

***dataTaker***<sup>®</sup>

Intelligent Data Logging Products

# ***DT80*** **User's Manual**



A complete guide to *DT80*:

- data acquisition
- data logging
- programming
- sensor wiring
- communications

[www.datataker.com](http://www.datataker.com)

# DT80 User's Manual

© Copyright 2005 Datataker P/L.  
UM-0085-A0

## Warranty

Datataker Pty Ltd warrants the instruments it manufactures against defects in either the materials or the workmanship for a period of three years 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 or to one of its authorized dealers.

This warranty excludes all other warranties, either express or implied, and is limited to a value not exceeding the purchase price of the instrument.

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

Where Datataker P/L supplies to the customer equipment or items manufactured by a third party, then the warranty provided by the third party manufacturer remains.

## Trademarks

dataTaker is a registered trademark of Datataker Pty Ltd.

All other brand and product names are trademarks or registered trademarks of their respective holders.

## Related Software Products

DeLogger, DeLogger Pro, DeTransfer, DeLoad, DeView  
dataTaker ActiveX, dataTaker LabVIEW™ instrument driver

## DT80 Firmware Covered in This Manual

This version of the *DT80 dataTaker User's Manual* (UM-0085-A0) applies to *DT80s* running **version 5.02 (or later)** firmware.

## WARNING

dataTaker products are not authorized for use as critical components in any life support system where failure of the product is likely to affect the system's safety or effectiveness.

## List of Major Tables

Table 1: DT80 *Channel* Types [\(P29\)](#)

Table 2: *DT80* System Variables [\(P32\)](#)

Table 3: DT80 *Channel* Options [\(P38\)](#)

Table 9: *DT80* Parameters [\(P112\)](#)

Table 10: *DT80* Switches [\(P113\)](#)

Table 11: *DT80* PROFILE Details [\(P115\)](#)

Table 12: *DT80* Resets [\(P119\)](#)

Table 13: *DT80* TEST Report [\(P121\)](#)

Table 14: *DT80* Delete Commands — Summary [\(P166\)](#)

Table 15: *DT80* Retrieval Commands — Summary [\(P168\)](#)

Table 16: ASCII Characters [\(P170\)](#)

Table 17: RS-232 Pinouts [\(P171\)](#)

Table 18: *DT80* Error Messages [\(P176\)](#)

# Contents

<b>Part A — The DT80</b> .....	<b>11</b>
<b>DT80 Concepts</b> .....	<b>11</b>
What is the <i>DT80</i> ? .....	11
DT80-Friendly Software .....	11
About This Manual .....	12
A Tour of the DT80's Interfaces .....	12
Getting Started.....	12
Power .....	12
Switch On! .....	13
Connecting to a Host Computer.....	13
Sending Commands .....	13
Localisation.....	13
Ways of Using the <i>DT80</i> .....	14
Fundamental Inputs and Ranges .....	14
Fundamental Input Ranges .....	14
Accuracy of the DT80.....	15
Derived Measurement Ranges .....	15
Analog Channels — Introduction.....	15
Input Terminals.....	15
Multiplexers .....	15
Gain Ranges and Attenuators.....	16
Analog Input Configurations .....	16
Sensor Excitation .....	17
Digital Channels — Introduction.....	17
Serial Channel – Introduction.....	18
<b>Programming the DT80</b> .....	<b>18</b>
Specify Channel Types .....	18
Add Channel Options .....	18
Test Each Sensor.....	18
Schedule Commands.....	18
Jobs .....	19
Scaling and Calculations.....	19
Reducing Data.....	19
Alarms .....	19
IFs.....	19
Data Logging.....	19
Handle With Care .....	19
Retrieving Data.....	19
Examples of Things You Can Do with Channels .....	20
USB memory devices .....	20
<b>Format of Returned Data</b> .....	<b>21</b>
Real-time data .....	21
Free Format Mode /h.....	21
Fixed Format Mode /H.....	21
Logged Data .....	22
Native Format.....	22
Fixed Format .....	22
<b>Guidelines for Successful Data Gathering</b> .....	<b>22</b>
The Procedure.....	22
Ground Loops.....	22
Grounds, Ground Loops and Isolation .....	23
Grounds are Not Always Ground .....	23
Ground Loops.....	23
Avoiding Ground Loops .....	23

Isolation .....	23
Noise Pickup .....	23
Self-Heating of Sensors .....	24
<b>Part B — Channels .....</b>	<b>25</b>
<b>Channel Definitions .....</b>	<b>25</b>
<b>Channel Numbers .....</b>	<b>25</b>
Channel Number Sequence .....	26
<b>Channel Types .....</b>	<b>26</b>
Internal Channel Types (in detail) .....	30
Time & Date .....	30
Text .....	30
Internal Maintenance .....	30
System Timers .....	31
System Variables .....	31
<b>Channel Options .....</b>	<b>32</b>
Overview .....	32
A Special Channel Option — Channel Factor .....	33
Multiple Reports .....	33
Mutually Exclusive Options .....	34
Order of Application .....	34
Default Channel Options .....	34
Channel Option Table .....	35
<b>Part C — Schedules .....</b>	<b>39</b>
<b>Schedule Concepts .....</b>	<b>39</b>
What are Schedules? .....	39
Schedule Syntax .....	39
Schedule ID .....	39
Schedule Name .....	40
Schedule Options .....	40
Schedule Trigger .....	41
Channel List .....	41
A Simple Schedule .....	41
Groups of Schedules — Jobs .....	42
<b>Types of Schedules .....</b>	<b>42</b>
General-Purpose Report Schedules (RA, RB, ...RK) .....	42
Trigger on Time Interval .....	42
Trigger on External Event .....	43
Trigger on Internal Event .....	43
Trigger on Schedule-Specific Poll Command .....	44
Trigger While .....	44
Continuous Report Schedules (No Trigger) .....	45
Special-Purpose Report Schedules .....	45
Polled Report Schedule (RX) .....	45
Immediate Report Schedules .....	46
Statistical Report Schedules .....	46
<b>Working with Schedules .....</b>	<b>48</b>
Entering Schedules into the DT80 (BEGIN-END) .....	48
Using Immediate Schedules in Programs .....	48
Time Triggers — Synchronizing to Midnight .....	48
Retrieving Entered Schedules and Programs .....	49
Triggering and Schedule Order .....	49
Changing a Schedule Trigger .....	49
Naming Schedules .....	49

