating the temperature represented by the signal from the device.

7F(1000) – a low frequency signal connected to analog channel 7 is less than the default minimum measurable frequency of 33 Hz. The Channel Factor sets maximum period to 1000mSec, enabling a minimum measurable frequency of 1 Hz.

The data stored in Channel Variable number 3 is simply scaled up by the factor of 2.2.

Intrinsic Functions

The *dataTaker* has 7 Intrinsic Functions that can be used to convert data in various ways.

The Intrinsic Functions are applied as Channel Options for the input channel or Channel Variable (See Section III – Channel Options). The Intrinsic Functions are mutually exclusive , such that only one Intrinsic Function can be applied to each channel or Channel Variable.

The Intrinsic Functions supported by the *dataTaker* are listed in the following table

Channel Option	Function	Unit Text Modifier
F1	Inverse of data (1/x)	(Inv)
F2	Square root of data ($\sqrt[a]{x}$)	(Sqrt)
F3	Natural Logarithm of data (Log_n)	(nLog)
F4	Logarithm Base 10 of data (Log_{10})	(Log)
F5	Absolute value of data (|x|)	(Abs)
F6	Square of data (x^2)	(Squ)
F7	Grey code conversion (8 bit)*	(Gc)

* (See Section III –The Digital Input Channels

When Intrinsic Functions are applied to input channels or Channel Variables, the data is returned with the units text appropriate to the channel, appended with the units text modifier for the Intrinsic Function.

Using DeTransfer, Intrinsic Functions are applied to input channels and Channel Variables for example

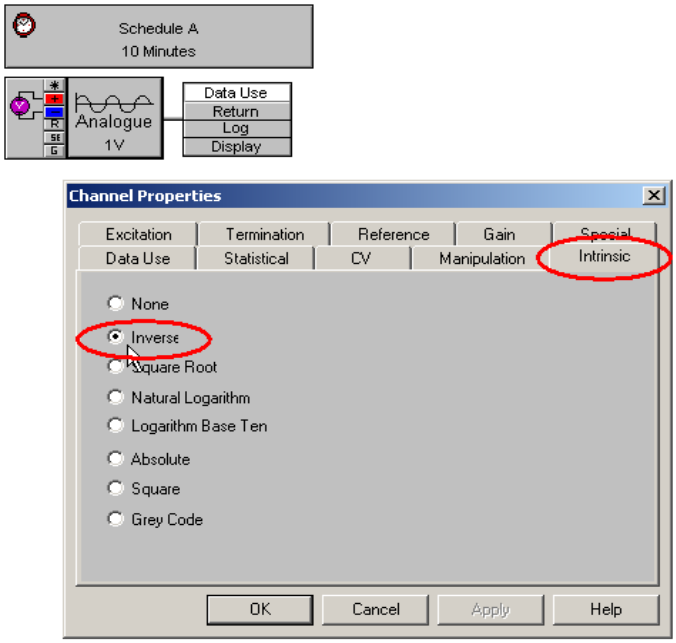
```
BEGIN
 RA10M  1V(F1)  2V(F2)  3V(F3)  4V(F4)
END
```

which will return data in the format

```
1V  0.036 mV (Inv)
2V  4.758 mV (Sqrt)
3V  2.38 mV (nLog)
4V  4.356 mV (Log)
```

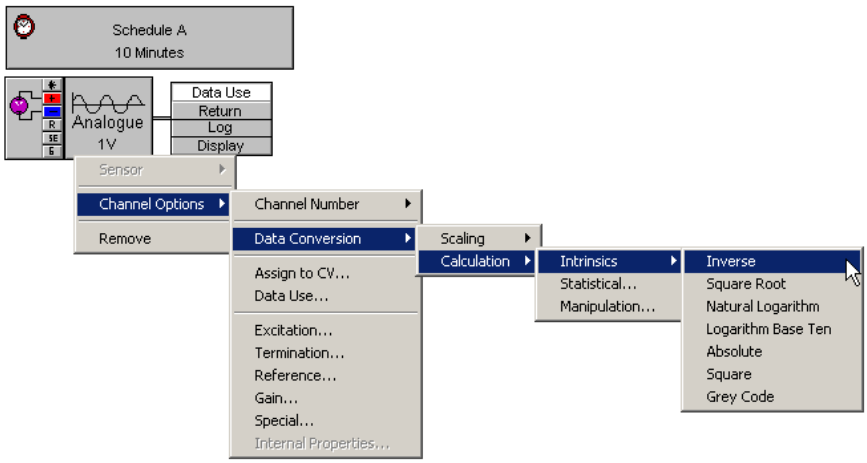
The channel number and units text are returned with the data according to the settings of the (/n /N) and (/u /U) Switches, Parameter22 and Parameter24 (See Section III – Format of Returned Data).

Using DeLogger, the Intrinsic Functions are applied to input channels and Channel Variables in the



in the Program Builder by first defining the channel or Channel Variable, then right click on the Data Use icon, select Properties…, select the Intrinsic tab in the Channel Properties dialog, and select the required function.

Intrinsic Functions can also be selected by right clicking on the channel icon, and follow the walking menu through Channel Options:Data Conversion:Calculation:Intrinsics and select the required Function from the list of Functions.



Introduction to Polynomials and Spans

The Polynomials and Spans extend the flexibility of the *dataTaker*, allowing the raw data for almost any type of sensor to be linearized to engineering units that are appropriate to the physical parameter being measured.

Polynomials are used to convert sensors with either curvilinear (non-linear) and linear calibrations, to engineering units.

Spans are used to convert sensors with linear calibrations to engineering units. Spans are particularly suited to current loop transmitters, which by definition have linear scaling.

The Polynomial and Span declarations are stored in a common area of *dataTaker* memory. A maximum of 20 Polynomials, 20 Spans, or any combination (eg. 17 Polynomials and 3 Spans) can be defined.

Polynomials

A Polynomial describes the relationship between the output signal from a sensor (which is plotted on the X axis of a graph), and the physical parameter that the sensor is measuring (which is plotted on the Y axis of a graph).

They are used to convert signals from sensors that have both curvilinear (non-linear) or linear calibrations.

The *dataTaker* supports a maximum of 20 polynomials, which are numbered 1 to 20. The Polynomial declarations share the internal storage memory with the Span declarations.

The polynomials can be defined as a 1st, 2nd, 3rd, 4th or 5th order polynomial, and units text can be defined appropriate to the new data.

Higher polynomial orders describe greater degrees of non-linearity in the relationship between the sensor signal and the physical parameter measured.

The polynomials are described mathematically as follows

- first order (linear) $Y = a + bX$
- second order $Y = a + bX + cX^2$
- third order $Y = a + bX + cX^2 + dX^3$
- fourth order $Y = a + bX + cX^2 + dX^3 + eX^4$
- fifth order $Y = a + bX + cX^2 + dX^3 + eX^4 + fX^5$

where

Y	is the calculated data in terms of the physical parameter measured
X	is the raw data measured from the sensor
a, b, c, d, e, f	are the polynomial coefficients

Declaring Polynomials

The Polynomial coefficients are entered into the *dataTaker* in the general format

Yn=a,b,c,d,e,f"units text"

where

Y	is the polynomial identifier
n	is the polynomial number (1 – 20)
a, b, c, d, e, f	are the polynomial coefficients
"units text"	is the user definable units text

The Polynomial number must be in the range of 1 to 20, and must not be a number used for a Span (see below).

The "units text" string is optional, and allows engineering units appropriate for the physical parameter being measured to be defined. The units text string is limited to a maximum of 7 characters, and may contain control characters which are entered in the two character format of eg. ^G (Bell) ^M (CR) ^J (LF) etc.

The dynamic range for the calculated data from polynomials is ±1.00e-18 to ±1.00e18.

An example of a curvilinear calibration between the output voltage and the applied load for a load cell is illustrated below

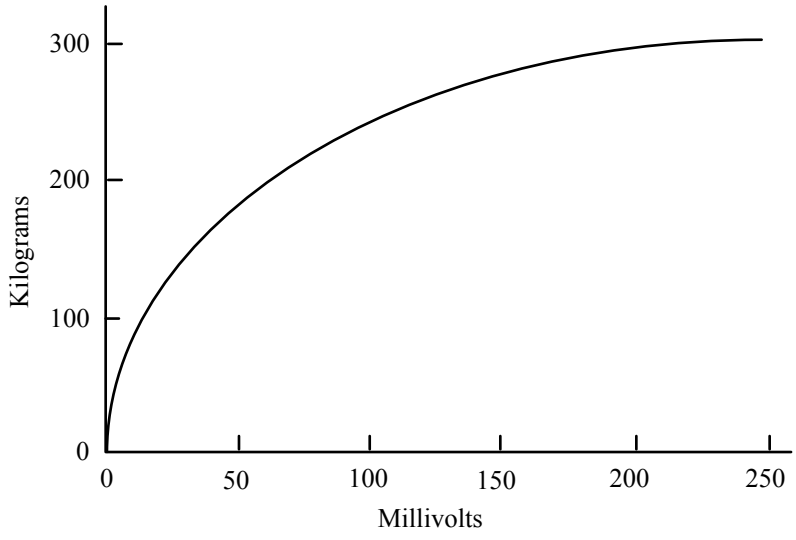


Figure 129 - A Load Cell Calibration

The polynomial for the curvilinear relationship between the millivolt output and applied load for the load sensor is obtained by a regression for example

$$Y=1.42+7.04X-0.099X^2+0.001X^3-2.88e-6X^4+3.93e-9X^5$$

The Polynomial is defined to the *dataTaker* in terms of its coefficients – the powers of X do not need to be entered because these are implied and the *dataTaker* understands this.

Using DeTransfer, the polynomial is entered into the *dataTaker* as polynomial number 1 as follows

```
Y1=1.42,7.04,-0.099,0.001,-2.88e-6,3.93e-9"Kgm"
```

where the polynomial coefficients only are listed. The coefficients are separated by commas, and there must be no included spaces. Any zero coefficients must be included to enable the following coefficients to be given the correct ascendancy.

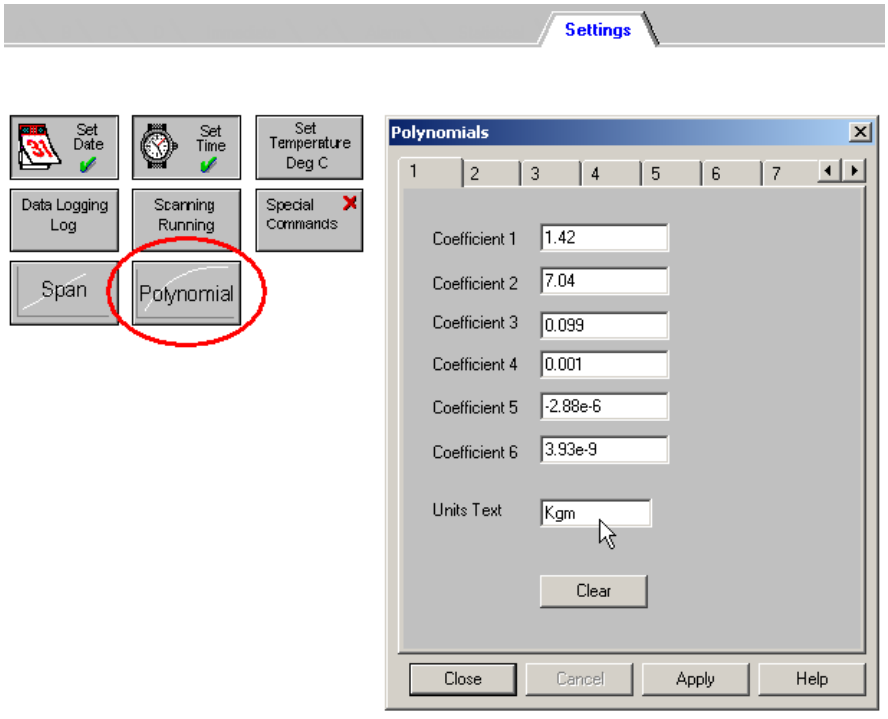
The polynomial coefficients can be entered either in floating point format (1.42, 0.099), or exponential format (1.55e-3, 3.0e2)

While polynomials up to the 5th order (6 coefficients) are permitted, lower order polynomials can be used and are entered as follows

```
Y2=3.67,-1.22"KWatts"
```

which in this case describes a second order polynomial, which is a linear relationship.

Using DeLogger, Polynomials are entered in the Program Builder. Click on the Settings tab, right click on the Polynomial button and select Properties... to open the Polynomials dialog. Click on the tab for the Polynomial number you are defining, and Enter the coefficients and units text for the new Polynomial. Click Apply to store the values, and either click on the tab for the next Polynomial number to define, or click OK to finish.



DeLogger only allows 10 Polynomials and 10 Spans to be defined rather than any mix totalling 20 as the *dataTaker* allows.

Using Polynomials

Defined Polynomials are specified as a Channel Option for input channels, Channel Variables, etc to linearise the original data.

Using DeTransfer, this is illustrated in the following program

```
Y1=1.42,7.04,-0.099,0.001,-2.88e-6,3.93e-9"Kgm"
Y2=3.67,1.22"KWatts"
Y9=84.65,3.9"m/sec"

BEGIN
 RA15M
  5V(Y1)   6V(Y2)   8F(Y9,"Speed")
END
```

which will result in data of the format

```
5V   224.37 Kgm
6V   85.63 Kwatts
Speed   26.431 m/sec
```

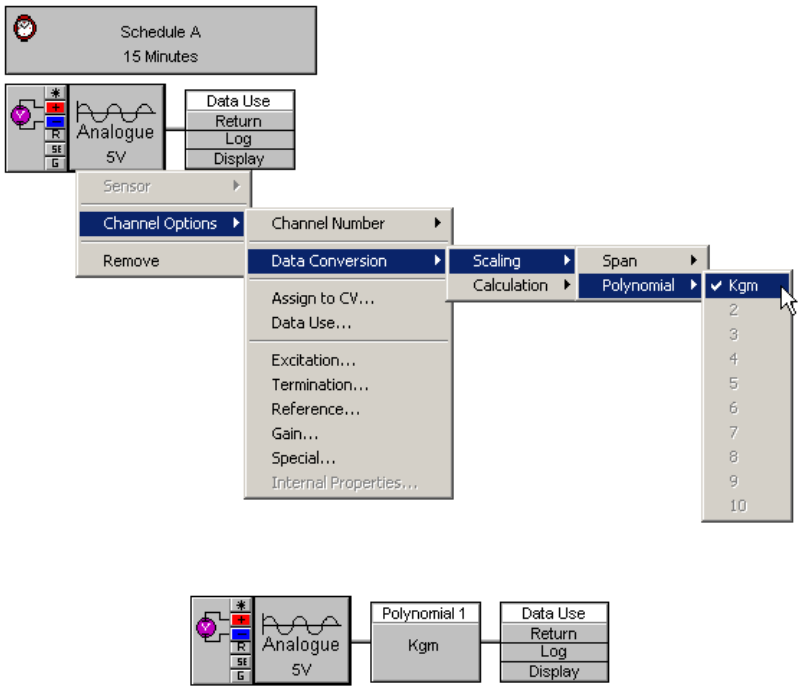
The channel number and units text are returned with the data according to the settings of the (/n /N) and (/u /U) Switches, Parameter22 and Parameter24 (See Section III – Format of Returned Data).

Polynomials can be specified as a Channel Option to sequences of channels using DeTransfer, as follows

```
R1H   1..5V(Y1)   6..10R(Y2)
```

where Polynomial Y1 is specified as a Channel Option to channels 1 through 5, and the Polynomial Y2 is specified as a Channel Option to channels 6 through 10.

Using DeLogger, defined Polynomials are specified as a Channel Option for input channels, Channel variables, etc in the Program Builder. Firstly, enter the input channel or Channel Variable into the work area. Right click on the channel icon, and follow the walking menu through Channel Options:Data Conversion:Scaling:Polynomial and select the required Polynomial from the list of defined Polynomials.



The channel will then change to show the Polynomial being applied between the channel icon and the data use icon.

Using Polynomials for Other Purposes

Polynomials can also be used to carry out various mathematical processing on data, and to attach custom units text to data.

A number of standard mathematical functions which can be implemented using polynomials, for example

squaring of data	Y1=0,0,1
cubing of data	Y1=0,0,0,1
scaling data by a constant multiplier	Y1=0,*multiplier*
adding a constant offset to data	Y1=*offset*,1

Dummy Polynomials can be used to add custom units text to data. This is particularly useful with Channel Variables which do not have units text. For example

```
Y1=0,1"Widgets"
BEGIN
 RA5M
  10CV(Y1)=(1CV+2CV)/(3CV-5.67)
END
```

where Y1 is defined as a dummy polynomial (does not change the magnitude of the data) but has a custom units text which is applied to any data that the Polynomial is applied to.

Obtaining the Polynomial for a Sensor

The polynomial calibration for sensors can usually be obtained from the manufacturer of the sensor.

However in some cases it may be necessary to calculate the sensor polynomial from calibration tables or lookup tables supplied by the manufacturer. This can be done using one of the many statistical software packages available, which have functions for calculating polynomials or for fitting polynomials by least squares regression. The Microsoft Excel spreadsheet package has these functions.

Spans

A Span describes the linear relationship between the output signal from a sensor (plotted on the X axis of a graph), and the physical parameter that the sensor is measuring (plotted on the Y axis of a graph).

Spans are used to convert signals from sensors which have linear calibrations, into units in terms of the physical parameter being measured. Spans are particularly suited to converting data from current loop transmitters which by definition have linear scaling.

The *dataTaker* supports a maximum of 20 spans, numbered 1 to 20. The Span declarations share internal memory with the Polynomial declarations.

Spans are described mathematically as follows

$$S = (\text{Slope})X - \text{Offset}$$

$$= \frac{(b-a)}{(d-c)}X - (b-d)\frac{(b-a)}{(d-c}$$

where

- S is the calculated data in terms of the physical parameter
- a is the lower physical co-ordinate
- b is the upper physical co-ordinate
- c is the lower signal co-ordinate
- d is the upper signal co-ordinate

Declaring Spans

Spans are entered into the *dataTaker* in the general format

```
Sn=a,b,c,d"units text"
```

where

S	is the span identifier
n	is the span number (1 - 20)
a	is the lower physical co-ordinate
b	is the upper physical co-ordinate
c	is the lower signal co-ordinate
d	is the upper signal co-ordinate
"units text"	is the user definable units text

The span number must be in the range of 1 to 20, and must not be a number used for a Polynomial (see above).

The "units text" string is optional, and allows the engineering units for the physical parameter being measured to be defined. The units text string is limited to a maximum of 7 characters, and may contain control characters which are entered in the two character format of eg. ^G (Bell) ^M (CR) ^J (LF) etc.

The dynamic range for the calculated data from spans is ±1.00e-18 to ±1.00e18.

An example of a linear calibration between the output voltage and temperature for an infrared detector is illustrated below

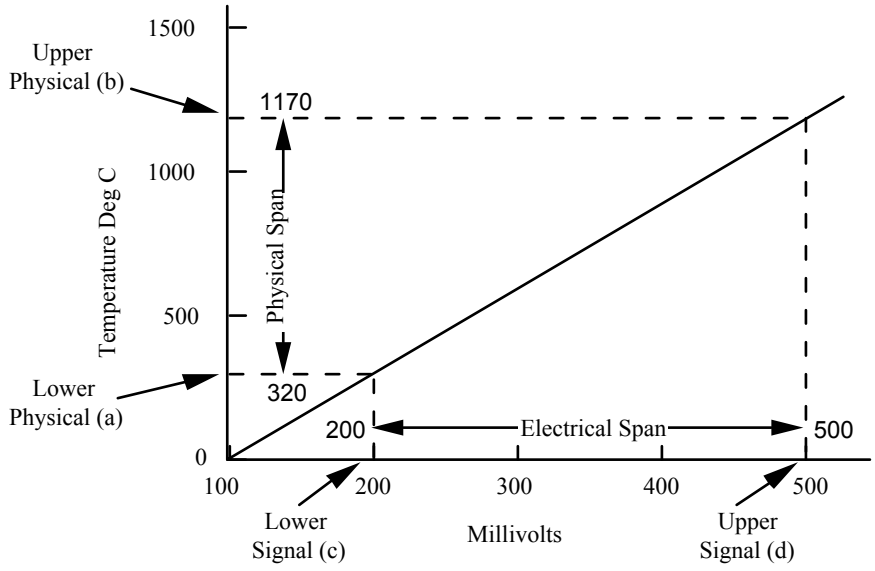


Figure 130 – Infrared Detector Calibration

The infrared detector has an output of 200mV for a temperature of 320°C, and 500mV for a temperature of 1170°C.

Using DeTransfer, the Span for the linear relationship between the temperature and millivolt output signal from the infrared detector is entered into the *dataTaker* as follows

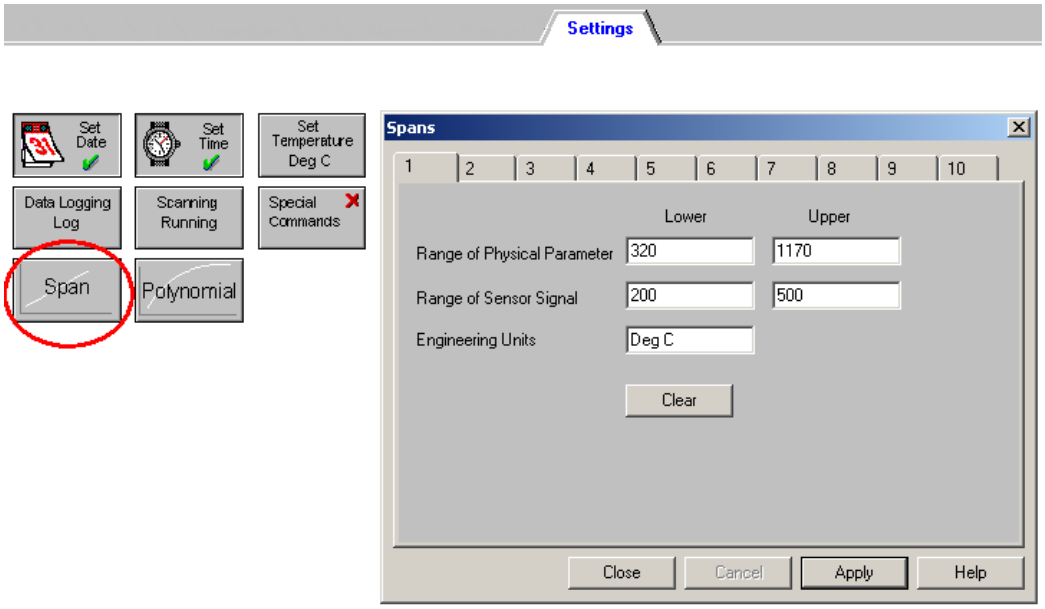
```
S1=320.0,1170.0,200.0,500.0"Deg C"
```

The Span co-ordinates are separated by commas, and there must be no included spaces. Embedded zero co-ordinates must be included to enable following co-ordinates to be given correct ascendancy.

The span co-ordinates may be entered either in floating point format (12.63, 0.001), or in exponential format (1.55e-3, 3.0e2)

Using DeLogger, Spans are entered in the Program Builder. Click on the Settings tab, right click on the Span button and select Properties… to open the Span dialog. Click on the tab for the Span number you are defining, and Enter the co-ordinates and units text for the new Span.

Click Apply to store the values, and either click on the tab for the next Span number to define, or click OK to finish.



DeLogger only allows 10 Polynomials and 10 Spans to be defined rather than any mix totalling 20 as the *dataTaker* allows.

Using Declared Spans

Defined Spans are specified as a Channel Option for input channels, internal channels, Channel Variables, etc to linearise the original data.

Using DeTransfer, this is illustrated in the following program

```
S1=0.0,1500.0,100.0,600.0"Deg C"
S15=20,100,-2500,2500"l/min"

BEGIN
 RA10M
  1V(S1)   3V(S15)
END
```

which will result in data of the format

```
1V   845.36 Deg C
3V   55.438 l/min
```

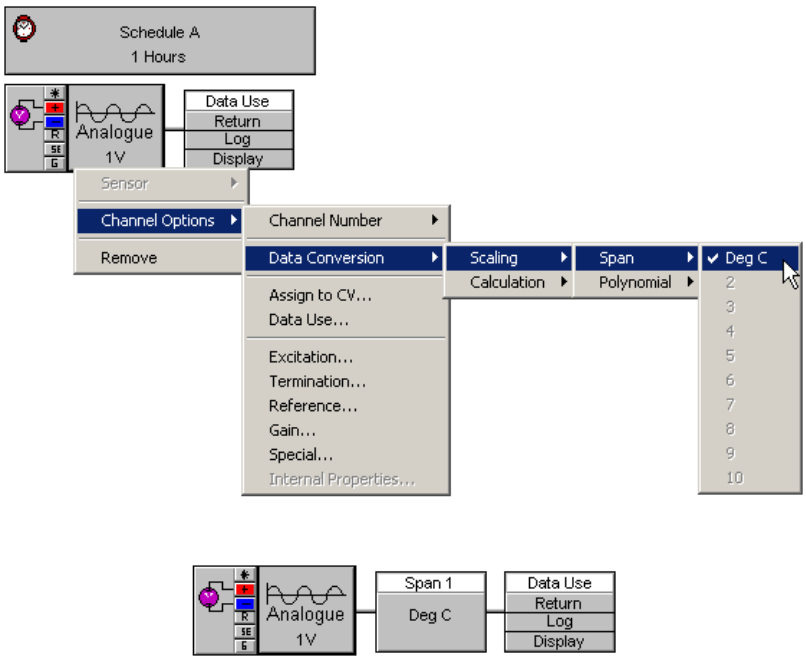
The channel number and units text are returned with the data according to the settings of the (/n /N) and (/u /U) Switches, Parameter22 and Parameter24 (See Section III – Format of Returned Data).

Spans may also be specified as a Channel Option to sequences of channels using DeTransfer as follows

```
R1H   1..5V(S1)   6..8R(S2)
```

the Span S1 is specified as a Channel Option to input channels 1 - 5, and Span S2 is specified as a Channel Option to input channels 6 - 8.

Using DeLogger, defined Spans are specified as Channel Option for input channels, Channel variables, etc in the Program Builder. Firstly, enter the input channel or Channel Variable into the work area of the appropriate Schedule. Right click on the channel icon, and follow the walking menu through Channel Options:Data Conversion:Scaling:Span and select the required Span from the list of defined Spans.



The channel will then change to show the Span being applied between the channel icon and the data use icon.

Spans for Current Loops

While the *dataTaker* allows the Spans to be used for any sensor type, Spans have particular relevance to current loop inputs. The current loop signal is in the range of 4 - 20mA that spans the measurement range of the sensor connected to the current loop transmitter.

The *dataTaker* returns current loop measurements as the percentage of the 4-20mA range that the particular current level represents. For example a 12mA signal in the 4 - 20mA range is returned as 50.00%.

When declaring Spans for current loop inputs, only the two physical span co-ordinates need to be declared. The *dataTaker* defaults the signal span co-ordinates to 0% and 100 %, which is the data range for current loops.

Using DeTransfer, declaring and using Spans for current loop inputs is illustrated below

```
S5=0.0,250.0"KPa"

BEGIN
 RA15M
  7L(S5)
END
```

In this example the sensor pressure range of 0 to 250KPa corresponds to 4 - 20 mA or 0 to 100% of output by the current loop device.

Spans for current loop devices can also be specified as a Channel Option to sequences of channels using DeTransfer as follows

```
R1H  3..5L(S1)
```

where Span S1 is specified as a Channel Option to channels 3 to 5.

The channel number and units text are returned with the data according to the settings of the (/n /N) and (/u /U) Switches, Parameter22 and Parameter24 (See Section III – Format of Returned Data).

Checking the Declaration of Polynomials and Spans

The Polynomials and Spans that have been declared to *dataTaker* can be checked at any time by the STATUS command. The general STATUS command includes information of the number of Polynomials and Spans defined as part of the general report. The STATUS4 command lists the individual definitions of Polynomials and Spans (See Section III – Status Command)

Using DeTransfer, the Polynomials and Spans is checked using the STATUS command as follows

```
STATUS
```

which produces a general report which includes details similar to

```
dataTaker 0 Version 7.00
A B C,none Scan Schedules Active, Halted
3,0 Alarms Active,Halted
2 Polynomial/Spans Defined
Logging is OFF
166350,0 Internal Data Points Free,Stored
79265,1670 Card Data Points Free,Stored
4090,0 Program Characters Free,Stored
/a/C/d/E/f/h/J/K/l/M/N/o/Q/R/S/t/U/v/w/x/y/Z
```

and the command

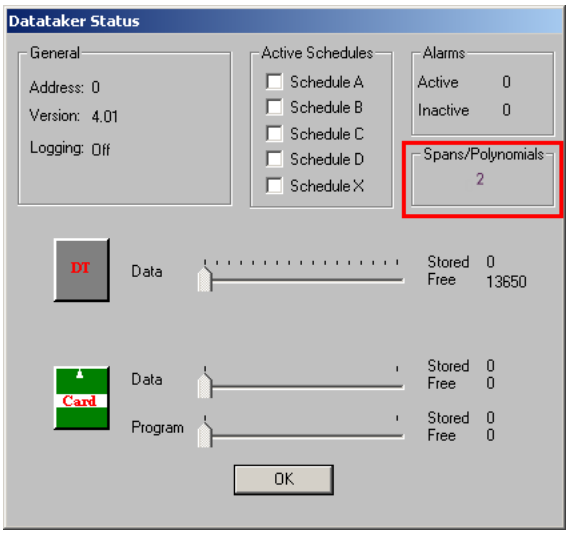
```
STATUS4
```

produces a detailed report of defined Polynomials and Spans similar to

```
2 Polynomials/Span Limits Defined
Y3=1.0e-2,1.21,1.03e1,2.2657"KWh"
S8=0,1000,4,20"Deg C"
```

Using DeLogger, the number of defined Polynomials and Spans can be checked by the dataTaker : Status option from the main menu bar, or the Status button on the main toolbar.

However the actual definition of the Polynomials and Spans are not shown here. These can be found in the Program Builder where the Polynomials and Spans were originally defined.



Obtaining the Span for a Sensor

The span calibration for sensors and current loop transmitters can usually be obtained from the sensor manufacturer.

This information is usually provided either as direct statement of the physical parameter and signal range end points, or a tabulation of the sensor calibration.

If a sensor is being used to measure a physical parameter over a smaller range than the full range of the sensor, then a span can be declared for this sub-range to provide greater resolution.

Channel Variables and Calculations

The *dataTaker* data loggers have a powerful calculations facility to evaluate expressions in real time. The results of calculations can be returned to the host computer, logged to the internal memory or memory card, displayed, used in Alarms, used to set the initial count of counters, used to set the range of System Timers, etc.

dataTaker expressions comprise Channel Variables, constants and intrinsic functions only. While input channels cannot be specified directly in expressions, the data from input channels is first read and assigned to Channel Variables which in turn are used in an expression.

Calculations provide many opportunities in *dataTaker* applications, including

- computation of parameters from single input channel data which involves fixed expressions
- support for sensors with calibrations which cannot be evaluated by Polynomials or Spans
- allow parameters which involve a number of different real time data to be evaluated in real time
- the volume of stored data can be reduced by combining the readings from a number of input channels in real time
- allow alarms to be used for parameters which involve a number of different real time data
- allow alarm setpoints, counter ranges, System Timer ranges, etc. to be dynamically altered in real time
- dynamically changing the High Speed Counter 1 output channel frequency in real time
- manage digital output channels by Boolean logic

Calculations can be used in applications for an infinite number of purposes. Some examples of categories for the use of calculations in applications are listed below as a guide

- calculate the statistical data for a parameter from a number of distributed sensors, for example spatial average and range of temperature
- calculate corrected data for parameters by adjusting for variations in influencing factors, for example correcting fluid flow by Boyles Law
- calculate derived parameters from fundamental data, for example wind sigma from wind speed and direction
- calculate integrals involving multiple input channel data, for example spatial heat sum
- calculate elapsed times, down times, up times, etc.
- calculate theoretical time profiles for a parameter for comparison with measured time profiles, for example temperature rise profile for a furnace

All calculations are performed in the Data Schedules, in the same way as the scanning of input channels.

When calculations are performed in Data Schedules, data can be sequentially collected from the input channels and then used in subsequent calculations in each pass. This procedure will occur each time the Schedule triggers.

Calculations can also be performed in the Immediate Schedule, in which case the calculation is only performed once when the schedule is entered. This operation is similar to performing calculations at the command level, and is the basis of using calculations to initialise Channel Variables, to run calculations from Alarms and to run calculations directly entered by the host.

All calculations involve the use of Channel Variables, to pass real time input channel data to the expression, and/or to receive results of the expression evaluation.

Channel Variables

The *dataTaker* has 100 internal Channel Variables which are used for temporary storage of data. The Channel Variables are floating point registers within the *dataTaker* memory, and are cleared when the *dataTaker* is initially powered up, is hardware reset or executes the RESET command.

Channel Variables are used for a number of purposes including

to receive the data from input channels by assignment at the time of scanning of the channels

to pass input channel data to expressions

to receive the results of expression evaluations

to pass input channel data and results of expressions to the host computer, the internal memory and memory cards, and the display.

to pass input channel data and results of expressions to alarms, counters, System Timers, etc.

The Channel Variables are treated in the same manner as physical input channels, and are included in Immediate Schedules and Triggered Schedules to return the Channel Variable data to the host, to log the Channel Variable data, and to display the Channel Variable data.

The data formatting functions of the channel options, the Switch commands and the Parameter commands can also be applied to the Channel Variables to specify the format of returned real time and stored data.

There are 100 Channel Variables available in the *dataTaker*, which are numbered 1 to 100.

The general formats for the Channel Variables are as follows

nCV(options)

n..mCV(options)

where

CV	is the Channel Variable identifier
n	is a single Channel Variable number
n..m	is a sequence of Channel Variables numbers
(options)	are optional channel options to be applied

The channel options apply to situations where the Channel Variable is used to return, log or display data previously assigned either from input channel scans or expression evaluation.

Units Text for Channel Variables

Channel Variables do not have directly definable units text, because of the infinite range of engineering units which can be appropriate to the results of calculations.

However defined units text can be applied to Channel Variables using 'dummy' Polynomials, as illustrated in the following example

```
Y1=0,1"KWh"
BEGIN
 RA5M
  5CV(Y1,"Power Consumption",FF2)
END
```

where in this case the Polynomial does not change the actual value of the Channel Variable, but adds the units text. The data returned from the Channel Variable will be similar to

```
Power Consumption  57.32 KWh
```

The advantage of this approach is that the Polynomial can also be used to re-scale the value of the Channel Variable if required.

Note also that Channel Options can also be used to name the Channel Variable and format the returned data.

Assigning Input Channel Data to Channel Variables

Input channels cannot be directly included in expressions. The input channel data must first be assigned to Channel Variables, which are then used in expressions. The Channel Variables effectively pass the data to the expression evaluator.

The assignment of input channel data to Channel Variables is performed by including the Channel Variable, with an assignment operator, as a Channel Option to the input channel to be read.

The input channel specification complete with the Channel Variable assignment is included in Schedules, which control the periodic scanning of the input channel and the assignment of the data to the Channel Variable.

The data from input channels is assigned to Channel Variables in a number of ways.

Direct Assignment

=nCV assign the input channel reading to the Channel Variable, overwriting any previous value.

Addition Assignment

+=nCV add the input channel reading to the value stored in the Channel Variable.

Subtraction Assignment

–=nCV subtract the input channel reading from the value stored in the Channel Variable.

Multiplication Assignment

✳=nCV multiply the value stored in the Channel Variable by the input channel reading.

Division Assignment

/=nCV divide the value stored in the Channel Variable by the input channel reading.

Using DeTransfer, input channel data is transferred to Channel Variables by commands for example

```
1TK(=3CV)
```

instructs the *dataTaker* to return the temperature read for a type K thermocouple connected to analog input channel 1, and assign the reading to Channel Variable 3 replacing any previous value.

```
2C(R,+=9CV)
```

instructs the *dataTaker* to read low speed counter channel 2, reset the counter when it is read, and add the count to the value which is stored in Channel Variable 9 (this procedure overcomes the problem of the 16 bit low speed counter registers overflowing at 65535 counts).

```
2F(/=23CV)
```

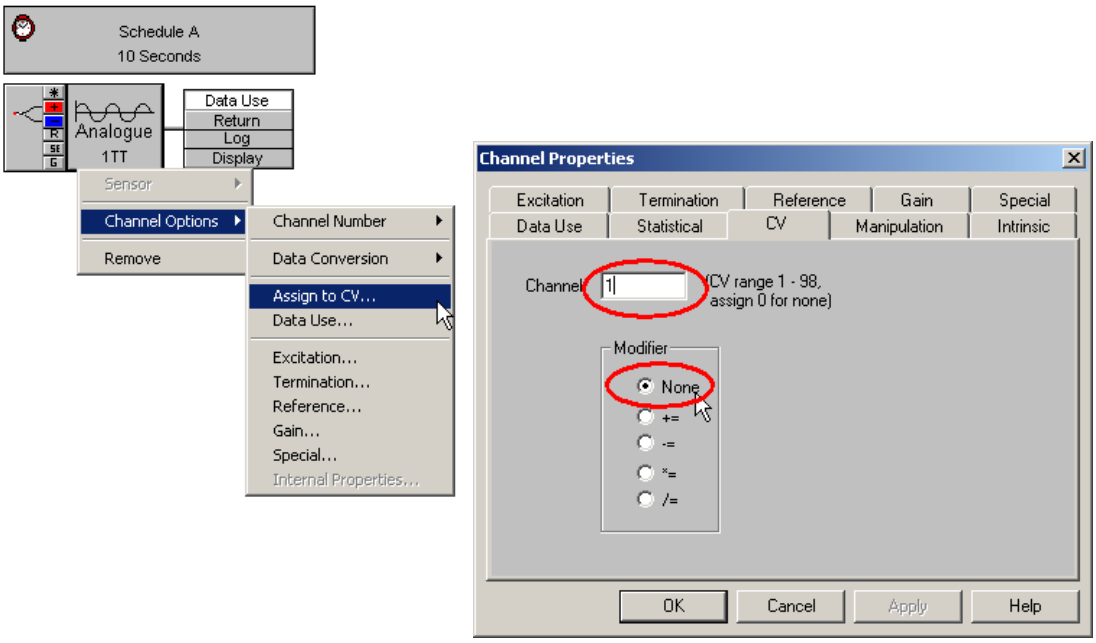
instructs the *dataTaker* to read the frequency on analog channel 2, and divide the value stored in Channel Variable 23 by the value of the new reading.

The assignment of input channel data to Channel Variables is made when the embracing Schedule (RA, RB, RC, RD, RX) executes. For example

```
BEGIN
 RA10S
  1TT(+=1CV)
END
```

in this case the temperature sum will accumulate every 10 seconds.

Using DeLogger, input channel data can be assigned to Channel Variables by the following program in the Program Builder. Firstly define the input channel , then right click on the channel icon and select Channel Options:Assign to CV… This opens the Channel Properties dialog with the CV tab in view. Enter the number of the Channel Variable to receive the reading, and select the modifier (operator) to use. The 'None' modifier means direct assignment.



If the input channel specification contains other data manipulation Channel Options such as Polynomials, Spans, Intrinsic Functions, differences and integrals, these have precedence over assignment to Channel Variables. The Channel Variable receives the result of all data manipulations of the input channel data.

This is illustrated by the following program example

```
BEGIN
 RA5M
  3V(Y1,DF,=12CV)
END
```

where the Channel Variable is assigned the difference between this and the previous channel reading which have been converted by the polynomial.

The order of application of the various data manipulation functions is maintained irrespective of the order of their listing in the channel options list. The assignment of channel data to Channel Variables always follows processing of the data manipulation, such that the Channel Variable receives the final result.

Assigning Statistical Data to Channel Variables

The various statistical data that is produced by the Statistical Sub Schedule can also be assigned to Channel Variables. When a Channel Variable is included as a Channel Option for input channels that are statistically scanned, then the statistical result is assigned to the Channel Variable rather than the individual data readings. This is illustrated by the following example

```
BEGIN
 RS5S
 RA10M
  3V(AV,=1CV)(MX,=2CV)(MN,=3CV)
END
```

in which the voltage connected to analog input channel 3 is sampled every 5 seconds (RS5S). Every 10 minutes (RA10M) the average, the minimum and the maximum values are calculated and assigned to the Channel Variables 1CV, 2CV, and 3CV respectively.

Assigning Results of Expressions to Channel Variables

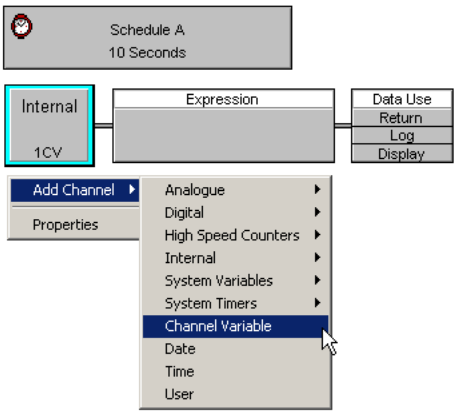
Results of expressions are generally assigned to Channel Variables. The Channel Variable is entered as the component to the left of the assignment operator of the equation, and the expression is entered as the component to the right of the assignment operator

Using DeTransfer, the result of expressions is assigned to the Channel Variables for example

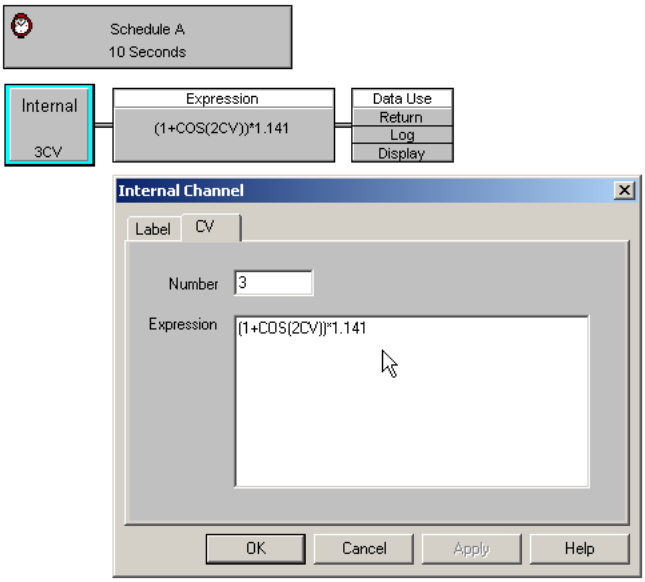
```
3CV=(1+COS(2CV))*1.141
```

where the expression is evaluated and assigned to 3CV.