Halting & Resuming Schedules.....	49
Locking Schedules .....	49
Deleting Schedules .....	50
Special Commands in Schedules.....	50
Conditional Processing — IF... Command .....	50
Conditional Processing — Boolean Expressions.....	51
Unconditional Processing — DO... Command .....	51
<b>Part D — Jobs.....</b>	<b>53</b>
<b>Part E — Manipulating Data .....</b>	<b>56</b>
<b>Channel Options — Statistical.....</b>	<b>56</b>
Average (AV) .....	56
Standard Deviation (SD) .....	56
Maximum and Minimum (MX and MN).....	56
Integration (INT).....	57
Histogram (Hx:y:m.nCV) .....	57
Rainflow Cycle Counting.....	58
<b>Channel Options — Scaling.....</b>	<b>61</b>
Channel Factor (f.f) .....	61
Intrinsic Functions (Fn).....	61
Spans (Sn) .....	62
Polynomials (Yn) .....	62
Thermistor Scaling (Tn).....	63
Channel Variables (nCV).....	63
<b>Calculations (Expressions).....</b>	<b>65</b>
Conditional Calculations.....	65
<b>Combining Methods .....</b>	<b>65</b>
<b>Part F — Logging and Retrieving Data.....</b>	<b>67</b>
<b>Format of Returned Data.....</b>	<b>67</b>
Character Pairs — Carriage Return + Line Feed.....	67
Two Format Modes for Returned Data .....	67
Free-Format Mode /h .....	67
Fixed-Format Mode /H .....	68
<b>Logging Data.....</b>	<b>69</b>
LOGON and LOGOFF Commands .....	69
Logging issues .....	69
Data Storage Issues.....	70
Storage Status.....	70
Data Storage Capacity — Readings/MB.....	70
Halt and Go During Data Logging .....	70
Deleting Logged Data.....	70
Moving , Copying and Archive Logged Data.....	71
The DT80 File System.....	71
<b>Retrieving Logged Data.....</b>	<b>73</b>
Retrieving Logged Data — USB memory device Transfer.....	73
Retrieving Logged Data — Comms Unload .....	73
Unload Commands.....	74
The U Unload Commands .....	74
The U( ) Unload Commands .....	74
The U[ ] Unload Commands .....	75
Labelling the End of Unloaded Data .....	76
Quitting an Unload.....	76
<b>Part G — Alarms.....</b>	<b>77</b>

<b>Alarm Concepts .....</b>	<b>77</b>
Alarm Number .....	78
Alarm Input .....	78
Alarm Condition .....	79
Alarm Delay Period .....	79
Alarm Digital Action Channels .....	80
Alarm Action Text .....	80
Alarm Action Processes .....	82
Combining Alarms .....	83
Polling Alarm Data .....	84
<b>Logging and Retrieving Alarms .....</b>	<b>84</b>
Logging Alarm States .....	85
Logging Alarm States — What's Logged, What's Returned .....	86
Retrieving Logged Alarm States .....	86
The A Unload Commands .....	86
The A( ) Unload Commands .....	87
The A[ ] Unload Commands .....	88
Deleting Logged Alarm Records .....	88
 <b>Part H — DT80 Front Panel .....</b>	 <b>90</b>
Display .....	90
Displaying Channels and Alarms .....	90
Bar Graph .....	91
Controlling what is shown on the display .....	91
Transient Messages .....	91
Display Backlight .....	92
User defined functions .....	92
The FUNCTION command .....	92
Selecting Functions .....	92
Default Functions .....	92
Keypad operation .....	93
Direction Keys .....	93
OK (Edit) Key .....	93
Cancel (Function) Key .....	93
Special Key Sequences .....	93
Entering Bootstrap Mode .....	93
Status Indicator Lights .....	93
Sample Indicator .....	93
Disk Indicator .....	93
Attn Indicator .....	93
 <b>Part I — Communications .....</b>	 <b>95</b>
Automatic Comms Port Arbitration .....	95
Password Protection — Comms Ports .....	95
 <b>USB Communications .....</b>	 <b>95</b>
Installing the USB Driver .....	96
Using the USB Connection .....	96
 <b>RS-232 Communications .....</b>	 <b>97</b>
Quick Start .....	97
DT80 RS-232 Basics .....	97
Host RS-232 Port Pinout .....	97
Automatic Device Detection .....	97
Host RS-232 Port Commands .....	97
Flow Control .....	98
Echo .....	99
Input Buffer (How the DT80 Receives and Processes a Program) .....	100
Comms Wakes the DT80 .....	100
DT80 Direct (Local) RS-232 Connection .....	100
Setting Up a Direct Connection .....	100

DT80 Modem (Remote) RS-232 Connection.....	100
DT80-to-Modem Cable.....	101
Modem Initialization.....	101
Modem Initialization Conditions.....	101
Modem Initialization Settings.....	101
AT Command Set.....	102
Modem Automatic Baud Rate Selection.....	102
Modem Communications Protocol.....	102
Powering the DT80's Modem.....	102
Modem Communications Operation.....	103
Dialling In.....	103
Dialling Out.....	103
Modem Status.....	103
Setting Up a Remote Connection.....	103
Installing the Host Computer's Modem.....	104
Using the Modem Connection.....	104
Visits to Site.....	104
<b>DT80 Ethernet Communications .....</b>	<b>104</b>
Ethernet Concepts.....	104
IP Address.....	104
IP Subnet Mask, IP Gateway.....	105
Ethernet Settings are Preserved.....	106
IP Port Number.....	106
Network Adapter Address.....	106
Ethernet Commands.....	106
DT80 Ethernet Setup.....	106
<b>DT80 FTP Communications .....</b>	<b>108</b>
<b>DT80 PPP Communications.....</b>	<b>108</b>
<b>Part J — Configuration.....</b>	<b>109</b>
<b>    Configuring the DT80 .....</b>	<b>109</b>
Parameters.....	109
Reading Parameters.....	109
Setting Parameters.....	109
Switches.....	112
Viewing Switch Settings.....	112
User Startup Defaults.....	113
User Startup Profile.....	113
USER.INI (User Initialization File).....	113
PROFILE... Commands.....	115
Startup Job.....	116
ONRESET.DXC.....	116
ONINSERT.DXC.....	116
Protecting Startup Files.....	117
Setting the DT80's Clock/Calendar.....	118
Setting the DT80's Time (T=).....	118
Setting the DT80's Date (D=).....	118
Setting Date and Time Together (DT=).....	118
<b>    Resetting the DT80 .....</b>	<b>119</b>
Wait after RESET.....	119
Manual Reset Button.....	120
Factory Defaults.....	120
LEDs and Messages After a Reset.....	120
<b>    TEST Commands DT80 .....</b>	<b>121</b>
Test Report (DT80 Health).....	121
<b>    Event Log .....</b>	<b>121</b>
Unloading the Event Log.....	121
Clearing the Event Log.....	122
<b>    STATUS Commands .....</b>	<b>122</b>

STATUS .....	122
STATUSn .....	122

**Part K — Hardware and Power ..... 124**

<b>Inputs and Outputs .....</b>	<b>124</b>
DT80 Front Panel .....	124
DT80 Wiring Panel .....	124
DT80 Side Panel .....	125
<b>MEMORY .....</b>	<b>125</b>
Storage Capacity .....	125
USB memory device Commands .....	125
<b>INSIDE THE DT80 .....</b>	<b>126</b>
Accessing the main battery .....	126
Accessing the lithium memory backup battery .....	127
Mounting the DT80 .....	128
Dimensions, Clearances .....	128
<b>Power .....</b>	<b>129</b>
<b>POWERING THE DT80 .....</b>	<b>129</b>
Operating Environment .....	129
Internal Power (Main Battery) .....	129
Main Battery is Disconnected for Shipping .....	129
Main Battery Life .....	129
External Power .....	130
Solar Charging .....	130
Internal Memory-Backup Battery .....	130
Battery Guidelines for Long-Term Storage .....	130
Internal Main Battery During DT80 Storage .....	130
Internal Memory-Backup Battery During DT80 Storage .....	131
<b>LOW-POWER OPERATION .....</b>	<b>131</b>
Power .....	131
Always Trying to Sleep .....	131
Controlling Sleep .....	132
Extending Battery Life .....	132
Low-Power Programs .....	132