Using DeLogger, the result of expressions is assigned to the Channel Variables in the Program Builder. Firstly define the Channel Variable to receive the result of the expression



Right click on the Expression icon, and enter the expression and Channel Variable number



The use of Channel Variables with expressions is discussed in detail in the Calculations section below.

Assigning Channel Variables to Channel Variables

Channel Variables can be assigned to other Channel Variables for various purposes including to temporarily store data, to carry the data to different calculations, to reverse the sign of the data, etc.

Channel Variables can be assigned to other Channel Variables by either of two methods.

• one Channel Variable can be assigned to another Channel Variable by an expression, as illustrated by the following examples

```
1CV=2CV
15CV=-23CV
```

• one Channel Variable can be assigned to another Channel Variable by specifying the second Channel Variable as a channel option of the first Channel Variable, as illustrated by the following examples

```
9CV(=3CV)
15CV(+=12CV)
```

where in each case the value of the Channel Variable specified as the 'channel' is assigned the value of the Channel Variable specified as the channel option.

The second example illustrates the use of the assignment operators which are available for assignment via channel options, and is equivalent to the expression

```
12CV=12CV+15CV
```

Channel Options for Channel Variables

Channel Options can be used with Channel Variables to perform data manipulation, scaling, statistical, data destination and data formatting functions that are available for the input channels. The Channel Options that are appropriate to Channel Variables are listed below

reset (R) the Channel Variable to zero. For example

```
12CV(R)
```

• scale the value stored in Channel Variables by a scale factor (f.f), Polynomials (Yn), Spans (Sn) or Intrinsic Functions (Fn). For example

```
52CV(Y3)

18CV(10.5,F3)
```

• return the difference (DF), rate of change (RC and RS) or integral (IB) between successive assignments to Channel Variables. For example

```
5CV(DF)

15CV(IB)
```

• return the average (AV), standard deviation (SD), minimum (MN), maximum (MX), date of minimum and maximum (DMN and DMX), the time of minimum (TMN and TMX) and histogram (Hx:y:n..mCV) for Channel Variables. For example

```
16CV(AV)(SD)(MN)(MX)

7CV(H:-100:100:20..30CV)
```

- assign unique names to Channel Variables. For example

```
33CV("Corrected Fuel Flow")
```

  set the numeric format of Channel Variable data to Floating Point (FFn), Exponential (FEn) or Mixed (FMn). For example

```
52CV(FF2)
```

  set destinations for the data from Channel Variables for not returned (NR), not logged (NL), not displayed (ND). For example

```
27CV(NL,ND)
```

  set Channel Variables as working or intermediate variables (W) for which data is to be retained internally for further calculation, use in alarms, etc. For example

```
BEGIN
 R1M
  1CV(W)=(5CV+6CV)/2
 ALARM3(1CV>125)"Oven overheating"
END
```

- define a bar graph (BGx:y) for displaying Channel Variable data. For example

```
8CV(BG0:100,"Slow    %    Fast")
```

Using the Channel Variables

Channel Variables are used in the same manner as input channels in Data Schedules, the Statistical Sub Schedules and Alarms. Assignment to Channel Variables, returning of Channel Variable data, logging of Channel Variable data and displaying of Channel Variable data is carried out when the associated Schedule triggers and executes.

Channel Variables do not have user definable units text, however the polynomials can be used to append units text to returned, logged and displayed data (See Units Text for Channel Variables above).

Using Channel Variables in Schedules

Channel Variables are only processed whenever the embracing Data Schedule or Alarm Schedule runs. When Channel Variables are included in Schedules then data can be sequentially assigned to the Channel Variables from the input channels, and used in subsequent calculations in each pass. This procedure will occur each time that the Schedule runs.

When Channel Variables are included in Immediate Schedules, the Channel Variable is processed once when the schedule is entered. This is similar to performing calculations at the command level, and is the basis of initialising Channel Variables, assigning values to Channel Variables from Alarm, and assigning values to Channel Variables directly from the host.

Using DeTransfer, a program using Channel Variables in a Schedule is illustrated below

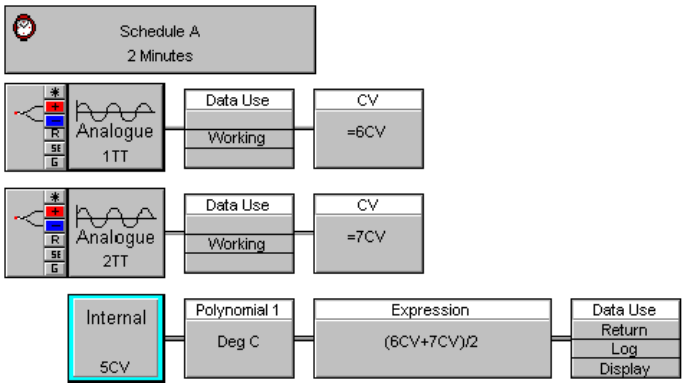
```
Y1=0,1"Deg C"
BEGIN
 RA2M
  1TT(=6CV,W)
  2TT(=7CV,W)
  5CV("Mean Temp",Y1,FF1)=(6CV+7CV)/2
END
```

which measures two temperatures and temporarily saves the data in 6CV and 7CV, calculates the average temperature and assigns this to 5CV, and returns 5CV. This sequence repeats every 2 minutes, and returns data each time the Schedule runs that is similar to

```
Mean Temp  23.5 Deg C
Mean Temp  24.6 Deg C
Mean Temp  27.1 Deg C
```

Note : The Working (W) Channel Option is used to define working steps where the temperatures are measured and assigned to Channel Variables for use in the later calculation. If the W channel option is omitted from these steps, then the raw temperature data will also be returned to the host. A dummy Polynomial is used to append units text to the average temperature data.

Using DeLogger, this same program is entered into the Program Builder as follows



DeLogger does not provide a way to name or label a Channel Variable channel.

Using Channel Variables as the Test Data in Alarms

Channel Variables can be used in the Alarms as the test data that is tested for alarm state (See Section III - Alarms). The use of Channel Variables as the test data for Alarms allows the results of calculations to be tested in alarms for range, etc.

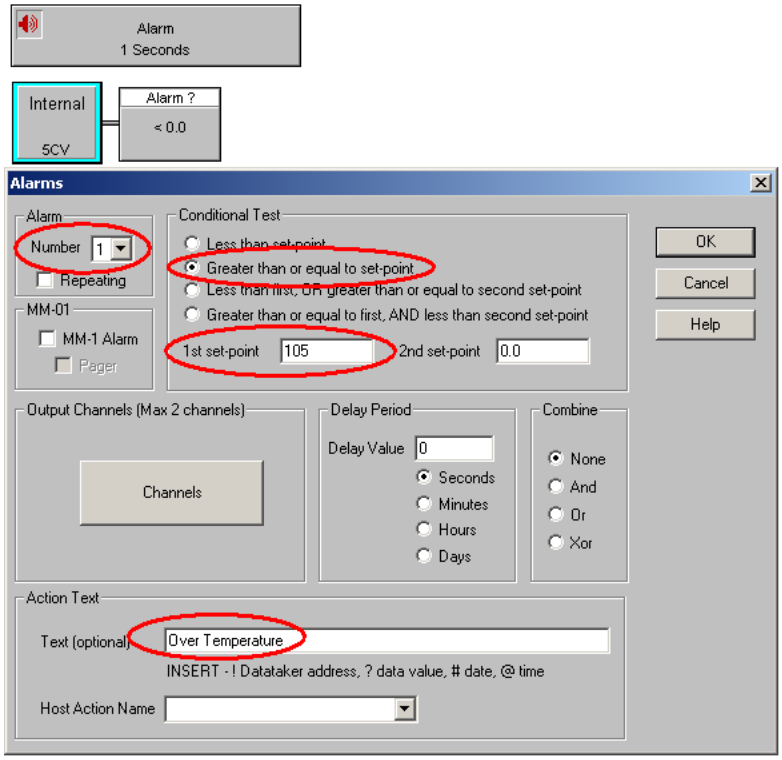
The following DeTransfer program for example measures two temperatures and calculates the average temperature, which is then tested in an alarm.

```
BEGIN
 RA5S
  1TT(=6CV,W)
  2TT(=7CV,W)
  5CV(W)=(6CV+7CV)/2
 RZ5S
  ALARM1(5CV>105)"Over Temperature"
END
```

Note that in this type of program the Schedule (RA5S) in which the raw data is collected and the calculation performed must execute at the same interval as the Alarm Schedule (RZ5S).

The Alarm Schedule is executed after the Data Schedule in each scan, and so data passed forward in this way is appropriate to the current test for the alarm.

Using DeLogger, this alarm step can be added to the program above in the Program Builder as follows



Channel Variables can also be used in Alarms to test data from the same input channel in several alarms. This increases the speed of alarm processing, because the input channel only needs to be read once rather than for each alarm.

This use of Channel Variables in this way is illustrated in the following DeTransfer example

```
BEGIN
  ALARM1(1V(=1CV)>0.5)"Over 0.5 Volts"
  ALARM2(1CV>0.6)"Over 0.6 Volts"
  ALARM3(1CV>0.7)"Over 0.7 Volts"
END
```

This approach can also be used to test input channel data which is continuously varying.

Using Channel Variables as Setpoints in Alarms

Channel Variables can also be used within the Alarms as setpoints, against which the test data is tested for alarm state (See Section III – Alarms). This allows the results of calculations to be used as range limits that can be dynamically changed as the program runs, for the testing the test data.

The following program for example measures two temperatures and tests one against the other in an alarm.

```
BEGIN
 RA5S
  1TT(=6CV,W)
  2TT(=7CV,W)
 RZ5S
  ALARM1(6CV>7CV)"Uneven Temperature"
END
```

Note that in this type of program the Data Schedule (RA5S) in which the raw data is collected must execute at the same interval as the Alarms Schedule (RZ5S).

Using Channel Variables to Transfer Statistical Data to Alarms

Statistical data (AV, MN, MX, SD, etc) cannot be tested in an Alarm directly (See Section III – Alarms). To test statistical data, the data must first be obtained in a Data Schedule/Statistical Sub Schedule, and assigned to a Channel Variable which is then tested in an Alarm.

This same approach must be used when testing difference, rate and integral data in Alarms.

The following program example demonstrates this approach

```
BEGIN
 RS5S
 RA1M
  3PT385(SD,=1CV,W)(AV,=2CV,W)
 RZ1M
  ALARM1(1CV>1.0)"Excess Variability"
  ALARM2(2CV>125)"High Temperature"
END
```

The raw data is collected every 5 seconds (RS5S), and the statistical data is calculated every 1 minute (RA1M). Note that in this type of program the Data Schedule (RA1M) in which the statistical data is calculated must execute at the same interval as the Alarms Schedule (RZ1M).

Using Channel Variables with Date and Time

Date, Time and System Timers of the real time clock can be assigned to Channel Variables for temporary storage of time points, and for use in calculations based on time such as elapsed time, down time, up time, rates, etc.

Date is stored in Channel Variables as the number of days elapsed since 01/01/1989. This format is irrespective of the Date format defined by the Parameter31 command.

Time of day is stored in Channel Variables as the number of seconds elapsed since the last midnight, or as the number of seconds for the current day. This format is irrespective of the Time format defined by the Parameter39 command.

Data for each of the System Timers is stored in Channel Variables as the number of counts registered in the particular timer, in the units of time for the particular timer.

Assigning Date, Time and System Timers to Channel Variables is illustrated in the following examples

```
D(=1CV)
T(=2CV)
2ST(=3CV)
```

The following program illustrates the use of Channel Variables to calculate the on time and the off time for a switch monitored by a Schedule Triggered by Digital Event

```
1CV(W)=0              'Clear ON time
2CV(W)=0              'Clear OFF time
T(=6CV,W)             'Program start time
BEGIN
 R1+E                 'Switch turned ON
  T(=5CV,W)
  2CV=2CV+(5CV-6CV)   'Total OFF time
 R1-E                 'Switch turned OFF
  T(=6CV,W)
  1CV=1CV+(6CV-5CV)   'Total ON time
END
```

This example program is only suitable for use over a single day. If the program runs longer than one day, then calculation of elapsed times will have to account for the Time resetting to zero at midnight.

Calculations

The *dataTaker* can evaluates expressions in real time. The results of these calculations can be returned to the host computer, logged into memory, displayed, used in Alarms, used to set initial count of counters, used to set the range of System Timers, etc.

The *dataTaker* calculations are based on expressions which can contain channel variables, numeric constants and mathematical functions.

Input channels cannot be specified directly in expressions. The data from input channels must first be sampled and assigned to Channel Variables, which in turn can be used in the expressions (see Channel Variables above).

The use of calculations provides many opportunities in applications, including

- computation of parameters from single input channel data which involves fixed expressions
- support for sensors with calibrations which cannot be evaluated by polynomials or spans
- allow parameters which involve a number of different real time data to be evaluated in real time
- the volume of stored data can be reduced by combining the readings from a number of input channels in real time
- allow alarms to be used for parameters which involve a number of different real time data being calculated to a single value
- allow alarm setpoints, counter ranges, System Timer ranges, etc. to be dynamically altered in real time
- dynamically changing the High Speed Counter 1 output channel frequency in real time
- manage digital output channels by Boolean logic

Some examples of categories for use of calculations in applications are listed below as a guide

- calculate the statistical data for a parameter from distributed sensors, for example spatial average and range of temperature variation in a room
- calculate corrected data for parameters by adjusting for influencing factors, for example correcting flow for temperature and pressure according to Boyles Law
- calculate derived parameters from fundamental data, for example wind sigma from wind speed and direction
- calculate elapsed times, down times, up times, rates, etc.
- calculate theoretical profiles for a parameter for comparison with measured profiles, for example temperature rise profile of furnaces

All calculations are performed in Data Schedules, in the same way as the scanning of input channels.

Expressions Operators

The *dataTaker* expression evaluator supports the standard range of arithmetic, relational and logical operators, and supports a number of mathematical functions. The expression operators supported by the *dataTaker* are listed below

Arithmetic	Addition	+
	Subtraction	–
	Multiply	*
	Divide	/
	Modulus	%
	Exponent	^

Relational	Less than	<
(1 is true)	Less than or equal to	<=
(0 is false)	Equal to	=
	Not equal to	!=
	Greater than or equal to	>=
	Greater than	>
Logical		AND
($^a\neq$0 is true)		OR
(=0 is false)		XOR
		NOT
Functions	Absolute	ABS()
	Logarithm base 10	LOG()
	Natural logarithm	LN()
	Sine	SIN()
	Cosine	COS()
	Tangent	TAN()
	ArcSine	ASIN()
	ArcCosine	ACOS()
	ArcTangent	ATAN()
	Square Root	SQRT()
Precedence	Parentheses	( )

The SIN(), COS() and TAN() functions require arguments in radians, where 1 radian = 57.296 degrees. The ASIN(), ACOS() and ATAN() functions return data in radians.

Polynomials, spans and intrinsic functions cannot be used directly in expressions, but can be applied as channel options to the Channel Variable associated with expressions to further process the result of an expression.

Operator Precedence

The order of evaluation of the components of an expression follows an operator precedence as follows

( )

NOT – (unary)

* / ^ %

+ –

> >= < <= = !=

AND OR XOR

= (assignment)

The order of precedence is in descending order down the list.

When more than one operator has the same level of precedence, then evaluation follows the order left to right in the list above.

The parentheses ( ) operator can be used to define a particular order of evaluation, with expressions in the parentheses being executed first.

For example the two expressions below will evaluate as follows

```
1CV=2+3*10

1CV=(2+3)*10
```

the first expression will evaluate as 32, while the second expression will evaluate as 50.

Parentheses can be nested to further define the order of evaluation, as illustrated in the following example

```
5CV=((1CV+(2CV-0.0231))/COS(3CV*4CV))/3.3
```

Expressions can be entered directly into the logger from DeTransfer, and can be included in programs developed in the Program Builder of DeLogger.

Execution of Calculations

All calculations are evaluated in Schedules, in the same way as the scanning of input channels.

When calculations are included in Schedules, the input channels to be scanned and calculations to be evaluated can be mixed in any order within the Schedule. When the schedule runs, the input channels are scanned and the calculations are evaluated in that same order as the order of entry.

Therefore data can be sequentially collected from the input channels and used in subsequent calculations in each pass. This sequence will repeat each time the Schedule runs. Several of the examples given in this chapter employ this sequencing.

The individual Schedules are executed in the order RA, RB, RC, RD then RX when the triggers for two or more of the Schedules coincide. Therefore with careful planning, the data collected in one Schedule can also be synchronously used in calculations within another Schedule .

Calculations can also be performed in the Immediate Schedule, in which the calculation is performed once when the schedule is entered into the logger. This has the effect that the calculations are performed at command level. This feature is the basis of using calculations to initialise Channel Variables, to run calculations from Alarms, and to execute calculations that are directly entered by the host.

Maximum Number of Calculations

All calculations are evaluated in Schedules, and Channel Variables receiving the results of calculations are entered into the internal channel table with the input channels for data acquisition and Alarms.

The Channel Variables which receive the results of calculations share that part of the channel table which is allocated to the data acquisition Schedules.

Therefore the number of calculations which can be performed in an application is governed by the proportion of the internal channel table that is allocated to the Data Schedules (defined by Parameter30), and the number of input channels being scanned.

For example, if Parameter30=40, then there is space for 40 Alarms and space for up to 70 channel accesses (input or output) and calculations. This could be distributed as 20 channel accesses and 50 calculations, or 35 channel accesses and 35 calculations, etc.

The expressions for calculations are collectively stored in a separate area of memory, and are limited to a total of 3848 characters.

Conditional Calculations

The *dataTaker* has no formal commands for conditional evaluation of expressions. However conditional evaluation can be performed by using Relational or Boolean expressions, or Alarms.

Boolean expressions employing Boolean logic can be used to return results that are dependent on a condition being true or false. Boolean expressions return a value of 0 if the result of the relational test is false, and return a value of 1 if the result of the relational test is true.

The following expressions will evaluate

```
(2CV>=100)

(15CV=25.5)
```

The first Boolean expression will return 1 if the value of 2CV is greater than or equal to 100, and will return 0 if the value of 2CV is less than 100.

The second Boolean expression will return 1 if the value of 15CV is equal to 25.5, and will return 0 if the value of 15CV is greater than or less than 25.5.

The value returned by Boolean expressions can be used to control evaluation of conditional expressions, that are arranged to add the results of two sub-expressions of which one will evaluate as false and one will evaluate as true.

The following example illustrates the use of Boolean expressions for conditional evaluation.

```
2CV=(1CV*2*(1CV<100))+(1CV*4*(1CV>=100))
```

If 1CV is less than 100, then the first Boolean expression will return a value of 1, and the second Boolean expression will return a value of 0. The conditional expression will therefore reduce to

```
2CV=(1CV*2*(1))+(1CV*4*(0))
or
2CV=(1CV*2*(1))+0
```

and 2CV will be assigned the value of 1CV*2.

If however 1CV is greater than or equal to 100, then the first Boolean expression will return a value of 0, and the second Boolean expression will return a value of 1. The conditional expression will therefore reduce to

```
2CV=(1CV*2*(0))+(1CV*4*(1))
or
2CV=0+(1CV*4*(1))
```

and 2CV will be assigned the value of 1CV*4.

The BASIC language equivalent of this conditional expression is

```
IF 1CV<100 THEN 2CV=1CV*2 ELSE 2CV=1CV*4
```

The same conditional evaluation can also be performed by 2 Alarms as follows

```
ALARM1(1CV<1000)"[2CV=1CV*2]"
ALARM2(1CV>1000)"[2CV=1CV*4]"
```

Alarms must be used with care in conditional calculations

the Alarm Schedule (RZ) must trigger at the same intervals as any Schedules which use the result

the Alarms are executed after the Schedule, and so the result of the conditional calculation will not be available until the next iteration of the Schedule

the Alarms execute the action commands each time that the alarm state becomes true and not while the alarm state remains true (unless the repeating Alarms ALARMR are used)

Standard Conditional Calculations

Several standard conditional calculations are listed below that can be used as a guide for *dataTaker* programs. This list is not exhaustive, however it provides a number of models for further development.

maintain a maximum value

```
2CV(W)=-99999     'Initial maximum
BEGIN
 RA10S
  1V(=1CV,W)
  2CV=2CV*(1CV<=2CV)+1CV*(1CV>2CV)   'Keep largest value
END
```

maintain a minimum value

```
2CV(W)=999999      'Initial minimum
BEGIN
 RA10S
  1V(=1CV,W)
  2CV=2CV*(1CV>=2CV)+1CV*(1CV<2CV)   'Keep smallest value
END
```

count the number of readings outside of a range

```
2CV(W)=0      'Initial count
BEGIN
 RA10S
  1V(=1CV,W)
  2CV=2CV+((1CV<100)OR(1CV>500))   'Add 1 if value<100 or value>500
END
```

count the number of readings within a range

```
2CV(W)=0      'Initial count
BEGIN
 RA10S
  1V(=1CV,W)
  2CV=2CV+((1CV>100)AND(1CV<200))   'Add 1 if 100>value<200
END
```

correct sensor readings for errors according to range

```
BEGIN
 RA1M
  1V(=1CV,W)
  1CV=1CV-0.5*((1CV>=25)AND(1CV<35))'Subtract 0.5 if 25<value<34.99
  1CV=1CV-1.5*((1CV>=35)AND(1CV<45))'Subtract 1.5 if 35<value<44.99
  1CV=1CV-2.5*((1CV>=45)AND(1CV<55))'Subtract 2.5 if 45<value<54.99
END
```

count elapsed ON time and elapsed OFF time for a digital input

```
2CV(W)=0    'Clear elapsed ON time
3CV(W)=0    'Clear elapsed OFF time
BEGIN
 RA1S
  1DS(=1CV,W)
  2CV=2CV+(1CV>0.5)       'ON time
  3CV=3CV+(1CV<0.5)       'OFF time
END
```

Combining Calculations

The different scaling and calculation functions of the *dataTaker* can be used together as illustrated in the following programs.

This program calculates the cross channel average of 10 temperatures

```
BEGIN
 RA10S
  1CV(W)=0  'Clear 1CV
  1..10TT(+=1CV,W) 'Sum the 10 temps into 1CV
  1CV=1CV/10'Divide by 10 for average
END
```

This program calculates the mean and standard deviation of wind direction, and the mean and standard deviation of wind speed. The inputs are wind speed and direction.

```
'Program to calculate the
'    - mean wind speed
'    - mean wind direction
'    - std devn of wind speed
'    - std devn of wind direction

'Define the wind sensor calibrations
Y1=0,0.050"m/s"                'Wind speed 0-1000mV = 0-50 m/s
Y2=0,0.360"Deg"                'Wind dirn  0-1000mV = 0-360 deg

'Define units text for calculations
Y3=0,1"m/s"
Y4=0,1"Deg"

'Clear accumulators
10..16CV(W)=0

BEGIN
 'Scan wind set every 5 secs
 RA5S
  1V(Y1,=1CV,W)                 'Read wind speed and save
  2V(Y2,=2CV,W)                 'Read wind dirn and save
  2CV(W)=2CV/57.29              'Convert wind dirn to radians
  10CV(W)=10CV+COS(2CV)         'Sum of COS(wd)
  11CV(W)=11CV+SIN(2CV)         'Sum of SIN(wd)
  12CV(W)=12CV+1CV              'Sum of ws
  13CV(W)=13CV+(1CV*1CV)        'Sum of ws * ws
  14CV(W)=14CV+(1CV*COS(2CV))   'Sum of ws * COS(wd)
  15CV(W)=15CV+(1CV*SIN(2CV))   'Sum of ws * SIN(wd)
  16CV(W)=16CV+1.0              'Number of scans

 'Calculate wind data every minute
 RB1M
  D                             'Calculation date
  T                             'Calculation time
  20CV(W)=14CV/16CV             'Mean ws * COS(wd)
  21CV(W)=15CV/16CV             'Mean ws * SIN(wd)

  'Calculate mean wind speed
  22CV(W)=SQRT((20CV*20CV)+(21CV*21CV))
  22CV("Mean Wind Speed",Y3,FF2)

  'Calculate mean wind direction
  23CV(W)=ATAN(21CV/20CV)*57.29
```

```
  'Find quadrant for mean wind direction
  23CV(W)=23CV+((20CV>0)AND(21CV<0))*360      '4th quadrant
  23CV(W)=23CV+((20CV<0)AND(21CV<0))*180      '3rd quadrant
  23CV(W)=23CV+((20CV<0)AND(21CV>0))*180      '2nd quadrant
  23CV(W)=23CV+((20CV>0)AND(21CV>0))*0        '1st quadrant
  23CV(W)=23CV-(12CV=0)*(23CV+1)              'No wind, return -1
  23CV("Mean Wind Dirn",Y4,FF2)

  'Calculate std devn of wind speed
  24CV(W)=13CV-((12CV*12CV)/16CV)
  24CV(W)=SQRT(24CV/16CV)
  24CV("SD Wind Speed",Y3,FF2)

  'Calculate std devn of wind dirn
  25CV(W)=10CV/16CV
  26CV(W)=11CV/16CV
  27CV(W)=SQRT((25CV*25CV)+(26CV*26CV))
  27CV(W)=SQRT(-2.0*LN(27CV))*57.29
  27CV("SD Wind Dirn",Y4,FF2)

  'Clear accumulators
  10..16CV(W)=0
END
LOGON
```

Calculation Errors

When data from input channels is used in calculations, then any measurement errors which result in the input channel returning the universal error value of 99999.9 will also force the result of the calculation using that data to 99999.9.

Forcing of the result of the calculation to 99999.9 can be disabled, and the value 99999.9 is used in the calculation which will return a result based on this value.

The method for handling over-range channel data in calculations is determined by the Over Range Error Carry Switch as follows

- /J Carry error through to result of calculations (Default)
- /j Substitute error with 99999.9 and perform calculation

The Over Range Error Carry Switch defaults to /J when *dataTaker* is powered up, is hardware reset or executes a RESET command.

The following examples demonstrate the function of the Over Range Error Carry Switch

```
/J
1V(=1CV)
2CV=1CV/2
```

any over-range errors are carried through, and the expression returns 99999.9

```
/j
1V(=1CV)
2CV=1CV/2
```

any over-ranged data is substituted with 99999.9, and the expression returns 49999.95.

Histogram

The histogram is based on the statistical standard, the Normal Distribution, and is a method of reducing large volumes of data into a statistical description of the variability of that data set.

The histogram is defined to have an overall range of values, and a number of classes of equal interval value within the overall range. For example a histogram may have a range of 0 to 100 units, and have 10 classes each of 10 units where class 1 is 0 – 9.99 units, class 2 is 10 – 19.99 units, class 3 is 20 – 29.99 units, etc

Each of the values in the data set is inspected for range, and is counted in the class interval in which its value fits. The result is a count or frequency or values from the data set which fall into each of the class intervals. Hence the alternate name for histogram – frequency distribution.

A Histogram or Frequency Distribution is illustrated in Figure 131 below.

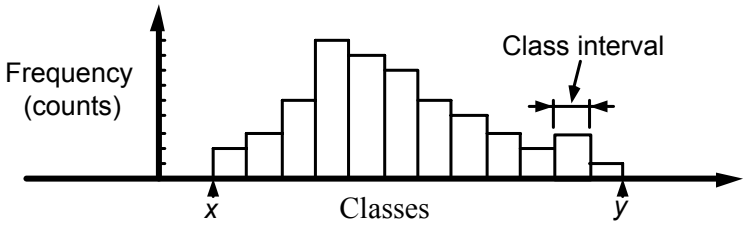


Figure 131 – The Histogram or Frequency Distribution

The *dataTaker* Histogram is implemented as a Channel Option, and can be used with any channel type to progressively build up a histogram of a data set as the data set is collected. The class frequencies are counted into a sequential group of Channel Variables.

The histogram is specified as a Channel Option in the format

Hx:y:n..mCV

where

x	is the lowest data value of interest
y	is the highest data value of interest
n	is the first Channel Variable to store counts (nCV)
m	is the last Channel Variable to store counts (mCV)

The overall range of the histogram classes is set by x and y, and the number of classes is set by the number of Channel Variables between n..m.

The *dataTaker* allocates all but 3 of the Channel Variables for classes, and class intervals are calculated by the *dataTaker* for this number of Channel Variables over the range.

The *dataTaker* also stores 3 other counts into Channel Variables as follows

the number of readings under the range specified (<x)

the number of readings over the range specified (>y)

the total number of readings, including those outside of the range

Programming the Histogram

Using DeTransfer, a histogram of a measured temperature over 5 classes is programmed in the example below. It is necessary to allocate 8 Channel Variables, 5 for the classes, 1 for under range frequency, 1 for over range frequency, and 1 for total number of readings

```
1TT(H25:35:1..8CV)
```

In this example a histogram will be generated with 5 temperature classes with intervals of 2°C as follows

1CV	counts for first class (25 - 27°C interval)
2CV	counts for second class (27 - 29°C interval)
3CV	counts for third class (29 - 31°C interval)
4CV	counts for fourth class (31 - 33°C interval)
5CV	counts for fifth class (33 - 35°C interval)
6CV	number of samples under range (<24°C)
7CV	number of samples over range (>35°C)
8CV	total counts or the sum of 1..7CV

The complete program requires two basic activities. The input channel is sampled over a period, and the resulting data is stored into the appropriate classes. At some longer interval the Channel Variables are returned or logger, to recover the Histogram data. For example

```
BEGIN
 RA30S
  1TK(H0:100:1..13CV,W)    'Collect primary data
 RB1H
  1..13CV(R)               'Return the class frequencies
 END
```

The Channel Variables can be cleared each time Schedule B returns the frequencies, or not cleared and left to continue to accumulate frequencies, depending on requirements.

The frequency counts that are generated can be normalised using the total number of values as follows

```
1CV=1CV/8CV
2CV=2CV/8CV
3CV=3CV/8CV
etc.
```

DeLogger does not support the Histogram function.

Logging Data to Memory

The *dataTaker* data loggers have two types of memory for logging data

internal SRAM memory. The internal memory for Series 1 and Series 2 *dataTakers* stores 13,650 data items, and for Series 3 loggers stores 166,530 data items

memory cards that can be inserted into the logger. Series 1 and 2 *dataTakers* accept 512 Kbyte and 1 Mbyte cards, and Series 3 loggers accept 512 Kbyte, 1 Mbyte, 2 Mbyte and 4 Mbyte.

The Data Logging commands are similar for storing data in the internal data memory, and for storing data in a memory card. When a memory card is not inserted in the logger, the Data Logging commands apply to the internal data memory. When a memory card is inserted, the Data Logging commands apply to the memory card. The memory card has the highest priority for storage of data.

The internal data memory is used as a temporary data storage buffer for the memory card when memory cards are used. The internal data memory cannot be used as a separate area of memory when a memory card is installed.

The following descriptions of the Data Logging commands refer to a 'data memory'. If a memory card is inserted, then the data memory is the memory card. If however a memory card is not inserted, then the data memory is the internal memory.

Controlling Logging of Data

The data produced by the Schedules is returned to the host computer by default, and is not logged into the internal data memory or memory card by default. Data logging has to be specifically enabled.

The logging of data can be controlled globally for all input channels scanned, and can be controlled for individual channels.

Logging Data - Global Control

Logging of real time Schedule data into the internal data memory or a memory card is globally enabled and disabled by the Data Logging commands as follows

LOGON	Enables data logging
LOGOFF	Disables data logging (Default)

The Data Logging commands can be included in programs, or can be entered separately as direct commands during system configuration.

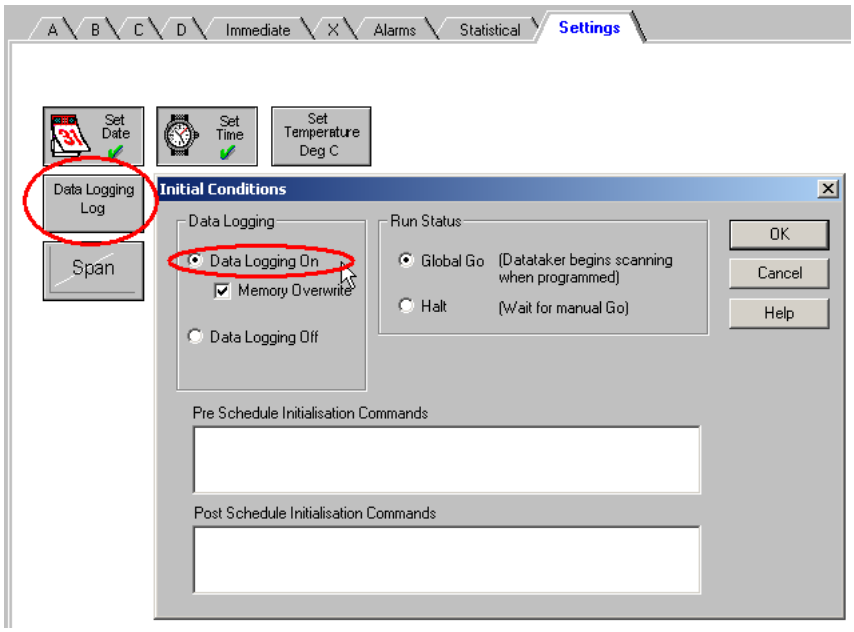
The Data Logging command defaults to LOGOFF after a power up, or after execution of a RESET command or a CLEAR command.

The LOGON command must be entered after all schedules have been entered, because it 'locks' the internal channel table. If you attempt to enter Schedules after a LOGON command, then an 'E4-clear data memory' error will be returned.

Using DeTransfer, data logging is globally enabled in the following program

```
BEGIN
 RA10S
  1..5TK
END
LOGON
```

Using DeLogger, data logging is automatically globally enabled in programs. This is controlled from the Program Builder Settings tab. Right click on the Data Logging button, and select Data Logging On or Data Logging Off as required. The default is Data Logging On.



DeLogger automatically inserts a LOGON command into the program if Data Logging On is selected, and does not insert a LOGON command if Data Logging Off is selected.

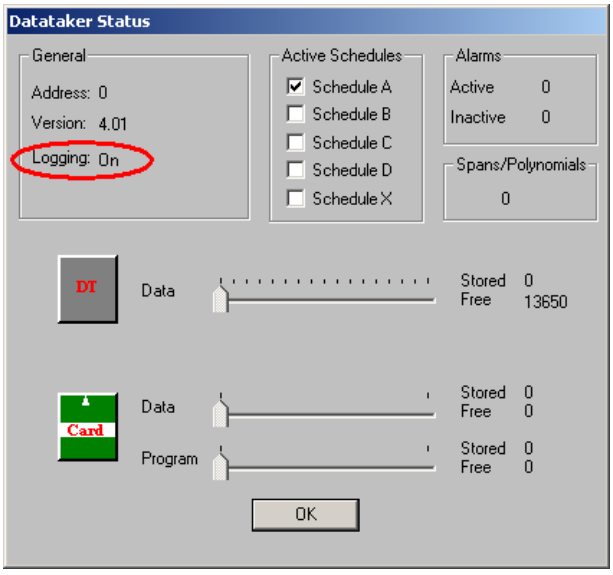
The current state of the Data Logging command can be checked at any time by either the STATUS or the STATUS5 commands (See Section III – Status Command).

Using DeTransfer this can be checked by the command for example

```
STATUS5
```

returns the response of either `Logging is ON` or `Logging is OFF`, depending on whether the last Data Logging command entered was LOGON or LOGOFF respectively, or a CLEAR or RESET command has been executed.

Using DeLogger, the state of the Data Logging command can be checked by selecting dataTaker:Status on the main menu or clicking the Status button on the toolbar, to open the dataTaker Status dialog. The state of the Data Logging command is shown as Logging On or Logging Off.



Logging Data - Individual Control

The logging of real time Schedule data can also be controlled for individual channels, using the No Log (NL) channel option (See Section III – Channel Options).

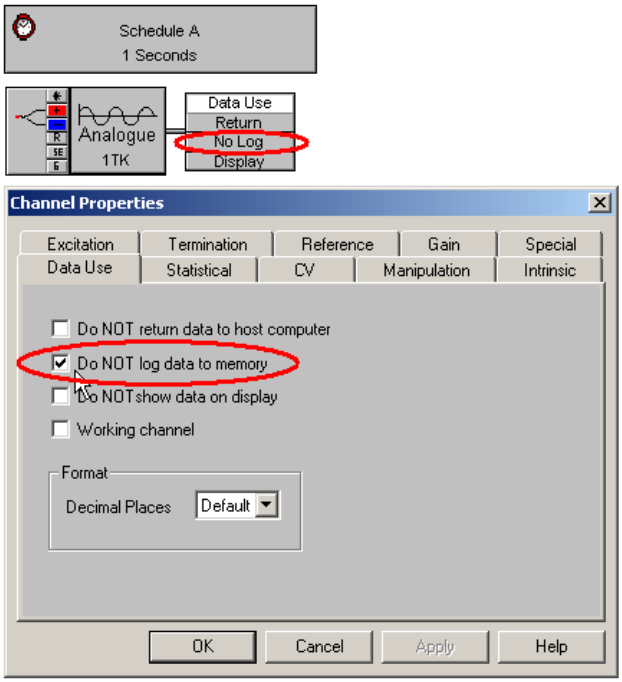
The NL Channel Option is applied to channels and Channel Variables that should not log real time data to the internal memory or memory card. NL only has effect when the LOGON command is enabled.

Using DeTransfer, the No Log Channel Option is used as follows

```
BEGIN
 RA2S
  1TK
  2TK(NL)
END
LOGON
```

Data logging is enabled by the LOGON command, and data will be logged for the first thermocouple but not for the second thermocouple. However data will be returned and displayed for both channels.

Using DeLogger, , the No Log Channel Option can be set in the Program Builder. After the channel is defined, right click on the Data Use icon and select Properties… The Channel Properties dialog opens with the Data Use tab open. Check Do NOT log data to memory.



Controlling Return of Data to the Host Computer

The data produced by the Data Schedules can be returned to the host computer, logged into the internal data memory or memory card if one is inserted, and displayed if a display is available

Sometimes it is not appropriate to return Schedule data in real time, particularly if it is being logged. The following commands allow control of returning data to the host computer.

Returning Data – Global Control

The return of real time Schedule data to the host computer is globally controlled by the Return Data Switch. The switch has no effect on the return of Logged data or Alarms data.

The Return Data Switch as defined as follows

/R	Enables return of Schedule data to the host computer (Default)
/r	Disables return of Schedule data to the host computer

The Return Data Switch defaults to /R whenever the *dataTaker* is initially powered up, hardware reset or executes a RESET command.

The current setting of the Return Data Switch can be checked at any time by the STATUS command or the STATUS9 command (See Section III – Status Command).

DeTransfer and DeLogger generally manage the Return Data Switch automatically, to suit the operation of the *dataTaker* / software interface. However it may be necessary to manage the switch if you are supervising the logger from other software, or if you are using some form of telemetry.

Returning Data - Individual Control

The return of real time Schedule data to the host computer can also be controlled for individual channels and Channel Variables, with the No Return (NR) channel option (See Section III – Channel Options).

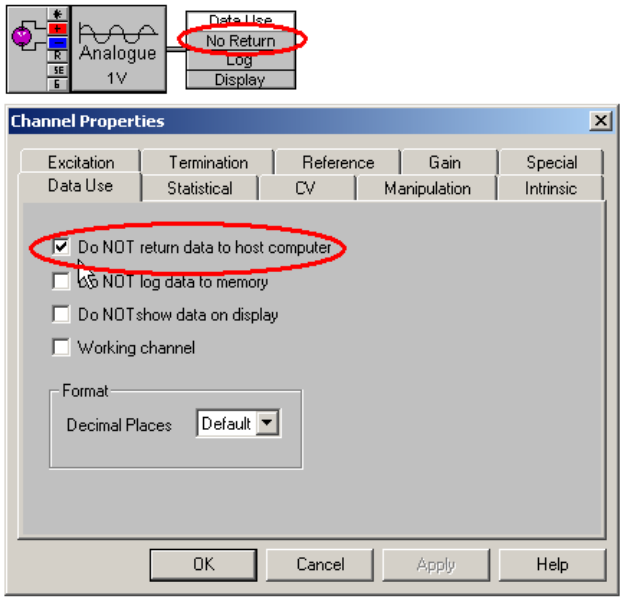
The NR Channel Option is applied to channels and Channel Variables that should not return real time data to the host computer. NR only has effect when the Return Data Switch is enabled.

Using DeTransfer, the Return Data Switch and No Return Channel Option are used as follows

```
/R
BEGIN
 RA2S
  1TK(NR)
  2TK
END
LOGON
```

The Return Data Switch is enabled, and data will be returned from the second thermocouple channel but not from the first thermocouple channel. The data can still be logged and displayed for both channels.

Using DeLogger, the No Return Channel Option can be set in the Program Builder. After the channel is defined, right click on the Data Use icon and select Properties… The Channel Properties dialog opens with the Data Use tab open. Check Do NOT return data to host computer.



Setting the No Return option here will prevent this data being displayed or charted, etc in DeLogger.

Controlling Displaying of Data

The data produced by the Data Schedules can be returned to the host computer, logged into the internal data memory or memory card if one is inserted, and displayed if a display is available

Sometimes it is not appropriate to display Schedule data in real time, particularly if it is being logged. The following commands allow control of displaying data on the display panel.

The display of realtime data can be controlled for individual channels, using the No Display (ND) channel option (See Section III – Channel Options). By default all of the channel data is displayed.

The ND Channel Option is applied to channels and Channel Variables that should not display real time data on the *dataTaker* display panel. NL only has effect when the LOGON command is enabled.