**Part L — Sensors and Channels ..... 133**

<b>Analog Channels .....</b>	<b>133</b>
Analog Sensors and Measurement .....	133
4–20mA Current Loops .....	133
Frequency .....	134
Thermocouples .....	134
Using Thermocouples with the DT80 .....	135
Thermocouple Types .....	135
Grounded Thermocouples .....	136
Accuracy — Thermocouple Techniques .....	136
Thermistors .....	136
RTDs .....	137
IC Temperature Sensors .....	137
Bridges .....	138
Bridge Excitation (Lead Compensation) .....	138
Scaling .....	139
Strain Gauges .....	139
Humidity Sensors .....	139
Analog Logic State Inputs .....	140
DT80 Analog Sub-System .....	141
DT80 Ground Terminals .....	141
<b>DIGITAL CHANNELS .....</b>	<b>141</b>



Bidirectional Digital I/O Channels .....	142
Using Digital Inputs .....	142
Channel Types .....	142
Channel Options .....	142
Connecting to Digital Inputs .....	143
Other Considerations .....	143
Using Digital Outputs .....	144
Channel Types .....	144
Channel Options .....	144
Digital Output Operation .....	144
Connecting to Digital Outputs .....	145
Other Considerations .....	145
High Speed Counter Channels .....	146
Using Counter Inputs .....	146
Channel Types .....	146
Channel Options .....	146
Connecting to Counter Inputs .....	146
Phase Encoders .....	147
Other Considerations .....	147
Examples .....	147
<b>SERIAL CHANNEL .....</b>	<b>148</b>
Connecting to the Serial Channel .....	148
Setting Serial Channel Parameters .....	148
Serial Channel Commands .....	149
Serial Channel Operation .....	150
The Control String .....	150
Serial Data Transmission and Reception .....	150
Control String – Output Actions .....	151
Control String – Input Actions .....	152
Control String – Example .....	154
Schedules .....	155
Serial Sensor Power Control .....	155
Serial Channel State .....	155
Serial Channel Debugging Tools .....	156
Serial Channel Examples .....	156
Configuring the Serial Channel .....	157
<b>WIRING CONFIGURATIONS — ANALOG CHANNELS .....</b>	<b>157</b>
Voltage Inputs .....	157
Shared-Terminal Voltage Inputs .....	158
Independent Voltage Inputs .....	158
Current Inputs .....	158
Independent Current Input with External Shunt .....	158
Independent Current Input using the internal shunt .....	159
Shared-Terminal Current Inputs with External Shunts .....	159
Independent current using internal shunt and external excitation .....	160
Resistance Inputs .....	160
4-Wire Resistance Inputs .....	160
3-Wire Resistance Inputs .....	160
2-Wire Resistance Inputs .....	161
Bridge Inputs .....	161
6-Wire BGV Inputs .....	161
4-Wire BGV Inputs .....	161
4-Wire BGI Inputs .....	162
3-Wire BGI Input .....	162
AD590-Series Inputs .....	162
2-Wire AD590-Series Inputs .....	163
LM35-Series Inputs .....	163
3 & 4-Wire LM35-Series input - full temperature range .....	163
3 and 4-Wire LM35-Series Inputs – restricted temperature range .....	163

LM135-Series Inputs.....	163
4-Wire LM135-Series Inputs .....	164
<b>WIRING CONFIGURATION — DIGITAL CHANNELS.....</b>	<b>164</b>
Digital Input Wiring configurations.....	164
Digital output wiring configurations.....	165
<b>Part M — Reference.....</b>	<b>166</b>
<b>COMMAND SUMMARIES .....</b>	<b>166</b>
<b>GETTING OPTIMAL SPEED FROM YOUR DT80.....</b>	<b>168</b>
Best Speed .....	168
<b>ASCII-DECIMAL TABLE.....</b>	<b>170</b>
<b>RS-232 STANDARD.....</b>	<b>171</b>
<b>CABLE DETAILS .....</b>	<b>171</b>
<b>UPGRADING DT80 FIRMWARE.....</b>	<b>172</b>
Recommended Preparation .....	172
Firmware Upgrade — Host USB or RS232 Port .....	173
In Case of a Failed Upgrade .....	174
<b>ERROR MESSAGES.....</b>	<b>174</b>
<b>Glossary .....</b>	<b>177</b>
<b>Index .....</b>	<b>187</b>

# Part A — The DT80



Figure 1: The dataTaker DT80

## DT80 Concepts

---

### What is the DT80?

The *dataTaker DT80* data acquisition and logging instrument is a tool to measure and record a wide variety of quantities and values in the real world.

With the *DT80* basic measurement tasks are easy. For example, sending the command line

```
RA5S 1..4TJ LOGON
```

declares a report schedule (**RA**) that reports every five seconds (**5S**) the temperatures on four type J thermocouples connected to the *DT80*'s analog input channels 1 to 4 (**1..4TJ**), and stores the results in memory (**LOGON**).

Recovering the logged data is even easier. For example, sending the single-character command

```
U
```

(the **UNLOAD** command) to the *DT80* returns time-stamped data to your computer in a format ready to be imported into the preferred program. The connection between the *DT80* and the host computer could be via Ethernet, USB, RS232 or modem.

Alternatively, you could insert a USB "memory stick", and select the **COPYDATA** option using the built-in keypad and LCD display.

The *DT80* can be programmed to carry out extremely powerful tasks. To do this, it will be necessary to be familiar with more of the set of *dataTaker* commands. Explore the features that are available.

---

### DT80-Friendly Software

Although any terminal software can be used to communicate with the *DT80*, *dataTaker DT80*-friendly software packages incorporate so many productivity features specific to data acquisition, data logging and the *DT80* that make it pointless to use anything else. For example:

**DeLogger** has a totally graphical interface, which means that knowledge of the *dataTaker* programming language is not required. Instead, supervise the *DT80* just by clicking on icons and making selections from menus and dialog boxes. In addition to standard text output, the capability to display and print real-time and logged data in dynamic table, chart and mimic (meter) views, load data into a fully-featured spreadsheet, and replay saved data to any of the dynamic views.

**DeLogger Pro** is the big brother of DeLogger. It has the added features of modem support, a database data storage option, the ability to connect to more than one data site at a time, enhanced mimic screens, additional spreadsheet and graphical analysis tools, and e-mail and web publishing capabilities.

**DeTransfer** is the easiest host software to use with the *DT80* programming language. Its non-graphical interface provides complete access to all of the *DT80*'s capabilities, and it has separate send and receive windows, which are the basis of its exceptional and unique functionality. If your preference is a command-line interface, then DeTransfer is ideal.

**DeView** works in conjunction with DeTransfer. It graphs real-time data and unloaded data on the computer, like traces on a chart recorder.

**DeLoad** is a software package which allows the access to your data via a single click of an icon. Loggers can also be programmed with a simple drag and drop on an icon.

**dataTaker ActiveX** this is a software package to allow you simple access to the logger in applications such as Visual Basic, Visual C and VBA etc.

**dataTaker Instrument driver for LabVIEW™** A comprehensive set of drivers combined with the appropriate documentation to allow the logger to be implemented in a LabVIEW application.

dataTaker recommends starting with DeLogger. It's included on the CD provided with your *dataTaker* data logger. A "Getting Started" video is also provided on the disk. Then graduate to DeLogger Pro if extra capabilities are required.

If you are comfortable with the idea of programming the logger using its command language then you may prefer to create DT80 programs directly, using DeTransfer.

---

## About This Manual

This manual is intended for all users of the DT80. It describes:

- how to connect sensors and other devices to the DT80's input and output channels.
- how to program the DT80 to collect and return data as required.
- how to manage the data that the DT80 collects.

The main focus of this manual will be on directly programming the DT80 using its command language. However, most of the concepts discussed here also apply when building programs using tools such as DeLogger.

---

## A Tour of the DT80's Interfaces

The DT80's interfaces with the outside world are grouped into three main areas:

### User Interface

On the top panel of the DT80 you will find controls which allow the user to interact with the unit during operation – without requiring a host computer:

- A 2-line LCD display shows status messages, measured values, and a menu of pre-defined functions
- Six keypad buttons allow the user to navigate between the various displayed options
- Three status LEDs are provided – the blue **Sample** LED flashes each time a measurement is taken, the green **Disk** LED indicates internal flash disk activity, and the red **Attn** LED indicates various warning conditions.
- A USB socket allows connection of a USB memory device, which provides a convenient way to retrieve data from the DT80 (or load a program onto it)

### Sensor Interface

On the sloping front panel of the DT80 there are two rows of terminal blocks – digital channels on the left, analog channels on the right. The green terminal blocks can be quickly unplugged from the DT80 without unscrewing the sensor cabling.

This interface includes:

- 8 digital input/output/counter channels (**1D – 8D**)
- an input to wake the DT80 from low power "sleep" mode (**WK**)
- 4 counter inputs (or two phase encoder inputs) (**1C – 4C**)
- a pair of voltage free relay contact outputs (**RELAY A** and **B**)
- an RS232/422/485 compatible serial port (**Tx, Rx, RTS** and **CTS**)
- 4 analog input channels (**1 – 4**)
- an external excitation input (**EXT \***)

### Communications/Power Interface

On the left side panel you have a variety of connectivity options:

- 10-Base-T Ethernet for connection to a host computer or local area network
- USB for high speed connection to a host computer
- RS232 for connection to host computer or modem
- two alternative DC power connectors – a standard plug-pack socket (DC jack) and a 4-pin terminal block

For more details, see [COMMUNICATIONS](#)

---

## Getting Started

### Power

POWERING THE DT80 ([P129](#)) discusses the ways to provide power to the DT80. The simplest option is to plug in the supplied AC adaptor.

The DT80 includes an internal 6V lead-acid battery which can power the logger if the main external supply is interrupted.

**Important** The *DT80* is shipped with its main internal battery disconnected. We recommend the battery is connected as soon as practical so that it can charge from the mains adaptor or other external power source. This is achieved by simply plugging the green power connector: See ...

## Switch On!

When power is connected, you should observe:

- the LCD backlight switches on
- a brief clicking sound as the unit performs an initial self-calibration
- **DT80 restarted / Power loss** is displayed on the LCD
- the three front panel LEDs flash a few times then the red **Attn** LED continues to flash.

The *DT80* is warning you that its power has been interrupted. Press any of the front panel keys to clear this indication. The **Attn** LED should stop flashing and the display should now read: **DT80 V5.02 / No current job**. This indicates that:

- the version of *DT80* firmware in use is "5.02" (this number may vary), and
- no user program (or "job") has been loaded

The *DT80* is now idle and waiting for instructions.

## Connecting to a Host Computer

In order to program the *DT80*, it is generally necessary to connect it to a "host" computer. The easiest option here is to use the supplied USB cable. Other options are to use a "null-modem" (cross-over) RS232 cable, or to connect the logger to an Ethernet network. See Figure 17 Anatomy of a sample *DT80* program ([P53](#)) for more details of the different communications options.

Very briefly, connecting the *DT80* via USB involves the following steps:

1. Install the required dataTaker software (DeLogger and/or DeTransfer) on the host PC.
2. Connect the USB cable between the *DT80* and the PC.
3. The Windows "New Hardware Found" wizard will then run automatically (if required) to install the necessary drivers.
4. Launch DeTransfer (or DeLogger)
5. In DeTransfer (or DeLogger), create a "connection"; this involves selecting the port to use when communicating with the *DT80*. A "virtual COM port" (e.g. COM5) will have been assigned by the USB driver.
6. Press the "Connect" button in DeTransfer (or DeLogger).

The above is only an brief overview. See USB Communications ([P95](#)) for detailed, step by step instructions.

The remainder of this manual will assume you have successfully established a connection between the host PC and the *DT80*.

---

## Sending Commands

The *DT80* is programmed by sending it textual **commands**. Commands are executed by the *DT80* only after it receives a carriage-return character (↵).

Commands are not case-sensitive; that is, they may be entered using either uppercase or lowercase characters.

In this manual all commands are shown in **UPPERCASE**. Responses from the *DT80* are shown *like this*.

The general categories of commands are:

- **channel definitions** ([P11](#)) (e.g. **2TK("Kiln temp",FF4)**) – these define **what** measurements are to be taken, **how** they are to be acquired and how the measured values are to be presented.
- **schedule definitions** ([P25](#)) (e.g. **RA(DATA:2MB)10S**) – these define **when** a set of measurements are to be taken and **where** the results are to be stored
- **job management commands** ([P12](#)) (e.g. **BEGIN, END, SHOWPROG**) – these allow a set of schedule and channel definitions to be grouped into a single program, or "job", which can then be treated as a unit.
- **data management commands** ([P12](#)) (e.g. **U** (unload), **COPYDATA, DELALARMS**) – these allow logged data points and alarms to be retrieved, displayed or deleted.
- **configuration commands** ([P115](#)) (e.g. **PROFILE, Pn** (parameter), **/char** (switch)) – these allow various aspects of the *DT80*'s operation to be adjusted to suit particular requirements.

Jobs (sets of commands) are stored in the *DT80*'s internal file system along with the data they generate. Different jobs can be loaded under manual or program control. In addition, the *DT80* can automatically run a particular job every time it is reset or powered up. See Startup Job ([P116](#)).

---

## Localisation

Many different aspects of the *DT80*'s operation can be customised. Some of these relate to the locale in which it is operating – in particular the local mains frequency and date/time format. For best performance it is recommended that these settings (especially the mains frequency) be configured and saved before taking any serious measurements.

The DT80 parameter **P11** specifies the local mains frequency, in Hz (default 50Hz). When taking an analog measurement, the DT80 integrates over one or more complete mains periods, in order to minimise any mains-related noise pickup.

The parameter **P31** specifies the date format: **1** for European (DD/MM/YYYY), **2** for North American (MM/DD/YYYY) and **3** for ISO (YYYY/MM/DD) (default is European format).

These (and any other settings) can be applied in a "set and forget" fashion by entering them into the DT80's **startup profile**. For example, the following commands will set up the DT80 for North American mains frequency and date format:

```
PROFILE"PARAMETERS", "P11"="60"
PROFILE"PARAMETERS", "P31"="2"
SINGLEPUSH
```

(The **SINGLEPUSH** command resets the DT80, which is necessary in order to apply profile settings.) For explanations of parameters and profiles See [P115](#)

## Ways of Using the DT80

The DT80 can be deployed in many ways depending on factors such as location, data volume and power availability:

- on-line to a host computer – data is returned in real-time as it is acquired
- periodic downloading to an on-line host
- periodic downloading to a portable computer
- periodic downloading by modem to a host computer, initiated by either the computer or the DT80
- data recovery (and programming) using removable USB memory devices

The method of deployment influences the fine tuning of the DT80's programming. As a general rule, it is better to recover data as often as reasonably possible so that sensor failures, program faults and so on are detected earlier.

## Fundamental Inputs and Ranges

The DT80 can directly measure the following **fundamental inputs**:

- voltage
- current
- resistance
- frequency
- digital input state
- pulse count
- phase encoder position

Many other quantities can be measured by connecting appropriate **sensors** which convert a physical quantity into something that the DT80 can measure. The DT80 directly supports:

- 4-20mA current loop sensors (0 to 100%)
- temperature sensors (thermocouples, RTDs, thermistors, IC sensors)
- bridges and strain gauges

This list can be extended by means of user specified scaling calculations.