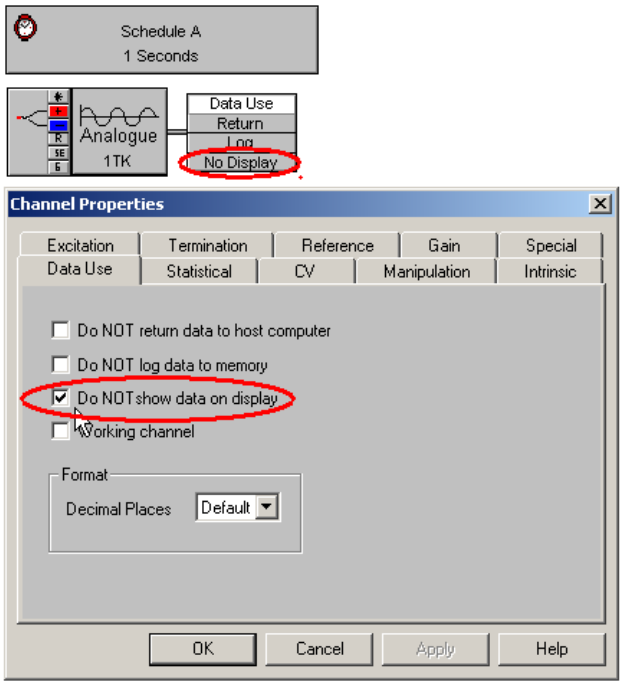
Note : The No Display channel option only applies to *dataTaker*s fitted with a display panel. If the ND channel option is used for a *dataTaker* without a display, then an 'E3 - channel option error' will be returned.

Using DeTransfer, the No Display Channel Option is used as follows

```
BEGIN
 RA2S
  1TK(ND)
  2TK(NR,NL)
END
LOGON
```

Data will be logged and returned for the first thermocouple channel but not displayed. Data will be displayed for the second thermocouple channel, but not logger or returned.

Using DeLogger, , the No Display Channel Option can be set in the Program Builder. After the channel is defined, right click on the Data Use icon and select Properties… The Channel Properties dialog opens with the Data Use tab open. Check Do NOT show data on display.



Individual channels can also be disabled from display from the keypad on the display panel itself (See Section III – Keypad and Display Panel).

Controlling Intermediate Data

Often programs include steps where input channel data is loaded into Channel Variables, etc. for further calculation. This intermediate or 'working' data is not to be returned, logged or displayed.

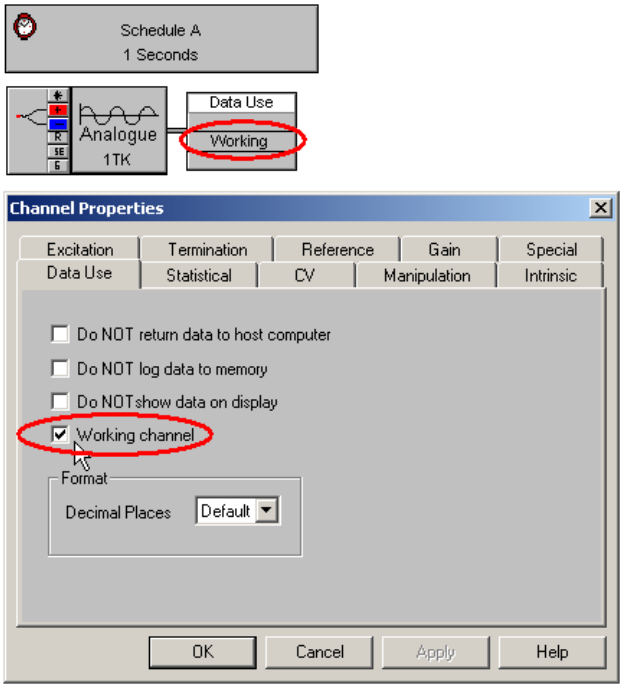
Such channels can have the NR, NL and ND channel options applied, or can have the Working channel (W) option applied which is a short form of NR, NL and ND.

Using DeTransfer, the Working Channel Option is used as follows

```
BEGIN
 RA2S
  1TK(W,=1CV)
  2TK
END
LOGON
```

Data from the thermocouple on channel 1 is saved into the Channel Variable, and not returned, logged or displayed. Data from the thermocouple on channel 2 goes to all destinations.

Using DeLogger, , the Working Option can be set in the Program Builder. After the channel is defined, right click on the Data Use icon and select Properties... The Channel Properties dialog opens with the Data Use tab open. Check Working channel.



Modes of Data Logging

The internal data memory and the memory cards can log data in either of two modes as follows

as a stack, where the data is written to the data memory until the memory is filled, and then data logging ceases.

as a circular buffer, where the data is written to the data memory until the memory is filled, and then new data is written over the oldest data.

The mode of data logging is defined by the Data Memory Overwrite Switch as follows

/o	Overwrite disabled, data store is a stack (Default)
/O	Overwrite enabled, data store is a circular buffer

The default setting for the Data Memory Overwrite Switch is /o, and the data memory functions as a stack. This default setting is selected following initial power up, or execution of a RESET command.

The current setting of the Data Memory Overwrite Switch can be checked at any time by the STATUS command or the STATUS9 command (See Section III – Status Command).

The Data Memory Overwrite Switch should not be changed after memory overwrite has begun.

Data Store Overwrite Switch Disabled (/o)

When data is being logged to the data memory, logging will continue until the data memory fills.

When the data memory fills, the *dataTaker* then responds according to the setting of the Data Store Overwrite Switch. If the switch is disabled (/o), the *dataTaker* responds as follows

- if there is no memory card installed in the logger, then data logging ceases and the message 'E5–data memory full' is returned to the host computer
- if a memory card is installed, the memory card is made invalid and data is then logged into the internal memory until that fills. When the internal memory is filled, data logging ceases and the message 'E5–data memory full' is returned to the host computer.
- if the memory card is removed, then data logging continues into the internal data memory. If a new or cleared memory card is later inserted, the data stored in the internal data memory is then transferred into the card and the internal memory is cleared. Data logging continues into the memory card until that fills.

In each case data that is acquired after the internal data memory is filled, will be lost. This will produce a discontinuity in the data if a new or cleared memory card is later inserted to allow further data logging. If memory cards are used in this mode of memory configuration, it is important that card replacement be managed carefully to avoid this discontinuity of data.

When all available data memory is filled, the Schedules continue to run and data will continue to return to the host continues if enabled by the Return Data Switch /R. Data logging remains enabled (LOGON) such that if a new or cleared memory card is later inserted, then data logging can continue to the card.

Data Store Overwrite Switch Enabled (/O)

When data is being logged to the data memory, logging will continue until the data memory fills.

If the Data Store Overwrite Switch is enabled (/O), then the data memory behaves as a circular buffer. When the memory fills, new data is progressively written over the oldest data. The data memory therefore contains a continuously updated recent history of data.

When the data memory fills, the *dataTaker* then responds as follows

- if there is no memory card installed, then data logging continues in the internal memory with the newest data being written over the oldest data. If a new or cleared memory card is inserted, the data stored in the internal data memory is transferred into the card, and the internal memory is cleared. Data logging then continues in the card in overwrite mode.
- if a memory card is installed, then data logging continues in the memory card with the newest data being written over the oldest data. The internal data memory is not used in this mode.
- if the memory card is removed, then the data is logged into the internal data memory in overwrite mode. If a new or cleared memory card is later inserted, the data stored in the internal data memory is transferred into the card, and the internal memory is cleared. Data logging then continues in the card in overwrite mode.

If a memory card is used in the overwrite mode, and is removed, the data is then logged into the internal data memory in overwrite mode. If the memory card is not replaced before the newest data commences

to overwrite the oldest data in the internal memory, then there will be a discontinuity in the data between memory cards.

Entering New Schedules When Logging

When data is logged into the internal data memory, the data is stored without details of channel number, input mode or engineering units. When the logged data is later read from the internal memory and returned to the host computer, the internal channel table created from the original Schedule lists is used to interpret the stored data.

Similarly when data is logged into a memory card, the data is stored without details of channel number, input mode or engineering units. A copy of the internal channel table is also stored in the memory card. This allows the memory card to be read and the data interpreted by another *dataTaker*, or by the MCI-04 Memory Card Interface.

This technique was developed to increase the data storage capacity of given memory size, more than could be stored in ASCII text as is seen at the host computer.

However this technique has important implications. If a new program is entered into the logger, then the original data could no longer be interpreted.

Therefore if data has been stored in the internal data memory, then entry of new program is rejected and the *dataTaker* will return the error message 'E4–clear data memory'. The *dataTaker* will also return this error if LOGON is enabled, even if there is no data in the memory yet.

However if there is a memory card inserted in the logger that contains data from the current program, then entry new program will process as follows

- if the new program is identical to the current program, then the new program is accepted and executed. New data is appended to data already stored in the memory card. This scenario is typical of a logger losing power during an application, and being restarted when power is resumed.
- if the new program is different in any way to the current program, then the memory card is made invalid. If the internal memory is clear, then the new program is accepted and executed and new data is stored in the internal memory only. However if the internal memory contains data, then the new program is rejected and the *dataTaker* will return the error message 'E4–clear data memory'

These actions preserve the original channel table until the data in the data memory has been returned (See Section III – Returning Data from Memory) and the internal data memory or memory card is cleared.

Priorities for Returning and Logging Data

When data from Schedules is logged to internal data memory or a memory card (LOGON), and also returned to the host computer (/R), the logging of data has priority over the returning of data.

Each time the Schedules run, the data is logged immediately, and is then returned to the host if time permits. However if the next Schedule trigger occurs before all of the data is returned, then returning of data is aborted and the Schedule runs.

This priority can be reversed if required by the Data Priority Switch, as follows

/Y	Return all data to host when data logging is enabled, ignore new triggers
/y	Return data to the host when data logging is enabled, only if time permits

The Data Priority Switch defaults to disabled (/y) following initial power up, or execution of a RESET command.

If data is not being logged to the internal data memory or a memory card, then all data will be returned to the host. If new Schedule triggers occur before all of the data is returned, then all new Schedule triggers are ignored until return of the original data set completes. There is no alternative to this.

Checking the Data Memory Configuration

The current settings of the various Data Logging commands can be checked at any time by the STATUS command (See Section III – Status Command).

Using DeTransfer, the status of the logger can be returned by the command

```
STATUS
```

which will include information of the current settings of the data logging commands

```
Logging is OFF
/o
/R
/y
```

These settings can be also be checked specifically by

the STATUS5 command for the setting of LOGON / LOGOFF

the STATUS9 command for the settings of the Data Memory Overwrite Switch, Return Data Switch and Data Priority Switch.

Checking Data Storage

The amount of data stored in the internal memory, and in the memory card, can be checked

by the STATUS command

the System Variables for internal memory and memory card data storage

Checking Data Storage in the Internal Memory

The STATUS command or STATUS6 command will return the amount of data stored, and the amount of free data space, in the internal data memory.

Using DeTransfer, the status of the internal memory can be returned by the commands

```
STATUS
STATUS6
```

which will include information about the internal data memory

```
103895,62635 Internal Data Points Free,Stored
```

Data storage in the internal data memory can also be returned by the System Variables as follows

1SV - returns the amount of free internal memory, in terms of data points

2SV - returns the amount of stored internal memory, in terms of data points

The amount of data currently stored in the internal data memory can be returned as part of the Schedule data, and can be used in Alarms to warn of full memory conditions.

For example

```
R10M  1..10V 2SV LOGON

ALARM1(1SV<100)"Memory Card is Full^M^J"
```

Checking Data Storage in the Memory Card

The STATUS command or STATUS7 command will return the amount of data stored, and the amount of free data space, in the memory card.

Using DeTransfer, the status of the memory card can be returned by the command

```
STATUS
STATUS7
```

which will include information about the internal data memory

```
79265,1670 Card Data Points Free,Stored
```

Note 1: When a valid memory card is inserted, and is not filled, then no will be data stored in the internal memory. The internal memory in this situation is a buffer for memory card change-over.

Note 2: When an invalid memory card is inserted, then stored data may be indicated for both internal memory and memory card.

Data storage in the memory card can also be returned by the System Variables as follows

3VS - returns the amount of free card memory, in terms of data points

4SV - returns the amount of stored card memory, in terms of data points

The amount of data currently stored in the internal data memory can be returned as part of the Schedule data, and can be used in Alarms to warn of full memory conditions.

For example

```
R10M  1..10V 4SV LOGON

ALARM1(3SV<100)"Memory Card is Full^M^J"
```

Logging Data to the Internal Memory

The *dataTaker* is able to store data into the internal data memory, as the data is produced by the Schedules reading input channels, reading internal channels, performing calculations, etc.

The *dataTaker* also supports removable memory cards, that can be inserted into the memory card socket of the logger and used to store data (See Section III – Logging Data to Memory Cards).

The Internal Data Memory

The *dataTaker* has 512 Kbytes of internal memory, of which approximately 12 Kbytes is used for system memory and approximately 500 Kbytes is used for data memory.

The *dataTaker* is able to store 166,530 data points or readings in the internal data memory. Stored data is written serially into the data memory, and the memory is managed as a First In First Out buffer.

The data memory can be configured as a data stack, where data logging stops when the memory becomes full, and new data is lost. Alternatively, the data memory can be configured as a circular buffer, where the newest data overwrites the oldest data when the memory becomes full.

Data is logged into the internal data memory as the Schedules run, producing the data by reading input channels, performing calculations, etc.

Once programmed, the *dataTaker* can scan input channels, perform calculations, and log the data to memory independent of a host computer. The computer can later be reconnected to the logger to recover the logged data.

The internal memory is backed by a ½AA lithium battery, which maintains the internal memory when the *dataTaker* loses power supply. If the logger slowly loses power supply (not less that 2 seconds), such as a battery power supply failing, then the logger will preserve the environment during the outage such that the internal data will be available when power is returned. However if power loss is sudden, then data stored in the internal memory will be lost.

Capacity of the Internal Data Memory

The *dataTaker* can store up to 166,530 data points or readings in the internal data memory. All data is stored in a compact 3 byte floating point format. Each 3 byte storage element is referred to as a 'data point equivalent' of data memory.

When the Schedules run, the data is produced as blocks of data. Each block contains the data for a single iteration of one Schedule.

When each block of data is logged into the internal data memory, the block is preceded by a header. This header is used to store information about the particular Schedule, and a timestamp (the date and time) for when the Schedule ran, that produced the block of data.

Storage of the header also consumes 3 bytes of memory.

The number of executions of Schedules that can actually be logged into the internal data memory varies depending on the number of channels being scanned, and calculations performed, and hence the number of data points stored in each block of data.

For example if the data from one channel is being logged, then there is a header associated with each block of data containing one data point. In this case there is a header overhead of 50%, resulting in the storage of approximately 83,250 data points total.

Alternatively if the data from 9 channels is being logged, then there is a header associated with each block of data containing 9 data points. In this case the header overhead is 10%, resulting in the storage of approximately 149,800 data points.

The amount of data stored in the internal data memory, and the amount of internal data memory free or unused, can be checked at any time by the STATUS command or the STATUS6 command (See Section III – Checking Data Storage and Section III – Status Command).

Clearing Internal Data Memory

Reading the data from the internal data memory does not clear the data memory. Consequently, data can be returned from the internal data memory as many times as required.

The internal data memory is cleared of data by the command

CLEAR

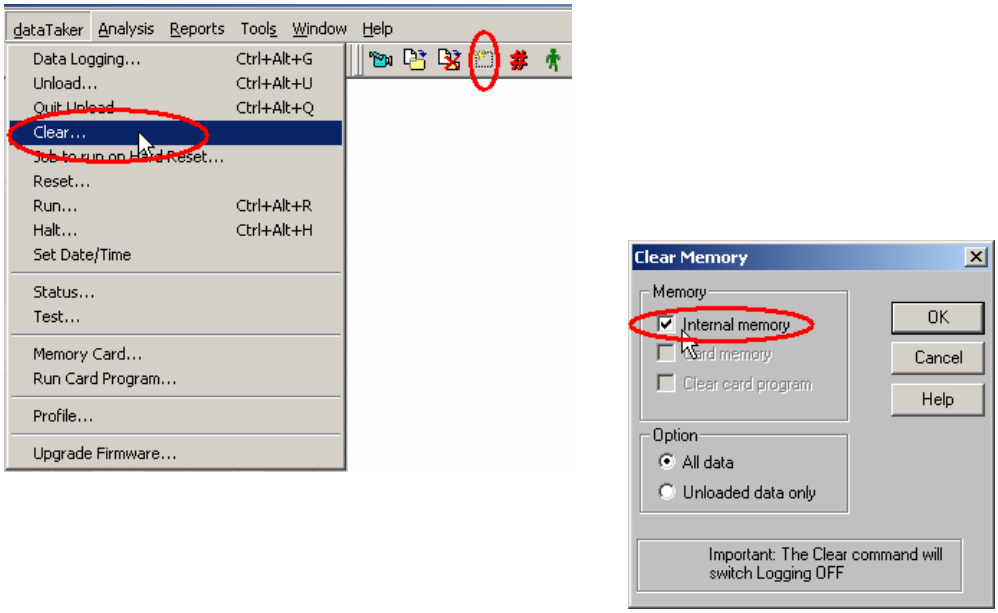
The CLEAR command also changes the Data Logging command to LOGOFF. The Data Logging command must be set to LOGON again before data logging can re-commence.

Using DeTransfer, the internal memory is cleared by the command for example

```
CLEAR
```

Using DeLogger, the internal memory is cleared by selecting dataTaker:Clear… from the main menu bar, or clicking the Clear button on the main toolbar. This will open the Clear Memory dialog.

Check Internal memory, and click OK to clear the internal data memory in the logger. There is also the option to clear either All data, or Unloaded data only. Normally this is set to clear All data,



Whenever there is data stored in the internal data memory, new Schedules cannot be entered into the logger until the data memory has been cleared by a CLEAR command.

The CLEAR command does not clear the data area of an inserted memory card. The data area of memory cards is cleared by the CDATA command.

Logging Data to Memory Cards

The *dataTaker* supports external memory cards, which can be plugged into the logger and used to store data. The logger can progressively store data into the memory card as the data is produced, or data can be transferred from internal memory to the memory card if one is inserted after data has been stored in internal memory.

Whenever a memory card is inserted into the *dataTaker*, the card is powered from the logger power supply. Some memory cards also have a lithium cell, which powers the card for up to 3 years while the card is removed from the logger. Other memory cards have a rechargeable cell, which recharges while the card is inserted in the logger. These cards may suffer a discharged battery, in which case the card should be inserted in the logger for several days to recharge. Under normal use however, this should never be a problem.

When the memory cards are inserted into the *dataTaker*, insertion should be swift and the memory card should be fully inserted into the memory card socket. The memory card must not be left partially inserted in the memory card socket.

During the insertion of a memory card, the logger's microprocessor is temporarily halted. This halt state must be kept to a minimum to avoid loss of time, and loss of any characters currently being received on the RS232 COMMS or NETWORK (*dataTaker 500/600* series loggers) serial interfaces.

Format of *dataTaker* Memory Cards

The *dataTaker* uses a proprietary format for memory cards, which is not compatible with any other formats. In particular the memory card format is not compatible with Microsoft DOS file format, and so these cards cannot be read directly if plugged into the PCMCIA or PC Card slots of computers.

The normal procedure is to read the *dataTaker* memory cards either

using a *dataTaker* data logger

using the *dataTaker* MCI-04 Memory Card Interface

The *dataTaker* MCI-04 Memory Card Interface is supplied with MCI Assistant software, which supports the interface in Microsoft Windows to transfer data from the memory cards to a file compatible with DeLogger and other applications. DeLogger 4 versions 4.2.16 or later also have this interface integrated into the package.

Data Transfer to Memory Cards

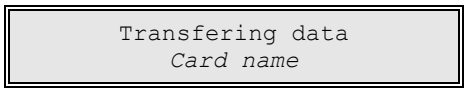
A memory card can be inserted into a *dataTaker* that has been already collecting data into internal memory. When the memory card is inserted, the *dataTaker* will transfer the data from the internal memory to the card of the card is clear.

The transfer of 166,530 readings from the internal data memory to a memory card takes 3 seconds . Therefore to be safe, do not remove a memory card for 5 seconds after inserting if you are transferring data.

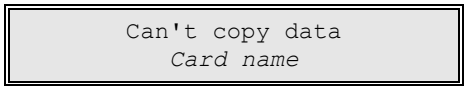
When data is transferred from the internal memory to the memory card, the data is written to the card then verified. If the memory card is removed during this transfer or verification period, then the check will fail. The whole of the data being transferred is retained in the internal memory, until another memory card is inserted. What is on the memory card is anyone's guess!!!

The worst case result of untimely removal of a memory card is that there could be a copy of the data on the memory card, and a copy in the internal data memory.

During data transfer, *dataTaker*s fitted with a display will display the message



If the memory card already contains data from another program when inserted into the logger, then no transfer occurs. *DataTakers* fitted with a display will display the message



As a general guide, the following procedures should be adopted to ensure maximum data integrity

whenever data is to be collected from the internal data memory, the memory card should be inserted for 3 seconds minimum, 5 seconds preferred.

whenever a memory card is removed during data acquisition, try to remove the card during the interval between scans, as can be observed on the 'Converting LED'.

Function of the Internal Data Memory with Memory Cards

The *dataTaker* has 512 Kbytes of internal battery backed memory, of which approximately 12 Kbytes is used for system memory and approximately 500 Kbytes is allocated to data memory.

When there is no memory card inserted, the internal memory is used as the data memory for logging of data. However whenever a memory card is inserted, the internal data memory is used as a data buffer for the memory card. The data buffer can store 166,530 readings.

The function of the internal data memory as a data buffer for the memory card is illustrated by the following scenario

a memory card is inserted into the *dataTaker* and data is logged to the card

the memory card may be removed for reading elsewhere, or for entering a program, etc.

data acquired during while the memory card is removed is buffered into the internal data memory

when the memory card is later inserted, the data in the internal memory is transferred to the card

It is the users responsibility to ensure that the internal data buffer does not overflow while the memory card is withdrawn, otherwise data will be lost.

Capacity of Memory Cards

The *dataTaker* Series 3 loggers support SRAM memory cards with capacities up to 4 Mbytes.

The *dataTaker* Series 1 and 2 loggers support SRAM memory cards with capacities up to 1 Mbyte.

The memory cards are automatically initialised when plugged into the logger (see below), and the available memory is divided as follows

4 Kbytes (4090 characters) is allocated to the program area for storing *dataTaker* commands

8 Kbytes is allocated for storing system and card management information

the balance of the card memory is allocated for data storage. The size of the data area depends on the card capacity.

All data is stored in the memory card in floating point format. The storage of each data point consumes 3 bytes of the card memory, irrespective of the magnitude of the data. This 3 byte format is referred to as a 'data point equivalent' of data memory.

The number of data points or readings which can be stored in the range of memory card sizes that are supported by the *dataTaker* is summarised in the table overleaf

Capacity	System Area (Kbytes)	Program Area (Characters)	Data Area (Kbytes)
512 Kbytes	8	4090	170,666
1 Mbytes	8	4090	343,980
2 Mbytes	8	4090	693,420
4 Mbytes	8	4090	1,392,300
Internal Memory			166,530

When input channels are scanned, the data is produced as blocks of data. Each block contains the individual channel data for a single iteration of one Schedule.

When each data block is logged into the memory card, the data block is preceded by a header. This header is used to store information about the particular Schedule, and the timestamp (the date and time) for when the Schedule ran, that produced the block of data.

The header consumes 3 bytes of memory.

The number of executions of Schedules that can actually be logged into a memory card varies depending on the number of channels being scanned, calculations performed, and hence the number of data points stored in each block of data.

For example if the data from one channel is being logged into a 1 Mbyte memory card, then there is a header associated with each block of data containing one data point. Therefore there is a header overhead of 50%, resulting in the storage of approximately 172,700 data points.

Alternatively if the data from 9 channels is being logged into a 1 Mbyte memory card, then there is a header associated with each block of data containing 9 data points. In this case the header overhead is 10%, resulting in the storage of approximately 310,800 data points.

The amount of data stored in a memory card, and the amount of free or unused memory card capacity, can be checked by the STATUS command or STATUS7 command (See Section III – Checking Data Storage and Section III – Status Command).

Initializing Memory Cards

Before a memory card can be used by the *dataTaker* to store data and program commands, the memory card must first be initialised or formatted.

Initializing New Memory Cards

Whenever a new memory card is inserted into the *dataTaker*, the card is automatically initialised and the data storage mode is enabled.

Similarly whenever a memory card in which the battery has been replaced is inserted into the *dataTaker*, the memory card will be automatically initialised.

Note : Whenever a memory card battery is replaced, the card memory powers down and loses initialization. Therefore memory cards can be cleared of data in the field if necessary, by removing and replacing the battery.

Initialization of memory cards involves several processes

the integrity of the card memory is checked, and the card capacity is determined

the card memory is divided into system, program and data areas

the program and data areas are cleared

a unique system identification code is assigned to the card

a default user CardID label of "Untitled" is assigned to the card

the memory card is enabled for data storage

if data logging is enabled and data has been stored into the internal data memory before the new card was inserted, then this data is transferred to the data area of the memory card

There are no initialization or formatting commands that must be executed when inserting new memory cards.

Initializing Used Memory Cards

Memory cards which have previously been initialised and used to store data, can be re-initialised at any time by the CDATA and CPROG commands (see below).

The memory card can be tested by the CTEST command, which destructively tests the integrity of the card memory. All stored data and programs are destroyed by this test command.

These commands do not change the system identification code or the user identification.

Memory Card Identification

Whenever a memory card is initialised by the *dataTaker*, a unique system identification code is assigned to the card. This system identification code is also stored in the *dataTaker*, and is used to identify the memory card.

Memory cards can also be given a user identification label, which can be used to identify memory cards according to the user name, application, site, data collection period, etc.

The user identification label replaces the default identification label of 'Untitled', which is assigned during initialization of the memory card.

User identification of a memory card is entered by the Memory Card Identification Command, in the general format

CARDID="*label*"

where *label* is a text string of up to 40 characters.

The label text string can contain any ASCII characters including spaces and control characters. Control characters are entered in the format ^G (Bell), ^J (LF), ^M (CR). While two characters are required to enter control characters, only one character is stored internally.

Using DeTransfer, a memory card can be labelled by the command for example

```
CARDID="My Card"
```

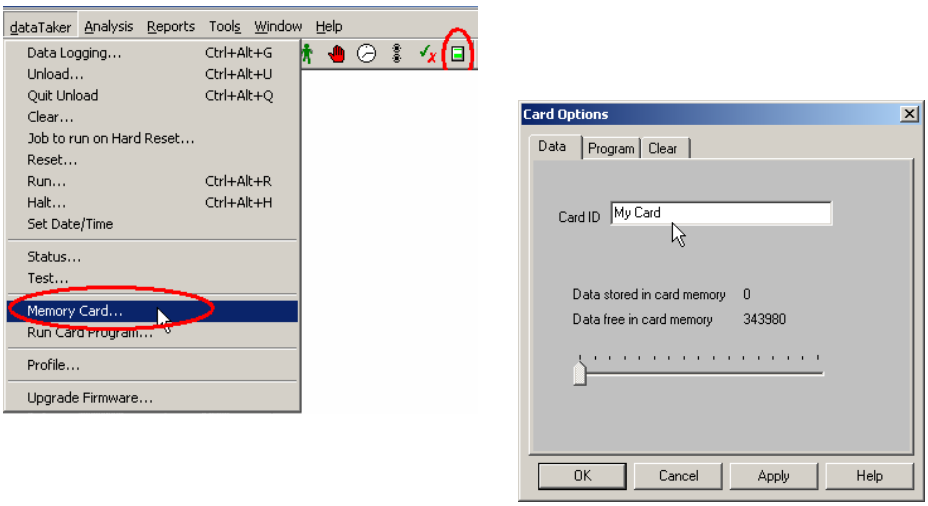
The user identification label for a currently inserted memory card can be returned by the command

```
CARDID
```

which for the above example would return

```
My Card
```

Using DeLogger, a memory card can be labelled by selecting dataTaker:Memory Card... on the main menu bar, or by clicking on the Memory Card button on the main toolbar, to open the Card Options dialog. Click on the Data tab, and enter your card ID into the Card ID field. Click Apply to label the card (which must be inserted into the logger!) and click Close



The user identification label of a memory card can be changed at any time, without effecting the data stored in the memory card.

The identification label is also displayed on the display when any memory card related messages are displayed.

Logging Data to Memory Cards

Whenever a memory card is inserted into the *dataTaker*, then all data will be logged into the memory card if data logging is enabled (LOGON), and the memory card is enabled for storing data.

Data can only be logged to memory cards which are enabled for data logging by the *dataTaker* into which the memory card has been inserted.

This policy has been designed to provide greatest security for data already stored in a memory card, and prevents accidental erasure or overwriting of potentially valuable data.

Enabling Memory Cards for Data Logging

There are no specific commands to enable a memory card for data logging. A memory card is enabled for data logging when it is inserted into the *dataTaker*, in response to a number of criteria as follows

if the memory card inserted is a new memory card, then it is initialised and enabled for data logging

if the memory card inserted is a memory card whose battery has been replaced, then it is initialised and enabled for data logging

if the data area of the memory card inserted has been cleared by the CDATA command elsewhere such as with another *dataTaker* logger or the *dataTaker* Memory Card Interface, then it is enabled for data logging

if the memory card inserted has data in the data area, and is cleared by a CDATA command from the host, then the memory card is enabled for data logging

if the memory card inserted has a CDATA command in the program area (see below), then this command is executed. The data area of the memory card is cleared, and the memory card is then enabled for data logging

Note : The CLEAR and RESET commands cannot be used to clear the data area of a memory card. These commands only clear the internal data memory.

When a newly inserted memory card has data stored in the data area of the card, and there are no commands either in the card or from the host to clear that data, then data cannot be logged to that card.

When a memory card containing data is inserted into the *dataTaker*, the error message 'E17-clear card data' is returned to the host computer.

The single fundamental rule that determines whether a newly inserted memory card can be enabled for data logging, is that the data area of the card must be clear of data.

Logging Data to the Internal Memory When a Memory Card is Inserted

Whenever a memory card is inserted into the *dataTaker*, data will always be logged to the memory card if possible. Data will only be logged to the internal data memory in particular circumstances as follows

when the memory card has filled, the data is buffered into the internal data memory for transferring into another memory card when the present card is replaced

when an unfilled memory card is removed, data is buffered into the internal memory for transferring into another memory card when inserted

when a newly inserted memory card contains data and is not enabled for data logging, the data is buffered into the internal memory for transferring into another memory card when inserted

The internal data memory cannot be deliberately used for data storage when a memory card which is enabled for data logging is installed in the *dataTaker*.

Commencing Data Logging to Memory Cards

A procedure for commencing a data logging application where data is to be logged into a memory card inserted in the *dataTaker*, is suggested below

insert the memory card into the *dataTaker*, and clear the data and program areas of the memory card as required using the CDATA and CPROG commands. These commands are described in the sections below

or insert a new memory card

enter the program into the *dataTaker*. The logging of data to the memory card will commence at the next Schedule trigger when data will first be produced.

Removing and Changing Memory Cards While Logging Data

Most data logging applications involving the use of memory cards often require that the memory cards are periodically removed and changed. Removal and changing of memory cards during a data logging task allows for a variety of scenarios for data collection and transfer.

However the three most common scenarios are as follows

removing a memory card from a remotely located *dataTaker*.
The memory card is transferred to a central facility where the data is read and processed. The same card is cleared of data after reading, and later returned to the logger.

changing the memory card of a remotely located *dataTaker*.
The original memory card is removed from the logger for transfer to a central facility where the data will be read and processed. A cleared memory card is installed into the logger at the time that the original card is removed.

remove the memory card from a local *dataTaker*, and insert memory cards from other *dataTaker*s for reading. The local *dataTaker* can continue a local data logging application while reading the 'foreign' memory cards, logging the local data into the internal memory. A cleared card is later inserted into the local *dataTaker* to continue the long term local data logging application.

The implications of each of these scenarios are discussed in the sections below.

Re-inserting the Same Memory Card

If a memory card is removed from a *dataTaker*, and then re-inserted into the *dataTaker* which is still running the same program, then the logger

will transfer any data from the internal memory to the memory card, appending this data to that already in the card

will continue to log data to the memory card

This scenario is similar to the scenario where a logger running a program and logging data to a memory card is powered down, powered up and reprogrammed – the logger will continue to log data to the same memory card as previous.

The *dataTaker* generates a 'unique signature' for each program entered, and writes this signature to the new/cleared memory card before logging the first data. Each time the logger attempts to write data to a memory card, it checks the signature stored on the memory card with that for the current program

- if there is no signature on the card, then the logger accepts this as a new/cleared card and writes the current program signature to the card, then stores data to the card beginning at the start of card data memory area. This is the situation of first use of a new or cleared memory card.
- if the two signatures match, then the logger appends the new data block to the existing data records in the card. This is the situation of continued use of an inserted card, or first use of a valid card that has been removed and re-inserted.
- if the two signatures do not match, then the logger invalidates the memory card and stores the new data block in the internal memory. This is the situation of a card being exchanged with one that has 'foreign' data on it.

While a memory card is removed from the *dataTaker*, the data is logged into the internal memory. If the internal data memory fills or overwrites while the card is removed, then data will be lost. It is the user's responsibility to manage the continuity of data. The data buffered in the internal memory of the logger while the card was removed, is copied to the memory card when the cleared card is re-inserted.

The inverse of this rule is where a *dataTaker* running a program and logging data to memory card, is reset and a new program entered. In this case the memory card and current program signatures will be different and the card will be invalidated by the logger. The memory card will not be used even though it has not been removed from the logger.

In this case data buffered in the internal memory is not copied to the memory card, and current data continues to be buffered into the internal memory.

When a memory card containing 'foreign' data is inserted into the *dataTaker* during a data logging application, the card can still be read and cleared by commands from the host computer to the logger. Only data logging to the card is disabled.

Inserting a Different Memory Card

Whenever the memory card is exchanged during a data logging task, the replacement memory card is tested for suitability for data logging as follows

- if the memory card is a new card, the card is initialised, any data in the internal memory is transferred to the memory card, and data logging continues to the card
- if the memory card is already initialised and has no data in the data area, any data in the internal memory is transferred to the memory card, and data logging continues to the card
- if the memory card contains data in the data area, then data cannot be logged to the memory card. The current data continues to be logged into the internal memory, and can be recovered when a new or cleared memory card is inserted.

When a memory card containing 'foreign' data is inserted into the *dataTaker* during a data logging application, the card can still be read and cleared by commands from the host. Only data logging to the card is disabled.

Controlling the Transfer of Data to Memory Cards

When a memory card is inserted into the *dataTaker*, the program area of the memory card is searched first and any commands found are downloaded into the logger and executed (See Section III – Programming from Memory Cards).

Following the execution of any commands present, and if the data area of the card is clear, then any data in the internal data memory is automatically transferred to the memory card.

However the automatic transfer of data can be controlled if required, to synchronize data transfer with programs stored on the memory card.

Forcing the Transfer of Data

The transfer of data can be forced at any time during the execution of a card program, rather than automatically after execution of the card programs. The command

COPY

instructs the *dataTaker* to transfer data from the internal memory to the memory card when the command is executed, rather than later in the normal course of events.

The following example illustrates the use of the COPY command in a card program. This program is stored in the program area of a memory card.

```
;COPY
;CLEAR CSCANS
;R10S  1..10TK  LOGON
```

When the memory card is inserted into a logger, this program in the memory card will

transfer the data from the internal data memory to the data area of the memory card

clear the internal memory and channel table, disable data logging ready for a new program

load the new program Schedules and enable logging again.

If the COPY command had not been used in the above program, the internal memory would have been cleared by the CLEAR command before the *dataTaker* could automatically transfer the data.

The COPY command is only appropriate for use in programs stored in the memory card. The command has no use when entered directly from a host computer to the *dataTaker*.

Preventing the Transfer of Data

The automatic transfer of data from the internal data memory to the memory card can be prevented by the command

NOCOPY

which instructs the *dataTaker* not to transfer data from the internal data memory to the data area of the inserted memory card. The NOCOPY state remains active until the memory card which contains the command is removed. When another memory card is inserted, automatic data transfer is again enabled.

The following example illustrates the use of the NOCOPY command in a card program. This program is stored in the program area of a memory card.

```
;NOCOPY
;ALARM5(1TK>100)"Temperature Alarm"
;ALARM6(2TK>105)AND
;ALARM7(3TK<140)4DSO
```

The NOCOPY command is used in this program to instruct the logger not to transfer data from the internal data memory to the memory card at any time while this memory card is inserted in the logger.

This programming example is typical of a memory card being used as a program only card, to enter new alarms or alarm setpoints, change Schedule triggers, etc. while the current program continues.

The data from this operation is logged to the internal memory while this memory card is inserted, but can be copied into the next memory card that is inserted if valid.

The NOCOPY command is only appropriate for use in programs stored in the memory card. The command has no use when entered directly from a host computer or terminal.

Clearing Data from Memory Cards

The data area of memory cards can be cleared of stored data by the command

CDATA

If a host computer is not available, a memory card can be cleared by removing and replacing the internal battery, to allow the card memory to power down.

When the memory card is next inserted into the logger, the card is recognised as a new card and is automatically initialised again for data logging.

Note 1: The CLEAR command cannot be used to clear the data area of a memory card. The CLEAR command only clears the internal data memory of the *dataTaker*.

Note 2: The RESET command cannot be used to clear the data area of a memory card. This command resets the *dataTaker*, and clears the internal data memory.

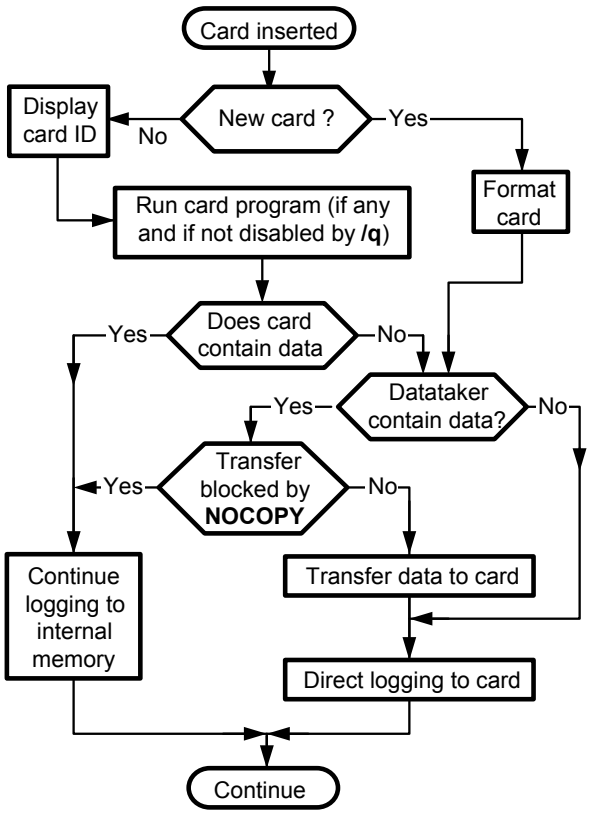
Memory Card Test

Memory cards can be tested for integrity by the *dataTaker*. Insert the memory card intothe logger, and run the CTEST command on the logger. This will perform a write:read test of the card memory, then reformat the card. The *dataTaker* will then return a response of whether the card passed or failed CTEST.

Beware : CTEST will erase all data and programs stored on the memory card.

Summary of Memory Card Insertion

The following flow chart indicates how the memory card is managed on insertion.



Returning Data from Memory

The *dataTaker* has many options available for the returning of data from internal memory and memory cards. When data is unloaded from the memory the memory it is NOT cleared, and can therefore be unloaded in many different formats as many times as required.

Reading of data from memory cards has priority over reading data from the internal data memory. However commands are available to select the internal memory or an inserted memory card for reading.

If data is being logged to the internal data memory, and a memory card that contains 'foreign' data is inserted, then that data can be read from the memory card while local logging continues to the internal memory. Logged data can read be from any inserted memory card, regardless of its validity for data logging.

The Unload Command

Data logged into the internal data memory or into a memory card is returned to the host computer by the Unload command.

The Unload command by default unloads all data from the memory, beginning from the first record. Extensions to the Unload command allow particular data to be unloaded from data memory in terms of the source, schedule and period.

The Unload command is specified in the general format

U *source schedule (start time,start date)(end time,end date)*

where

U	specifies the Unload command	
source	I	- unloads data from the internal memory
	M	- unloads data from the memory card
	none	- unloads data from a memory card if inserted, otherwise from internal memory
schedule	specifies the schedule to be unloaded	
	A, B, C, D, X	- specified schedule is unloaded
	none	- all schedules are unloaded.
start	sets time and date to unload from	
	time	- unload from specified time
	date	- unload from specified date
	BEGIN	- unload from the beginning of data
	LAST	- unload from the end of the last unload
	none	- unload from the beginning of stored data
end	sets time and date to unload to	
	time	- unload to specified time
	date	- unload to specified date
	END	- unload to the end of data
	LAST	- unload to the last unload
	none	- unload to the end of stored data

Each of the above specifiers is optional, however the (*end*) can only be included if the (*start)* is included.

Data is unloaded from the oldest data first. Schedules are merged chronologically and in the order of X, A, B, C, and D.

Each of the Unload command specifiers is discussed in the following sections.

Unloading Data from a Specific Source

Logged data can be selectively unloaded from the internal memory or from an inserted memory card.

If the *source* is not specified, then the Unload command will unload data from memory as follows

if no memory card is inserted, data is unloaded from the internal memory

if a valid memory card is inserted, then the data is unloaded firstly from the memory card, and then from the internal memory if data for the current task has been logged there. This is only appropriate for the stack mode of data logging (/o).

if an invalid memory card is inserted, then the data is unloaded from the memory card only. Any data in the internal memory cannot be unloaded until the invalid memory card is removed

However the *source* of data can be specified for the Unload command as the internal memory (UI) or an inserted memory card (UM).

If a memory card is inserted which has data from another *dataTaker* or from another program run on the same *dataTaker*, then the current data in internal memory can still be unloaded using the UI Unload command.

If the UM Unload command is entered when no memory card is inserted, then the 'E40-no data found' message is returned.

Unloading Data from a Specific Schedule

Normally data is unloaded by the Unload command from all of the Schedules in chronological order of the data block timestamps. Where more than one Schedule have data blocks with the same timestamp, then data for the timestamp is unloaded in the Schedule order RX, RA, RB, RC then RD.

However data can be unloaded for specific Schedules as follows

UX - unload data for the RX Schedule only

UA - unload data for the RA Schedule only

UB - unload data for the RB Schedule only

UC - unload data for the RC Schedule only

UD - unload data for the RD Schedule only

If a Repeat Scan Schedule is specified for which no data has been logged, then the 'E40-no data found' message is returned.