### Fundamental Input Ranges

The following table lists the available measurement ranges and resolutions for the fundamental input types.

Input Type	Range	Resolution
DC Voltage	±30 mV	0.25 µV
	±300 mV	2.5 µV
	±3000 mV	25 µV
	±30 V	250 µV
DC Current Internal Shunts(100Ω)	±0.3 mA	2.5 nA
	±3 mA	25 nA
	±30 mA	250 nA
External Shunts (typically 20~200Ω)	any range	depends on shunt
Resistance	100 Ω	1.5 mΩ
	1000 Ω	15 mΩ
	10,000 Ω	150 mΩ
Frequency	0.1 to 20,000 Hz	0.0002%
Digital Bit	0 or 1	1
Counter	-2,147,483,648 to	1 count

	2,147,483,647 counts	
Phase Encoder	-2,147,483,648 to 2,147,483,647 counts	1 count

## Accuracy of the DT80

Maximum measurement error is given by:

$$\text{error} = (\text{reading} * \text{Basic Accuracy}) + (\text{FullScale Reading} * 0.01\%)$$

where *Basic Accuracy* is as specified in the following table:

	5°C to 40°C	-45°C to 70°C
DC voltage measurement	±0.1%	±0.35%
DC current measurement	±0.15%	±0.45%
DC resistance measurement	±0.1%	±0.35%
Frequency measurement	±0.1%	±0.25%

## Derived Measurement Ranges

The following table indicates typical measurement ranges and resolutions for derived measurements using external sensors:

Input Type	Range	Resolution
4-20mA Loop	0 to 100%	0.01%
Temperature	-250.0 to 1800 °C	depends on sensor
Strain Gauges and Bridges	±10 <sup>4</sup> ppm	1 ppm
	±10 <sup>5</sup> ppm	10 ppm
	±10 <sup>6</sup> ppm	100 ppm
Analog State	0 or 1	1

## Analog Channels — Introduction



### Input Terminals

The DT80 provides four analog input channels, numbered 1 to 4. Depending on the wiring configuration used, these allow between 4 and 12 separate voltages to be measured.

Each analog input channel on a DT80 is a 4-wire connection (Figure 2 (P15)) that allows voltage, current, resistance and frequency to be measured. These are the fundamental signals output by most sensors. It is not necessary to use all four terminals on each channel—two are often adequate.

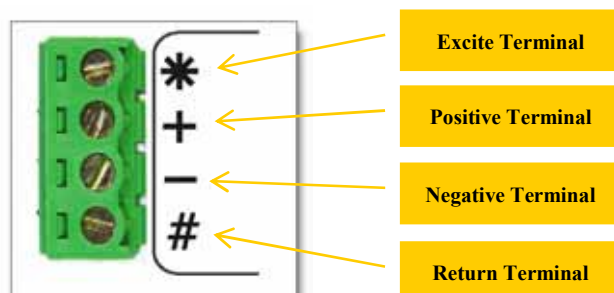


Figure 2: Analog input channel terminal labels

The exact function of each terminal varies depending on how the channel is programmed. In general terms:

- The \* ("Excite") terminal can be a voltage input, or it can provide sensor excitation current (for example, for resistance measurement).
- The + ("Plus") and - ("Minus") terminals are voltage inputs
- The # ("Return") terminal is normally used as a common or return terminal. It can also be used as a current input, using the DT80's internal shunt resistor.

### Multiplexers

The DT80's analog input channels are **multiplexed**. The required input terminals are first connected to the input of the DT80's instrumentation amplifier and analog to digital converter, then a measurement is taken. The next channel to be



sampled is then switched through to the amplifier and ADC, and so on.

**Channel definition commands** in the DT80 program determine which terminals are used for a particular measurement. For example, the channel definition **1+V** measures the voltage between the + and # terminals on channel 1.

## Gain Ranges and Attenuators

The DT80's instrumentation amplifier has three switchable gain settings. These give three basic voltage measurement ranges (3V, 300mV and 30mV full scale)

The DT80's default is for its instrumentation amplifier to automatically change gain range to suit the input signal applied to it by the multiplexers.

If the amplitude of your input signals are known, then the gain can be set manually. Do this by applying the **GLx** (gain lock) channel option, which disables autoranging for that channel and sets the gain to a fixed range.

The analog inputs also include switchable 10:1 **attenuators**, which effectively provide a fourth range (30V).

Note however, that the autoranging process does not affect the attenuator setting. Each channel definition command specifies (either implicitly or explicitly) whether the attenuators should be on or off.

**Warning** Knowledge of the output signal type and magnitude for each sensor is essential. Make sure that the input signal to the DT80 does not exceed the input voltage rating. As a general rule, the voltage on any analog input terminal should be within  $\pm 30V$  or  $\pm 3V$  (depending on whether the channel's attenuators are on or off) relative to the **AGND** terminal.

## Analog Input Configurations

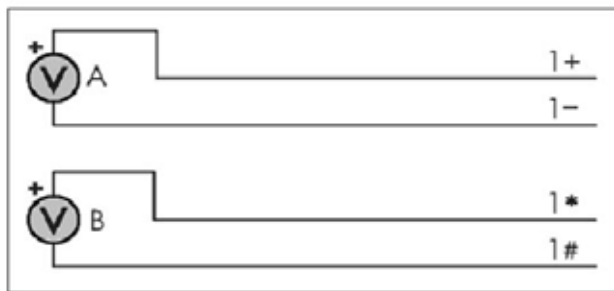
The basic quantity that the DT80 measures is voltage. Voltages can be measured using two different input configurations:

- **independent** analog inputs
- **shared-terminal** analog inputs

### Independent Analog Inputs

Sensors and signals connected using the independent configuration are often simply called "inputs" (sometimes also known as "basic", "default", "unshared", "differential" or "double-ended" inputs).

An independent input is one that connects to its own terminals and does not share any of those terminals with any other inputs. For example, in *Figure 3*, sensor A is connected to channel 1's + and - terminals, and sensor B is connected to the other two terminals of the channel. In other words, each sensor's terminals are independent of the other's — no terminal is used by both sensors.



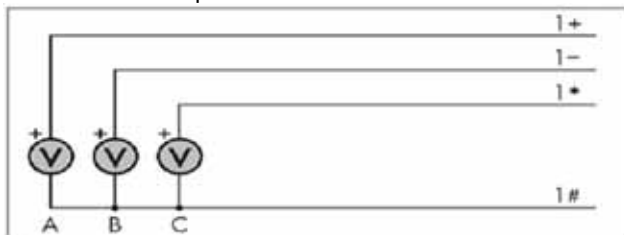
*Figure 3* Wiring one or two independent inputs to a single channel (voltage inputs used as example)

For an independent input, the signal voltage is measured between a pair of terminals and neither terminal is necessarily at ground potential.

Note that each analog input channel can support **two** independent voltage inputs. In the above example, the channel definition **1V** will read sensor A while **1\*V** will read sensor B. The channel definition syntax is fully described in Channels ([P25](#)).

### Shared-Terminal Analog Inputs

Sometimes called "single-ended" inputs, a shared-terminal input is one that shares one or more of its terminals with another input. For example, in *Figure 4* ([P16](#)), the three sensors share channel 1's # terminal. Each of the three inputs is a shared-terminal input.