Unloading Data for a Specific Period

When data from Schedules is logged into data memory, the date and time that the Schedule ran is automatically recorded in the header of each block of data.

These timestamps allow data to be unloaded from the data memory for specific periods of time, even if Time and Date 'channels' were not included in the original Schedules.

The period for which data is to be unloaded from the data memory is defined as the data occurring between a *start* point and an *end* point. The *start time* and *start date* can be defined as follows

BEGIN - data is unloaded from the first record in the memory, which is the oldest data

time - data is unloaded from time, must be in the format defined by Parameter39

date - data is unloaded from date, must be in the format defined by Parameter31

LAST - data is unloaded from the end point of the last data unload. This allows data logged since the last unload to be unloaded.

None - data is unloaded from the first record in the data memory, which is the oldest data

The *end time* and *end date* can be defined as follows

END	- data is unloaded to the last record in memory, which is the newest data
time	- data is unloaded to time, which must be in the format defined by Parameter39
date	- data is unloaded to date, which must be in the format defined by Parameter31
LAST	- data is unloaded to the end point of the last data unload. This allows data unloaded earlier to be unloaded again to the same end point.
None	- data is unloaded to the last record in the data memory, which is the newest data

Several considerations apply to the specification of the *start* point and the *end* point as follows

if a *start time* is specified but not a *start date*, then the current date is assumed

if a *start date* is specified but not a *start time*, then the first timestamp for the *start date* is assumed

if an *end time* is specified but not an *end date*, then the current date is assumed

if an *end date* is specified but not an *end time*, then the last timestamp for the *start date* is assumed

if only the *start* point is specified, then the end of data memory is assumed as the *end* point

an *end* point cannot be specified alone, a BEGIN or *start* point must also be specified

If a period is specified for which no data has been logged, or the *end* point is earlier than the *start* point, then the 'E40-no data found' message is returned.

Including Time and Date with Unloaded Data

When data from Schedules is logged into the data memory, the date and time that the Schedule ran is automatically recorded in the header of each block of data.

These timestamps allow data to be unloaded from the data memory with time stamps, even if Date and Time 'channels' were not included in the original Schedules.

Return of the date and time stamps is controlled by the Date Switch and the Time Switch as follows (See Section III – Format of Returned Data and Section III – Switch Commands)

/D	Enable return of date stamp with data
/d	Disable return of date stamp with data (Default)
/T	Enable return of time stamp with data
/t	Disable return of time stamp with data (Default)

Following initial power up, a hardware reset, or a RESET command, the Date Switch and Time Switch are disabled (/d/t).

The Date Switch and Time Switch can be changed at any time, even during and between unloads of the data memory.

Unloading Data Manually with DeTransfer

The following examples illustrate the use of the Unload command for returning data from the internal memory, or from an inserted memory card.

These commands are used directly with DeTransfer, and the unloaded data is returned into the upper Receive screen of the package and must be saved manually.

```
U
U(BEGIN)
U(BEGIN)(END)
```

unloads data from the internal data memory or memory card as appropriate for all Schedules, beginning at the oldest data and ending at the latest data.

```
UI
UI(BEGIN)(END)
```

unloads data from the internal data memory for all Schedules, beginning at the oldest data and ending at the latest data.

```
UMB
UMB(BEGIN)(END)
```

unloads data from a memory card for Schedule B only, beginning at the oldest data and ending at the latest data.

```
/D/T
UA(BEGIN)(12:00:00,29/05/92)
```

unloads data from the internal data memory or memory card for Schedule A, beginning at the oldest data and ending at the specified time and date. The unloaded data is timestamped.

```
UMA(9:00,28/05/1992)(17:00,29/05/92)
```

unloads data from a memory card for the Schedule A between the specified times and dates.

```
UIX(BEGIN)(18:45:00)
```

unloads data from the internal memory for Schedule X, beginning at the oldest data and ending at the specified time on the current date.

```
UC(LAST)(END)
```

unloads data from the internal data memory or memory card for Schedule C, beginning after the last data block unloaded by a previous Unload command and ending at the latest data. This allows unloading of data logged since the previous unload.

```
P31=0
P39=2
UC(15.75,123)(10.0,125)
```

the start time, start date, end time and end date are specified per the settings for Parameter31 and Parameter39.

Unloading Data with DeTransfer

DeTransfer has a semi-automated method for unloading logged data fro data loggers. DeTransfer creates the appropriate Unload commands and manages unloading of data.

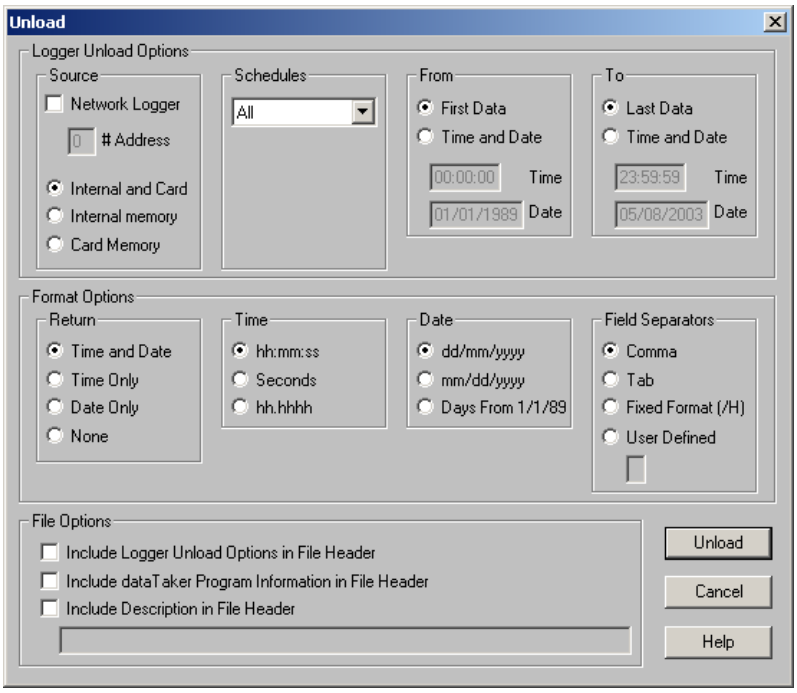
Select Data Transfer : Unload on the main menu bar, to open the Unload dialog.

The Unload dialog has various panels for defining the parameters for unloading data from the data logger, including data source, schedules, start time and date, and end time and date.

There are also panels too select the timestamps, and to format the timestamps. There is also a panel to define the format for the unloaded data, and to include some details in the top of the data file.

Set the various panels as required, and click the Unload button to commence unloading data. DeTransfer creates the appropriate Unload command from the settings of the various panels, and sends this to the

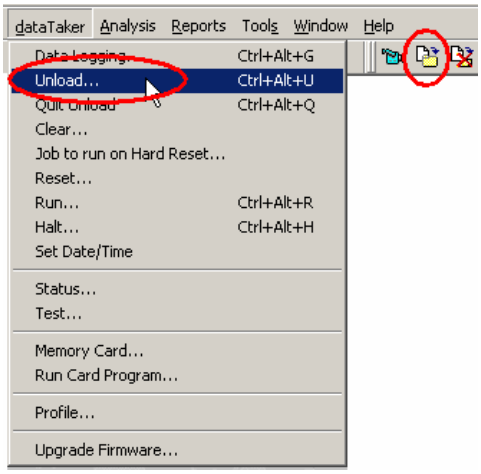
logger. The data is saved into a file, which you have the opportunity to name when the unload is completed.



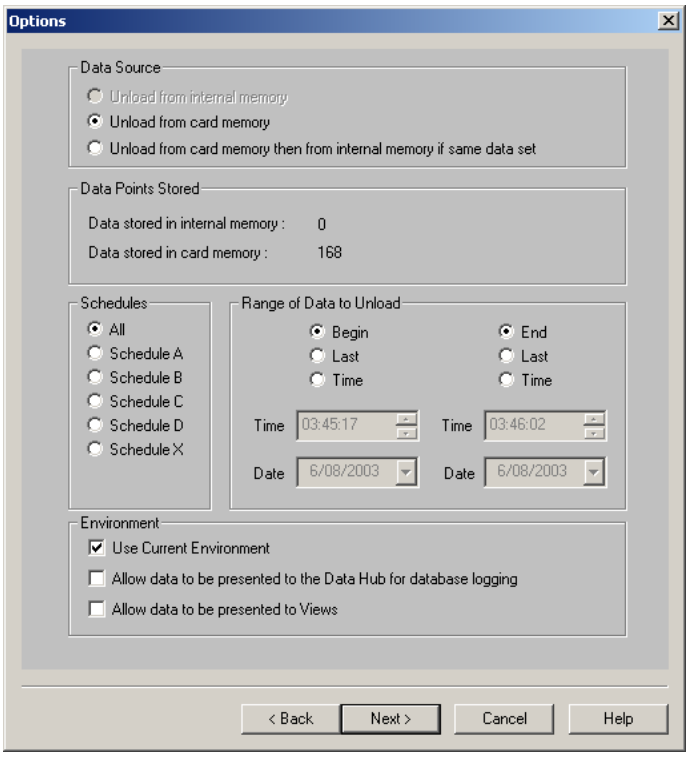
Unloading Data with DeLogger

DeLogger also provides support for unloading data.

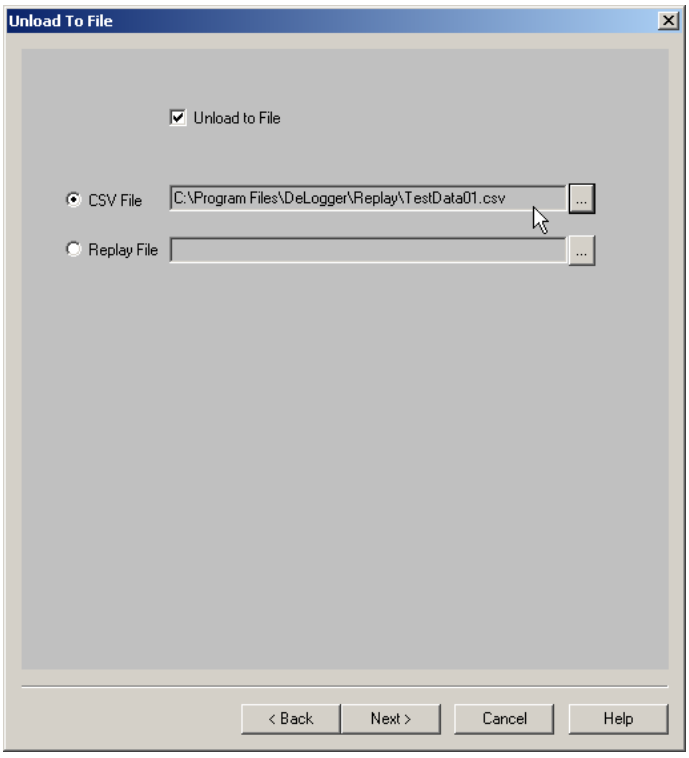
Select dataTaker:Unload… from the main menu bar, or click on the Unload button on the main tool bar to open the Unload Options dialog.



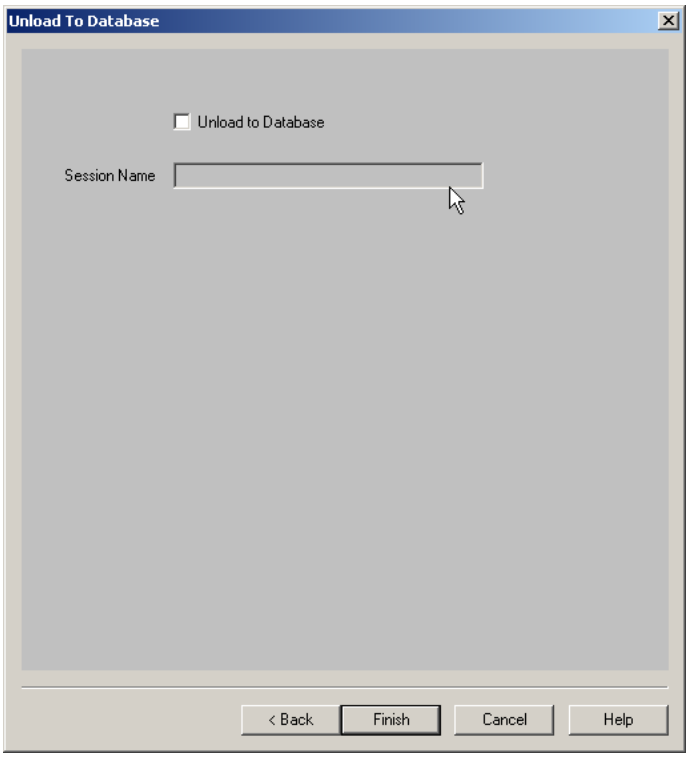
The Unload Options dialog provides various panels to specify where to unload data from, which Schedules to unload data for, and the timestamp range for data to unload.



When you have set the panels as required, click on the next button to open the Unload to File dialog. If you wish to unload the data to a file, then check the Unload to File check box, and select the file format. CSV File is suitable to exporting to many applications including Microsoft Excel, however cannot be used in DeLogger. Replay File is intended for use in DeLogger. After selecting a file format, enter a name for the file and click the Next button.



The Unload to Database then opens (DeLogger Pro only), in which you can set up a database session into which to store the unloaded data.



Finally, click the Finish button to unload of data from the logger into the data storage in DeLogger.

Format of the Unloaded Data

Data is returned to the host computer by the Unload command in the currently defined data item format, and currently defined data block format.

The format of unloaded data can be configured in the same way as data returned in real time. The data format can be changed between successive unloads (See Section III – Format of Returned Data).

If you are using the automated facility in DeTransfer or DeLogger to unload data, then these tools will set the logger into Fixed Format mode (/H), and then re-construct the data into the required format within the computer.

Disabling of System Switches

During an Unload operation in Free Format mode (/h), several of the Switch commands are internally disabled to protect the integrity of the unloaded data as follows

the Return Data Switch is disabled (/r) to prevent any current real time scan data from being inserted into the unloaded data stream. However logging of any current scan data will continue during an Unload operation if enabled by LOGON.

the Command Echo Switch is disabled (/e) to prevent commands received during an Unload operation from being echoed into the unloaded data stream.

the Messages Switch is disabled (/m) to prevent any error messages which may be generated during the Unload operation from being inserted into the unloaded data stream.

When unloading is completed, or is Quit (see below), these system switches are returned to their settings prior to beginning the Unload operation. The switches cannot be changed during the unload operation.

If data is unloaded from memory in the Fixed Format mode (/H), then this mode automatically presets all of these switches.

Quitting Data Unloading

When an Unload operation is in progress, the unload can be aborted by the Unload Quit command

Q

Using DeTransfer, an unload is quit by

```
Q
```

Using DeLogger, an unload is quit by selecting *dataTaker*:Quit Unload on the main menu bar, or by clicking on the Quit button on the main toolbar.

There may be a short delay in DeTransfer and DeLogger between sending the Quit command, and the return of stored data actually stopping.

This is because the output buffer of the *dataTaker* is generally full during an unload operation, and the contents have still to be returned to the host. Further, there may be buffered data in the computer application as well which has to be cleared.

The Quit command actually stops the copying of stored data into the output buffer of the logger. Any data upstream of this function is not stopped by the Quit command.

Labelling the End of Unloaded Data

A special character can be output from the *dataTaker* following the last data item of an unload. This character can be used by the host computer to identify the end of an unload operation, to close data files, etc.

The end of unload character is an ASCII character defined by the Parameter25 command. It can be any character in the range ASCII 0 — 127. The default is ASCII 0 (NULL), for which no character is sent after the last data item.

For example the command

```
P25=42
```

defines the end of unload character as the asterisk (*) character (ASCII 42).

An example of the last block of unloaded data terminated by the end of stored data character declared by Parameter25=42 is as follows

```
Date 29/05/2003
Time  11:37:30
3V -12.15 mV
8C  2267 Counts
20TK  1759.8 Deg C
*
```

Clearing the Data Store

Unloading the data from the internal data memory or memory card does not clear the data memory. Consequently, data can be returned from the data memory as many times as required.

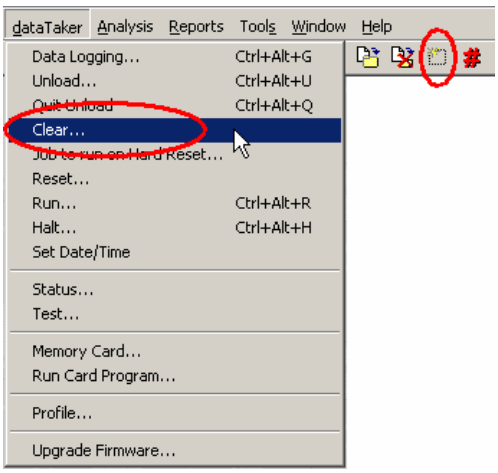
The internal data memory is cleared by the CLEAR command. This command does not clear the data area of an inserted memory card.

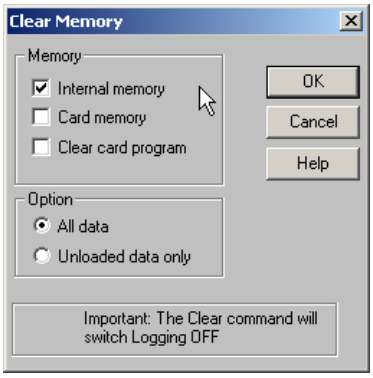
When there is data stored in the internal memory of the *dataTaker*, new programs or Schedules cannot be entered into the logger until the internal data memory has been cleared by a CLEAR command. This is because the internal channel table is required to interpret the data memory during Unload operations.

The CLEAR command sets the Data Logging command to LOGOFF, and must be set to LOGON again before data logging can recommence.

The data area of an inserted memory card is cleared by the CDATA command. This command does not clear the internal data memory.

Using DeLogger, the internal data memory and memory card are cleared by selecting dataTaker:Clear… from the main menu bar, or by clicking the Clear button on the main toolbar.





This opens the Clear Memory dialog. Check the clear option(s) required, and click the OK button.

Programming from Memory Cards

Memory cards can be used to program the *dataTaker*, independently of a host computer. This feature is particularly useful where *dataTaker* must be remotely deployed, remote host facilities are unavailable, or loggers are programmed by unskilled personnel.

The memory cards have a program area which can store up to 4090 characters of *dataTaker* program and commands.

Whenever a memory card containing a program or commands in the program area is inserted into the *dataTaker*, these are automatically read and executed by the logger.

A number of the *dataTaker* commands are not appropriate for use from the memory card. These include the Unload command, the STATUS command, the TEST command, etc. which only return data or system information to a host computer.

Recording Commands into a Memory Card

dataTaker commands or programs are recorded into a memory card by installing a memory card into any logger, and then sending the program or commands from a host computer.

Commands are directed to the program area of the memory card by prefixing each command with the semicolon character in the general format

; *command*

The following program for example

```
;RESET
;/m /n /u P22=44 P24=13
;BEGIN
; RA5M
;  1V 3..5TT 2HSC
;  ALARM1(1V>1000)4DSO
;  ALARM2(5TT>50.0)"Temperature Alarm"
;END
;LOGON
```

is copied directly to the program area of the inserted memory card. The commands are not executed by the *dataTaker*.

When *dataTaker* commands are entered into the program area of memory cards, the commands are not checked for syntax or executed to check for run-time errors. Therefore it is advisable to test memory card programs before using them in the field.

When commands that are copied to the program area of a memory card by the logger, these commands are appended to any commands that may already be stored in the area.

If a memory card is not inserted in the *dataTaker* when ;commands are entered, then the commands are lost.

Enabling Memory Card Programming

Remote programming of the *dataTaker* from programs and commands stored in a memory card is controlled by the Execute Card Commands Switch as follows

/Q Programming by memory card commands is enabled (Default)

/q Programming by memory card commands is disabled

Following initial power up, a hardware reset, or a RESET command, the Execute Card Commands Switch is enabled.

The current setting of the Execute Card Commands Switch can be determined by the STATUS command or the STATUS9 command (See Section III — Status Command).

Executing Commands from Memory Cards

When a memory card is inserted into a *dataTaker*, the program area of the memory card is searched for programs or commands. If there are any programs or commands stored in the program area, then these are downloaded into the *dataTaker* and executed under the following conditions

inserting the card into a logger that has the Execute Card Commands Switch enabled (/Q)

resetting the logger via the Reset Hole while the card is inserted

powerdown/powerup resetting the logger while the card is inserted

resetting the logger by RESET command from the host computer while the card is inserted

sending the command RUNPROG from the host computer while the card is inserted

*dataTaker*s fitted with a display, display the message

```
Card Programming
card name
```

If the Execute Card Commands Switch is disabled at the time of memory card insertion, then commands stored in the program area of the memory card are not executed.

The stored commands are executed each time that the memory card is inserted. Therefore if the card is inserted, withdrawn, and then reinserted, then the commands will be executed twice. This may be overcome by disabling the Execute Card Commands Switch with a /q command within the card program.

Controlling the Transfer of Data to Memory Cards

When a memory card is inserted into the *dataTaker*, the program area of the memory card is searched first and any commands found are downloaded into the logger and executed.

Following the execution of any commands present, and if the data area of the card is clear, then any data in the internal data memory is automatically transferred to the memory card.

However the automatic transfer of data can be controlled if required, to synchronize data transfer with programs stored on the memory card.

Forcing the Transfer of Data

The transfer of data can be forced at any time during the execution of a card program, rather than automatically after execution of the card programs. The command

COPY

instructs the *dataTaker* to transfer data from the internal memory to the memory card when the command is executed, rather than later in the normal course of events.

The following example illustrates the use of the COPY command in a card program. This program is stored in the program area of a memory card.

```
;COPY
;CLEAR CSCANS
;R10S  1..10TK  LOGON
```

When the memory card is inserted into a logger, this program in the memory card will

transfer the data from the internal data memory to the data area of the memory card

clear the internal memory and channel table, disable data logging ready for a new program

load the new program Schedules and enable logging again.

If the COPY command had not been used in the above program, the internal memory would have been cleared by the CLEAR command before the *dataTaker* could automatically transfer the data.

The COPY command is only appropriate for use in programs stored in the memory card. The command has no use when entered directly from a host computer to the *dataTaker*.

Preventing the Transfer of Data

The automatic transfer of data from the internal data memory to the memory card can be prevented by the command

NOCOPY

which instructs the *dataTaker* not to transfer data from the internal data memory to the data area of the inserted memory card. The NOCOPY state remains active until the memory card which contains the command is removed. When another memory card is inserted, automatic data transfer is again enabled.

The following example illustrates the use of the NOCOPY command in a card program. This program is stored in the program area of a memory card.

```
;NOCOPY
;ALARM5(1TK>100)"Temperature Alarm"
;ALARM6(2TK>105)AND
;ALARM7(3TK<140)4DSO
```

The NOCOPY command is used in this program to instruct the logger not to transfer data from the internal data memory to the memory card at any time while this memory card is inserted in the logger.

This programming example is typical of a memory card being used as a program only card, to enter new alarms or alarm setpoints, change Schedule triggers, etc. while the current program continues.

The data from this operation is logged to the internal memory while this memory card is inserted, but can be copied into the next memory card that is inserted if valid.

The NOCOPY command is only appropriate for use in programs stored in the memory card. The command has no use when entered directly from a host computer or terminal.

Using the CDATA Command from the Program Area

The data area of memory cards can be cleared of stored data by the command

CDATA

The CDATA command can be included in a memory card program to clear the memory card of data stored in the data area, when inserted into a *dataTaker* to collect the data stored in the internal memory.

For example the card program

```
;CDATA
;COPY
```

will clear the data area of the memory card, and transfer data from the internal data memory.

Clearing the data area of a memory card with the CDATA command enables the memory card for data logging. The COPY command in this example is not necessary, because data transfer would occur after all of the commands in the program area are executed. However the command can be included if preferred.

A memory card containing this program is typical for applications where a memory card is used to periodically collect data from remote *dataTaker*s, to transfer the data to a central computing facility.

Caution : Use of the CDATA command in this way has potentially damaging consequences. If a memory card containing this program is inserted into another *dataTaker* to unload the data, then the CDATA command will again be executed and the memory card will be cleared !!

The Execute Card Commands Switch must be disabled (/q) in the second *dataTaker* to prevent execution of the card commands.

The preferred method is to clear the memory card by the CDATA command from the host, or clearing the card with the MCI-04 Memory Card Interface, after the data has been unloaded.

Forcing Execution of the Memory Card Program

When a memory card is inserted into the *dataTaker*, the program area is searched and any commands or program found are executed. However once the card is inserted, the program will not automatically run again while the card remains inserted.

The program in the memory card can be executed in response to a command from the host computer

```
RUNPROG
```

This will force the program stored in an inserted memory card to be read and executed.

The command can be entered at any time from the host, and can be used for memory card control when operating the *dataTaker* remotely via modems.

However the command has limited application when included in a program stored in a memory card.

Executing RESET Command from the Program Area

The RESET command can be included in the program area of the memory card, to reset the *dataTaker* before entering a new program.

However use of the RESET command in this manner will also clear any data stored in the internal data memory.

If a memory card program contains the RESET command, then this reset does not cause the card program to be read again (unlike a commanded reset from a host computer). The prevents the *dataTaker* getting into an infinite loop.

Clearing the Program Area

The program area of a memory card is cleared by the command

```
CPROG
```

Checking the Program Area

The number of characters of program stored in the program area of a memory card can be returned by the STATUS command which returns the response

```
3406,684 Program Characters Free,Stored
```

The program and commands stored in a memory card can be returned separately by the STATUS8 command, which returns a listing of the commands as follows

```
4008,82 Program Characters Free,Stored
;CSCANS CALARMS
;/m /n /u P22=44 P24=13
;ALARM2(5TT>50.0)"Temperature Alarm"
;RA5M 1V 3..5TT 2HSC
;LOGON
```

Example of a Card Program

The following example provides a guide to the general format for programming a *dataTaker* from a memory card.

```
;' Program Card for Boiler House
;
; H
; COPY
; CLEAR                    'Clear internal memory
; CSCANS CALARMS           'Clear Scan schedules & Alarms
;
; /u/n/O/T/D               'Set switches
; P22=44                   'Set Parameters
; Y1=4.5,0.213"KPa"        'Polynomials for Pressure sensor
; S2=0,10,0,50"L/m"        'Span limit for Flow sensor

;BEGIN
; RA2S
;   1TK("Steam In Temp.")
;   2TK("Steam Out Temp.")
;   3V(Y1,"Steam Pressure")
;   4V(S2,"Water Flow")
;
; RB5M
;   1TK(AV,"Steam In Average")
;   2TK(AV,"Steam Out Average")
;
; RS2S                     'Statistical Sampling rate
;
; RZ1S                     'Alarm Sampling rate
; ALARM1(1TK("Steam In LOW")>110)1DSO"Steam In HIGH[GA]"
; ALARM2(1TK<110)"[HA]"
;
; LOGON                    ' Enable logging
;
; END
; HA GB GS GZ
```

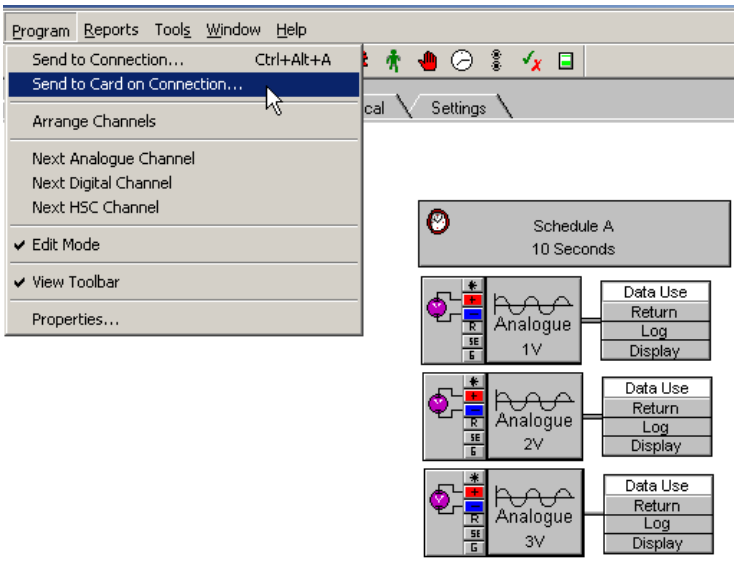
When the memory card is inserted, the following procedure occurs

- the scanning by the *dataTaker* is halted (H)
- internal memory data is transferred to the card memory (COPY)
- the internal memory is cleared (CLEAR)
- the Scan Schedules and Alarms are cleared (CSCANS, CALARMS)
- The new program is read, and execution of the new task begins

Storing Programs in to Memory Cards from DeLogger

Programs developed in the Program Builder of DeLogger can be stored into memory cards for remote use. Firstly create your program in the Program Builder, keeping in mind that the program will be used in a logger not connected to a computer.

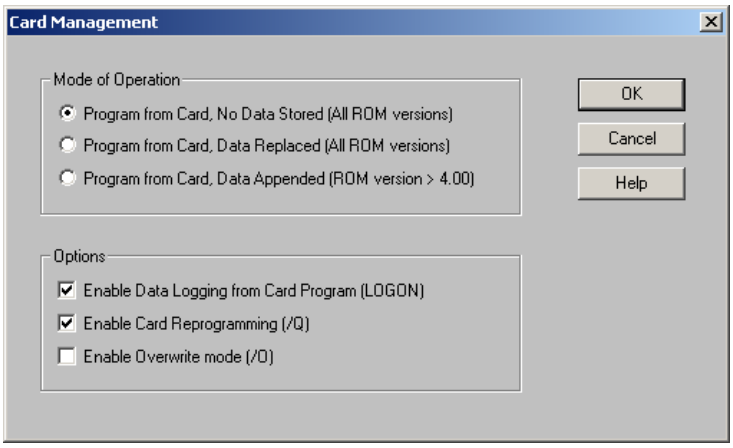
To send the program to a memory card inserted in a *dataTaker* connected to DeLogger, select Program:Send to Card on Connection… from the main menu bar.



This will open the Card Management dialog.

This dialog has 3 predefined modes of operation which configure the card program for 3 commonly used roles of card programming of loggers. The dialog also provides control of the data logging command, the Memory Overwrite switch, and the Execute Card Commands switch.

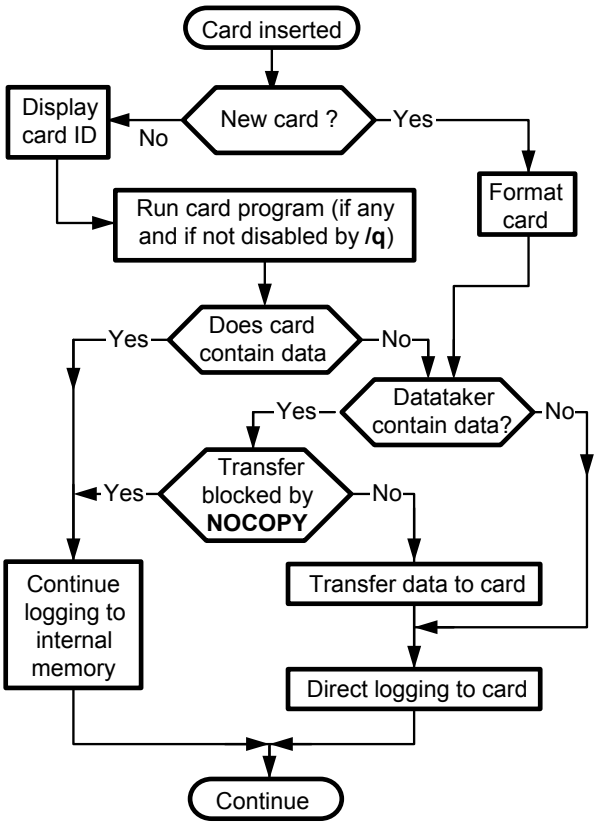
Select the required mode of operation and options settings, then click OK button to send the program to the memory card.



Once the memory card is programmed, the card program should be test run before taking into the field. The program can be run by selecting *dataTaker*:Run Card Program… on the main menu bar.

Summary of Memory Card Insertion

The following flow chart indicates how a memory card containing a *dataTaker* program is managed on insertion.



STATUS Command

The STATUS command produces a report of the current configuration of the *dataTaker*.

The STATUS reports are returned to the host computer, and some are displayed on the display panel (See Section III - Keypad and Display) as 8 predefined Status Screens. Status reports cannot be logged.

General STATUS

Using DeTransfer, a general status report for the *dataTaker* is returned to the host computer by the STATUS command as follows

```
STATUS
```

which produces a response similar to

```
dataTaker 0 Version 7.08
A B C,none Scan Schedules Active,Halted
3,0 Alarms Active,Halted
4 Polynomials/Spans Defined
Logging is ON
166530,0 Internal Data Points Free,Stored
217630,125350 Card Data Points Free,Stored
4090,0 Program Characters Free,Stored
/a/C/d/E/f/h/J/K/l/M/N/o/Q/R/S/t/U/v/w/x/y/Z
```

for an programmed logger.

If the Units Text Switch is disabled (/u), then the format of the status report is truncated to data only. The Parameter22 and Parameter24 commands can be used with this format to define the delimiters for the status information.

If the Units Text Switch is disabled (/u) and Parameter22=13 and Parameter24=13, the general status report is returned without text

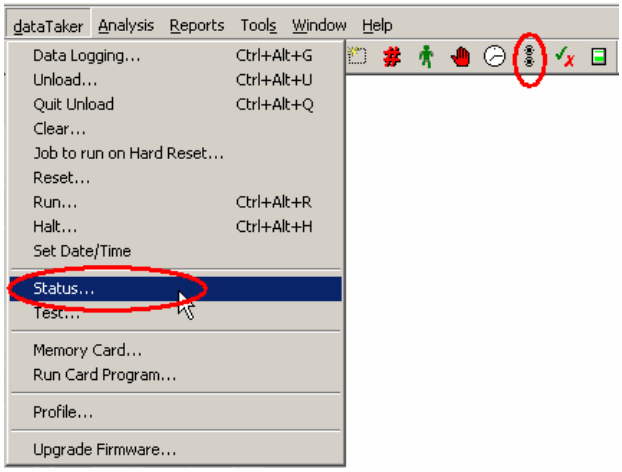
```
0,7.08
A B C,none
3,0
4
0
166530,0
217630,125350
4090,0
/a/C/d/E/f/h/J/K/l/M/N/o/Q/R/S/t/u/v/w/x/y/Z
```

With the Units Text Switch disabled (/u) and Parameter22=44 and Parameter24=13, the general status report is returned in a single line (although shown here with line wrap)

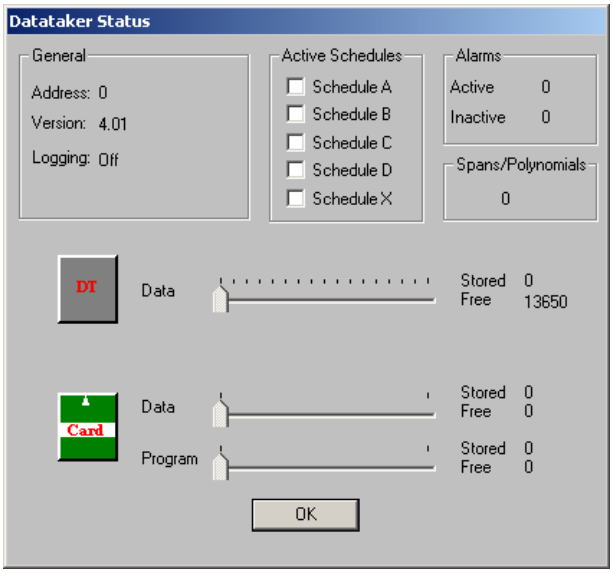
```
0,7.00,A B C,none,3,0,4,0,166530,0,217630,125350,4090,0
/a/C/d/E/f/h/J/K/l/M/N/o/Q/R/S/t/u/v/w/x/y/Z
```

These formats of the status report facilitates reading the *dataTaker* status into dedicated host software. This computer compatible format of the status report can also be defined for the individual status reports described below.

Using DeLogger, the general status report is available by selecting dataTaker:Status... from the main menu bar, or by clicking the Status button on the toolbar,



which produces a dialog similar to the following which details the various status information, in this case for an unprogrammed logger.



Individual STATUS

Individual status reports can also be returned using the STATUS command, which provide greater detail of the configuration.

These individual status reports are not supported by DeLogger, however these could be obtained by issuing the following commands from the Entry Screen of the Text View.

Individual status reports are returned by the command

```
STATUSn
```

where

n is the individual status level required (1 - 13)

The individual status reports to which the 13 status levels relate are listed below

STATUS1 – *dataTaker* network address and firmware version

STATUS2 – lists Schedules active and halted

STATUS3 – lists Alarms active and halted

STATUS4 – lists polynomials and spans defined

STATUS5 – reports current setting of LOGON/LOGOFF

STATUS6 – reports data memory free and data stored in the internal memory

STATUS7 – reports data memory free and data stored in the memory card

STATUS8 – lists the program stored in the memory card

STATUS9 – lists current switch settings

STATUS10 – lists information about the current program

STATUS11 – lists the program on the memory card

STATUS12 – lists the time and date of the first and last data points stored in the internal memory

STATUS13 – lists the time and date of the first and last data points stored in the memory card

The status reports returned by the individual STATUS commands is illustrated in the following examples.

These individual status reports are not supported by DeLogger.

STATUS1

The STATUS1 command returns the *dataTaker* Network address and Firmware version number as follows

```
STATUS1
```

```
dataTaker 0 Version 7.08
```

STATUS2

The STATUS2 command lists the Triggered Schedules that are currently defined, active and halted, as follows

if there is no memory card inserted, the Schedules currently defined in the *dataTaker* are returned

if an invalid memory card containing other data is inserted, then the Schedules used to acquire the data in the memory card are returned. These Schedules may be different to those which are currently defined in the *dataTaker*.

```
STATUS2
```

```
A B C,none Scan Schedules Active,Halted
RA5M  1..3V  8..10TT
RB1H  1..4DS  1..4C
RC15M  4..6L(S2)
```

STATUS3

The STATUS3 command lists the currently defined alarms, active and halted, and the Alarms Schedule as follows

```
STATUS3
```

```
3,1 Alarms Active,Halted
RZ20S
ALARM3(1V>1000.0)"Voltage Over-range"
ALARM4(5TT<>100,105/60S)"Boiler Fault"
ALARM22(1C><256,512)"[RA1S 1C]"
alarm25(T>10:00:00)"Start Logging [GA GB]"
```

Active alarms are shown with the ALARM command in upper case, and inactive alarms are shown with the alarm keyword in lower case.

STATUS4

The STATUS4 command lists the currently defined polynomials and spans as follows

```
STATUS4
```

```
3 Polynomials/Spans Defined
Y1=2.364,1.002,2.2,0.945"KPa"
S5=0.0,1000.0,0.0,100.0"Deg C"
S6=20.0,100.0,0.0,500.0"% RH"
```

STATUS5

The STATUS5 command returns the current setting of the LOGON or LOGOFF commands as follows

```
STATUS5
```

```
Logging is ON
```

or

```
Logging is OFF
```

STATUS6

The STATUS6 command returns the amount of data memory free and the amount of data stored for the internal memory as follows

```
STATUS6
```

```
166530,0 Internal Data Points Free,Stored
```

STATUS7

The STATUS7 command returns the amount of data memory free and the amount of data stored for the memory card as follows

```
STATUS7
```

```
217630,125350 Card Data Points Free,Stored
```

However if there is no memory card inserted in the *dataTaker*, then the STATUS7 command returns

```
No Card Inserted
```

STATUS8

The STATUS8 command returns the number of program characters free and stored in the memory card, and lists the program if there is one stored in the memory card, as follows

```
STATUS8
```

```
4008,82 Program Characters Free,Stored
;CSCANS CALARMS
;/m /n /u P22=44 P24=13
;BEGIN
;RA5M
;1V  3..5TT  2HSC
;ALARM2(5TT>50.0)"Temperature Alarm"
;END
;LOGON
```

if there is a program stored in the memory card, or returns

```
4090,0 Program Characters Free,Stored
```

if there is no program stored in the memory card.

However if there is no memory card inserted in the *dataTaker*, then the STATUS8 command returns

```
No Card Inserted
```

STATUS9

The STATUS9 command lists the current switch settings as follows

```
STATUS9
```

```
/a/C/d/E/f/h/J/K/l/M/N/o/Q/R/S/t/U/v/w/x/y/Z
```

STATUS10

The STATUS10 command details the data that will be returned by the current *dataTaker* program.

```
STATUS10
```

```
27113,1989,1,0,"",<A,"2S",H,<"Dry bulb","",0,0,5,4,3>, <"Wet
bulb","",0,0,5,4,3>,<"Humidity",%RH",0,0,5,4,3> >,<B>,<C>,<D>,<X>
```

the items returned are in the order

program ID
base year
time resolution
card status
$ string definition
followed by Schedule fields that identify individual channels, their format and their units

For a detailed description, consult the "*dataTaker* Advanced Communications Manual".

STATUS11

The STATUS11 command details the data that will be returned by the program in the memory card.

```
STATUS11
```

```
0,0,0,0,"",<A>,<B>,<C>,<D>,<X>
```

The format of the data returned is the same as for STATUS10 command. The above example indicates the memory card does not contain a program.

STATUS12

The STATUS12 command returns the time and date of the first and last data points stored in the *dataTaker*'s internal memory.

```
STATUS12
```

```
08:00:00 on 19/06/2003, 14:25:00 on 20/06/2003 Internal data Start, End times
```

STATUS13

The STATUS13 command returns the time and date of the first and last data points stored in the memory card.

```
STATUS13
```

```
08:00:00 on 19/06/2003, 14:25:00 on 20/06/2003 External data Start, End times
```

Fixed Format Data Mode

The STATUSn commands return status in a different format if the Fixed Format Mode for data is enabled (/H). The format is described in the "*dataTaker* Advanced Communications Manual".

RESET Command

Whenever the *dataTaker* is powered up or hardware reset, the logger executes an initialisation and system reset routine.

During a system reset, the following operations are performed by the *dataTaker*

check the ROM and the RAM memory, and initialise the internal data storage memory

reset the Parameter variables to the default values (See Section III – Parameter Commands)

reset the Switches to the default settings (See Section III – Switch Commands)

calibrate the analog to digital converter, checking for calibration errors (See Section III – TEST Command)

initialise the digital input circuits and counters, checking for any malfunctions (See Section III – TEST Command)

search the internal serial bus for any Channel Expansion Modules, keypad displays, signal conditioning boards, etc. and configure the logger accordingly for those found

Caution : Resetting the *dataTaker* will erase all data stored in the internal memory, erase all schedules and alarms, and erase all polynomial and span declarations. However resetting the logger does not clear data or programs from an inserted memory card.

The reset takes approximately 600mS to complete, and commands should not be entered until the reset function is complete. This delay applies to a power up, hardware or commanded reset.

Using the RESET Command with DeTransfer

Using DeTransfer, a system reset is commanded as follows

```
RESET
```

Whenever a system reset is performed, the *dataTaker* returns the sign on message to the host computer or terminal as follows

```
dataTaker 0 Version 7.00
Initialising...Done
```

All checking of ROM and RAM memory, initialisation of interfaces, setting of defaults, checking of digital input circuits, and calibration of the analog to digital converter, occurs between the 'Initialising...' and 'Done' messages being returned.

Any errors which occur during reset and calibration produce error messages at this time. The error messages are displayed directly by DeTransfer in the upper Receive window, and may be any combination of the following

```
E13-digital failure
E30-calibration failure nn
E33-CSIO bus failure
E46-ROM checksum error
E60-RAM n failure
```

The meaning and cause of each of these initialisation error messages is discussed in the *dataTaker* Technical/Service Manual.

These error messages indicate that the *dataTaker* has failed in some manner. If you receive any of these error messages then disconnect all sensors, actuators, etc. from the *dataTaker* and enter the RESET command again. If there are no error messages now, then

check that there are no large offset or common mode voltages on the input signals

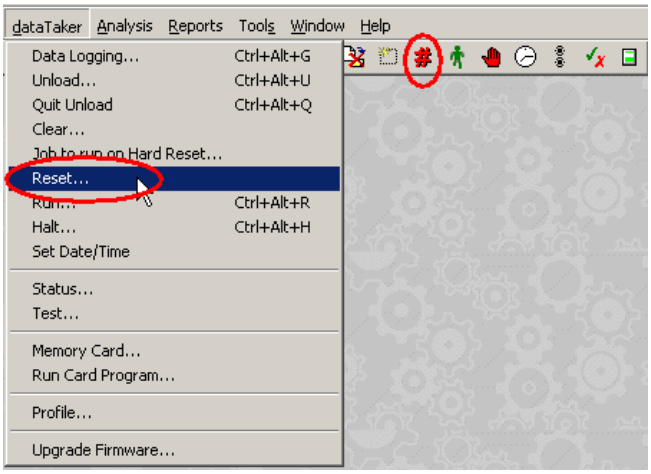
check that there are no ground loops either between the logger and sensors, and between the logger and the host computer

check that the logger power supply is of the correct voltage and current capacity

If the error messages persist, then the logger should be returned to a service centre for repair.

Using the RESET Command with DeLogger

Using DeLogger, a system reset is performed by selecting dataTaker:Reset… from the main menu bar, or by clicking the Reset button on the main tool bar



DeLogger displays any error messages that occur during reset and calibration in the Events Log window, which is usually at the bottom of the screen (make sure View:Show Error Events and View:View Events are checked on the main menu bar).

TEST Command

The TEST command is used to force the *dataTaker* to recalibrate and self-test the analog to digital converter, and to report the status of the analog to digital converter and various internal parameters.

Using TEST Command with DeTransfer

Using DeTransfer, the system TEST is commanded as follows

```
TEST
```

and returns the test information similar to the following

```
dataTaker 524  Ver 7.08
Vos (mV) 0.118
Vfo  (V) 7.875
Fc (kHz) 17.778
CMRR(db) 98.4
Vos3(mV) 0.314
Tos      1.0022
Ios (nA) 0
Ibia(nA) 8
Ibat(mA) -102.1
Vbat (V) 6.3
Vos*(uV) -11
Vos+(uV) 41
Vos-(uV) 27
VosR(uV) -10
Vosd(uV) 44
Ics1(mA) 2.5012
Ics2(uA) 249.93
PASS
```

The meaning of each of these test data is described in the table overleaf.

These test data provide diagnostic assistance if a *dataTaker* is malfunctioning. Their interpretation for fault diagnosis is described in the *dataTaker* Technical/Service Manual.

The format of the test data can be changed with the Units Text Switch and the Parameter24 command. If the Units Text Switch is enabled (/U) then the verbose format of the test parameters as illustrated above is returned.

If the Units Text Switch is disabled (/u) then the brief format of the test data as illustrated below is returned (without the line wrap). The separator between individual test parameters is defined by the Parameter22 command, which by default is a space character

```
524 3.35 0.116 7.875 17.777 98.2 0.317 1.0022 0 8 -102.4 6.3 -10 42 27 -11
46 2.5011 249.92 PASS
```

If the separator character is set to a comma (Parameter22=44) then the test parameters are returned as illustrated below (without the line wrap)

```
524,3.35,0.117,7.875,17.777,98.8,0.315,1.0022,0,8,-102.2,6.3,-11,41,26,
-9,45,2.5 011,249.92,PASS
```

Self Checking of Test Parameters

The TEST command checks each of these test data against internal limits. These limits are listed in the table overleaf. The *dataTaker* returns a general message at the end of the test report to indicate overall condition of the logger, as follows

```
PASS
```

or

```
FAIL
```

The FAIL message is returned if one or more of the test parameters does not fall within the respective limits. If the Units Text Switch is enabled (/U), then the message '- fail' is also appended to each test parameter that has failed. For example the test parameter result

```
CMRR(db) 80.3 - fail
```

indicates that the common mode rejection is unacceptable.

The meaning and allowable limits for the test parameters included in the TEST report are discussed in detail in the *dataTaker* Technical / Service Manual. Diagnostic and service procedures for correcting out of range test parameters is also described.

Individual TEST Data

Individual test data can also be returned by the command

```
TESTn
```

where

n is the individual test line level required
(See the following *dataTaker* Test Table for the individual test data n)

Using DeTransfer for example

```
TEST4
```

returns individual test data for line 4 of the general TEST data, which is the common mode rejection ratio (CMRR) test data

```
CMRR(db) 98.4
```

Repeating TEST Data

Test data can be returned repeatedly using the TESTR or TESTnR commands, which will continually repeat the entire test data or an individual test data item. Repeating test reports are returned by the general commands

TESTR

or

TESTnR

See the following *dataTaker* Test Table to identify the individual test data n.

dataTaker Test Table

Test Data	n	Description	Limits
dataTaker 524 7.00	0	dataTaker config & firmware version	
Vos (mV) 0.009	1	Input offset voltage	±1mV
Vfo (V) 7.308	2	Input voltage for VCO=0Hz	6.0 to 8.50V
Fc (kHz) 18.200	3	VCO centre frequency	11.46 to 23.87kHz
CMRR(db) 99.6	4	Common mode rejection ratio	>90db
Vos3(mV) 0.238	5	Three wire input offset voltage	-1.9 to 3.1mV
Tos 1.0023	6	Terminator attenuation for T channel option	0.99 to 1.01
Ios (nA) -3	7	Input bias current offset	±30nA
Ibia(nA) 15	8	Input bias current	±90nA
Ibat(mA) 0.5	9	Battery current: + charging, - discharging	-500 to +600mA
Vbat (V) 6.6	10	Battery voltage	5.4 to 13.0V
Vos*(uV) -95	11	Multiplexer * single ended offset voltage	-600 to +110µV
Vos+(uV) 33	12	Multiplexer + single ended offset voltage	±180µV
Vos-(uV) 10	13	Multiplexer - single ended offset voltage	±180µV
VosR(uV) 66	14	Multiplexer R single ended offset voltage	-110 to 600µV
Vosd(uV) 5	15	Multiplexer differential offset voltage	±180µV
Ics1(mA) 2.5012	16	Current source 1 current	0.5mA to 10mA
Ics2(µA) 249.93	17	Current source 2 current	1µA to 500µA
PASS	23	Test pass or fail	

Using DeTransfer, the command

```
TEST16
```

will repeatedly return the data for excitation current source 1 (2.500 mA).

The repeating TEST can be stopped by the Halt command, by a Carriage Return (hit the Enter key).

Battery Status

The test parameters returned by the TEST command also contain information of the internal battery status.

The internal battery can be any of the following

1.2Ah gel cell lead acid battery mounted under the top cover of a *dataTaker 500/600*

4Ah gel cell lead acid battery connected to BAT+ and BAT– of the *dataTaker 500/600*

17Ah alkaline battery connected to the internal battery connector of the *dataTaker 500/600*

4Ah gel cell lead acid battery connected to the Lead Acid + and – terminals of a *dataTaker 50*

17Ah alkaline battery connected to the Alkaline + and – terminals of a *dataTaker 50*

The test parameter Ibat reports the battery current as follows

if the Ibat data is positive, then the battery is being charged from an external power supply

if the Ibat data is negative, then the battery is powering the logger and is discharging

The test parameter Vbat reports the battery voltage.

dataTaker Configuration Code

The first test data returned by the ***TEST*** Command is the *dataTaker* configuration, and the version number of the firmware within the logger as follows

```
dataTaker 524 Ver 7.08
```

The configuration code describes the composition of the *dataTaker* as follows

First digit — 5 = DT5xx series logger

Subsequent digits
1 = DT50 series CMOS multiplexer I/O board
2 = DT500 series CMOS multiplexer I/O board
3 = DT500 series relay multiplexer I/O board
4 = Keypad display module in logger or external panel mount module
5 = Vibrating wire conditioning board
6 = Channel expansion module

The subsequent digits are in the order that respective components of the particular *dataTaker* are configured to the internal serial bus. The components are reported in the order

– 5 (always the first digit)
– signal conditioning board
– multiplexer I/O board
– Channel expansion board(s)
– Keypad and display module (integrated or external)

Some examples of common configuration codes follow

51	*dataTaker* 50 with CMOS multiplexer board
514	*dataTaker* 50 with CMOS multiplexer board and external panel display
52	*dataTaker* 500 with CMOS multiplexer board
524	*dataTaker* 500 with CMOS multiplexer board and external keypad display
	dataTaker 600 with CMOS multiplexer board and integral keypad display
53	*dataTaker* 505 with relay multiplexer board
534	*dataTaker* 505 with relay multiplexer board and external keypad display
	dataTaker 605 with relay multiplexer board and integral keypad display
553	Geologger 515 with vibrating wire conditioning board and relay multiplexer board
5534	Geologger 515 with vibrating wire conditioning board, relay multiplexer board and external keypad display
	Geologger 615 with vibrating wire conditioning board, relay multiplexer board and integral keypad display
5366	*dataTaker* 505 with relay multiplexer board and 2 channel expansion modules

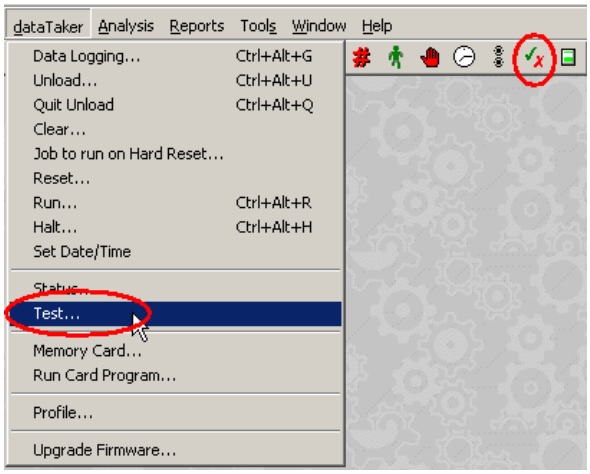
5264 *dataTaker* 500 with CMOS multiplexer board, 1 channel expansion module and external keypad display

dataTaker 600 with CMOS multiplexer board, 1 Channel expansion module and integral keypad display

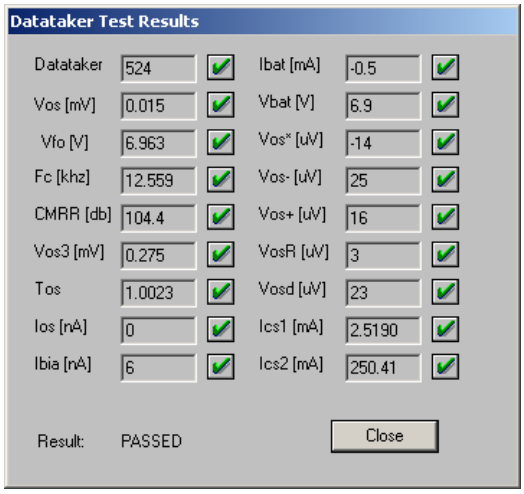
553664 Geologger 515 with vibrating wire conditioning board, relay multiplexer board, 2 channel expansion modules and external keypad display

Using the TEST Command with DeLogger

The TEST command is available by selecting *dataTaker*:Test… from the main menu bar, or by clicking the Test button on the main toolbar



and DeLogger displays the test data in the following dialog



The green checkmarks indicate that the respective test data are within acceptable limits. Any test data that is outside acceptable limits is marked with a red cross.

The overall test result `Pass` or `Fail` is displayed at the bottom of the dialog.

DeLogger does not support individual TESTn commands directly, however these could be obtained by issuing the commands from the Entry Screen of the Text View.

Switch Commands

Switches provide a means for setting various system variables within the *dataTaker*, which configure the logger as required. Switches and Parameters (See Section III – Parameter Commands) provide the bulk of the system configuration.

The Switches have two values – enabled and disabled – in contrast to the Parameters which allow system variables to be set to any value within a defined range.

The Switches are enabled and disabled by commands in the general format

/Switch	Enables the Switch
/switch	Disables the Switch

where

/	is the Switch command identifier
S (upper case)	indicates switch is to be enabled
s (lower case)	indicates switch is to be disabled

The Switches have default settings, and are initialised to these default settings whenever the *dataTaker* is powered up, is hardware reset or a RESET command is executed. The Switches can also be forced to their default settings by the Default Switches command //.

The Switches can be changed at any time, producing an immediate effect.

The Switch commands for the *dataTaker* are listed by function and alphabetically in the following tables. These tables also indicate the default setting for each of the switches.

The particular function of each Switch command are also detailed and discussed in the relevant sections throughout this manual.

Description of the Switch Commands

The function and operation of each of the Switch commands is briefly described below, and where appropriate reference is made to sections elsewhere in this manual for further description of use.

/A – Alarms Display Switch

The display of alarm state and messages on the display panel is normally disabled (/a).

The state of individual alarms can be enabled using the List/Edit keys of the keypad, and display of the state all alarms can be enabled by /A (See Section III – Alarms).

/C – Channel Identifier Switch

Data returned to the host computer is normally prefixed with the channel number (/N) and channel type identifier (/C) to identify the source of the data. This switch only has effect if the Channel Number Switch is enabled.

The channel identifier can be disabled (/c) to reduce the length of the data string (See Section III – Format of Returned Data).

/D – Date Switch

The date or day number that channels are read or calculations performed can be automatically returned with real time and logged data (/D). The format is defined by Parameter31.

The date or day number is normally not returned with the real time or logged data (/d). (See Section III – Real Time Clock).

Alphabetical Listing of the Switches

Switches		Function	Default Setting
Enable	Disable		
/A	/a	Display alarm state and messages	/a
/C	/c	Return channel identification with data	/C
/D	/	Prefix date/day to returned and logged data	/d
/E	/e	Echo command characters	/E
/F	/f	Fix internal scan table and schedules	/f
/H	/h	Fixed format data mode	/h
/J	/j	Over range error carry	/J
/K	/k	Analog to digital converter autocalibration	/K
/L	/l	Return logger address with data	/l
/M	/m	Return warnings and error messages	/M
/N	/n	Return channel number with data	/N
/O	/o	Overwrite oldest data in data store	/o
/Q	/q	Programming from memory card	/Q
/R	/r	Return data to host computer in real time	/R
/S	/s	Synchronize schedules to real time	/S
/T	/t	Prefix time to returned and logged data	/t
/U	/u	Return units text with data	/U
/V	/v	Speaker control (Geologger)	/v
/W	/w	Return intermediate/working data	/w
/X	/x	Display progressive min and max	/x
/Y	/y	Priority for returning data	/y
/Z	/z	Return alarm messages	/Z
/ /		Set all switches to defaults	

Functional Listing of the Switches

Switches		Function	Default Setting
Enabled	Disabled		
Schedules			
/F	/f	Fix internal scan table and schedules	/f
/S	/s	Synchronize schedules to real time	/S
/R	/r	Return data to host computer in real time	/R
/Y	/y	Priority for returning data	/y
Real Time Clock			
/D	/d	Prefix date/day to returned and logged data	/d
/T	/t	Prefix time to returned and logged data	/t
Alarms			
/A	/a	Display alarm state and messages	/a
/Z	/z	Return alarm messages	/Z
Communications			
/E	/e	Echo command characters	/E
ADC Settings			
/K	/k	Analog to digital converter autocalibration	/K
Data Logging			
/O	/o	Overwrite oldest data in data store	/o
Memory Card Programming			
/Q	/q	Programming from memory card	/Q
Format of Returned Data			
/C	/c	Return channel identification with data	/C
/H	/h	Fixed format data mode	/h
/L	/l	Return logger address with data	/l
/M	/k	Analog to digital converter autocalibration	/M
/N	/n	Return channel number with data	/N
/U	/u	Return units text with data	/U
Miscellaneous			
/J	/j	Over range error carry	/J
/V	/v	Speaker control (Geologger)	/v
/W	/w	Return intermediate/working data	/w
/X	/x	Display progressive min and max	/x
//			

/E – Echo Switch

All commands received by the *dataTaker* are normally echoed back to the host computer (/E).

Echo can be disabled (/e) if required. Echoed commands can be used to determine if the commands are properly received (See Section III – Communication Protocols and Commands).

/F – Fix Channel Table Switch

When new Schedules are entered into the *dataTaker*, these normally replace or overwrite any existing schedules (/f), unless data has been logged for the original Schedule.

However existing Schedules can be protected by /F, preventing new schedules from being entered. Fixing the internal channel table prevents both triggers and schedule lists from being changed. (See Section III – Schedules)

/H – Data Format Mode Switch

Data is normally returned to the host computer in a user definable Free Format Mode (/h).

However a predefined Fixed Format Mode can be selected (/H), which provides a predictable *dataTaker* environment in which to operate (See Section III – Format of Returned Data).

/J – Over Range Error Carry Switch

When data from input channels is used in calculations, then any measurement errors which result in the input channel returning the universal error value of 99999.9 will also force the result of all subsequent calculations to 99999.9 (/J).

Forcing of the result of the calculation to 99999.9 can be disabled (/j), and the value 99999.9 is used in the calculation which will return a result based on this data (See Section III – Channel Variables and Calculations).

/K – Autocalibration Switch

The *dataTaker* normally checks its analog to digital converter against internal references before each scan, and if error limits are exceeded then the analog to digital converter is automatically re–calibrated (/K).

Automatic calibration requires approximately 600mS to complete, and can be disabled (/k) to prevent gaps in data, particularly during rapid data acquisition (See Section II – Analog Processing).

/L – Logger Address Switch

Blocks of data returned to the host computer can be prefixed with the address of the logger(/L) to identify the origin of data.

The logger address is normally not returned with the blocks of data (/l) from the logger (See Section III – Format of Returned Data).

/M – Messages Switch

When the *dataTaker* encounters syntax errors in entered commands, or encounters error conditions while executing an application, the logger normally returns warning or error messages to the host computer (/M), depending on the severity of the condition.

Return of warning or error messages can be disabled (/m) to prevent these from disrupting host software, etc. (See Section III – Warning and Error Messages).

/N – Channel Number Switch

The data returned to the host computer is normally prefixed with the channel number (/N) and the channel type (/C) to identify the source of the data.

The channel number can be disabled (/n) to reduce the length of the data string (See Section III – Format of Returned Data).

/O – Overwrite Switch

The internal data memory or a memory card is normally managed as a data stack (/o), and data logging ceases when the memory is filled.

A circular buffer or overwrite mode can be selected (/O), for which the newest data progressively overwrites the oldest data after the data memory is filled. (See Section III – Logging Data to Memory)

/Q – Execute Card commands Switch

When memory cards containing commands or programs are inserted into the *dataTaker*, then the commands are normally downloaded and executed (/Q).

Automatic down loading and execution of memory card programs can be disabled (/q) to protect the current application (See Section III – Programming from Memory Cards).

/R – Return Data Switch

Data resulting from scanning input channels, executing calculations, etc. is normally returned to the host computer in real time (/R).

However this can be disabled (/r) if required (See Section III – Format of Returned Data).

/S – Synchronization Switch

Whenever Data Schedules and Alarm Schedules triggered by time are entered into the *dataTaker*, the first execution of the Schedule normally occurs on the next multiple of the trigger interval since last midnight (/S).

If synchronization is disabled (/s), then the first execution occurs at the end of the first trigger interval following entry of the Schedule (See Section III – General Schedule Commands).

/T – Time Switch

The time at which channels are read or calculations performed can be returned automatically with real time and logged data (/T).

The time is normally not automatically returned with the real time or logged data (/t). (See Section III – The Real Time Clock).

/U – Units Text Switch

Data returned to the host computer is normally suffixed with the units of measurement (/U) to identify the type of data.

The units can be disabled (/u) to reduce the length of the data string (See Section III – Format of Returned Data).

/V – Speaker Switch

The *Geologgers* have a speaker and headphone jack for listening to the operation of vibrating wire sensors during fault diagnosis. Signal output to the speaker or headphones is selected by enabling the switch (/V).

The switch is normally disabled (/v) and there is no signal output to the speaker or headphones.

/W – Intermediate Data Switch

Channel Variables normally return the current value when assigned new values as channel options or from expressions. This can be prevented by the W channel option where Channel Variables are assigned values as part of internal calculations, use in alarms, etc.

The W channel option can be over–ridden (/W) allowing intermediate or working data to be returned from Channel Variables for debugging purposes. The W channel option is not normally over–ridden (/w) so that Channel Variables with the W channel option do not return data (See Section II – Channel Variables and Calculations).

/X – Max Min Display Switch

Whenever input channels are statistically scanned, the display of the maximum and minimum data on the display panel is normally updated at the Triggered Schedule interval when other statistical data is calculated (/x).

Display of the maximum and minimum data can be progressively updated at the Statistical Sub–Schedule interval as the channels are scanned (/X). (See Section III – Statistical Sub–Schedule).

/Y – Data Priority Switch

When data from input channels is returned to the host computer in real time and is logged to memory, then scanning of new data normally has priority over returning data for the previous scan (/y).

Priority for returning data for the previous scan can be selected over new scans (/Y). The scan rate will then be dictated by the format of data returned and the communications baud rate (See Section III – General Schedule Commands).

/Z – Alarm Messages Switch

When alarms with alarm messages test true, the message is normally returned to the host computer (/Z).

Return of alarm messages can be disabled (/z) to suit requirements of host software or modems (See Section III – Alarms).

Setting Switches from DeTransfer

The Switch commands are included in commands and programs sent from DeTransfer for example

```
/m/e/n/c/u    'Disable text in data
/A/z          'Show alarms only on display
P22=44
BEGIN
 RA1S
  1..5TK
  ALARM1(1TK>100)"Column 1 Over Temp"
  ALARM2(2TK>100)"Column 2 Over Temp"
END
/O            'Rolling memory
LOGON
```

Normally the Switches are set at the start of the program, however can be set at any time either in the program or by separate command after the program has been entered into the logger.

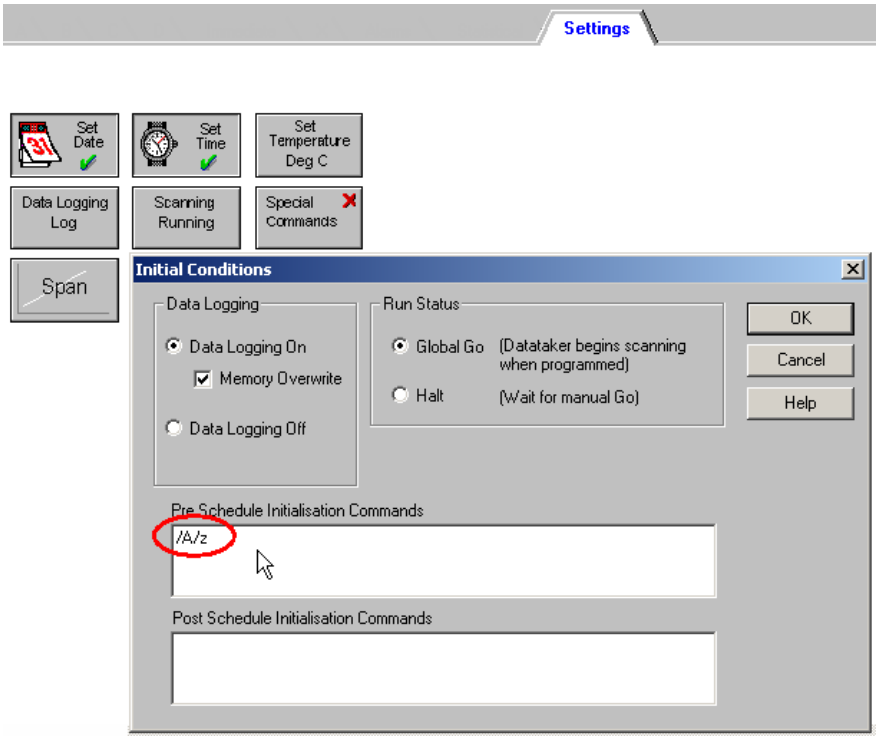
Setting Switches from DeLogger

DeLogger automatically manages all Switches which determine format of returned data, communications, etc. that directly affect the DeLogger / *dataTaker* interface.

DeLogger also provides the user opportunity to set some Switches in dialogs, for example the Memory Overwrite Switch can be set in the Settings tab of the Program Builder (see below).

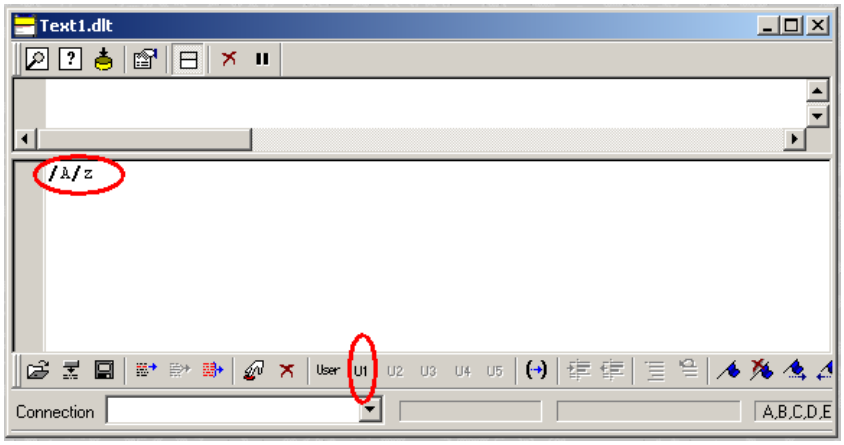
However DeLogger does not provide direct support for managing Switches that change configurations of the logger but do not affect the DeLogger interface, for example the Switches which manage the display panel.

These Switches can be set as required by the user from the Program Builder. In the Program Builder click on the Settings tab, then right click on the Special Commands button and select Properties...The Initial Conditions dialog opens, and Switch commands can be entered into the Pre Schedule Initialization and Post Schedule Initialization input fields.



The Switch commands in the Pre Schedule Initialization field are sent to the logger before the program schedules, and the Switch commands in the Post Schedule Initialization input field are sent to the logger after the program schedules.

Switch commands can also be sent directly to the logger from the Entry Screen of the Text View, and by appropriately defined User buttons at the bottom of the Entry Screen.



Parameter Commands

Parameters provide a means for setting various system variables within the *dataTaker*, which configure the logger as required. Parameters and Switches (See Section III – Switch Commands) provide the bulk of the system configuration.

The Parameter commands however allow these system variables to be set to a value within a range, in contrast to the Switch commands which have only two possible values – enabled and disabled.

The Parameter commands can be used for either altering the value of the system variables directly, or to return the current settings of these system variables.

The general formats of the Parameter commands are as follows

To set a Parameter

```
Pn=value
```

To return a Parameter setting

```
Pn
```

where

n	is the Parameter number
value	is the setting for the Parameter

The Parameter commands for the *dataTaker* are listed by function and numerically in the following tables. The tables also indicate the permissible range for each Parameter, and the default setting.

Parameters have a default setting, and are initialised to these during a power up, following a hardware reset, or when a RESET command is executed.

The purpose and function of each of the configurations set by the Parameter commands is detailed and discussed in the relevant sections elsewhere in this manual.

The Parameter commands can generally be changed at any time, producing an immediate effect. The exception is Parameter30 which must be set before any Data Schedules or Alarm Schedules are entered.

Setting of Parameters

Parameters are defined in the units indicated in brackets beside the description of each Parameter in the tables below. For example Parameter2 in the table below

2	Internal temperature trim (0.001°C)	–30000 to 30000	0°C

has units of 0.001°C. Therefore in this case to apply a +0.5°C trim to the internal temperature sensor, Parameter2 is set to 500.

Parameter settings can only be within the range specified for each Parameter in the tables below. If a setting is entered which is outside of the range for the Parameter, then an 'E8–Parameter read/set error' message is returned.

Several Parameters are read only, and cannot be set to values. These Parameters report the frequency of various errors associated with network and host communications, and the frequency of lost counts on the digital input channels.

Numerical Listing of Parameters

Parameter	Function	Permitted Values	Default Value
0	Calibration interval zero voltage (1 μV)	0 to 10000	4μV
1	2.5000V reference trim (10 μV)	–30000 to 30000	0
2	Internal temperature trim (0.001 °C)	–30000 to 30000	0°C
3	Reference resistor trim (1 mªΩ)	–30000 to 30000	1mΩ
4	Lost count flag (read only)	Number errors	0
7	Network turn around time (13.75 mS)	0 to 30000	1(13.75mS)
9	Remote network errors (read only)	Number errors	0
10	ADC settling period (1 mS)	0 to 3000	10
11	Mains frequency (1 Hz)	48 to 1000	50/60Hz
12	Transmission errors (read only)	Number errors	0
13	Digital inputs sampling period (1 mS)	0, 10 to 100	50mS
14	Password timeout (1 Sec)	1 to 30000	300sec
15	Low power operation (mode)	0, 1 or 2	0
16	ADC warm up time after wake (100 mS)	1 to 255	1 (100mS)
17	Delay to low power after comms (1 Sec)	1 to 255	30 sec
18	Display auto scroll time (1 Sec)	1 to 255	2 sec
19	Status screens to display (bit map)	0 to 255	255
20	Wake schedules (bit map)	0 to 255	0
21	Return data to address number	0 to 128	128
22	Data delimiter character (ASCII)	1 to 127	13 (CR/LF)
23	Number of calibration cycles	1 to 10	3
24	Schedule delimiter character (ASCII)	1 to 127	13 (CR/LF)
25	Unload finished character (ASCII)	0 to 127	0
26	XOFF timeout before automatic XON (secs)	30	1 to 254
27	Input to counter 1HSC (mode)	0 to 3	0
30	Number of alarms permitted	0 to 110	20
31	Date Format	0,1 or 2	1 or 2
32	Number of significant digits returned	0 to 9	5
33	Fixed data item field width	0 to 80	0
36	Temperature units	0, 1, 2 or 3	0 (°C)
38	Decimal point character (ASCII)	0 to 127	46 (.)
39	Time format	0, 1 or 2	0
40	Time separator for HH:MM:SS (ASCII)	1 to 127	58 (:)

Functional Listing of Parameters

Parameter	Function	Permitted Values	Default Value
ADC Settings			
0	Calibration interval zero voltage (1 μV)	0 to 10000	4μV
1	2.5000V reference trim (10 μV)	–30000 to 30000	0
2	Internal temperature trim (0.001 °C)	–30000 to 30000	0°C
3	Reference resistor trim (1 mªΩ)	–30000 to 30000	1mΩ
10	ADC settling period (1 mS)	0 to 3000	10
11	Mains frequency (1 Hz)	48 to 1000	50/60Hz
23	Number of calibration cycles	1 to 10	3
Digital Channels			
13	Digital inputs sampling period (1 mS)	0, 10 to 100	50mS
27	Input to counter 1HSC (mode)	0 to 3	0
Alarms			
30	Number of alarms permitted	0 to 110	20
Power Mode			
15	Low power operation (mode)	0, 1 or 2	0
16	ADC warm up time after wake (100 mS)	1 to 255	1 (100mS)
17	Delay to low power after comms (1 Sec)	1 to 255	30 sec
20	Wake schedules (bit map)	0 to 255	0
Data Format			
22	Data delimiter character (ASCII)	1 to 127	13 (CR/LF)
24	Schedule delimiter character (ASCII)	1 to 127	13 (CR/LF)
25	Unload finished character (ASCII)	0 to 127	0
32	Number of significant digits returned	0 to 9	5
33	Fixed data item field width	0 to 80	0
36	Temperature units	0, 1, 2 or 3	0 (°C)
38	Decimal point character (ASCII)	0 to 127	46 (.)
Real Time Clock			
31	Date Format	0,1 or 2	1 or 2
39	Time format	0, 1 or 2	0
40	Time separator for HH:MM:SS (ASCII)	1 to 127	58 (:)

Functional Listing of Parameters

Parameter	Function	Permitted Values	Default Value
Data Logging			
25	Unload finished character (ASCII)	0 to 127	0
Display			
18	Display auto scroll time (1 Sec)	1 to 255	2 sec
19	Status screens to display (bit map)	0 to 255	255
Network			
7	Network turn around time (13.75 mS)	0 to 30000	1(13.75mS)
9	Remote network errors (read only)	Number errors	0
21	Return data to address number	0 to 128	128
Miscellaneous			
4	Lost count flag (read only)	Number errors	0
12	Transmission errors (read only)	Number errors	0
14	Password timeout (1 Sec)	1 to 30000	300sec

Description of Parameter commands

The function and operation of each of the Parameter commands is briefly described below, and where appropriate reference is made to sections elsewhere in this manual for further description of use.

P0 – Calibration interval – zero voltage (1 µV)

Sets the allowable drift for the zero voltage reference before automatic recalibration occurs. The drift can be changed in increments of 1 µV, over the range of 0 to 100mV (See Section II – Analog Processing).

P1 – 2.5000V reference voltage trim (10 µV)

A software trim for the 2.5000V voltage reference for range calibration. This can be used to adjust the *dataTaker* accuracy when calibrating against an external standard reference voltage. The software trim is applied in increments of 10µV over the range of ±300mV
(See Section II – Analog Processing).

P2 – Internal temperature trim (0.001°C)

A software trim for the internal LM35 temperature sensor to adjust accuracy. This can be used to adjust internal thermocouple reference junction temperature compensation where a temperature gradient across the logger is suspected. The software trim is applied in increments of 0.001°C over the range of ±30°C (See Section II – Analog Processing).

P3 – Reference resistor trim (1 mOhm)

A software trim for the internal 100.0ª ±0.1% reference resistor used for resistance range calibration. This can be used to adjust the *dataTaker* accuracy when calibrating the logger against an external standard resistance. The software trim is applied in increments of 1mªOhm over the range of ±30ª (See Section II – Analog Processing).

P4 – Lost count flag (read only)

This is a read only Parameter, for which any positive value will indicate that counts have been lost by the low speed counters. Loss of counts can occur if counts are received while the *dataTaker* is in sleep mode (See Section II – The Low Speed Counters).

P7 – Network turn around time (13.75 mS)

Sets the turn around time for the RS485 network. This normally does not have to be changed unless RS485 line drivers, modems, radios, etc. are used on the network to extend the networking distance. The turn around time is set in increments of 13.75mS over the range 0 to 412.5 Sec. A turn around time of 300mS for a radio is set by Parameter7=22 (See Section III – Network Commands).

P9 – Remote network errors (read only)

This is a read only Parameter, for which the value indicates the number of remote network errors that have occurred (See Section III – Network Commands).

P10 – ADC settling period (1 mS)

The time between the selection of an analog input channel and the beginning of the analog to digital conversion can be adjusted. This time sets the period during which sensors can settle after being selected. Changing the ADC settling period will affect the sampling rate of the *dataTaker*.

The ADC settling period is set in increments of 1mS, over the range of ±30 Sec. This Parameter can also be accessed as System Variable 7 (7SV) (See Section II – Analog Processing and Appendix E).

P11 – Mains Frequency (1 Hz)

The ADC sampling period can be set to improve rejection of local noise. The ADC sampling period is normally set to the local mains frequency (defined by the country DIP switch setting), however it can be set in increments of 1 Hz over the range of 48 to 1000Hz.

Increasing the mains frequency setting will in turn decrease the ADC sampling period, and increase the ADC sampling rate (See Section II – Analog Processing and Appendix E).

P12 – Transmission errors (read only)

This is a read only Parameter, for which the value indicates the number of transmission errors that have occurred in protocol mode communications (See Section III – Communications Protocol).

P13 – Digital input sampling period (1 mS)

The sampling interval of the digital input channels (and keys of the display keypad) can be set, which determines the minimum detectable pulse width. Setting this Parameter to shorter periods will increase the sensitivity of the digital inputs, but decreases the overall performance of the logger because more time is spent sampling the digital inputs. Setting the Parameter13 to 0 disables digital inputs. (Section II – The Digital Input Channels).

P14 – Password timeout (1 Sec)

When a password is defined, the *dataTaker* will automatically SIGNOFF after this period following last communications. The password timeout period is set in increments of 1 Sec over the range of 1 to 30000 Secs. The default is 300 Secs or 5 minutes. (See Section II – The RS232 COMMS Port and Section III – Communications Protocols and Commands).

P15 – Low power operation

The mode for the *dataTaker* low power operation can be set as follows

P15=0 Automatic selection of low power mode based on battery current (Default)

P15=1 Force low power mode

P15=2 Force normal power mode

(See Section II – Powering the *dataTaker*).

P16 – ADC warm up time (100 mS)

The minimum time from wake–up to the first analog to digital conversion to be defined. Changing this Parameter is usually only appropriate for sensors with long powerup settling times. The ADC warm up time is normally 100mS, and can be increased in increments of 100mS over the range of 0 to 25 Secs (See Section II – Analog Processing).

P17 – Delay to low power mode (1 Sec)

The delay period before the *dataTaker* enters low power mode following communication, external wake or keypad input, can be set. This delay is normally 30 Sec, and can be changed in increments of 1 Sec over the range of 0 to 255 Secs (See Section II – Powering the dataTaker).

P18 – Display auto scroll time (1 Sec)

Sets the time for display of each screen when the display is in auto scroll mode. The auto scroll time is normally 2 Secs, and can be set in the range of 0 to 255 Secs. (See Section III – Keypad and Display).

P19 – Status screens to display (bit map)

The status screens that are to be displayed on the display panel is set by a bit map.

By default Parameter19=255 and all status screens are enabled for display. Setting Parameter19=0 disables all status screens from the display (See Section III – Keypad and Display).

P20 – Wake Schedules (bit map)

A bit map or mask of the Schedules that cannot wake the dataTaker when they next trigger.

By default Parameter20=255 and all Schedules can wake the logger. Setting Parameter20=0 disables all of the Schedules from waking the logger (See Section II – Powering the dataTaker).

P21 – Return data to address number

The return address for all data and reports sent by a logger in a network is normally the logger from which commands are received. However this can be forced to an alternative address.

By default (P21=128) all data from a networked logger is returned to the RS232 port of the logger which issued the command. (See Section III – Network Commands).

P22 – Data delimiter character (ASCII)

Defines the ASCII character to delimit the data items in data blocks when units text is disabled (/u). The character can be any character in the range ASCII 1 to ASCII 127. The default is CR, to which a LF is automatically appended, and successive data items are on successive lines of a terminal or printer (See Section III – Format of Returned Data).

P23 – Number of calibration cycles

Sets the number of calibration cycles to be performed during an autocalibration. This is a compromise between calibration speed and accuracy, and can be increased to a maximum of 10 cycles to reduce effects of local noise during autocalibration (See Section III – Analog Processing).

P24 – Scan delimiter character (ASCII)

Defines the ASCII character to delimit the successive data blocks when units text is disabled (/u). The character can be any character in the range ASCII 1 to ASCII 127.

The default is CR, to which a LF is automatically appended (See Section III – Format of Returned Data).

P25 – Unload finished character (ASCII)

Defines the ASCII character which indicates the last item of data from the data storage memory. The character can be any character in the range of ASCII 1 to ASCII 127. The default is ASCII 0 and no character is returned (See Section III – Returning Data from Memory and Section III – Format of Returned Data).

P26 – XOFF Timeout

When the logger is XOFFed by a host computer, the logger will automatically restore the XON state after a period of time.

By default this period is 30 seconds, and can be set in the range of 1 to 254 seconds. Setting P26=0 disables the timeout, and setting P26=255 the logger ignore XOFFs.

P27 – Input to Counter 1HSC

The input for high speed counter 1HSC can be setup as follows

P27=0	input is from terminal C1, gate is permanently enabled
P27=1	input is from an internal 32768 Hz source, gate is signal on C1
P27=2	input is from terminal C1, gate is signal on C2
P27=3	input is from an internal 1024 Hz source, gate is permanently enabled

The default is P27=0 for counting external signals (See Section III – High Speed Counters).

P30 – Number of alarms permitted

Defines the maximum number of alarms that can be entered. The maximum number of alarms can be from 0 to 110, with the reverse being the maximum number of input channels and calculations that can be entered in Data Schedules. This Parameter MUST be set before any Alarms or Data Schedules are defined (See Section III – Alarms).

P31 – Date Format

P31=0	Day Number
P31=1	DD/MM/YYYY (default Rest of the World)
P31=2	MM/DD/YYYY (default North America)

(See Section III –The Real Time Clock).

P32 – Number significant digits returned

Sets the number of significant digits in data returned. By default all data is returned to a maximum of 5 significant digits, and can be set in the range of 1 to 9 significant digits (See Section III – Format of Returned Data).

P33 – Fixed data item field width

Each item of data returned to the host computer may have a fixed field width capacity and may be set to between 1 and 80 characters. By default each item of data returned is of variable width, which is

dependent on the magnitude of the signal read. The setting of Parameter 33 allows for fixing the number of characters returned with each data item.