*Figure 4* Shared-terminal voltage inputs sharing a channel's # terminal (voltage inputs used as example)

In a shared-terminal configuration, a sensor's "return" or "negative" wire is usually connected to the channel's # terminal. The remaining sensor wire (the "positive" or "signal") is connected to any of the channel's other three terminals.

For shared-terminal inputs, the channel number is given a suffix indicating the terminal to which the positive wire is connected. For example, a shared-terminal Voltage input applied to channel **1** between the + and # terminals (*Figure 4* [P16](#))



is recognized by the channel definition [1+V](#).

### Which Analog Input Configuration Should I Use?

- **Shared input (single-ended)** wiring uses the # terminal as a common sense point between multiple sensors. Each of the +, - and - inputs are measured with respect to the # input. The main advantage of shared inputs is that the number of measurement points per channel is increased – each DT80 analog channel can measure three separate voltages.
- **Unshared (differential)** inputs do not share any measurement wires. Unshared inputs allow for easier connection to sensors where there is a **common mode voltage** (an unwanted voltage offset applied to both sensor wires relative to ground). Because unshared input pairs are totally independent from one other, different sensors can have different common mode voltages without affecting measurement accuracy.
- **Shielded** input cable may be helpful when the signal source has a high output impedance or when noise pickup (especially from power cables) is a problem. Ensure the shield is only connected to the ground at one point (see [P22](#)) – usually the logger # terminal on the same channel as the sensor.

**Important** Unshared (differential) inputs can effectively remove the unwanted common mode component from the input signals **provided that** the maximum input voltage for each terminal is not exceeded (max.  $\pm 3V/30V$  for attenuators off/on, relative to **AGND**).

### Sensor Excitation

Many sensors require **excitation** (electrical energy) so that they can provide an output signal. For example, to read the temperature of a thermistor, excitation current is passed through the thermistor to generate a voltage drop that can be measured.

The *DT80* can provide

- Voltage source of 4.5V via 1k $\Omega$ . Useful for powering some sensors however the supply is not regulated and consequently liable to drift with temperature
- 200 $\mu$ A Default current source for resistance measurement. Very stable over environmental temperature range.
- 2.5mA Default source for RTD and bridge measurement. Very stable over environmental temperature range.
- User supplied external excitation **Ext \*** terminal. ([P38](#)) The user can provide an external excitation which is appropriate to the sensor being used.

See the Excitation category in the **Table 3: DT80 Channel Options** ([P38](#)) table.

---

## Digital Channels — Introduction



The DT80 provides:

- 4 bidirectional digital I/O channels (**1D-4D**) with open drain output driver and pull-up resistor;
- 4 bidirectional digital I/O channels (**5D-8D**) with tri-stateable output driver and weak pull-down resistor (these are SDI-12 compatible, however SDI-12 operation is not currently supported by the DT80 firmware);
- 1 voltage free latching relay contact output (**RELAY**)
- 1 LED output (**Attn**)
- 4 hardware counter inputs (**1C-4C**) which can be used as independent counter channels or as two quadrature (phase encoder) inputs

As with analog channels, **channel definition commands** are used to specify which digital inputs are to be measured and/or what digital output states are to be set. For example, the command **1DS** will read the digital state (0 or 1) on channel **1D**, while **3DSO=0** will set channel **3D** low.

A transition on a digital channel can be used to trigger a schedule. This allows a series of measurements to be made (or commands executed) in response to an incoming digital pulse.

The DT80 can count the number of pulses received on any digital input. The four dedicated counter inputs provide additional capabilities

- a higher maximum count rate
- the ability to keep counting even if the logger is in low-power "sleep" mode
- optional low-level (5mV) input threshold levels
- optional decoding of phase-encoded input signals

For more details on the digital channels ([P141](#))

---

## Serial Channel – Introduction

The DT80's serial channel allows a wide variety of sensors and devices to be controlled and polled. The serial channel:

- supports RS232, RS422 and RS485 signal levels
- supports point-to-point or multi-drop operation
- features programmable output (poll) strings and a variety of options for parsing returned data
- can trigger execution of a schedule in response to received data

For more details on the serial channel ([P148](#))

# Programming the DT80

When creating a program to send to the *DT80*, typically the work will follow this order:

### Specify Channel Types

The input channels are very versatile, but the *DT80* does not automatically know what type of sensor is connected — it must be informed. A channel is defined by a **channel type** that determines how the DT80's multiplexer is set up and how the readings are to be processed. There are more than thirty different channel types (see Table 1: DT80 *Channel Types* ([P29](#))).

A particular physical channel can be read using different channel types. For example, a thermocouple can be read as a thermocouple or as a voltage. The command

`1TK 1V`

returns both a temperature and a voltage based on two readings of the same sensor.

In very general terms, when working with the *DT80*, firstly select the most appropriate **channel type** for each sensor from Table 1: DT80 *Channel Types* ([P29](#)) table. The Wiring Configuration column shows appropriate wiring configurations; connect the sensors accordingly.

### Add Channel Options

Then, use **channel options** to modify channel function. In a channel definition these are listed in round brackets immediately after the channel type. The Channel Options ([P32](#)) table describes the channel options.

### Test Each Sensor

Next, it is recommended that each sensor is tested by declaring a simple schedule. For example

`RA1S 2PT385(4W)`

returns every one second (`RA1S`) the temperature of a platinum resistance temperature sensor (`PT385`) connected as a 4-wire resistance (`4W` channel option) on channel `2`.

### Schedule Commands

Program the *DT80* by sending individual commands to it, by sending several commands all on the one line, or by sending a program.

Program the *DT80* by sending **schedules** and other commands to it from any of the following:

- a host computer
- a USB memory device (any program present on the USB memory device can be automatically downloaded to the *DT80* when the card is inserted)
- an alarm (the *DT80* can re-program itself if an alarm occurs)

Sent commands are not processed by the *DT80* until it receives a carriage return (*dataTaker* supervision software inserts this character automatically — it is not necessary to type it every time). Note the following:

- The input buffer is 254 characters, so command lines must not exceed this length.
- Each command must be separated by one or more spaces, tabs or carriage returns.
- All schedules must be entered on one line or placed between the BEGIN and END keywords.

### Schedules in More Detail

A **schedule** is a list of channels preceded by a scan trigger specification — see *Figure 6* ([P39](#)).

As a general rule when creating schedules, don't instruct the *DT80* to read channels more frequently than is really necessary. For example, temperatures generally change slowly so rapid reading does not provide extra useful information.

Up to eleven different schedules can be declared, each with a different trigger based on a time interval or a digital input event. The schedule's trigger can be changed at any time. The trigger can also be modified by the program itself (see Alarm Action Text ([P80](#))).

A list of channels without a trigger specification can be entered at any time. These are scanned immediately, without affecting other schedules that may be operating. For more information, ([P46](#))

**Important** A schedule's channel list cannot be altered without re-entering all schedules. In fact, all schedules must be entered at the same time, either all on one line or between BEGIN and END keywords (see WORKING WITH SCHEDULES [\(P48\)](#)).

## Jobs

A *DT80* job is a logical "hold-all" for a group of schedules and other commands, and related data and alarms. Each job has a directory structure that organizes these components. The command **BEGIN** signifies the start of a job, and the command **END** signifies the end of the job. A job comprises all statements beginning with and including **BEGIN**, up to and including **END**.