Each data item is right justified and space padded. If the name or value is larger than the field width specified by the Parameter33 setting, then the name or value is truncated from the right. The $ string channel, Alarm messages, Test results and Status reports are not effected by the setting of Parameter33. (See Section III – Format of Returned Data)

P36 – Temperature units

The units of temperature for temperature data can be defined. The temperature units for data already stored in the data memory cannot be altered. The temperature units can be defined to be

P36=0	Deg C (Default)
P36=1	Deg F
P36=2	Deg K
P36=3	Deg R

(See Section II – Measuring Temperature).

P38 – Decimal point character (ASCII)

The character to be used as the decimal point in floating point data can be defined. Any ASCII character in the range of 1 to 127 is permitted, and the default is ASCII 46 (period).

P39 – Time Format

Defines the time format for returned and logged data as follows.

P39=0	HH:MM:SS (Default)
P39=1	Seconds
P39=2	Decimal hours

(See Section III– The Real Time Clock).

P40 – Time separator character (ASCII)

The ASCII character used as the separator in the HH:MM:SS time format can be defined. Any ASCII character in the range of 1 to 127 is permitted, the default is ASCII 58 (colon). (See Section III – The Real Time Clock).

Setting Parameters from DeTransfer

The Parameter commands are included in commands and programs sent from DeTransfer for example

```
/m/e/n/c/u
P22=44            'Data delimiter is comma
P10=2             'Faster channel settling
P11=1000          'Faster ADC sampling
BEGIN
 RA1S
  1..5V
END
LOGON
```

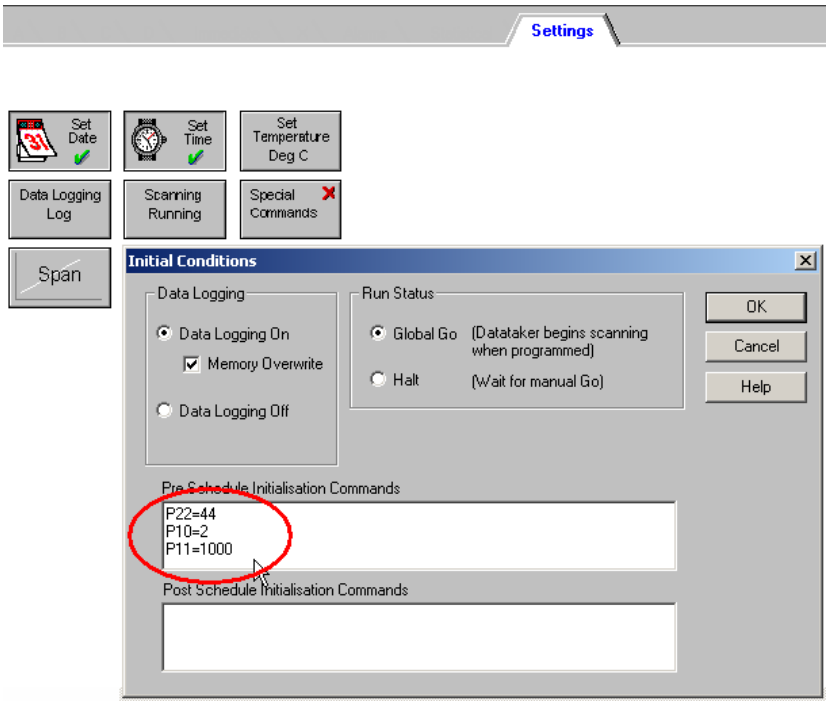
Normally the Parameters are set at the start of the program, however can be set at any time either in the program or by separate command after the program has been entered into the logger.

Setting Switches from DeLogger

DeLogger automatically manages all Parameters which determine the format of returned data, the logger communications, etc. that directly affect the DeLogger / *dataTaker* interface.

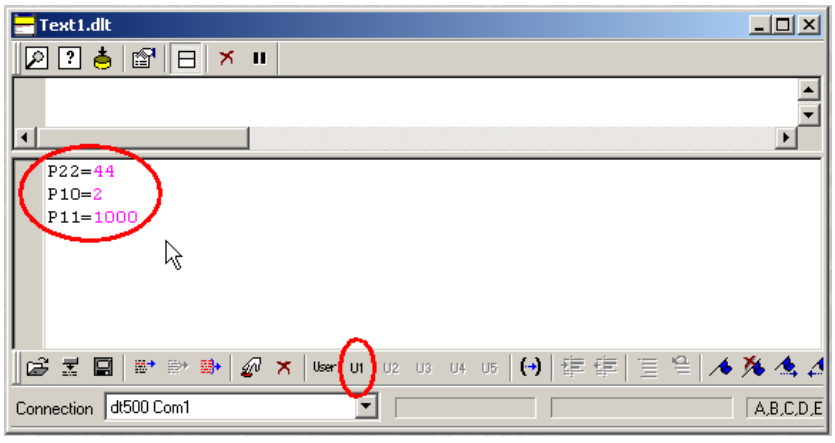
DeLogger does not provide direct support for managing Parameters that change configurations of the logger but do not affect the DeLogger interface, for example the Parameters which manage the analog to digital converter.

These Parameters can be set as required by the user from the Program Builder. In the Program Builder click on the Settings tab, then right click on the Special Commands button and select Properties...The Initial Conditions dialog opens, and Parameter commands can be entered into the Pre Schedule Initialization and Post Schedule Initialization input fields.



The Parameter commands in the Pre Schedule Initialization field are sent to the logger before the program schedules, and the Parameter commands in the Post Schedule Initialization input field are sent to the logger after the program schedules.

Parameter commands can also be sent directly to the logger from the Entry Screen of the Text View, and by appropriately defined User buttons at the bottom of the Entry Screen.



dataTaker Networks

Up to 31 *dataTaker 500/600* series loggers can be configured into a network via their NETWORK interfaces. The individual loggers in the network are supervised from a host computer connected to the RS232 COMMS interface of any of the loggers in the network. There is no slave / master hierarchy.

Host Computer Software

DeLogger 4 or DeLogger Pro 4 do not support the *dataTaker* network.

A *dataTaker* network can be supervised from DeTransfer, using the network addressing and commands described below.

Similarly, purpose written software can supervise a network using the network addressing and commands described below.

The *dataTaker* Address

The individual *dataTaker 500/600* series loggers in the network are identified by a unique address. This address is used

to direct commands to loggers on the network

to direct messages to the RS232 COMMS interface of loggers on the network

The *dataTaker* network address is prefixed to commands for particular loggers on the network.

The *dataTaker 50* does not have a network interface. However the *dataTaker 50* supports the address, and it is useful for identifying the logger in STATUS and STATUS1 commands, and for returning data prefixed with a logger address or number that identifies the source of data. The logger address is prefixed to returned data by enabling the Logger Address Switch (/L).

Setting the Address

The address of each of the *dataTaker* data loggers in a network is set by the 8 bit DIP switch which is located under the top cover of the logger. The logger network address can be set in the range 0 to 31.

When the top cover of the logger has been removed, the DIP switch can be found immediately behind the memory card socket. Bits 4, 5, 6, 7 and 8 of the DIP switch are used to set the address (See Section II – The NETWORK Interface).

Make sure that each logger in the network has a different address, otherwise conflicts will occur. The addresses do not need to be consecutive.

Directing Commands and Programs to Loggers

Commands and programs are directed to specific loggers in the *dataTaker* network by using the address of the target logger.

Directing Commands to the Local dataTaker

Whenever a command or program is to be sent to the logger in the network that the host computer is connected to, then no addressing is required.

```
BEGIN
 RA10M
   1V 2R 3F
END
LOGON
```

Commands without addresses are received and executed by the local *dataTaker*, and are not transmitted onto the network.

Directing Commands to the Remote dataTakers

When commands or programs are to be sent to a remote *dataTaker* data logger in a network, then the target logger must be identified by its unique address. The remote *dataTaker* address is prefixed to the command in the general format

#n *commands*

where

#	is the address identifier
n	is the remote logger address (0 – 31)
commands	is the command or commands to be sent

Specifying the address of a remote logger in a command is illustrated in the following example

```
#4 STATUS
```

Specifying an address of a remote logger in a program is illustrated in the following example

```
#4 BEGIN
#4  RA10M
#4    1V 2R 3F
#4 END
#4 LOGON
```

In this case the local logger places the command or program onto the network, which is then identified and processed by *dataTaker* number 24 (if present).

The # character indicates that the following command is for the remote logger at the address specified.

Broadcast Address for Commands

A broadcast address for commands and programs is also supported by the *dataTaker* network. The broadcast address allows commands to be sent to the local logger, and to all remote loggers in a network. This is despite the fact that the individual loggers in the network are set to different addresses.

The broadcast address is prefixed to the command in the general format

commands

where

##	is the broadcast address identifier
commands	is the broadcast command

Specifying the broadcast address for a command that is to be sent to all loggers in a network is illustrated in the following example

```
## /m/e/n/u  P22=44  T=10:30:00  CLEAR
```

Broadcasting commands to a network may be used to configure all loggers, to set the real time clock of all loggers, to clear the data memory of all loggers, etc. by a single command from the host computer.

If a broadcasted command results in the remote loggers returning data via the network, then because of the nature of the network protocol the data will return to the host in a random order from the networked loggers.

Communicating Messages to Other Devices on the Network

The host computer can communicate via the network with devices such as computers, terminals, printers and displays that are connected to the RS232 COMMS interfaces of remote loggers (remember that the host computer is connected to the RS232 COMM interface of the local logger).

Text strings or messages can be sent from the host computer to these devices by the command

@n *message*

where

@	is the message address identifier
n	is the remote logger address (0 – 31)
message	is the text string or message to be sent

The @ character indicates that the message is not a command for the remote logger at the address specified, but is a message for the device on the RS232 COMMS interface of the remote logger.

The address of the remote logger is prefixed to the text string or message as follows

```
@5 HELLO THERE^M^J
```

In this case the message "HELLO THERE CR/LF" is sent from the host computer via the local logger and network to a device connected to the RS232 COMMS interface of logger number 5.

Control characters are included in the message in the format ^M (CR), ^J (LF), ^[(ESC), etc. Note that CR/LF is not automatically appended to the message, and must be specifically included if required.

Broadcast Address for Messages

A special message broadcast address is also supported by the network. The message broadcast address allows a common message to be sent devices connected to the RS232 COMMS interfaces of all remote loggers in a network. This is despite the fact that the individual loggers in the network are set to different address.

The broadcast or wild card address for a message is prefixed to the message in the general format

@@ message

where

@@	is the message broadcast address identifier
message	is the broadcast message

Specifying the broadcast address for a message is illustrated in the following example

```
@@ HELLO THERE^M^J
```

In this case the text string "HELLO THERE CR/LF" is sent from the host computer via the local logger and network to the devices connected to the RS232 COMMS interfaces of all remote loggers.

Control characters are included in the message in the format ^M (CR), ^J (LF), ^[(ESC), etc. Note that CR/LF is not automatically appended to the message, and must be specifically included if required.

Path for Returned Data

Any data resulting from commands that were sent to a networked logger is returned to the local logger via its network interface. The local logger then transmits this data from its RS232 COMMS interface to the host computer (unless the data is redirected to an alternative address – see below).

Note : The local logger in the network must also have a unique address. Although addressing of the commands is not necessary when commanding the local logger, its address is used to indicate to remote

loggers where data resulting from remotely directed commands is to be returned for output to the host computer.

Redirecting Data to an Alternative Address

By default all data from a networked *dataTaker* is returned to the host computer via the RS232 COMM interface of the logger which issued the command(s) to the networked logger.

However data can be redirected to another logger in the network, and transmitted from the RS232 COMM interface of that logger. This redirection is controlled by the setting of Parameter21.

This has use where a networked *dataTaker* is programmed via one *dataTaker* in the network, and returns its data to a display terminal or printer connected to the RS232 COMM interface of another logger in the network. Setting Parameter 21 to the address of the second logger will redirect the data to that logger.

Specifying the address for data to be returned to

```
#4 BEGIN
#4  RA10M
#4    1V 2R 3F
#4 END
#4 LOGON

#4 P21=3
```

Assuming logger #4 received the command via logger #1, then in this example all data produced by the program will be directed to logger #3 rather than returned to logger #1.

Setting Parameter21 to its default (P21=128) will restore the returning of data to the logger which transmitted the command on the network.

Addressing Returned Data

Data that is returned to the host computer from the local logger, and from the remote loggers, is not normally prefixed by the address of the sending logger.

However the source of the data returned to the host computer from a network can be identified by two methods

enabling the Logger Address Switch (/L) (See Section III – Switch Commands)

using the Text Channel $ (See Section III – Text Channel).

The return of the *dataTaker* address is determined by the Logger Address Switch as follows

/L	Enable return of logger address
/I	Disable return of logger address (Default)

The Logger Address Switch defaults to disabled (/I) when the *dataTaker* is initially powered up, hardware reset or executes a RESET command.

The effect of enabling the Logger Address Switch (/L) is illustrated in the following example

```
#4 /L
#4 BEGIN
#4  RA15S
#4   1..5TK
#4   1..3DS
#4 END
```

In this case the logger number will be prefixed to the data returned to the host.

```
Logger 4
1TK  105.88 Deg C
2TK  107.76 Deg C
3TK  100.34 Deg C
etc.
```

Use of the Text Channel $ is to identify the source of data is illustrated in the following example

```
#4 $="Logger 4 - Boiler House"
#4 BEGIN
#4  RA15S
#4   1..5TK
#4   1..3DS
#4 END
```

In this case the Text Channel contains identification of a particular logger, and is returned as part of the data from that logger as follows

```
Logger 4 - Boiler House
1TK  105.88 Deg C
2TK  107.76 Deg C
3TK  100.34 Deg C
etc.
```

Networking and Low Power Mode

When the *dataTaker* data logger enters the low power mode (asleep), it will not wake when network activity begins. To ensure proper network operation the loggers must be kept awake by either externally powering the loggers, or setting Parameter15=2 to prevent the logger entering the low power mode (See Section III – Parameter Commands).

The Wake terminals of all loggers in a network can be linked together, such that when one logger wakes then all loggers wake.

Programming a *dataTaker* over the Network

The major difference between operation of a single *dataTaker* and a network of *dataTaker*s is that management of the returned data and alarms becomes more involved.

The best method for managing the network will depend on the goals

data logging

monitoring for alarm conditions

rapid real-time response

simplicity of programming

telemetry or modem connection

Don't underestimate the complexity of managing large volumes of data. Unloading data over a network link from a *dataTaker* which has a large memory card can take over an hour.

There are two broadly different methods for network management, part-time and real-time networks.

Part-Time Network

If the main task is data logging, and communications between the network and the host computer can be infrequent (hours, days or weeks), then it is reasonable to program and unload each logger in the

network individually. This is the same as using a single logger, except that each logger must be addressed.

Real-Time Network

Where the emphasis is for rapid response or for tracking conditions, network speed becomes vital. There are many ways the network can be managed. However as a general rule more speed leads to greater complexity in host software:

1. Polling channels one at a time. This method is simple but is slow. Polling over a network can take approximately 2 seconds before requested data is received.

However since only one operation is undertaken at a time, there can be no confusion about the source of the data.

For example when polling a channel (assuming /n/u)

```
#3  2V
```

Will return data as follows

```
156.54
```

The polling speed can be as low as one channel every two seconds.

2. Polling data back from installed alarms one at a time using the ?n command (See Section III – Alarms). This is similar to the first method, but will be faster because the logger returns the last reading of the test channel rather than initiating a new sampling.

3. Polling data back from groups of channels using the Schedule Triggered by Poll (RX)

To poll groups of channels, the appropriate Schedule is first entered into each logger in the network as illustrated below

```
#2  RX 2V 3..4TT(FF1) /u/n/L

#3  RX  1..4DS  /u/n/L
```

The networked *dataTaker*s are then polled by the Poll Trigger command as follows

```
#2  X

#3  X
```

and will return data similar to

For Logger #2

```
2:156.54 23.5 28.9
```

For Logger #3

```
3:0 1 1 0
```

This method retains most of the simplicity of method 1, but is faster by scanning up to 20 channels every two seconds.

4. Poll all alarms on a logger by the ?ALL command (See Section III – Alarms). This is similar to the previous method but is faster at approximately 30 channels every two seconds.

5. Data can be synchronously returned from loggers in a network which have been programmed by Triggered Schedules RA, RB, RC and RD (See Section III – Triggered Schedules).

This is the most flexible method as it allows each logger full control of running schedules, but to work it requires that the host software use's more complicated data routing and time stamping techniques.

6. The use of the Fixed Format data mode (/H) is recommended for real-time networks.

Network Error Messages

Various warnings and error messages are returned to indicate that errors in network communications have occurred.

If the local *dataTaker* is unable to send any commands or messages to remote *dataTaker*s because of network failure, then one or more network error messages (E38, E43, E44, E45) are returned to the host by the local logger. These error messages are detailed in Section III – Warnings and Error Messages.

However if a remote *dataTaker* is unable to return data to the host computer because of network failure, then the error cannot be reported since the link has failed. In this case the remote *dataTaker* records the occurrence of the error, which can be returned either by the Parameter9 command or as System Variable 6 (6SV). This can later be checked when the link is repaired.

For example the network errors logged by logger 15 can be read by the command

```
#15 P9

or

#15 6SV
```

to determine if any errors have occurred.

After reading the network error variable, the variable can be cleared by the command

```
#15 P9=0
```

Network errors can be logged with other data by including the system variable with other channels to be scanned in schedule lists. For example the command

```
#7 BEGIN
#7  RA10M
#7   1..5TT  10V  6SV
#7 END
#7 LOGON
```

will log the number of network errors with the channel data.

Alarms can also be raised as a function of network errors, to provide active indications that the network is failing or has failed.

For example the command

```
ALARM5(6SV>0)1WARN
```

will warn of network failures on a display panel.

Writing *dataTaker* Programs

The *dataTaker 50*, *dataTaker 500* and *dataTaker 600* series data loggers have an extensive and powerful command set which allows you to easily develop programs for both simple and sophisticated tasks.

This command set, combined with the range of different measurement capabilities of the *dataTaker,* allows the logger to be used in a wide variety of applications.

Programming style only applies if you are programming your *dataTaker* from DeTransfer, or other terminal emulation or communications software. Some third party data collection software packages may require that you develop the *dataTaker* program separately as a text file, which the application simply transmits to the logger.

If you are using DeLogger, then you can skip this chapter.

Style of Programming

The success or otherwise of your *dataTaker* application generally reduces to the way in which you have written your program.

There are only a few simple rules for program structure, and there is considerable flexibility to allow individuals to develop their own styles.

After writing many different programs for the *dataTaker* this author has adopted a general style which makes programs easier to initially develop and later to interpret. While the following is not intended to dissuade others from developing their own styles, it may help in the development of those styles.

Planning the Program

Before writing a program, you should first plan your program by working through the following steps

make sure you fully understand what the *dataTaker* is required to do, what sort of data you require, how you want it presented, etc. A flowchart will be helpful later.

make sure that you are fully aware of the various sensor(s) you will be using including
- whether these fulfil your needs
- how the sensors are connected to the *dataTaker*
- which if any channel options are required to support the sensors
- whether linearization is required, and if so in what form are the calibrations provided, etc
- preferably make test readings from each sensor so that later you can be confident that the sensors are not the cause of unexpected results.

if you intend to use calculations, also decide at this time how many channel variables you will require for each Schedule (don't be afraid to over-estimate), and allocate channel variables in blocks of say 10 consecutive variables to each Schedule (ie. 1..10CV for Schedule A, 11..20CV for Schedule B, etc.). This will allow you to easily recognise later which variables belong to which schedules.

if you intend to use alarms, also decide at this time what the setpoints will be and what actions you require in response to alarm states.

Writing the Program

The overwhelming temptation is to write a *dataTaker* program from the 'top down', however the best approach is to write programs from the core outwards as follows

1. Firstly enter a BEGIN and an END on successive lines. These will eventually bracket the Schedules, input channel declarations, calculations and alarms that will make up the core of the program.

2. Between BEGIN and END insert the schedule(s) required, in terms of the trigger interval and method of triggering, etc.

If there is more than one Schedule then insert these on separate lines and indented two spaces. If necessary add comments to the right of the Schedules to describe their purpose.

3. Then add the channels to be scanned to each schedule, including any channel options necessary to instruct the *dataTaker* how the signal is to be measured, and how the data is to be used, etc.

If there is more than one channel in each schedule then insert these on separate lines below the schedule and indented four spaces. If necessary add comments to the right of the channels.

4. Then add any calculations that you require to the Schedule(s). Generally program flow is such that you will collect the channel data first, and then process the data with calculations. Therefore calculations will usually follow the channels in each Schedule.

5. Then insert any alarms that are required to monitor input channels for out of range conditions, and define the Alarms Schedule if continuous monitoring is not required. Generally these commands can be inserted following the last Data Schedule, however this is not mandatory.

6. This completes the core of the program, and all that remains is to include initialization commands required, global data formatting required, and the data logging command if required.

7. Immediately above the BEGIN statement, insert any initialization of channel variables required.

8. Immediately above these commands, insert the sensor calibration polynomial and span declarations if any are required.

9. Above these declarations insert Switch and Parameter commands required to determine the global data format.

10. At the top of the program insert the RESET command, followed by a delay period. DeTransfer will pause for the delay period when sending the program to the logger, allowing time for the *dataTaker* to execute the RESET command. A delay period of 5 seconds is usually sufficient time for a reset to execute.

Using a RESET command at the top of the program is strongly recommended, because the logger will then have a predictable configuration for receiving the new program.

11. Above the RESET command insert a delay period of one second, which will ensure that the logger wakes (if asleep), and is ready to receive the following program.

12. Following the END command, insert the data logging commands if required.

13. Finally insert any global or schedule Halt and Go commands as required.

14. Add comments throughout the program to document your intentions. Remember to make your comments 'meaningful to an audience, not just to yourself', because either you or others may have to interpret your program months or years later.

15. Keep a folder of all documentation, sensors, calibrations, wiring configurations, etc. for the project / program.

The program skeleton illustrated overleaf embodies the general program outline described above

```
Wait for dataTaker to wake

RESET the dataTaker
Wait for the dataTaker to RESET

Data format definition

Sensor calibrations

Channel variable initialisations

BEGIN
  First scan schedule
    Channels
    Calculations

  Second scan schedule
    Channels
    Calculations

  Alarm
  Alarm
  Alarm
END
Data logging commands

Go and Halt commands
```

BEGIN ... END

In practice most multiple schedule programs will extend over several command lines. If Schedules are entered on successive command lines, then each overwrites or replaces the last.

A multiple line command block or program format can be used to enter multiple Schedules, which also improves readability of the application program.

All of the commands required to be entered which are distributed over more than one line must be placed between the keywords BEGIN and END. The rules of the BEGIN … END construct are listed below

each line can be up to 250 characters long, and must be terminated by a carriage return

blank lines can be used to improve layout

commands which are executed on entry can be included

Schedule Triggers can be defined on lines without channel lists

channels can be placed on successive lines without a Schedule header, and are included in the previous Schedule.

Channels on separate lines are placed into the last defined Schedule. If there is no Schedule defined then the channel is scanned immediately in an Immediate Schedule.

when a BEGIN command is received, all of the Schedules including the Alarm Schedule RZ are Halted, and any existing RA, RB, RC, RD , RS and RX schedules are deleted.

The exception is when data logging is enabled, the Schedules are fixed (/F), or there is data logged into memory. In these cases the BEGIN command is rejected, and an error message is returned.

when the END command is received , the original Halt - Go state of each Schedule is restored.

channels cannot be appended to schedules after the END command is issued. The full set of schedules must be re-entered.

data format commands, system commands, etc. which are not a part of Schedules can also be placed between BEGIN … END, although this is not mandatory

The action of the BEGIN ... END construct is illustrated below

```
BEGIN

  RA10S
   1TK("Oven Temp",FF1)
   2TK("Flue Temp",FF1)
   3..6TK("Product Temps",FF0)

  RB1S
   1C(S1,"Gas Flow")
   2C(S2,"Electricity")

END
```

where

on receipt of the BEGIN command all Schedules are Halted and all existing schedules deleted

the Schedule A comprises the six thermocouple channels

the Schedule B comprises the counter channels

on receipt of the END command the original settings for Halt - Go for each schedule are restored

note that tabs, spaces and blank lines can be used to improve layout and readability

DeTransfer provides for sending of single line commands (Send Line) and multiple line commands (Send All). Refer to the DeTransfer help for these details.

Commands Executed on Entry

With the exception of the Triggered Schedules alarms, all other commands are executed on entry. This is irrespective of whether these commands are entered outside of BEGIN . . . END, or between BEGIN . . . END .

Commands which are executed on entry include Switch commands, Parameter commands, Polynomial and Span declarations, and all system commands.

The Immediate Schedules are also executed on entry, which is a very useful programming tool (see below).

Commands that are executed on entry are only executed once, and never execute again for the life of the program.

Use of Immediate Scan Schedules

The Immediate Schedules are executed on entry, irrespective of whether the commands are entered outside of BEGIN . . . END or between BEGIN . . . END. The channels and expressions in the Immediate Schedule are only executed once, and only at the time of entry.

Care must be used when placing Immediate Schedules between BEGIN . . . END to ensure that they don't inadvertently become associated with Triggered Schedules.

For example

```
BEGIN
  1V 2V
  RA5S 5TT
END
```

in this case the channels 1V and 2V will be executed on entry as an Immediate Schedule

```
BEGIN
  RA5S 5TT
  1V 2V
END
```

in this case the channels 1V and 2V will be attached to the Schedule A.

Initialization of Channel Variables are in fact Immediate Schedules. These may be placed above the BEGIN Command (preferred), or after the BEGIN Command but before the first Triggered Schedule,

For example in the program

```
1..5CV(W)=0.0
BEGIN

END
```

the Channel Variable assignments are interpreted as being within an Immediate Schedule, and are initiated at the time of entry.

When Immediate Schedules are used as actions within Alarms, then the schedule is executed whenever the alarm becomes true. For example

```
ALARM3(5TT>150)"[21CV(W)=4.75]"
```

the assignment to the Channel Variable on each true alarm state is executed as an Immediate Schedule. However each invocation of the assignment is a newly entered Immediate Schedule, and not a re-running of an existing Immediate Schedule.

This is important to understand since there may be other Immediate Schedules executing from elsewhere in the program between successive invocations of the action command for this alarm.

Association of Channels to Schedules

Channels can be placed on successive lines between BEGIN . . . END without a Schedule definition on the same line. Each channel declaration is processed as follows

the channels are placed into the last defined Triggered Schedule

if there is no Triggered Schedule defined before this point in the program, the channel is placed into an Immediate Schedule and is executed immediately

Documenting Programs

Comments can be used in programs to describe the application, the intention of particular steps, etc. The rules for the use of comments in *dataTaker* programs are as follows

a comment is prefixed by an apostrophe (') character, and all text up to the next carriage return is treated as the comment

comments can be placed on separate lines

comments can be placed after commands on the same line

comments cannot be placed before commands on the same line

in DeTransfer, comments can also be preceded by \', which signifies not to transmit the comment

Examples of the use of comments in a program is illustrated below

```
'Boiler monitoring program for the dataTaker
'Author: Henry Higgins
'Created                  23/4/2003
'Last Modified     25/4/2003

CSCANS
CALARMS
CDATA

'Switch units text and channel numbers off
'Switch return of Time and Date on
     /u/n/T/D

'Define spans for Gas & Electricity
S1=0,50,0,100"M^3"
S2=0,20,0,1000"KW"

BEGIN              'Begin program
'Repeat Schedule A records temperatures
     RA10S
            4TT("Oven Temp",FF1)
            5TK("Flue Temp",FF1)
            3..6TK("Product Temps",FF0)

'Repeat Schedule B records Gas & Electricity
     RB1S
            1C(S1,"Gas Flow")
            2C(S2,"Electricity")

END                        'End of program

LOGON       'Log all channel to memory
```

A Program Example

The following program illustrates a *dataTaker* program in which the programming styles discussed above have been used.

```
'WIND_SET.DXC
'Program for monitoring a wind set and calculating the
'    - mean wind speed
'    - mean wind direction
'    - standard deviation of wind speed
'    - standard deviation of wind direction

'The anemometer has output of 0-1000 mV for 0-50 m/s windspeed
'The wind vane has output of 0-1000 mV for 0-360 deg direction

'Reset the dataTaker
RESET
\w5

'Synch the dataTaker clock to host computer clock
D=\d
T=\t

'Enable date and time stamps
/D /T
```

```
'Define the wind sensor calibrations (Polynomials)

Y1=0,0.050"m/s"       'Wind speed, 0-1000mV = 0-50 m/s
Y2=0,0.360"Deg"       'Wind direction  0-1000mV = 0-360 deg

'Define units for calcs (Polynomials with unity scaling)
Y3=0,1"m/s"           'Units text for wind speed
Y4=0,1"Deg"           'Units text for wind direction

'Clear accumulators used to sum intermediate data
10..16CV(W)=0

'Pause while the dataTaker completes initialisations
\w1

BEGIN
 'Schedule reads wind set and sums data every 5 seconds
 RA5S
  1V(Y1,=1CV,W)       'Scan wind speed and save
  2V(Y2,=2CV,W)       'Scan wind direction and save
  2CV(W)=2CV/57.29    'Convert wind direction to radians for
                      'dataTaker trig functions

  10CV(W)=10CV+COS(2CV)          'Sum of COS(wd)
  11CV(W)=11CV+SIN(2CV)          'Sum of SIN(wd)
  12CV(W)=12CV+1CV               'Sum of ws
  13CV(W)=13CV+(1CV*1CV)         'Sum of ws * ws
  14CV(W)=14CV+(1CV*COS(2CV))    'Sum of ws * COS(wd)
  15CV(W)=15CV+(1CV*SIN(2CV))    'Sum of ws * SIN(wd)
  16CV(W)=16CV+1.0               'Number of scans

 'Schedule to calc wind data every minute
 RB1M
  20CV(W)=14CV/16CV              'Mean ws * COS(wd)
  21CV(W)=15CV/16CV              'Mean ws * SIN(wd)

  'Calculate mean wind speed
  22CV(W)=SQRT((20CV*20CV)+(21CV*21CV))
  22CV("Mean Wind Speed",Y3,FF2)

'Calculate mean wind direction
  23CV(W)=ATAN(21CV/20CV)*57.29

  'Find quadrant for mean wind direction
  23CV(W)=23CV+((20CV>0)AND(21CV<0))*360     '4th quadrant
  23CV(W)=23CV+((20CV<0)AND(21CV<0))*180     '3rd quadrant
  23CV(W)=23CV+((20CV<0)AND(21CV>0))*180     '2nd quadrant
  23CV(W)=23CV+((20CV>0)AND(21CV>0))*0       '1st quadrant

  23CV(W)=23CV-(12CV=0)*(23CV+1)    'No wind speed, return -1.0

  23CV("Mean Wind Dirn",Y4,FF2)

  'Calculate standard deviation of wind speed
  24CV(W)=13CV-((12CV*12CV)/16CV)
  24CV(W)=SQRT(24CV/16CV)

  24CV("SD Wind Speed",Y3,FF2)
```

```
  'Calculate standard deviation of wind direction
  25CV(W)=10CV/16CV
  26CV(W)=11CV/16CV
  27CV(W)=SQRT((25CV*25CV)+(26CV*26CV))
  27CV(W)=SQRT(-2.0*LN(27CV))*57.29

  27CV("SD Wind Dirn",Y4,FF2)

  'Clear accumulators
  10..16CV(W)=0

END

LOGON

G
```

Keypad and Display Module

The Display Module is fitted to the *dataTaker* in 2 different ways depending on the *dataTaker* model as follows

The *dataTaker* models *600, 605* and *615* loggers have the display integrated into the body of the instrument

The *dataTaker* models *50, 500, 505* and *515* loggers can have an optional Panel Mount Display mounted externally in enclosure doors or walls, adjacent panels, etc. (See Section II – Installing the Panel Mount Display)

The *dataTaker* Keypad and Display Module has a 2 line by 16 character Liquid Crystal Display (LCD) which displays current channel data, alarm states and various system information.

The *dataTaker* Keypad and Display Module also has five keys for scrolling information in the display, controlling the backlight, and editing the scroll lists. The module keys can also be defined as function keys, for executing predefined *dataTaker* commands by key presses.

The Keypad and Display Module features include

indication of channel data and results of calculations in numerical or bar graph format

indication of alarm channel data in numerical or bar graph format

indication of various system information

3 LEDs for local indication of alarm states

a buzzer for local indication of alarm states

LCD backlight control

5 keys for scrolling information in the display and running function keys

Displaying Channel Data

The latest data from input channels and calculations in Data Schedules is automatically displayed on the LCD display as the Schedules run.

The top line of the display indicates the channel identification, the bottom line of the display indicates the most recent reading as a numeric value or bar graph.

If a unique name is given to the channel or calculation as a channel option, then this is displayed as the channel identification (See Section III – Channel Options). Otherwise the default channel identification of the channel number and channel type is displayed.

The default channel identification is displayed with the data as follows

```
Channel 1TK
159.3 Deg C
```

If a unique name is given to the channel as a channel option by the command for example

```
RA1S  1TK("Boiler Temp")
```

then this unique channel identification is displayed with the data as follows

```
Boiler Temp
159.3 Deg C
```

Following the entry of a Data Schedule, and before the first execution of the Schedule occurs, the display shows that data is pending as follows

```
Boiler Temp
------
```

When a Data Schedule with more than one channel or calculation is entered, or more than one Data Schedule is entered, the first channel or calculation of the first Schedule is displayed initially. The other channels and calculations can be displayed by scrolling with the scroll keys (see below).

The data display is updated each time the Schedule to which the channel belongs is executed. Therefore at any time the data that is displayed is the data for the most recent execution of the schedule.

Caution : Depending on the trigger interval for Schedules, the data displayed may be minutes, hours or even days old.

Controlling the Display of Individual Channel Data and Calculations

By default all channels and calculations in Data Schedules are enabled for display. The display of data for any channel or calculation can be disabled in a number of ways as described below.

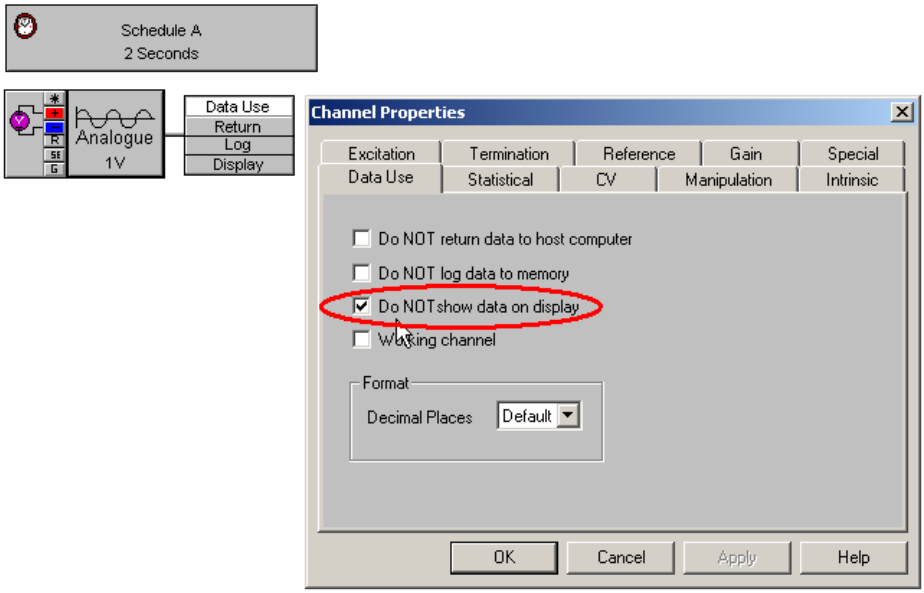
The display of channel data can be disabled by including the No Display (ND) channel option in the declaration of the channel (See Section II – Channel Options).

Using DeTransfer, display of individual channel and calculations data is disabled by the command for example

```
BEGIN
 RA2S
  1V(ND)   2..5PT385(4W,ND)   6TK
END
```

where only data for the thermocouple on channel 6 will be displayed. However the data for all channels will be returned to the host computer in real time.

Using DeLogger, the display of individual channel and calculations data is disabled in the Program Builder. Right click on the Data Use icon for the channel to disable, and select Properties… to open the Channel Properties dialog. Click on the Data Use tab, and check Do NOT show data on display.



The display of channel and calculations data can also be disabled by including the intermediate channel or working (W) channel option. The intermediate or working channel option also disables logging of the channel data, and returning of the channel data to the host.

The Intermediate Data Switch can over-ride the W channel option by /W, and allows channels declared with the W channel option to be displayed and returned to the host, but not logged. This allows display of intermediate or working data for debugging purposes. The default setting for the Intermediate Data Switch is /w.

The display of channel data can also be disabled by the List Edit keys (see below). Display of channel data disabled by the List Edit keys can later be re-enabled again by the List Edit keys. The List Edit keys have a toggling action for display of data for each channel and calculation. Channels and calculations defined with the ND or W channel option cannot be enabled for display with the List Edit keys.

Minimum and Maximum Data

The display update strategy for minimum and maximum data from the Statistical Sub-Schedule can be defined in two ways

Update the displays of minimum and maximum data whenever the associated Triggered Schedule (reporting) executes. This provides periodic update, and is the default.

Update the displays of minimum and maximum data when the Statistical Sub-Schedule executes. This provides progressive minimum and maximum information.

The update strategy is controlled by the Min Max Display Switch as follows

/x	Update the minimum and maximum displays when Triggered Schedule runs (Default)
/X	Update the minimum and maximum displays when Statistical Sub-Schedule runs

When the *dataTaker* is initially powered up, or executes a RESET Command, the Min Max Display Switch defaults to /x.

Displaying Alarms

Alarm test data and alarm state are by default not enabled for display. However alarm test data and alarm state can be enabled for display by the List Edit keys (see below).

The bottom line of the alarm display always indicates the most recent data for the alarm, as a numeric value or bar graph.

The information displayed on the top line of the alarm display depends on the way in which the alarm has been defined.

1. If the simplest form of Alarm is used, for example

```
ALARM1(1TT>60)1DSO
```

then the OFF state of the alarm is displayed as

```
Alarm 1 OFF
57.3 Deg C
```

and the ON state of the alarm is displayed as

```
Alarm 1 ON
62.3 Deg C
```

2. If the Alarm is defined with a text message, for example

```
ALARM1(1TT>60)"High Temperature"
```

then the OFF state of the alarm is displayed as

```
Alarm 1 OFF
57.3 Deg C
```

and the ON state of the alarm is displayed as

```
High Temperature
62.3 Deg C
```

3. If the Alarm is defined with a text message, and the test channel is defined with a unique name, for example

```
ALARM1(1TT("Low Temperature")>60)"High Temperature"
```

then the OFF state of the alarm is displayed as

```
Low Temperature
57.3 Deg C
```

and the ON state of the alarm is displayed as

```
High Temperature
62.3 Deg C
```

Specifying the ND channel option for the test channel can be used to permanently disable alarm channels from display (See Section II – Channel Options).

Bar Graph Display

The bottom line of the display can be used to display data as a Bar Graph. The bar graph is specified as a Channel Option, and over-rides the standard numerical data display.

The Bar Graph produces a bar scale of the channel or alarm data over a defined range, which is often easier to visually interpret than numeric data.

The general format of the Bar Graph Channel Option is as follows

channel(BGx:y,"Text")

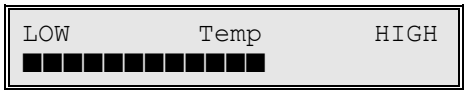
where

channel	is the input channel or calculation
BG	specifies the Bar Graph channel option
x	is the minimum value for display
y	is the maximum value for display
Text	is a unique name for the channel

Using DeTransfer, the Bar Graph display is commanded for example

```
BEGIN
 RA1S
  1TK(BG0:50,"LOW    Temp     HIGH")
END
```

will display data in the range of 0 to 50 Deg C.



The Bar Graph channel option can also be used for alarm test channels to display current alarm status in a more visual way as follows

```
ALARM3(1V(BG0:50,"Empty      Full")<5)"Empty  CAUTION  Full"
```

will display the alarm OFF state as



and will display the alarm ON state as



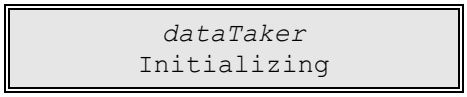
The Bar Graph Channel Option is not a useful format for returning data to the host. Therefore these channels should be disabled from returning data by the No Return (NR) channel option.

The Bar Graph Channel Option is not supported by DeLogger.

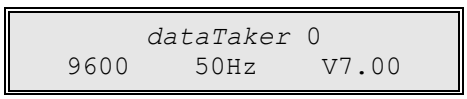
Status Screens

There are 8 status screens available, which display various system information. These can be viewed by the pressing the scroll keys .

Whenever the *dataTaker* is first powered up, or executes a RESET command, the display shows the initialization message



When initialization is complete, the display will change to



When a Schedule is entered and begins to execute, the first displayable channel or calculation in the schedule will replace the first status screen.

There are two levels of display control as follows

- Level 1 - Program control

W	Intermediate or Working channel option
ND	No Display channel option
P19	Parameter19, status screen control (see below)

These controls determine whether a channel, calculation, alarm or status screen is available for display. By default all channels and calculations included in Schedules, all current alarms and all status screens are available for display.

The screen item is then displayed only if enabled at the second level.

- Level 2 - Key control

List	Hold down the key for edit list mode
Scroll	Scroll up or down to select a screen
Light	Enable or Disable a display item (toggle action)

These controls allow an operator to enable or disable available screens for display. By default the status screens and data channels defined in Schedules are enabled for display, and the alarm screens are not enabled. Alarm screens must be manually enabled for display.

Controlling the Status Screens

There are 8 status screens that display various system information. By default all of the status screens are available for display and are enabled. Each status screen can be disabled using the List Edit keys (see below).