The *DT80* can store more than one job, but only one can be the current/active job. A job remains current in the *DT80* until

- reset the *DT80* (see RESETTING THE *DT80* [\(P119\)](#)), or
- send a new job to the *DT80*, or
- use the **RUNJOB "JobName"** command to make **JobName** the current job [\(P55\)](#)

## Scaling and Calculations

The *DT80* can scale the channel input data to engineering units by applying intrinsic functions, spans or polynomials. Arithmetic expressions provide cross-channel and other calculations. Various statistical functions, including averaging and histogram channel options, can be applied. See Channel Options — Scaling [\(P61\)](#).

## Reducing Data

In many instances the volume of the data recorded can be reduced by taking averages, maximums, minimums, standard deviations, histograms or integrals. See CHANNEL OPTIONS — STATISTICAL [\(P56\)](#).

Conditional statements can also be used to define when data is to be logged. See Trigger While [\(P44\)](#) and Alarm Condition [\(P79\)](#).

## Alarms

The *DT80*'s alarm facility is flexible and powerful. Alarms are used to warn of error conditions and to control the *DT80*'s operation. Alarms can

- allow logical comparisons with set-points
- control *DT80* digital state outputs
- initiate execution of *dataTaker* commands
- trigger the sending of messages to the host computer.

Executing *DT80* commands from an alarm can be particularly useful in modifying the *DT80*'s programming in response to changes in input(s). [\(P77\)](#)

## IFs

The *DT80*'s **IF** facility allows powerful program control. See Conditional Processing — IF... Command [\(P50\)](#).

## Data Logging

The *DT80* stores measurements in its internal data store and in removable USB memory device.

Logging begins only after you issue the **LOGON** command. Time and date stamping is automatic.

By default, the *DT80* overwrites the oldest data with new data once the memory is full. If you prefer to have the logger stop logging once the memory is full then you need to set the no-overwrite schedule option (**NOV**) — see Attn Indicator, Schedule Options.

## Selective Logging

To selectively log channels and schedules:

- For channels, use the **NL** channel option — see Disabling Data Logging for Specific Channels [\(P69\)](#).
- For schedules, use the **LOGONx** & **LOGOFFx** commands — see LOGON and LOGOFF Commands [\(P69\)](#).

## Handle With Care

**Important** The *DT80* does everything possible to avoid data loss caused by careless use. However, it does respond to resets and **DEL...** commands instantly (see Table 14 [\(P166\)](#)).

**DEL...** commands erase information without question. **DEL...** commands activate the moment they are sent to the *dataTaker* data logger, so please use with care. (See Deleting Logged Data [\(P70\)](#).)

## Retrieving Data

The *DT80* can do two things with the data it measures:

- Return it immediately to the host computer, where it can be seen arriving on-screen. This monitoring function is data return in **real time**.

- Store it in its internal memory and/or an inserted USB memory device ready for retrieval (unload) to the host computer at a later time. This is data **logging**.

The *dataTaker DT80* can carry out these functions separately, or at the same time.

### Retrieving Real-Time Data

The *DT80*'s default is to return data to a connected host computer instantaneously — that is, as it is measured. (To override this send the `/r` switch to the data logger; see `/R` ([P113](#)) Table 10([P113](#))). Store this real-time data as a file on the computer if required.

**Data Return Modes — Real-Time Data** Real-time data can be returned to the host in either free-format mode (the *DT80*'s default) or fixed-format mode. ([P21](#))

### Retrieving Logged Data

Data stored in a *DT80*'s internal memory or USB memory device can be retrieved (returned, unloaded) by means of the Host RS-232 port, the Ethernet port, or the USB port. Data can be retrieved for an individual schedule or all schedules, or for all jobs or an individual job. Here are a few useful commands when retrieving logged data:

Command	Function
<code>U</code>	begins to unload stored data
<code>A</code>	begins to unload stored alarms
<code>Q</code>	terminates unload

See

- RETRIEVING LOGGED DATA([P73](#))
- the Table 15: *DT80* Retrieval Commands — Summary ([P168](#)).

**Data Return Mode — Logged Data** The *DT80* always returns logged data to the host in fixed-format mode. See fixed-format mode. ([P21](#))

## Examples of Things You Can Do with Channels

Read a channel once. For example, sending the command

`2TT`

to the *DT80* instructs it to read channel 2 as a type T thermocouple. It returns data in the standard format [2TT 449.3 degC](#)

Read channels repeatedly. For example, sending the command

`RA1S 2..4TT`

to the *DT80* instructs it to read channels 2, 3 and 4 as type T thermocouples (`2..4TT`) every second (`RA1S`) and return data in the standard format

```
2TT 451.5 degC
3TT 563.2 degC
4TT 487.8 degC
```

```
2TT 451.9 degC
3TT 569.8 degC
```

↓

**Important** It is recommended that sensors are wired to channels. Also calibrate and test them using a schedule command such as the one above prior to entering the full program.

Change the format of the returned data. For example, sending the parameter and switch commands

`P22=44 /n/u`

to the *DT80* instructs it to change the data separator to ASCII 44, the comma, and disable channel number and units. The returned data looks like

```
452.0,565.4,451.0
452.3,566.2,450.5
```

↓

## USB memory devices

The *DT80*'s USB port supports USB memory devices, which can be used

- as removable data storage. See Logging ([P69](#))
- as a medium for transferring logged data from the internal memory of a *DT80* to a computer (see Retrieving Logged Data — USB memory device Transfer ([P73](#)))
- to upgrade a *DT80*'s operating system (see UPGRADING DT80 FIRMWARE ([P172](#)))
- to load a start-up job into a *DT80* (see Startup Job ([P116](#))).

Data is stored on the USB memory device in a Windows-compatible file structure — see Directory Structure of USB memory devices ([P72](#)).

# Format of Returned Data

The DT80 can:

- return data to a host computer as it is measured (**real-time data**), and/or
- store data in memory to be retrieved at a later date (**logged data**)

You can control whether data is returned or logged on a per channel, per schedule or global basis.

---

## Real-time data

The DT80 can be configured to return real-time data in one of two possible formats:

- free format mode
- fixed format mode (also known as "host mode", or "formatted mode")

The `/h` switch command selects free format mode (which is the default); `/H` selects fixed format mode.

### Free Format Mode /h

In free format mode, data is returned as human-readable ASCII text. Various settings are available to control how the data is presented. By default, each channel is printed on a separate line, prefixed by its name (either a standard DT80 channel name e.g. "3TK", or a user-specified name e.g. "Inlet temp") and followed by appropriate units.

Thus the following program:

```
RA30S 1V("Pressure~kPa") 2TK 5DS("Valve state")
```

would result in text similar to the following text being sent to the active communications port.

```
Pressure 102.3 kPa
2TK 98.0 degC
Valve state 1 State
```

```
Pressure 107.3 kPa
2TK 98.2 degC
Valve state 1 State
```

and so on.

By applying various formatting settings you can get different results. One possible example would be:

```
/n/c/u/T P33=10 RA30S 1V("Pressure~kPa",FF2) 2TK(FF2) 5DS("Valve state")
```

which would format the data thus:

```
12:46:00.029      102.32      97.98      1
12:46:30.017      107.34      98.22      1
```

In this example, `/n/c/u` are **switch commands** ([P112](#)) that have been used to switch off output of channel numbers, channel names and units. The `/T` switch causes each data record to be prefixed by a timestamp. `P33=10` is a **parameter setting** ([P109](#)) that sets each data value to a fixed width (10 characters). Finally, the `FF2` channel option specifies that the channel value is to be rounded to 2 decimal places.

For more details on the various options for controlling the presentation of free format data ([P38](#)).