Selected or all status screens can also be disabled for display by defining Parameter19, which provides access to an 8 bit map that controls display of the status screens. Each bit controls a different status screen, defined as follows

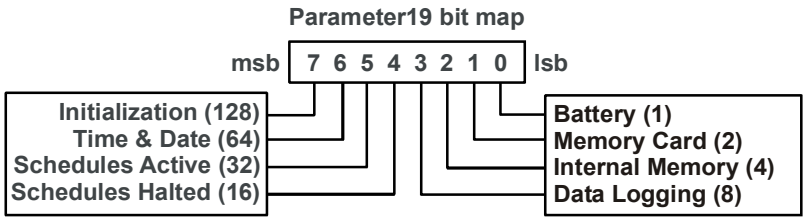


Figure 131 – Parameter19 Bit Map for Display of *dataTaker* Status

Parameter19 defaults to 255 (11111111), that enables all status screens for display. If Parameter19 is set to 0 (00000000), then no status screens will be available for display. To enable selected status screens, set Parameter19 to the sum of the number defined in the diagram above for the required screens.

For example, to make only the Time/Date and Battery screens available for display, Parameter19 is set to P19=65, where 65 (01000001) is the sum of 64 and 1, and only the Time/Date and Battery status screens will be available for display.

Any status screens disabled by this method cannot be made available for display by the List Edit keys.

Panel Keys

There are 5 keys on the Keypad and Display Module, which provide for scrolling of the status, data and alarm screens, and control of the display backlight. The layout is displayed below

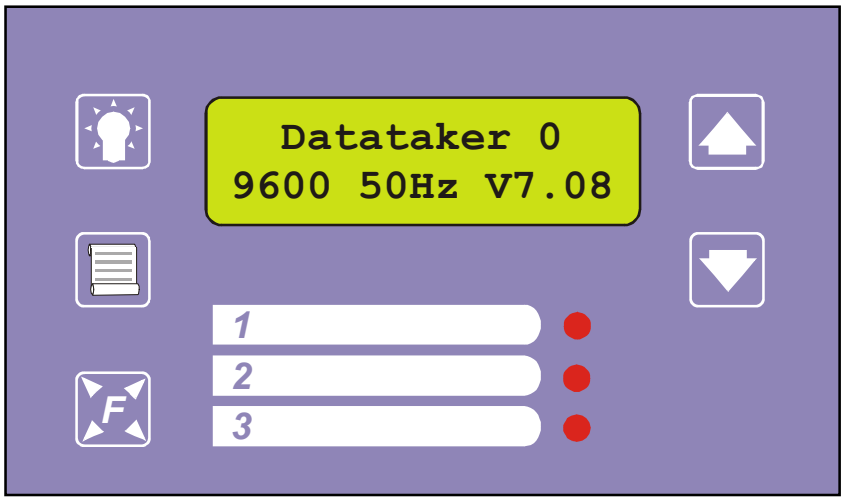


Figure 132 - Keypad and Display Module Keys



Scroll Keys

The Scroll Keys allow scrolling through the available screens, to locate a screen for viewing. Holding a Scroll Key down for more than three seconds starts auto-scrolling in the direction of the Scroll Key being held. The scrolling speed is set by Parameter18 in seconds (See Section III - Parameter Commands). By default P18=2 seconds. Pressing either Scroll Key again will stop the auto-scroll.



List Key

When the List Key is held down, the display is held in List Edit mode. In this mode access is possible to all of the available screen items by using the scroll keys. While the List Key is held down, pressing the Light Key will toggle the currently displayed screen between enabled and disabled (flashing).

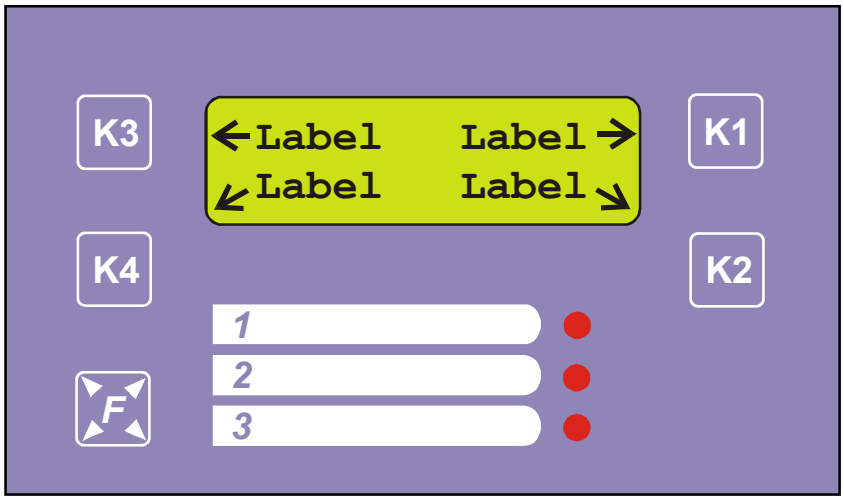


Light Key

The light key toggles the display backlight on and off. When the *dataTaker* enters the low power mode (sleep) the backlight is automatically switched off, and does not switch on again when the *dataTaker* next returns to normal power mode. When the List Key is held down, pressing the Light Key will toggle the currently displayed screen between enabled (stable) and disabled (flashing).

Function Shift Key

When the Function Shift Key is pressed, the remaining four keys become user definable function keys K1 to K4, that are located as follows



Each function key has an associated 8 character label which is displayed adjacent to the key. These labels are user definable as described below.

Function keys K1 and K2 have a default definition, which is initialized at power up, or following a RESET command, as follows

```
                    LED on →
                    LEDoff ↘
```

Function keys K3 and K4 have no default definition.

When a function key K1 to K4 is pressed while the Function Shift Key is held down, the function for the function key is executed and the display identifies the key and its label as follows

```
        Function 3
    ---> Halt <---
```

When the function has executed, the display returns to the previously displayed screen.

Defining Function Keys

Function keys can be assigned any *dataTaker* command or group of commands. The general command for function key assignment is as follows

Kn="Label",[commands]

where

K	is the Function Key specifier
n	is the function key number from 1 to 4
Label	is the key label up to 6 characters for display
command	is any set of space separated *dataTaker* commands, up to 255 characters

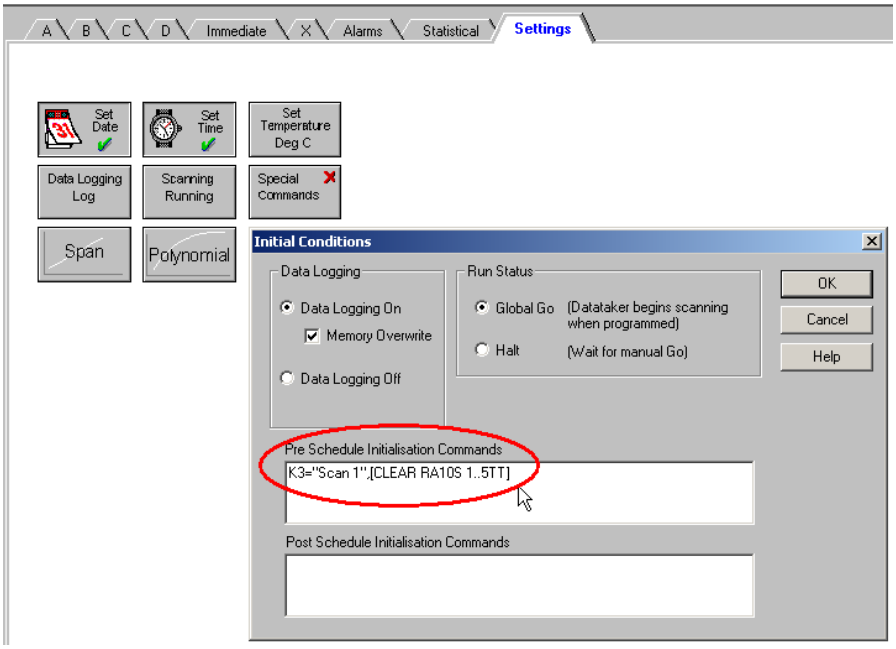
Using DeTransfer, the command for example

```
K3="Scan 1",[CLEAR RA10S 1..5TT]
```

defines Function Key F2. The command sequence is executed when the F2 key is pressed while the Function Shift Key is held down, displaying

```
Function 3
---> Scan 1 <---
```

Using DeLogger, Function Key assignments are entered in the Program Builder Settings tab.



In the Settings tab, right click Special Commands and select Properties... Enter the Function Key assignments into the Pre or Post Schedule Initialization Commands.

The default definitions for function keys K1 and K2 are

K1="LED on",[1..3WARN=1]

Kn="LEDoff",[1..3WARN=0]

Display Screens and Scroll Order

The various status screens, data screens and alarm screens for the *dataTaker* Keypad and Display Module are illustrated overleaf.

This illustration also details the movement of the display between the available screens when the Scroll Keys are pressed.

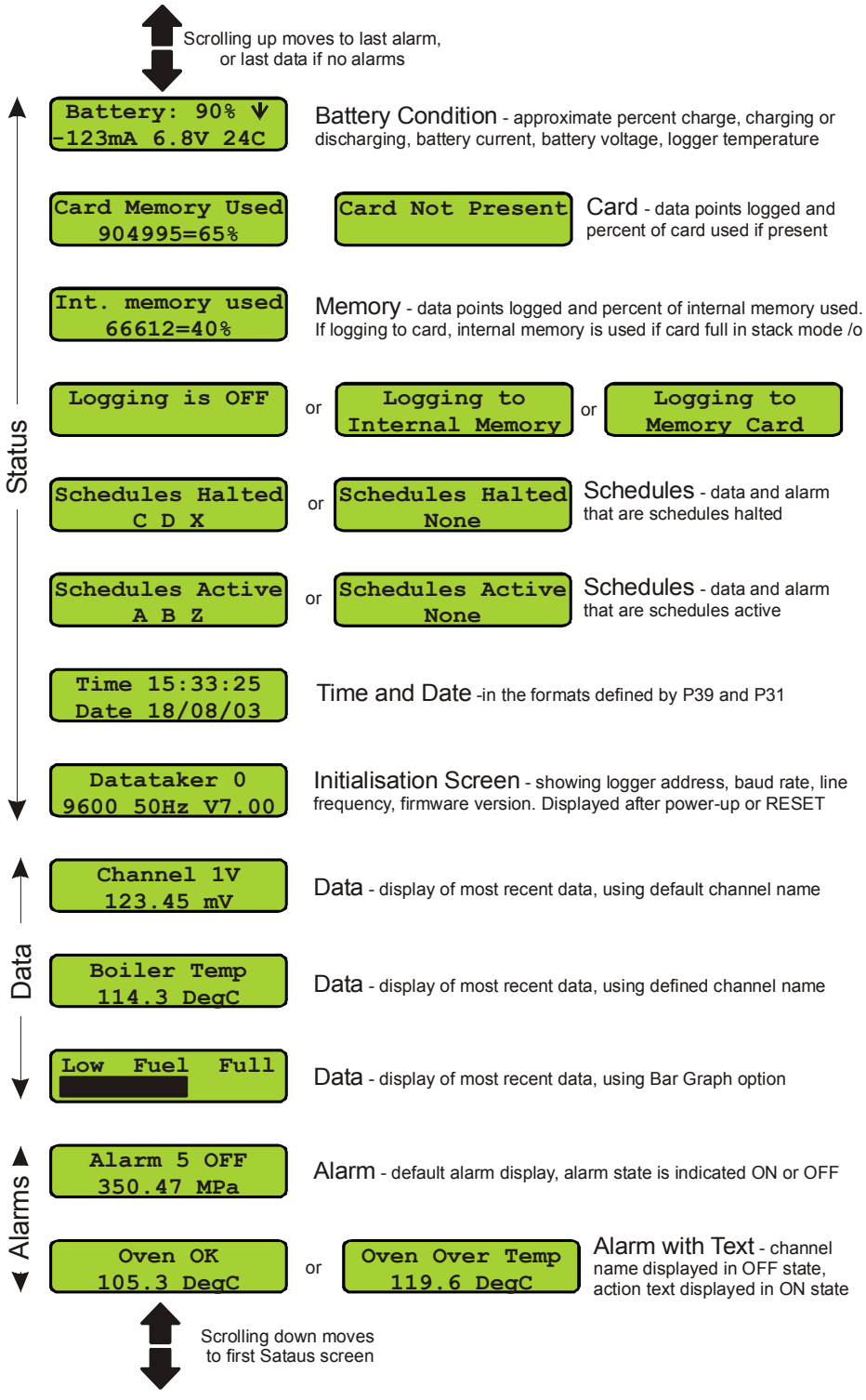


Figure 133 - Scrolling Order of the Display Screens

Warning Outputs

The Keypad and Display Module has 6 warning output channels, which can be used to provide local warning of alarm states.

The warning output channels are as follows

1WARN	Display panel LED 1
2WARN	Display panel LED 2
3WARN	Display panel LED 3
4WARN	Beeper
5WARN	Flash display backlight at approx 1Hz
6WARN	Turn display backlight on or off

The warning outputs are used by assigning a state to them as follows

```
1WARN=1
3WARN-0
4WARN=1
```

which turns LED 1 on. turns LED 3 off, and turns the beeper on respectively.

The Warning outputs are particularly useful when used in Alarms (See Section III – Alarms).

Using DeTransfer, the Alarm command for example

```
ALARM1(3SV("Card OK")<1000)"Change CARD[1WARN=1]"
```

will illuminate Warning LED 1 when the memory card is nearly full.

The Warning LEDs are supported in the alarms dialog of DeLogger, and can be used to indicate alarm states.

Warnings and Error Messages

Various warnings and error messages are returned by the *dataTaker* which report command and runtime errors, and warnings of potential errors.

The return of messages is controlled by the Messages Switch Command as follows

/M	Enable the return of error and warning messages (Default)
/m	Disable the return of error and warning messages

The default setting is /M, and is automatically selected when the *dataTaker* is powered up, is hardware reset or executes a RESET command.

During an Unload operation (See Section III – Returning Data From Memory), error reporting is disabled until the Unload is completed. In a network, errors in remote *dataTaker*s are reported back to the host computer.

If you are using DeTransfer, the format of the warnings and error messages returned to the host is determined by the setting of the Units Text Switch as follows

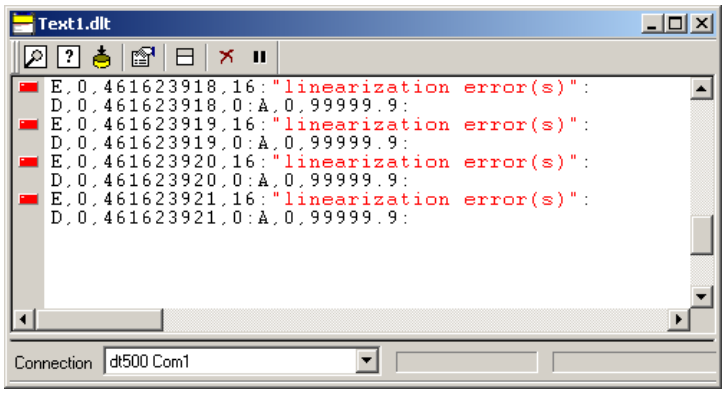
if the Units Text Switch is enabled (/U) the error number and a description is returned as follows

```
E12-channel list error
```

if the Units Text Switch is disabled (/u) the error number only is returned as follows

```
E12
```

If you are using DeLogger, then you do not need to worry about the format of error messages. DeLogger displayes most error messages from the logger in the Text View as shown below



and will log error messages into an error log file id enabled (see your DeLogger Users Manual).

The list the *dataTaker* error messages is shown overleaf.

These error messages are detailed in the *DT50, DT500* and *DT600* Series Users Manual supplied with you data logger, and includes an explanation of the possible causes for each error message.

E1–time set error
E2–input buffer full
E3–channel option error
E4–clear data memory
E5–data memory full
E6–data memory empty
E7–day set error
E8–parameter read/set error
E9–switch error
E10–command error
E11–input(s) out of range
E12–channel list error
E13–digital failure
E14–communications error
E15–assignment error
E16–linearization error
E17–clear card data
E18–STATUS command error
E19–card write error
E20–illegal character(s)
E21–illegal separator(s)
E22–statistical option error
E23–scan schedule error
E24–unload command error
E25–channel table full
E26–halt command error
E27–TEST command error
E28–go command error
E29–poly/span declaration error
E30–calibration failure nn
E31–test channel failure nn
E32–battery sample failure
E33–CSIO bus failure
E34–function key command error
E35–card faulty
E36–CLAST not valid
E38–address error
E40–no data found
E41–program area full
E42–no card inserted
E43–RS485 chip failure
E44–network transmission error
E45–remote logger not found or network error
E46–ROM checksum error
E47–user string error
E48–channel list fixed
E51–ALARM command error
E52–alarm text memory full
E53 -no statistical samples
E54-expression error
E55-expression memory full
E60-RAM *n* failure
E61-memory card failure

Appendices

Appendix A - ASCII Character Table

ASCII Character	Dec	Hex	Oct	Binary
NUL	0	00	000	00000000
SOH (CTRL A)	1	01	001	00000001
STX (CTRL B)	2	02	002	00000010
ETX (CTRL C)	3	03	003	00000011
EOT (CTRL D)	4	04	004	00000100
ENQ (CTRL E)	5	05	005	00000101
ACK (CTRL F)	6	06	006	00000110
BEL (CTRL G)	7	07	007	00000111
BS (CTRL H)	8	08	010	00001000
HT (CTRL I)	9	09	011	00001001
LF (CTRL J)	10	0A	012	00001010
VT (CTRL K)	11	0B	013	00001011
FF (CTRL L)	12	0C	014	00001100
CR (CTRL M)	13	0D	015	00001101
SO (CTRL N)	14	0E	016	00001110
SI (CTRL O)	15	0F	017	00001111
DLE (CTRL P)	16	10	020	00010000
DC1 (CTRL Q) XON	17	11	021	00010001
DC2 (CTRL R)	18	12	022	00010010
DC3 (CTRL S) XOFF	19	13	023	00010011
DC4 (CTRL T)	20	14	024	00010100
NAK (CTRL U)	21	15	025	00010101
SYN (CTRL V)	22	16	026	00010110
ETB (CTRL W)	23	17	027	00010111
CAN (CTRL X)	24	18	030	00011000
EM (CTRL Y)	25	19	031	00011001
SUB (CTRL Z)	26	1A	032	00011010
ESC (CTRL [])	27	1B	033	00011011
FS (CTRL |)	28	1C	034	00011100
GS (CTRL])	29	1D	035	00011101
RS (CTRL ^)	30	1E	036	00011110
US (CTRL _)	31	1F	037	00011111
SPACE	32	20	040	00100000
!	33	21	041	00100001
"	34	22	042	00100010
#	35	23	043	00100011
$	36	24	044	00100100
%	37	25	045	00100101
&	38	26	046	00100110
'	39	27	047	00100111
(	40	28	050	00101000
)	41	29	051	00101001
*	42	2A	052	00101010
+	43	2B	053	00101011
,	44	2C	054	00101100
-	45	2D	055	00101101
.	46	2E	056	00101110
/	47	2F	057	00101111

ASCII Character	Dec	Hex	Oct	Binary
0	48	30	060	00110000
1	49	31	061	00110001
2	50	32	062	00110010
3	51	33	063	00110011
4	52	34	064	00110100
5	53	35	065	00110101
6	54	36	066	00110110
7	55	37	067	00110111
8	56	38	070	00111000
9	57	39	071	00111001
:	58	3A	072	00111010
;	59	3B	073	00111011
<	60	3C	074	00111100
=	61	3D	075	00111101
>	62	3E	076	00111110
?	63	3F	077	00111111
@	64	40	100	01000000
A	65	41	101	01000001
B	66	42	102	01000010
C	67	43	103	01000011
D	68	44	104	01000100
E	69	45	105	01000101
F	70	46	106	01000110
G	71	47	107	01000111
H	72	48	110	01001000
I	73	49	111	01001001
J	74	4A	112	01001010
K	75	4B	113	01001011
L	76	4C	114	01001100
M	77	4D	115	01001101
N	78	4E	116	01001110
O	79	4F	117	01001111
P	80	50	120	01010000
Q	81	51	121	01010001
R	82	52	122	01010010
S	83	53	123	01010011
T	84	54	124	01010100
U	85	55	125	01010101
V	86	56	126	01010110
W	87	57	127	01010111
X	88	58	130	01011000
Y	89	59	131	01011001
Z	90	5A	132	01011010
[	91	5B	133	01011011
\	92	5C	134	01011100
]	93	5D	135	01011101
^	94	5E	136	01011110
_	95	5F	137	01011111
'	96	60	140	01100000

ASCII Character	Dec	Hex	Oct	Binary
a	97	61	141	01100001
b	98	62	142	01100010
c	99	63	143	01100011
d	100	64	144	01100100
e	101	65	145	01100101
f	102	66	146	01100110
g	103	67	147	01100111
h	104	68	150	01101000
i	105	69	151	01101001
j	106	6A	152	01101010
k	107	6B	153	01101011
l	108	6C	154	01101100
m	109	6D	155	01101101
n	110	6E	156	01101110
o	111	6F	157	01101111
p	112	70	160	01110000
q	113	71	161	01110001
r	114	72	162	01110010
s	115	73	163	01110011
t	116	74	164	01110100
u	117	75	165	01110101
v	118	76	166	01110110
w	119	77	167	01110111
x	120	78	170	01111000
y	121	79	171	01111001
z	122	7A	172	01111010
{	123	7B	173	01111011
|	124	7C	174	01111100
}	125	7D	175	01111101
~	126	7E	176	01111110
DEL	127	7F	177	01111111

Appendix B - ADC Sampling Speed

The conversion rate or sampling speed for the *dataTaker* analog to digital converter is dependent on a number of factors. These include

the time required to determine which Schedules and channels are to be scanned

the time required to test whether an autocalibration is required now, and if so the time required to perform an autocalibration

the time required to select each channel for conversion

the settling time after selecting a new channel, during which the sensor signal is allowed to stabilize

the time required for the actual analog to digital conversion of the input signal

the time required to change gain (range) if necessary, and to repeat the analog to digital conversion

the time required to convert the primary data from the analog to digital converter into electrical units

the time required to linearise data in electrical units to data in physical units using internal functions, or user defined polynomials or spans

The timing diagram for analog to digital conversion by the *dataTaker* is illustrated overleaf. The time components of the analog to digital conversion process are described below.

Note : The ADC timing given below assumes that return of data to the host in real time is disabled (/r). Any RS232 COMMS or NETWORK communications will increase these times, depending on baud rate.

Schedule Execution Setup Time

The schedule execution setup time is the period required to determine which schedules if any are due for execution, and to carry out autocalibration of the analog to digital converter if necessary.

If there are no schedules due for execution, then other functions such as communications, digital inputs, counters, etc. are checked for need for attention before the schedules are again checked. The process of checking schedules for eligibility for execution takes 5 mS.

When a schedule becomes due for execution, and auto-calibration is enabled (/K), then the zero voltage reference is checked for the potential need for an auto-calibration. This takes 35 mS.

If autocalibration is required, then this is carried out before the schedule executes and any input channels are sampled. Autocalibration takes a further 550 mS to complete.

Therefore the schedule execution setup time can vary from 5 mS if auto-calibration is disabled (/k), to 40 mS if autocalibration is enabled but not required, to 590 mS if an autocalibration is carried out.

Following the schedule execution setup time, the sampling of input channels in the schedule list begins. During sampling of the input channels, a number of time consuming processes occur.

Channel Selection Time

This is the time taken for the analog input multiplexer to select an analog input channel for sampling. The channel selection time is approximately 10 mS, and is fixed.

Channel Settling Time

The channel settling time is the time allowed between selecting a particular analog channel on the input multiplexer, and commencing analog to digital conversion. This period allows the output of the sensor to settle after beginning to output bias current to the logger following selection.

The channel settling time is 10 mS by default. However the channel settling time can be defined within the range of 0 to 30000 mS using the Parameter10 command, to match the output characteristics of the

sensors being monitored. The conversion rate or sampling speed of the *dataTaker* can be modified by altering the channel settling time (see below).

Analog to Digital Conversion Time

During the analog to digital conversion time, the input signal on the selected channel is converted to its basic digital equivalent. The conversion time is fixed at one mains or line cycle, which is 20 mS for a 50 Hz line frequency and 16.67 mS for a 60 Hz line frequency. The analog to digital conversion process is linked to the mains or line period to maximize line hum rejection.

The local mains or line frequency is primarily defined by the setting of bit 1 of the DIP switch, which allows selection for 50 or 60 Hz. Alternatively the line frequency can be set within the range of 48 to 1000 Hz using the Parameter11 command. The conversion rate or sampling speed of the *dataTaker* can be modified by altering the line frequency setting (see below).

Calculation and Linearization Time

Following analog to digital conversion of an input signal, the raw data is firstly calculated into the fundamental electrical units of mV, mA, Ohms or Hz, then linearized to physical units of temperature, strain, loop percentage or user defined polynomial or span as required.

The calculation and linearization of raw data for each channel is performed while the next channel in the schedule list is being converted (see ADC Timing Diagram). The calculation and linearization time for each channel does not influence the overall sampling speed, unless the conversion time is reduced to less than the calculation and linearization time (see below).

Signal or Sensor Type	Calculation and Linearization Time (mS)
Voltage	4.0
Current	5.0
Current Loop	6.0
Resistance – 2 & 3 wire	6.0
Resistance – 4 wire	7.0
Frequency	6.0
Period	5.0
LM35	4.5
LM335/AD590	5.0
LM34	6.0
RTD - 4 wire	9.0
Thermocouple – zero voltage reference	3.0
Thermocouple – junction temperature	4.5
Thermocouple – first in schedule	13.0
Thermocouple – subsequent in schedule	9.0
Thermistor	9.0
Digital state and counters	2.0
Calculation	2 - 10
Polynomial	Add 2 – 8
Span	Add 2 – 3
Averaging	Add 1 – 2
Minimum and Maximum	Add 1 – 2
Integration	Add 1 – 2

The calculation and linearization time varies considerably, from 4 mS for a voltage signal to 25 mS for a thermocouple with a user defined polynomial attached. Approximate calculation and linearization times for the different analog input channel types are listed in the table below.

Calculation and linearization times for calculations, polynomials and spans depends on the complexity of the expression.

Default ADC Sampling Speed

The analog to digital converter sampling speed is determined by the combination of channel selection, settling, conversion, calculation and linearization times.

At 50 Hz line frequency, the default channel selection, settling and conversion times total 45 mS. This combined time translates to a sampling speed of

23 samples/sec for 20 channels in a Schedule

20 samples/sec for 1 channel in a Schedule

At 60 Hz line frequency, the default channel selection, settling and conversion times total 37 mS. This combined time translates to a sampling speed of

25 samples/sec for 20 channels in a Schedule

22 samples/sec for 1 channel in a Schedule

Increasing ADC Sampling Speed

The conversion rate or sampling speed of the *dataTaker* can be increased to approximately 3 fold the default sampling speed.

Disable Autocalibration

Disabling the Autocalibration Switch (/k) will reduce the schedule execution setup time, because the logger no longer checks if a calibration is required and does not autocalibrate. Setting the calibration interval Parameter0 to a large value will prevent autocalibration from occurring, however the checks will still be done.

This time saving is minor, however if an actual autocalibration occurs during rapid data collection, then there will be a 500-600mS gap in the data set when the autocalibration occurred.

Increasing Mains or Line Frequency Setting

The *dataTaker* does not actually take its timing from the local mains or line frequency. The logger actually simulates the local mains or line frequency, which it then uses to time analog to digital conversion such that it is synchrony with the local frequency. This reduces mains or line hum induced errors.

Increasing the mains or line frequency setting of the logger decreases the simulated mains or line period. Since analog to digital conversion occurs over one mains or line period, this then decreases the analog to digital conversion time or increases the conversion rate. Note however that such actions will reduce hum rejection by the analog to digital converter, and will introduce some errors if the area is electrically noisy.

Using DeTransfer, the line frequency can be increased to 100 Hz (decreases ADC period to 10mS) by the command

```
P11=100
```

The total channel selection-settling-conversion time is reduced to 10 + 10 + 10 = 30 mS. This combined time translates to an increased sampling speed of a little more than 33 samples/sec (other factors not being limiting).

The mains or line frequency can also be set by assigning the period to System Variable 8 as follows

```
8SV=100
```

If the analog to digital conversion time is decreased to less than the linearization time, then the sampling speed will be determined by the settling and linearization times rather than by the settling and analog to digital conversion times. Under these circumstances the selection, settling and conversion of the next channel will not commence until the linearization of the previous channel has finished.

Decreasing the Channel Settling Time

The channel settling time is the time allowed between selecting a particular analog channel on the input multiplexer, and commencing analog to digital conversion. This period allows the output of the sensor to settle after beginning to output bias current to the logger following selection.

The channel settling time is 10 mS by default. However the channel settling time can be defined within the range of 0 to 30000 mS using the Parameter10 command, to match the output characteristics of the sensors being monitored.

Using DeTransfer, the channel settling time can be decreased to 1mS by the command

```
P10=1
```

The total channel selection-settling-conversion time is reduced to 10 + 1 + 20 = 31 mS. This combined time translates to an sampling speed of a little more than 32 samples/sec at normal mains frequency (other factors not being limiting).

However, combine this with increased mains or line frequency (see above), the total channel selection-settling-conversion time is reduced to 10 + 1 + 10 = 21 mS which translates to an sampling speed of a little more than 47 samples/sec.

The channel settling time can also be set by assigning the period to System Variable 7 as follows

```
7SV=1
```

Again, if the linearization time is the dominant factor, then reducing channel settling time will have no real effect.

Autoranging

When Schedule are entered into the *dataTaker*, all analog channels are set to an initial gain or range which is optimal for the particular signal type.

When each channel is subsequently sampled, it is autoranged to obtain a reading at the highest possible resolution. This range is then retained and becomes the initial range when the channel is next sampled.

This autoranging however reduces the apparent sampling rate. Input signals that are out of the currently selected range for that channel have to be re-converted at different ranges to obtain an acceptable reading. This may involve two or three successive conversions of the channel before suitable data is obtained, or the channel is deemed to be 'out of range'.

For example the second analog channel in the ADC Timing Diagram below is out of range on the first conversion, requiring a second conversion after a gain change. In this case the analog to digital converter took 70 mS to read this channel, rather than 40 mS if the first reading had been in range.

Therefore the sampling rate of the analog to digital converter will be slower when signals are continuously varying between ranges, than if signals stay in the same range.

Digital Channels

Digital input and counter channels in schedule lists are skipped during the channel selection phase. These channels are directly read and linearized following calculation and linearization of the previous analog channel, while the next analog channel is converting (see fourth and fifth channels in the ADC Timing Diagram below).

Digital and counter channels are read in less than 2 mS.

Data Destination

Following execution of a Schedule, resulting data is then logged into the data storage memory if data logging is enabled (LOGON), and returned to the host computer if data return is enabled (/R).

The time taken to log the data into the data storage memory is 2 mS plus 0.5 mS per data point.

The time taken to return the data to the host computer depends on the number of characters to be transmitted for each data item, and the setting for the RS232 COMMS baud rate.

However, transmitting of data to the host computer will generally slow the sampling rate to less than the default sampling rate. The increased sampling rates discussed above will only be achieved if returns to the host computer in real time are disabled (/r) and all data is logged during the data collection session.

Maximum Sampling Rate

The maximum sampling rate possible with the *dataTaker* 50, 500,600 series data loggers is a little over 70 samples per second.

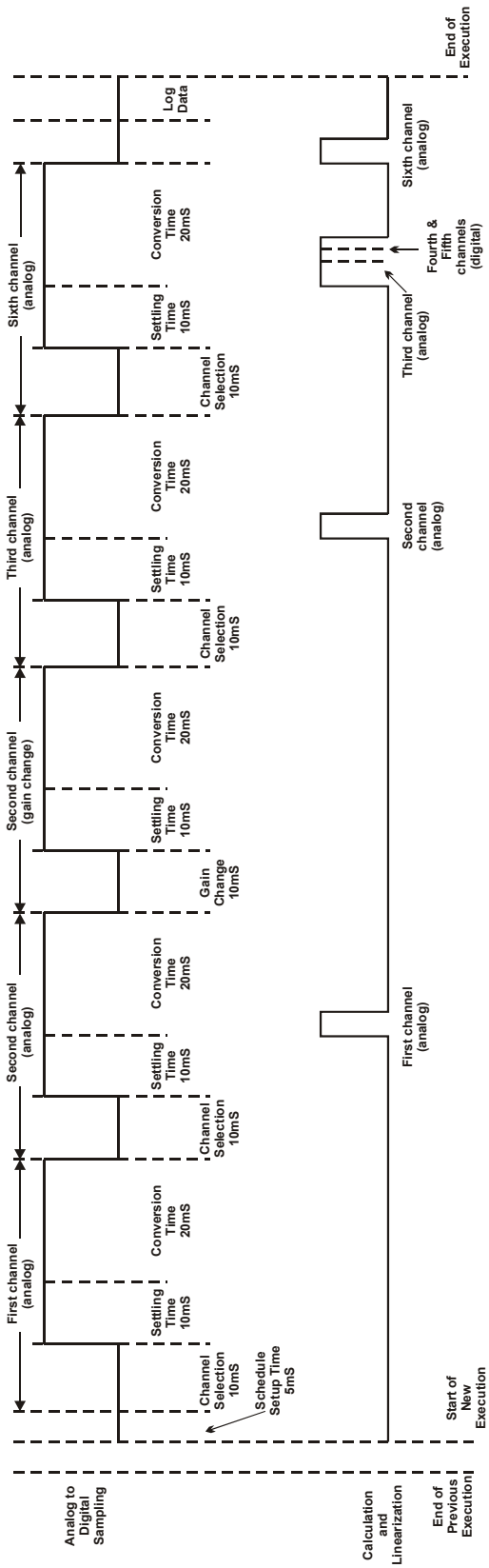
This is achieved by setting the mains or line frequency to the upper limit of P11=1000, setting channel settling time to the lower limit of P10=1, logging data in real time, disabling return of data to the host computer, and reading only simple voltage signals which have minimal calculation requirements.

Using DeTransfer, the maximum sampling rate can be increased by the commands

```
/r
/k
P0=100
P10=1
P11=1000
LOGON
```

Using DeLogger, these commands can be entered in the Program Builder Settings tab. In the Settings tab, right click Special Commands and select Properties... Enter the commands into the Pre or Post Schedule Initialization Commands.

ADC Timing Diagram



ADC Timing Diagram for the Schedule

1V 2V 3V 1DS 2DS 4V

The times assume a mains frequency 50Hz (P11=50) and a channel settling time of 10 mS (P10=10)

Index

1ST 371, 372, 373, 374
1WARN362, 363, 377, 571, 590
2.5 Volt Common 384
2.500 mA Current Output386
250.00 µA Current Output 386
2ST 371, 372, 373, 374, 465, 489
2WARN 362, 454, 590
3ST 371, 373
3WARN362, 363, 588, 590
4 Wire Measurement 383
4-20 mA Current Loops120, 122
4ST 371
4WARN 362, 363, 590
5WARN 362, 590
6WARN 362, 590
Accumulating Counters272, 278
AD590 Directly Powered and Measured Using External Shunts201
AD590 Directly Powered and Measured Using Internal Shunts199
AD590 Externally Powered and Measured Using External Shunts 203
ADC Gain..... 384
ADC Sample Period and Line Frequency 72
ADC Sampling Speed597
ADC Settling Period.... 74
ADC Timing Diagram598, 600, 602
Address, Logger . 67, 565
Addressing Returned Data568
Alarm
  Keywords..... 439
  Message Substitution Characters 449
Alarm Actions..... 447
Alarm Actions,Combining455
Alarm Comparison Operators443
Alarm Delay Period... 446
Alarm Logical Operators455
Alarm Number..... 440
Alarm Schedule457, 458, 466, 487
Alarm Setpoints 444
Alarm Test Data..... 441
Alarms Checking..... 462
Alarms for Controlling464
Alarms from the Internal Channel Table,Clearing435, 463
Alkaline Backup Battery50
Alphabetical Listing of the Switches550
Alternative Commands for XON and XOFF..... 300
Analog
  Channels of the Channel Expansion Modules..... 326
  Channels of the dataTaker 50 326
  Channels of the dataTaker 500/600 Series.....326
  Input Multiplexer ...68
  Input Precautions..79
  Input Types329, 330, 332
  Processing68, 86, 552, 559, 560, 561
  State Inputs.....339
  State Thresholds.156
Analog Devices AD590/592 Sensor198
Analog Input Channels71, 85, 313, 326
Analog Input Signals,Connecting. 79
Analog Logic State 156
Analog State as Differential Inputs157
Analog State as Single Ended Inputs 159
Analog States as Differential Inputs158
Analog to Digital Conversion Time598
Analog to Digital Converter72, 384
Appendices.....593
Applications354, 361, 363, 374, 418
Applications for System Timers.. 374
Arm Resistance.....231
ASCII Character Table62, 594
Assigning
  Channel Variables to Channel Variables.....485
  Date and Time to Channel Variables 368
  Input Channel Data to Channel Variables.....482
  Results of Expressions to Channel Variables.....484
  Statistical Data to Channel Variables.....483
Association of Channels to Schedules 576
attenuated input.157, 159
Attenuation Network..110
Autocalibration by Drift in the Zero Voltage Reference76
Autocalibration of the ADC. 75
Automatic Date and Time Recording368
Automatic Scaling.....467
Autoranging.....70, 600
Autoranging by Instrumentation Amplifier.....70
Avoiding Excessive Common Mode Voltages.....81
Avoiding Ground Loops83
Backlight293, 362, 363, 437, 447, 453, 580, 586, 590
Back-Light 362, 590
Back-Light Flashing362, 590
Backspace [BS].....305
Bar Graph Display394, 441, 583, 584
Battery8, 12, 14, 19, 25, 27, 30, 31, 39, 46, 47, 49, 50, 51, 53, 55, 66, 293, 294, 308, 349, 387, 435, 441, 509, 511, 512, 513, 515, 519, 546, 547, 561, 592
Battery Life..... 51, 52
Battery Life Chart..... 51
Battery Status 546
Baud Rate8, 9, 14, 19, 28, 30, 58, 60, 62, 63, 64, 299, 304, 313, 318, 353, 354, 554, 597, 601
Baud Rate DIP Switches63
BEGIN ... END429, 430, 574, 575
Block Format BEGIN ... END (DeTransfer).... 429
Bridge Excitation Current230
Bridge Excitation Voltage240
Bridge Excitation Voltage Channel391
Bridge Excitation Voltage Source239
Bridge Outputs to Engineering Units,Converting248
Bridges and Strain Gauges196, 230
Broadcast Address for Commands566
Broadcast Address for Messages567
Buzzer..... 362, 363, 590
Cable Type..... 60
CALARMS397, 430, 435, 439, 460, 463, 533, 540, 577
Calculating Microstrain for Full Strain Gauge Bridges233, 244
Calculating Microstrain for Half Strain Gauge Bridges235, 247
Calculating Microstrain for Quarter Strain Gauge Bridges238
Calculation and Linearization Time598
Calculation Errors 496
Calculation of Thermocouple Temperature.... 172
Calculations410, 467, 480, 484, 490, 492, 574
Calculations,Combining495
Calibrating the External Attenuation Network..... 110
Calibration11, 68, 75, 76, 167, 171, 172, 185, 199, 207, 213, 248, 249, 250, 326, 330, 371, 378, 438, 471, 474, 475, 479, 542, 543, 552, 557, 558, 559, 561, 573, 592, 597, 599
Capacity of Memory Cards512
Capacity of the Internal Data Memory 509
Card data .. 515, 517, 592
Card program518, 529, 530, 531, 532, 534
CARDID 514
Carriage Return [CR] 306

Caution51, 55, 64, 271, 282, 283, 300, 301, 457, 532, 542, 581
Cautions When Commanding the dataTaker from an Alarm ..453
Cautions When Switching Outputs from an Alarm..... 454
CDATA430, 510, 514, 515, 516, 519, 528, 531, 532, 577
CDATA Command from the Program Area..... 531
Changing System Variables377
Channel
Factor or Scale Factor 388
identification Text 394
Identifier302, 315, 549
Management Options325
Module..... 322
Name..... 316
Number285, 290, 302, 315, 316, 322, 549, 552
Options for Channel Variables 485
Options Table380, 381, 383
Selection Time ... 597
Settling Time 597
Variables234, 236, 239, 244, 247, 254, 256, 368, 369, 388, 392, 437, 441, 444, 445, 451, 468, 469, 473, 474, 476, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 492, 497, 498, 501, 502, 503, 504, 552, 554, 576
Variables and Calculations 234, 236, 239, 244, 247, 254, 256, 369, 392, 480, 552, 554
Channel Expansion Module11, 23, 34, 80, 95, 102, 263, 267, 270, 274, 277, 280, 283, 286, 288, 289, 290, 291, 322, 326, 336, 344, 356
Channel Expansion Module Digital Output Channels283
Channel Expansion Module Number286
Channel Expansion Module,Construction23, 291
Channel Expansion Modules326
Channel Expansion Modules (CEM-AD) 7, 11, 76, 79, 111, 286
Channel Options Grouping383
Channel Variable393, 481, 483, 484
Calculations234, 236, 239, 244, 247, 254, 256, 369, 392, 480, 552, 554
Channel Variables
Assigning 485
Date and Time.... 489
Schedules..... 486
Setpoints 445, 488
Setpoints in Alarms488
Test Data in Alarms487
Transfer Statistical Data to Alarms 489
Checking Data Storage507, 508, 510, 513
Checking Data Storage in the Internal Memory.....507
Checking Data Storage in the Memory Card.....508
Checking Entered Schedules434
Checking the Alarms .462
Checking the Data Memory Configuration ...507
Checking the Declaration of Polynomials and Spans 478
Checking the Program Area532
CLAST.....592
CLEAR499, 500, 510, 515, 518, 519, 527, 528, 530, 531, 533, 566, 588
Clearing
Alarms from the Internal Channel Table.....435, 463
Data from Memory Cards 519
Data Store.....527
Internal Data Memory 510
Program Area 532
Schedules from the Internal Channel Table ..434
Combining
Alarm Actions.....455
Calculations 495
Input Modes 335
Schedules ...428, 432
Command for Scanning at the Maximum Rate 410
Command Line Editing305
Command Separators306
Command Styles.....37
Commanding the Channel Expansion Module.....288
Commands Broadcast Address for566
Commands Executed on Entry .. 575
Commencing Data Logging to Memory Cards 516
Comments429, 430, 573, 576, 577
Common Mode
Voltage Limits 104
Voltage Range.....96
Common Mode Voltage79, 88, 90, 96, 104, 126
Common Mode Voltages81
Common Mode Voltages,Identifying83
Communicating Messages to Other Devices on the Network567
Communicating with the dataTaker 9
Communication Cable for Apple Macintosh Computers61
Communication Echo 302
Communication Failure,Identifying Causes of 30
Communication Protocols299, 301, 552
Communication Protocols and Commands299, 552
Communications Baud Rate299
Communications Cable27, 60, 61, 82
Communications Cable for Windows Computers... 60, 61
Conditional Calculations492
Constant Current Excitation of Bridges 230
Consumption..... 7, 47
Controlling
Display of Individual Channel Data and Calculations581
Displaying of Data503
Intermediate Data504
Logging of Data .. 499
Return of Data 308
Return of Data to the Host Computer 501
Status Screens ... 585
Transfer of Data to Memory Cards 517, 530
Waking of the dataTaker 48
Converting Bridge Outputs to Engineering Units248
Converting Current Loop Data to Physical Units.. 136
Converting LED29, 31, 35, 49, 53, 512
Copper RTD Types ... 187
Copper RTDs 187, 261
COPY518, 530, 531, 532, 533
Counte
Events,Schedules Triggered by 265, 336, 348, 352, 355, 415, 416, 417
Counter
Channels273, 279, 336, 344
Event Inputs 355
Events Command,Schedules Triggered by 415
Modes..... 344
Types..... 345
Counting Digital Output Switching361
CPROG..... 514, 516, 532
CSCANS397, 430, 434, 435, 439, 464, 518, 530, 533, 540, 577
CTEST 514, 519
Current excitation196, 230, 232, 233, 234, 235, 236, 237, 239, 240, 248, 250, 330
Current input 117
Current Loop Data to Physical Units,Converting136
Current Loops Basic Concepts of124
Current Loops Using External Current Shunts 130
Current Loops Using the Internal Shunts 128
Current Using External Current Shunts 116

Current Using the Internal Current Shunts 114
Currents114, 126, 130, 201, 204
CV 393, 481, 483, 484
Data Bits, Stop Bits and Parity299
Data Block Format318, 319
Data Block Format for Real Time Data 319
Data Block Format for Stored Data319
Data Block Formats,Examples320
Data Delimiter Character317
Data Destination 601
Data from Memory Cards,Clearing519
Data Item Description315
Data Item Format310, 318, 319
Data Logging12, 430, 499, 500, 507, 510, 528, 551, 559
Data Logging to Memory Cards,Commencing 516
Data Manipulation..... 389
Data Memory Configuration507
Data Numeric Format310
Data Storage507, 508, 510, 513
Data Storage in the Internal Memory507
Data Storage in the Memory Card508
Data Store Overwrite Switch Disabled (/o)..... 505
Data Store Overwrite Switch Enabled (/O)..... 505
Data Store,Clearing .. 527
Data to the Host Computer,Controlling 501
Data Transfer to Memory Cards .511
dataTaker..... 7
dataTaker 50..... 326
dataTaker 50 Address 67
dataTaker 50, 500 and 60010, 53, 68, 80, 85, 102, 127, 149, 156, 157, 159
dataTaker 50,Construction14, 19
dataTaker 500/600 Series 326
dataTaker 500/600 Series,Construction 19
dataTaker 505, 515, 605 and 61511, 20, 80, 102, 123, 127, 151, 156, 290
dataTaker Address321, 565
dataTaker Commands298
dataTaker Configuration Code547
dataTaker Features 8
dataTaker Input and Output Channels 10
dataTaker Networks66, 67, 565
dataTaker Test Table545, 546
Date & Time of Minimum & Maximum 391, 420
Date and Time Formats314
Date and Time in Alarms369
Date and Time Recording Automatic368
Date and Time Setpoints445
Date and Time to Channel Variables Assigning 368
Date or Day and Time Labels316
Day 371
Day Formats.....364
Day or Date , Setting .365
Decimal Day.....364
Declaration of Polynomials and Spans 478
Declared Spans.....476
Declaring Polynomials471
Declaring Spans.....475
Decreasing the Channel Settling Time 600
Default ADC Sampling Speed599
Defining Function Keys587
delay period48, 64, 257, 356, 357, 358, 359, 360, 362, 446, 561, 573
Delay Period and Reset358
Delete [DEL].....306
DeLogger 4 and DeLogger Pro 4 . 43
Description of Parameter commands 559
Description of the Switch Commands 549
DeTransfer and DeLogger31, 41, 62, 63, 297, 299, 304, 308, 449, 452, 502, 527
Diagnosing Faults with the Speaker257
Difference ..381, 390, 443
Difference and Statistical Data in Alarms 443
Differential Analog Channels331
Differential Analog Input Mode83, 87, 331
Differential Analog Inputs327
Differential Analog State Inputs.. 340
Differential and Single Ended Analog Inputs.....327
Differential Input96, 105, 118, 131, 150, 158, 163, 164, 173, 203, 208, 210, 213, 214, 217
Differential Measurement95, 96, 104, 105, 117, 130, 253
Differential Measurement of Externally Attenuated Voltage104
Differential Measurement of Low Level Voltage 95
Differential Measurement of Vibrating Wire Strain Gauges 253
Digital Bit Inputs 268, 337
Digital Bit Inputs,Connecting267
Digital Bit Output Channels281
Digital Byte Inputs269, 337
Digital Byte Inputs,Connecting268
Digital Byte Mask.....338
Digital Byte Output Channels283
Digital Channels290, 558, 600
Digital Event Inputs ...343
Digital Events,Schedules Triggered by 265, 336, 343, 412, 413, 414
Digital Input Channels263, 336, 469, 560
Digital Input Channels,Connecting266
Digital Input Circuit.... 263
Digital Input Modes ... 265
Digital Input Sample Period264
Digital Output Channels280, 356
Digital Output Channels of the CEM-AD 284
Digital Output Types . 356
Digital State Inputs.... 336
Digital While Inputs ... 343
DIP switch9, 14, 19, 28, 32, 33, 53, 54, 58, 62, 63, 64, 66, 67, 72, 299, 314, 364, 560, 565, 598
Direct Measurement of High Level Voltages 111
Directing Commands and Programs to Loggers 565
Directing Commands to the Local dataTaker..... 565
Directing Commands to the Remote dataTakers 566
Directly Connected High Level Voltages 111
Directly Connecting High Level Voltages 111
Directly Connecting Thermocouples to the dataTaker.. 172
Disable Autocalibration599
Disable Low Power Mode47
Disabling of System Switches526
Disabling the Return of Alarm Messages 449
Display of Individual Channel Data and Calculations,Controlling581
Display Screens and Scroll Order588
Displaying Alarms450, 582
Displaying Channel Data580
Displaying of Data,Controlling503
Divide 381, 393, 490, 495
Document Details 2
Documenting Programs430, 576
Enabling Memory Card Programming 529
Enabling Memory Cards for Data Logging 515
Enclosures 15, 20, 23
Entered Schedules Checking434
Entering Alarms 439
Entering Commands and Programs304
Entering Commands from a Memory Card 305
Entering Commands from the Network - dataTaker 500/600 loggers305
Entering Commands in the Low Power Mode 30, 304

Entering New Schedules When Logging 506
Error Messages101, 113, 123, 137, 148, 155, 166, 184, 197, 219, 229, 251, 552
Establishing Communications28
event 11, 30, 39, 49, 126, 343, 355, 402, 412, 413, 414, 415, 416, 419, 421, 422, 428, 432, 458
Event Inputs..... 355
events7, 11, 12, 263, 265, 298, 336, 343, 348, 352, 355, 371, 374, 396, 402, 403, 412, 414, 415, 417, 423, 437, 438, 458, 518, 530
Example
Card Program..... 533
Examples41, 314, 320, 384, 385, 386, 387, 388, 390, 392, 393, 394, 420, 430, 577
Data Block Formats320
Programming the Statistical Functions..... 420
Excitation Current188, 222
Excitation Voltage Channel Bridge391
Excite Terminal Excitation and Differential Inputs164
Excite Terminal Excitation Single Ended Inputs 165
Excite Terminal Function385
Executing Commands from Memory Cards..... 530
Executing dataTaker Commands from an Alarm..... 451
Executing RESET Command from the Program Area.. 532
Execution of Calculations492
Expansion34, 35, 547, 548
Expansion Module Digital Output Channels..... 283
Expansion Module Number286
Expansion Modules (CEM-AD)7, 11, 76, 79, 111, 286
Exponential Data Format311
Expression evaluation480, 481
Expressions358, 396, 399, 400, 402, 405, 408, 410, 412, 415, 426, 480, 481, 482, 484, 490, 491, 492, 493, 554, 575
Expressions Operators490
Extending the Minimum Temperature 221
Extending Triggered Schedules .404
External Attenuation Network110
External Attenuation of High Level Voltages..... 102
External Common92, 142, 383
external excitation..... 292
External Excitation163, 164, 291
External Excitation and Differential Inputs 163
External Excitation and Single Ended Inputs.....164
External Isothermal Block178
External Power Supply 49
External References..390
External Voltage Attenuator Networks 103
Externally Referenced Single Ended Inputs to Reduce Cabling ... 93
Extra Samples...258, 387
Factor Channel330, 388, 467, 468
Firmware Versions 2
Fixed Exponential digits394
Fixed Field Width.....317
Fixed Floating Point ..393
Fixed Format Data Mode541
Fixed Format Mode for Data301
Fixed Mixed output....394
Flashing Back-Light362, 590
Floating Point Data Format310
Force Low Power Mode47
Forcing Execution of the Memory Card Program.....532
Forcing the Transfer of Data518, 530
Format of dataTaker Memory Cards511
Format of Displayed Statistical Data424
Format of Returned Data301, 308, 393, 424, 461, 469, 473, 476, 478, 522, 526, 549, 551, 552, 553, 561, 562, 563
Format of Returned Statistical Data424
Format of the Unloaded Data526
Four Wire Resistance Measurement140, 141, 142
Four Wire Resistance Measurement with External Common142
Four Wire RTD Measurement190
Four Wire Thermistor Measurement224
Free Format Mode for Data301
Free Format Mode for Returned Data 309
Frequency and Period149
Frequency as Differential Inputs 150
Frequency as Single Ended Inputs152
Frequency input.150, 154
frequency measurement76, 149, 150, 151, 152, 153, 154, 155, 257, 387
Frequency Measurement149
Frequency vs Amplitude149
Full Bridge with Constant Current Excitation.....232
Full Bridge with Current Excitation,Interpreting the Data from 233
Full Bridge with Voltage Excitation241, 242
Full Bridge with Voltage Excitation,Interpreting the Data from 243
Full Strain Gauge Bridges, Calculating Microstrain for233, 244
function keys580, 587, 588
Function of the Internal Data Memory with Memory Cards512
Function Shift Key587, 588
Functional Listing of Parameters558, 559
Functional Listing of the Switches551
FW 253, 255, 260, 330
G 37, 71, 167, 169, 170, 171, 184, 328, 330, 332, 375, 381, 384, 385, 403, 411, 414, 417, 425, 431, 448, 459, 471, 475, 514, 579, 594, 595
GA 403, 411, 414, 417, 431, 432, 453, 461, 464, 465, 533, 539
Gain Lock..... 71, 384
Garbage Characters ... 30
Gauge Resistance 257
Gauge Temperature . 259
GB 403, 411, 414, 417, 431, 465, 533, 539
GC 403, 411, 414, 417, 431
GD 403, 411, 414, 417, 431
General Format for the Triggered Schedules 402
General Rules for Entering Commands 305
General Schedule Commands402, 428, 553, 554
General STATUS..... 536
Geologger252, 253, 254, 255, 256, 257, 258, 259, 547, 548, 550, 551
Getting Started2, 7, 25, 42
Global Control,Logging Data499
Global Number of Significant Digits312
Gray Code Conversion342
Grey code conversion (F7)342, 343, 469
ground loops58, 59, 79, 81, 82, 83, 543
Ground Loops . 81, 82, 83
Ground Loops via the Communication Cable (Series 1 dataTakers)81
Ground Loops via the Sensor Cables 82
Ground Loops,Identifying83
Grounding of Thermocouples183
GS 425, 431, 533, 594
Guard Output..... 385
H 36, 97, 99, 101, 106, 109, 110, 116, 119, 120, 122, 129, 132, 134, 136, 142, 143, 145, 146, 147, 151, 153, 155, 159, 161, 163, 174, 176, 177, 191, 193, 195, 196, 201, 203, 204, 206, 209, 211, 212, 215, 216, 217, 218, 219, 225, 227, 229, 233, 235, 238, 243, 246, 254, 256, 268, 269, 272, 273, 276, 278, 279, 301, 302, 308, 312, 375, 403, 408, 411, 414, 417, 425, 431, 446, 459, 485,

526, 527, 533, 540, 541, 550, 551, 552, 571, 594, 595
HA 403, 411, 414, 417, 431, 453, 461, 464, 465, 533
Half Bridge with Constant Current Excitation 234
Half Bridge with Current Excitation,Interpreting the Data from 235
Half Bridge with Voltage Excitation245
Half Bridge with Voltage Excitation,Interpreting the Data from 246
Half Strain Gauge Bridges, Calculating Microstrain for235, 247
Halting and Resuming
Alarm Schedule .. 459
All Alarms 459
Individual Alarms 460
Schedules411, 414, 417, 431
Schedules Triggered by Counter Events 417
Schedules Triggered by Digital Events 414
Schedules Triggered by Time 411
Statistical Sub Schedule 425
Triggered Schedules403
Halting Scanning..... 36
HB 403, 411, 414, 417, 431, 432, 465
HC 403, 411, 414, 417, 431
HD 403, 411, 414, 417, 431
High Level Voltages95, 102, 127, 150, 152, 157, 159, 208, 240, 241, 245
High Resistance Nickel RTDs259
High Speed Counter 1 Input349, 353
High Speed Counter 1 Output352
High Speed Counter Events415
High Speed Counter Initial Count349
High Speed Counter Inputs,Connecting 278
High Speed Counter Range350, 351
High Speed Counter Range and Initial Count..... 351
High Speed Counters8, 277, 345, 349, 351, 415, 417, 562
Histogram381, 391, 497, 498
Host Computer Software10, 565
Hour 371, 373
HS 425, 431
identification41, 87, 290, 309, 322, 403, 424, 514, 515, 550, 551, 569, 580
Identification Text Channel394
Identifier Channel302, 315, 549
Identifying Causes of Communication Failure 30
Identifying Common Mode Voltages83
Identifying Ground Loops83
Identifying Triggered Schedules .403
If You are Using DeLogger to Program Your dataTaker 297
If You are Using DeTransfer to Program Your dataTaker 297
Immediate Schedule48, 344, 376, 396, 399, 400, 401, 403, 404, 418, 480, 481, 486, 492, 574, 575, 576
Immediate Schedule Command. 399
Immediate Schedules in Programs400
Immediate Trigger Command400
Including Time and Date with Unloaded Data 522
Increasing ADC Sampling Speed599
Increasing Mains or Line Frequency Setting 599
Increasing the ADC Sampling Rate73
Individual Control,Logging Data . 501
Individual STATUS 537
Individual TEST Data 545
Initialising the System Timers373
Initializing Memory Cards513
Initializing New Memory Cards... 513
Initializing Used Memory Cards . 514
Input Bias Current 71
Input Channel Data to Channel Variables Assigning 482
Input Channels 11
Input Modes,Combining335
Input Multiplexer 68
Input Precautions 79
Input Signal Configuration and Mode of Operation.....383
Input Terminated 383
input termination71, 87, 96, 98, 105, 109, 110
Input Types329, 330, 332
Input Unterminated....383
Inserting a Different Memory Card517
Installation of Current Loop Transmitters 126
Installing the Channel Expansion Module.....286
Installing the dataTaker25
Installing the Panel Mount Display293, 580
Installing Thermocouples173
Installing Voltage Attenuator Networks 104
Instrumentation Amplifier70
Integral 381, 420
Integrate 381, 390
Integration .391, 420, 598
Interfacing to the dataTaker 41
Intermediate Data,Controlling504
Intermediate or Working Channel394
Internal Analog Channels378
Internal Attenuation ...385
Internal Calibration and Checks ... 76
internal channels174, 175, 177, 290, 298, 371, 378, 379, 396, 476, 509
Internal Channels290, 371
Internal Current Shunts114, 126
Internal Data Memory 509
Internal Data Memory,Clearing...510
Internal Memory When a Memory Card is Inserted,Logging Data to the 516
Internal Memory,Logging Data509, 516
Internal or Backup Battery .. 50
Internal Power Supply Backup Battery 39
Internal Temperature .. 76
interval11, 12, 52, 264, 270, 274, 277, 344, 391, 396, 402, 408, 409, 410, 418, 419, 421, 423, 424, 425, 426, 432, 433, 435, 458, 460, 461, 487, 488, 489, 497, 498, 512, 553, 554, 557, 558, 559, 560, 572, 581, 599
Intrinsic Functions388, 389, 441, 467, 468, 469, 470, 483, 485
Isolating the RS232 COMMS Serial Interface - dataTaker Series 1 Only..... 59
isolation..... 59, 124, 125
isothermal block171, 176, 178, 179, 180, 181, 182, 442, 443
Isothermal Block,Connecting181
Keypad and Display Module580, 586, 588, 590
Labelling the End of Unloaded Data527
Lead Acid Backup Battery dataTaker 50 50
Lead Acid Backup Battery dataTaker 500/600 Series.. 50
LED 362, 363, 377, 454, 571, 588, 590
Light Key 586
Limit to the Number of Statistical Scans 423
Line Feed [LF]..... 306
Linearization of Thermocouple Inputs 172
Linearization Time 598
List Key 293, 586
Lithium System Battery39
LM335 Directly Powered208
LM335 Externally Powered210
LM34 and LM35 Directly Powered213
LM34 and LM35 Externally Powered217
Load Cell..... 250, 471
Locking the Gain..... 71
Logging Data - Global Control499
Logging Data - Individual Control501
Logging Data to Memory499, 509, 511, 515, 553
Logging Data to Memory Cards.509, 511, 515
Logging Data to the Internal Memory 509, 516

Logging Data to the Internal Memory When a Memory Card is Inserted 516
Logging of Data,Controlling499
Logical Operators437, 440, 455, 456, 457, 490
LOGOFF453, 465, 499, 500, 507, 510, 528, 538, 539
LOGON430, 433, 452, 453, 455, 464, 465, 496, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 510, 515, 518, 526, 528, 529, 530, 533, 538, 539, 540, 554, 563, 565, 566, 568, 571, 577, 579, 601
Long Cables..... 60
Low Level Voltages95, 104, 107
Low Power Delay.. 31, 48
Low Power Mode30, 46, 271, 277
low power operation412, 415, 419, 560
Low Resistance Platinum RTDs.260
Low Speed Counter Events415
Low Speed Counter Inputs,Connecting 271
Low Speed Counters8, 270, 275, 345, 347, 560
Mains/Line Frequency 54
Management Options Channel...325
mathematical functions474, 490
Maximum Counting Rate270
Maximum Number of Calculations492
Maximum Sampling Rate601
Measurement Delay.. 257
Measurement Ranges and Accuracy 101, 110, 122, 137, 148, 155, 196, 229, 251
Measurement Ranges and Accuracy for Attenuated Voltages 110
Measuring
- 4-20 mA Current Loops 120, 122
- Analog Logic State156
- Analog State as Differential Inputs 157
- Analog State as Single Ended Inputs 159
- Analog States as Differential Inputs 158
- Bridges and Strain Gauges 196, 230
- Current Loops Using External Current Shunts. 130
- Current Loops Using the Internal Shunts..... 128
- Current Using External Current Shunts..... 116
- Current Using the Internal Current Shunts..... 114
- Currents114, 126, 130, 201, 204
- Frequency and Period 149
- Frequency as Differential Inputs 150
- Frequency as Single Ended Inputs 152
- Gauge Resistance257
- Gauge Temperature259
- High Level Voltages95, 102, 127, 150, 152, 157, 159, 208, 240, 241, 245
- Low Level Voltages95, 104, 107
- Resistance138, 146, 185, 189, 220, 222, 257
- Resistance Using Single Ended Measurement....146
- RTDs Using Single Ended Inputs 195
- Temperature34, 77, 167, 169, 181, 185, 198, 220, 443, 563
- Temperature with Integrated Circuit Temperature Sensors 198
- Temperature with RTDs 185
- Temperature with Thermistors 220
- Temperature with Thermocouples 169, 443
- the Bridge Excitation Voltage 240
- Thermocouples as Differential Inputs.....173
- Thermocouples as Single Ended Inputs.....174
- Vibrating Wire Strain Gauges 252

Mechanical Details 294
Memory Card Identification514
Memory Card Test.....519
Memory Cards,Commencing Data Logging to.....516
Memory Cards,Format of dataTaker511
Memory Cards,Logging Data509, 511, 515
Memory, Logging Data499, 509, 511, 515, 553
Messages Broadcast Address for567
Minimum and Maximum Data391, 420, 582
Minimum Temperature Measured220
Minute371, 372, 373, 374, 465, 489
Mixed Data Format....312
modem connection....569
Modes of Data Logging504
Modes of Operation and Networking9
Module Channel 322
Module Number.....289
Monitoring Contact Closures by Analog State 163
Monitoring Digital State267
Multiple Scan Schedules428
Multiplexer14, 20, 53, 68, 85, 102, 546
Multiplexer Powering...53
Multiplexer Types 20
Multiply 381, 393, 490
Mutually Exclusive Options383
Name Channel..... 316
National Semiconductor LM335 Sensor 207
National Semiconductor LM34 and LM35 Sensors. 212
Network Address 66
Network and Modems. 66
Network Error Messages571
NETWORK Interface65, 565
networking... 65, 304, 560
Networking and Low Power Mode67, 569
Nickel RTD Types..... 186
No Display394, 451, 503, 581, 585
No Log 394, 501
No Message..... 30
No Output..... 386
No Reference Junction Temperature Compensation. 180
No Return346, 350, 394, 449, 502, 503, 584
NOCOPY .. 518, 519, 531
Number Channel285, 290, 302, 315, 316, 322, 549, 552
Number of Alarms438, 441
Number of Calibration Samples....75
Number of Channels for Alarm Commands..... 438
Number of Channels for Schedules396, 434
Number of Data Bits ... 62
Number of Single Ended Inputs..332
Number of Statistical Samples423
Number of Stop Bits.... 63
Numerical Listing of Parameters 557
Obtaining the Polynomial for a Sensor 474
Obtaining the Span for a Sensor 479
Open Collector Digital Outputs ...280
Open Collector Digital Outputs,Connecting 281
Open Installation of the dataTaker26
Operator Precedence 491
Options38, 54, 179, 183, 199, 213, 230, 313, 322, 323, 324, 325, 331, 359, 380, 382, 383, 385, 388, 389, 391, 393, 394, 419, 441, 443, 481, 520, 534
Options for
- ADC Gain 384
- Channel Variables392
- Data Manipulation389
- Excite Terminal Function 385
- External References390
- Input Signal Configuration and Mode of Operation383
- Output Data Format393
- Resetting to Zero 388
- Scaling of Channel Data 388

Statistical Evaluation391
Options for Channel Variables485
Options Table Channel380, 381, 383
Order of Application .. 383
Order of Data Returned by Triggered Schedules 404
Order of Data Returned from Multiple Scan Schedules431
Other Applications of Alarms464
output channels7, 8, 9, 10, 11, 12, 14, 19, 23, 41, 263, 280, 281, 282, 283, 284, 285, 286, 293, 298, 322, 324, 336, 356, 357, 358, 359, 360, 361, 362, 396, 405, 437, 447, 451, 453, 454, 455, 480, 490, 590
Output Data Format .. 393
Over Voltage Protection113, 290
Panel Keys..... 586
Parameter Commands47, 48, 55, 64, 73, 74, 75, 76, 78, 167, 264, 298, 377, 397, 438, 542, 549, 556, 569, 586
Parameter Commands and Switch Commands..... 298
Parameters 556
Parameters from DeTransfer563
Parity 9, 28, 58, 62, 63, 64, 299
Parity Checking..... 63
Part-Time Network.... 569
PASSWORD protection64, 302, 303, 557, 559, 560
Path for Returned Data567
Phase Encoder263, 270, 274, 275, 276, 345, 354, 355
Phase Encoder Counter274
Phase Encoder Counter Range..355
Phase Encoder Initial Count354
Phase Encoder,Connecting263, 270, 274, 275, 276, 345, 354
Phase Encoding Basic Concepts of274
Planning the Program572
Platinum RTD Types. 185
PMD to the dataTaker 50,Connecting 293, 294
PMD to the dataTaker 500 Series,Connecting294
Polynomials72, 102, 106, 109, 110, 136, 172, 220, 249, 250, 251, 254, 256, 312, 342, 388, 389, 391, 410, 418, 420, 441, 467, 470, 471, 472, 473, 474, 476, 478, 480, 481, 483, 485, 486, 490, 491, 533, 536, 538, 539, 578, 597, 599
Polynomials and Spans470
Polynomials for Other Purposes.474
Portable Installation of the dataTaker 26
Power Consumption291, 481
Powering and Connecting IC Temperature Sensors198
Powering Relays 56
Powering Sensors 55
Powering Sensors from the dataTaker 55
Powering the dataTaker27, 31, 46, 304, 561
Preparing The Host Computer64
Pressure Transducer.248
Preventing Alarm Schedules from Waking the dataTaker460
Preventing Schedules from Waking the dataTaker 435
Preventing the Transfer of Data518, 531
Priorities for Returning and Logging Data ...62, 435, 506
Processing68, 86, 552, 559, 560, 561
Program Area 532
Program Area,Clearing532
Programming a dataTaker over the Network 569
Programming from Memory Cards517, 529, 553
Programming the dataTaker12, 297, 305
Programming the Histogram498
Programming the Statistical Functions,Examples 420
Protected Installation of the dataTaker 26
Protecting the Internal Channel Table 397
Protocols ...299, 301, 552
Pulse Output.....356, 358
Q 300, 434, 462, 478, 527, 529, 530, 536, 540, 550, 551, 553, 594, 595
Quarter Bridge with Constant Current Excitation 236
Quarter Bridge with Current Excitation,Interpreting the Data from 238
Quarter Strain Gauge Bridges, Calculating Microstrain for 238
Quitting Data Unloading527
Radians 491, 495, 578
Range of Frequency Measurement150
Range of System Timers373
rate of change389, 390, 485
Rate of Change 390
Reading
Accumulating Counters 272, 278
Differential Analog Channels 331
Differential Analog State Inputs 340
Digital Bit Inputs268, 337
Digital Byte Inputs269, 337
Directly Connected High Level Voltages.....111
High Speed Counters351
Input Channels.....11
Low Speed Counters347
Phase Encoder... 355
Real Time Clock... 33
Resetting Counter276
Resetting Counters272, 279
Same Channel for a Number of Inputs 335
Single Ended Analog Channels 331
Single Ended Analog State Inputs 341
Temperature of the dataTaker and the CEM 34
Thermocouple Isothermal Block 182
Real Time Clock11, 31, 32, 33, 314, 317, 319, 364, 367, 420, 445, 549, 551, 553, 558, 562, 563
Reallocating the Internal Channel Table 397
Real-Time Network... 570
Recording Commands into a Memory Card 529
Redirecting Data to an Alternative Address..... 568
Reference Channels. 442
Reference Junction Temperature Compensation91, 92, 169, 170, 171, 172, 173, 178, 179, 180, 181, 183, 291, 378, 381, 391, 443, 559
Reference Junction Temperature Sensor..... 178
Referenced to Analog Return90, 152, 195
Regulatory Compliances3
Re-inserting the Same Memory Card 516
Related Documents 2
Related Products 2
relational operators443, 444
Relay Multiplexer20, 68, 69, 85, 86, 102
Removing and Changing Memory Cards While Logging Data.516
Repeating TEST Data545
RESET command29, 47, 256, 299, 301, 302, 308, 315, 316, 318, 320, 321, 322, 344, 368, 397, 398, 404, 411, 414, 417, 425, 431, 433, 434, 435, 436, 438, 449, 451, 460, 463, 481, 496, 499, 500, 502, 505, 506, 515, 519, 522, 530, 532, 543, 549, 556, 568, 573, 584, 587, 591
RESET Command with DeLogger543
RESET Command with DeTransfer542
Resetting Counter..... 276
Resetting Counters272, 279
Resetting the dataTaker29, 300, 542
Resetting the dataTaker Receive Buffer 300
Resetting to Zero 388

Resistance138, 146, 185, 189, 220, 222, 257
Resistance Measurement with External Common,Four Wire. 142
Resistance Measurement,Four Wire140, 141, 142
Resistance Measuring Circuit138, 139
Resistance Using Single Ended Measurement .. 146
Resistances,Connection and Measurement of140
Resolution8, 12, 72, 73, 95, 101, 123, 137, 138, 148, 169, 171, 189, 196, 222, 233, 236, 239, 244, 247, 251, 311, 312, 313, 318, 364, 382, 384, 387, 479, 541, 600
Results of Expressions to Channel Variables Assigning 484
Resuming Scanning.... 37
Re-Triggering Schedules432
Return of Data,Controlling 308
Returned Data,Format of301, 308, 393, 424, 461, 469, 473, 476, 478, 522, 526, 549, 551, 552, 553, 561, 562, 563
Returning a Message to the Host Computer or Printer 447
Returning Data - Individual Control502
Returning Data From Alarm Input Channels..... 461
Returning Data from Memory506, 520, 562
RS232 COMMS Serial Interface28, 58, 83, 84
RS232 COMMS Serial Interface - dataTaker Series 1 Only58
RS232 COMMS Serial Interface Connector 58
RS232 Communications Parameters 62
RS232 Control and Handshake Lines 61
RTD 11, 172, 178, 179, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 228, 259, 260, 261, 262, 292, 324, 328, 330, 332, 382, 383, 391, 392, 410, 421, 467, 468, 598
RTD Measurement,Four Wire190
RTD Support by the dataTaker...185
RTDs Using Single Ended Inputs195
RTDs,Connection and Measurement of 189
Running Schedules at the Maximum Speed..... 410
RUNPROG 530, 532
Same Channel for a Number of Inputs 335
Sampling51, 68, 72, 73, 74, 75, 149, 150, 259, 270, 271, 318, 322, 376, 386, 387, 410, 418, 419, 420, 421, 459, 460, 557, 558, 560, 563, 570, 597, 598, 599, 600, 601
Scale Factor,Channel388
Scaling Data - Polynomials, Spans & Functions.....467
Scaling of Channel Data388
Schedule Execution Setup Time 597
Schedule Triggered by Digital Events Command.....412
Schedule Triggered by Poll360, 376, 400, 402, 403, 404, 405, 406, 407, 428, 570
Schedule Triggered by Poll Command 405
Schedule Triggered by Time Command 408
Schedules (STATUS2)434, 538
Schedules and Schedule Lists298
Schedules from the Internal Channel Table,Clearing .434
Schedules Triggered by Counter Events 265, 336, 348, 352, 355, 415, 416, 417
Schedules Triggered by Counter Events Command.....415
Schedules Triggered by Digital Events 265, 336, 343, 412, 413, 414
Schedules Triggered by Time408, 409, 410, 411
Schedules Triggered While Condition 265, 336, 343, 404, 426, 427, 458
Schedules,Combining428, 432
Scroll Keys 293, 586, 588
Second371, 372, 373, 374
Selecting Automatic Low Power Mode 47
Selecting the Power Mode . 47
Selection Time Channel597
Self Checking of Test Parameters545
Self Documentation of Commands307
Sending Commands Directly from DeLogger.....43
Sending Commands with a Program from DeLogger...45
Sequences323, 333, 335, 380, 441, 473, 476, 478
Serial Interfaces 57
Setpoints for Digital Input Channels446
Setting
Address.....67, 565
Day or Date.....365
Digital Bit Output Channels 281
Digital Byte Output Channels 283
Digital Output Channels of the CEM-AD.....284
High Speed Counter Initial Count 349
High Speed Counter Range 350, 351
High Speed Counter Range and Initial Count 351
Low Speed Counter Initial Count 345
Low Speed Counter Range 346, 347
Low Speed Counter Range and Initial Count 347
Mains/Line Frequency 54
Number of Alarms438, 441
of Parameters..... 556
Parameters from DeTransfer 563
Phase Encoder Counter Range 355
Phase Encoder Initial Count 354
Range of System Timers 373
Real Time Clock... 32
RS232 Communications Parameters..... 62
Significant Digits and Decimal Places for Individual Channels 312
Switches from DeLogger 554, 564
Switches from DeTransfer 554
Time 366
Settling Time..... 597
Sharing Current Loop Signals127
sign on message..... 542
Signal Routing 387
Signal Type Identifier 324
Significant Digits and Decimal Places for Individual Channels. 312
SIGNOFF.... 64, 303, 560
Single Ended Analog Channels..331
Single Ended Analog Input Mode89, 332
Single Ended Analog Inputs92, 328
Single Ended Analog State Inputs341
Single Ended Frequency Inputs Referenced to an External Common..... 154
Single Ended Frequency Inputs Referenced to Analog Return152
Single Ended Inputs83, 90, 91, 98, 99, 108, 109, 119, 121, 132, 134, 147, 152, 154, 159, 160, 161, 164, 165, 175, 176, 204, 206, 209, 211, 212, 215, 216, 218, 329, 332, 334
Single Ended Inputs for the dataTaker 50 329
Single Ended Inputs for the dataTaker 500/600 Series 329
Single Ended Inputs Referenced to an External Common83, 91, 99, 109, 121, 134, 154, 161, 176, 206, 212, 216, 218, 334
Single Ended Inputs Referenced to an External Ground332

Single Ended Inputs Referenced to Analog Return90, 98, 108, 119, 132, 159, 160, 175, 204, 211, 215
Single Ended Inputs Referenced to dataTaker Ground332
Single Ended Measurement of Attenuated Voltage107
Single Ended Measurement of Low Level Voltage 97
Single Ended Measurement of Vibrating Wire Strain Gauges. 254
Single Ended Measurement Referenced to an External Common99, 109
Single Ended Measurement Referenced to Analog Return98, 107, 119, 132
Single Ended Measurements Referenced to an External Common.. 120, 134
Single Ended Threshold Voltage 153
Single Line Format (DeTransfer) 428
Solid State Multiplexer20, 69, 86
Some Application Ideas417
Some Applications Ideas414
Some Examples of Data Item Formats 318
Sources of Error 196, 229
Spans102, 106, 109, 110, 136, 220, 249, 251, 254, 256, 342, 388, 434, 441, 462, 467, 470, 472, 474, 475, 476, 477, 478, 480, 483, 485, 536, 539
Spans for Current Loops477
Speaker Output..... 256
Special Channel Options387
Specifying Channels85, 322, 419
Specifying Channels for Statistical Functions 419
Specifying Differential Channels.331
Specifying Excitation Current139
Specifying Sequences of Single Ended Channels..... 333
Specifying Single Ended Channels332
Standard Conditional Calculations494
Standard Deviation391, 419, 420
State Inputs..... 339
State Thresholds..... 156
Statistical Data to Channel Variables Assigning 483
Statistical Data,Format of Displayed424
Statistical Data,Format of Returned424
Statistical Evaluation. 391
Statistical Sub Schedule337, 376, 391, 392, 396, 418, 419, 420, 421, 423, 424, 425, 426, 432, 483, 486, 489
Statistical Sub Schedule Command419
Statistical Sub Schedules within Multiple Triggered Schedules424
STATUS306, 434, 462, 478, 500, 502, 505, 507, 508, 510, 513, 529, 530, 532, 536, 537, 538, 565, 566, 592
STATUS Command ..536
Status Screens..536, 584
Status Screens,Controlling585
STATUS1 .....538, 565
STATUS10 538, 540, 541
STATUS11 .....538, 541
STATUS12 .....538, 541
STATUS13 .....538, 541
STATUS2 .....434, 538
STATUS3306, 462, 538, 539
STATUS4 ..478, 538, 539
STATUS5500, 507, 538, 539
STATUS6507, 510, 538, 539
STATUS7508, 513, 538, 539, 540
STATUS8 ..532, 538, 540
STATUS9502, 505, 507, 530, 538, 540
Stop bits 9, 63, 64, 299
Storage capacity.....506
Stored data30, 39, 293, 297, 309, 318, 319, 320, 480, 481, 490, 506, 508, 514, 519, 520, 527, 531
Storing Programs in to Memory Cards from DeLogger.533
Strain gauge8, 11, 188, 230, 231, 232, 234, 236, 239, 244, 247, 252, 253, 254, 255, 257, 259, 386
Strain Gauge Bridges233, 235, 238, 244, 247
Style of Programming 572
Subtract 381, 393, 494
Summary of Memory Card Insertion519, 535
Supervising the dataTaker ... 9
Switch Commands38, 298, 307, 522, 542, 549, 556, 568
Switches (/)27, 53, 58, 62, 63, 64, 66, 67, 68, 240, 264, 267, 270, 277, 282, 353, 526, 527, 533, 549, 550
Switches from DeLogger554, 564
Switches from DeTransfer 554
Switching Digital Outputs in Schedules 358, 360
Switching Outputs from an Alarm453
Switching the Digital Bit Outputs 356
Switching the Digital Byte Output359
Switching the Multiplexer Power for the dataTaker 50 53
Switching the Multiplexer Power for the dataTaker 500 dataTaker 60053
Switching the WARN Channels.. 362
Synchronizing Schedule Time Triggers to Last Midnight433
System Timer371, 372, 373, 374, 465, 489
System Timers364, 371, 372, 373, 374, 388, 437, 438, 441, 466, 468, 480, 481, 489, 490
System Variables371, 375, 376, 377, 378, 423, 437, 441, 468, 507, 508
System Variables in Alarms377
System Variables in Schedules ..376
Tab and Space Characters306
Temperature of the dataTaker and the CEM 34
Temperature with Integrated Circuit Temperature Sensors198
Temperature with RTDs185
Temperature with Thermistors220
Temperature with Thermocouples169, 443
Termination of Analog Inputs87
TEST 7, 83, 287, 306, 529, 542, 544, 545, 546, 547, 548, 592
TEST Command with DeLogger.548
TEST Command with DeTransfer544
Text Channel374, 375, 568, 569
The Wake Terminal 49
Thermistor Measurement,Four Wire224
Thermistor Support by the dataTaker 220
Thermistors,Connection and Measurement of224
Thermocouple
Alloys and Combinations 169
Measurement Accuracy 183
Reference Junction170, 171
Reference Junction Temperature 171
Reference Temperature 179, 391
Support by the dataTaker 171
Wire Colour 170
Zero Reference Channel 391
Zero Voltage..... 171
Thermocouple Isothermal Block .182
Thermocouple Temperature, Calculation of 172
Thermocouples 291
Thermocouples as Differential Inputs 173
Thermocouples as Single Ended Inputs 174
Thermocouples Basic Concepts of169
Three Wire Resistance Measurement 143
Three Wire RTD Measurement191, 192
Three Wire Thermistor Measurement 225, 226
Time 366
Time Format364, 446, 523, 563
Time Formats..... 364
Time separator character (ASCII)446, 563
Time,Schedules Triggered by408, 409, 410, 411
Trademarks..... 2
Transfer of Data to Memory Cards,Controlling517, 530
Transmission errors (read only)..560
Triggered Schedules,Identifying .403

Triggering Schedules265, 282, 360, 374, 396, 400, 401, 402, 403, 404, 405, 410, 411, 414, 417, 419, 426, 431, 438, 481, 538, 570, 575
Triggering the Schedule from an Alarm 407
Triggering the Schedule from the Host Computer 406
Trimming Errors in the Reference Voltage..... 77
Two Wire Resistance Measurement145
Two Wire RTD Measurement193, 194
Two Wire Thermistor Measurement227, 228
U 87, 105, 106, 183, 302, 316, 317, 318, 319, 320, 381, 383, 384, 404, 424, 431, 434, 461, 462, 469, 473, 476, 478, 520, 522, 536, 540, 544, 545, 550, 551, 553, 591, 594, 595
UA 521, 523
UB 521
UC 521, 523
UD 521
Units of Temperature 167
Units Text302, 316, 317, 319, 320, 404, 420, 424, 431, 481, 486, 536, 544, 545, 553, 591
Units Text for Channel Variables481, 486
Unload Command..... 520
Unloaded Data,Format of the526
Unloading
Data for a Specific Period 521
Data from a Specific Schedule 521
Data from a Specific Source 521
Data Manually with DeTransfer 522
Data with DeLogger524
Data with DeTransfer523
Unloading:521, 522, 523, 524, 527, 569
Use of Immediate Scan Schedules575
Using
Alarms for Controlling 464
Battery Life Chart..51
CDATA Command from the Program Area ...531
Channel Variables486
Channel Variables as Setpoints 445, 488
Channel Variables as Setpoints in Alarms.....488
Channel Variables as the Test Data in Alarms.....487
Channel Variables in Schedules 486
Channel Variables to Transfer Statistical Data to Alarms 489
Channel Variables with Date and Time.....489
Declared Spans ..476
External Isothermal Block 178
Externally Referenced Single Ended Inputs to Reduce Cabling 93
Immediate Schedules in Programs 400
Polynomials 473, 474
Polynomials for Other Purposes 474
RESET Command with DeLogger 543
RESET Command with DeTransfer 542
System Timers....371
System Variables in Alarms 377
System Variables in Schedules 376
TEST Command with DeLogger 548
TEST Command with DeTransfer 544
Thermocouples... 291
UX 521
Variables13, 369, 375, 376, 392, 399, 418, 419, 473, 477, 486, 490, 542, 549, 556, 572, 573
Vibrating Wire Strain Gauges252
Vibration Wire Support by the Geologgers..... 252
Voltage Output..... 385
Voltage to Temperature Conversion171
Waking of the dataTaker,Controlling48
WARN356, 362, 363, 453
WARN Channels..... 362
Warning. 2, 356, 552, 590
Warning Outputs356, 590
Warnings and Error Messages571, 591, 592
Warnings and Error Messages - Numerical Listing592
Warranty 2
When the Internal Battery Goes Flat 39
While Condition,Schedules Triggered 265, 336, 343, 404, 426, 427, 458
Wiring a dataTaker Network65
Writing dataTaker Programs572
Writing the Program.. 572
XON/XOFF Protocol . 299
Yellow Springs Inc Thermistors ..223
YSI Thermistor Types222
Zero Voltage Compensation172
Zero Voltage Reference180