### Fixed Format Mode /H

Fixed format mode is designed for use with *dataTaker* host software. Data is still returned in ASCII form, but the record format is **fixed** to allow it to be easily parsed by a computer. If `/H` is specified then both of the above examples will return data as:

```
D,081044,"JOB1",2005/03/29,12:46:00,0.029368,1;A,0,102.3220,97.97991,1;0071;065F
D,081044,"JOB1",2005/03/29,12:46:30,0.017032,1;A,0,107.3411,98.22000,1;0071;3BEB
```

In fixed format mode:

- all formatting commands (e.g. `FF2`, `/n`, channel names) are ignored – fixed settings are used
- all records are prefixed by a **header**, which specifies that this is a data record (D), from DT80 serial number 081044, running a job called "JOB1". This is followed a timestamp (date, time, and sub-second time). The 1 indicates that this is real-time data, the A identifies the schedule, and the 0 is the index within the schedule of the first data value.
- floating point data values are always specified to 7 significant digits
- each record includes an error-detection code (CRC) on the end. This allows host software to reject corrupted records.

Data records such as the above are only one of several types of fixed format message. A comprehensive description of all fixed format message types is beyond the scope of this manual.

---

## Logged Data

Logged data can be returned in two different formats:

- **native** format
- **fixed format** records

### **Native Format**

When the DT80 logs data to its internal memory, it stores it in fixed size data files, one for each schedule. These files have a **.DBD** file extension, e.g. **DATA\_A.DBD**.

One way of getting data out of a DT80, therefore, is to transfer relevant **.DBD** files to the host computer. These files can then be opened using tools such as *DeView*, which provides plotting facilities and can export the data in a form that can be loaded into spreadsheets.

Native format DT80 data files can be transferred to the host by:

- copying to a USB memory device. The simplest way to do this is to send the **COPYDATA** command, or select the **Copy Logged Data** option from the LCD function menu. This will take a "snapshot" of the stored data from all schedules in the current job and copy it to appropriate directories on the USB memory device.
- using FTP (File Transfer Protocol). This would involve taking a snapshot of the data using the **ARCHIVE** command, then using an FTP client program to copy the snapshot files.

The "snapshot" or "archive" files created by **COPYDATA** or **ARCHIVE** are compressed versions of the "live" data files, and have names like **2005-04-01T12-42-09.DBD**.

### **Fixed Format**

The other way to retrieve logged data is to send an **unload** command. This causes the DT80 to read the data file and output the data in the form of fixed format records (as described above). For example the command **U** will unload all data from all schedules in the current job.

For more details on unload commands, ([P74](#))

# Guidelines for Successful Data Gathering

## **The Procedure**

Data acquisition and data logging are orderly processes and should be undertaken in a systematic way. In order to obtain effective information efficiently, do the following:

- Identify the quantities to be measured.
- Select the sensors, considering measurement range, accuracy, stability, ruggedness and cost.
- Select the wiring configuration. For example, resistive sensors can be connected in 2, 3 or 4 wire configuration.
- Determine sensor output scaling, that is, the relationship between sensor output voltage/current/resistance/etc. and the actual quantity. For many sensor types this calculation is performed automatically by the DT80 – all you need to do is specify the appropriate channel type.
- Determine how data is to be processed, for example statistical functions such as max/min or histograms may be required.
- Decide on the sample frequency – don't sample faster than you need to.
- Calculate the volume of data to be collected.
- Decide on the method of data recovery and archiving – real-time data return or logging or both? Will logged data be unloaded via a comms port or collected using a USB memory device? How often?
- Consider the power consumption. If power resources are limited, low power sleep mode can be enabled.

Having defined the task, connect sensors and program the *DT80*.

## **Ground Loops**

Experience has shown that ground loops (sometimes called "earth loops") are the most common cause of measurement difficulties. Excessive electrical noise, unexpected offset voltages and erratic behaviour can all be caused by one or more ground loops in a measurement system.



## Grounds, Ground Loops and Isolation

### Grounds are Not Always Ground

Electrical grounds in a measuring system can be an elusive cause of errors.

In the real world, points in a system that one could reasonably consider at ground potential are often at different and fluctuating AC or DC potentials. This is mainly due to earthed neutral returns in power systems, cathodic corrosion protection systems, thermocouple effects in metal structures, lightning strikes and solar storms. Whatever the cause, the result can be loss of measurement integrity.

### Ground Loops

If grounds of different potential are connected by cabling used in the measuring system, ground currents flows — this is the infamous **ground loop**. The magnitude of the currents can be from milliamperes to tens of amperes, and in the case of a lightning strike, can be as high as five thousand amperes. Frequently, voltage drops along cables (caused by these current flows) are superimposed on the desired signal voltage.

A ground loop can arise when a measurement system has more than one path to ground. As *Figure 5* (P23) shows, this can be caused by

- connecting a sensor to a ground point that has a different potential to the ground of another sensor — a **sensor-to-sensor** ground loop is likely to flow through the return wires of the two sensors
- connecting the *dataTaker* to a ground point that has a different potential to the ground of one or more of the sensors or instruments connected to the *dataTaker* inputs — a **sensor-to-equipment** ground loop
- connecting the *dataTaker* to a ground point that has a different potential to the ground of the host computer — an **equipment-to-computer** ground loop.

In these situations, conduction paths can occur from one ground point to another through the sensor and/or equipment and/or computer, making measurement errors inevitable (particularly if sensor wires are part of the conduction path).

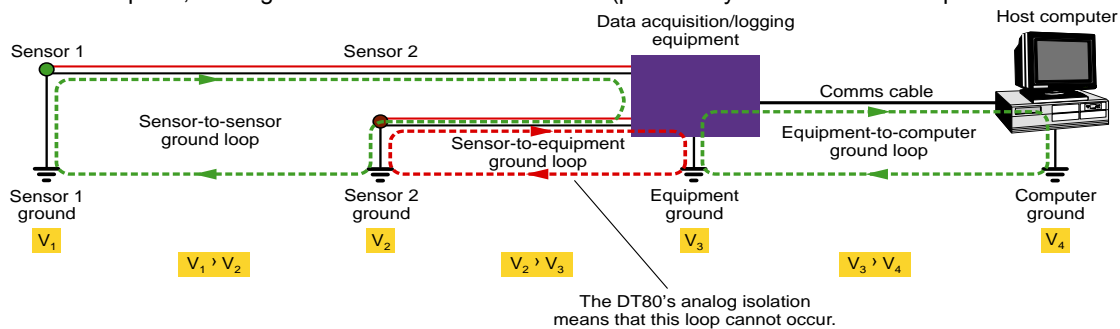


Figure 5: Some of the possible ground-loops in a measurement system

### Avoiding Ground Loops

Generally, avoidance is better than cure, so break the ground loop.

There are several strategies for minimizing the impact of ground potentials:

- Use only one system ground point.
- Ground sensors at one end of the sensor cable only. This should nearly always be at the DT80 end.
- Wire the DT80 analog sub-system making full use of its internal isolation — see DT80 Analog Sub-System (P141) and Figure 49 (P141).

### Isolation

The DT80 provides over 100V isolation of the analog sub-system from the rest of the DT80, but limits inter-channel isolation to 40V. This allows for fast multiplexing and comprehensive error detection and correction.

### DT80 Solves Ground-Loop Problems

There are three general areas of any measurement system that can give rise to ground loops (described in greater detail in Grounds, Ground Loops and Isolation (P141) — see Figure 5 (P23)). The analog section isolation built in to the DT80 removes the likelihood of ground-loop problems between sensors and the *dataTaker* data logger.

Of course, other ground-loop combinations are possible (sensor-to-computer, for example), but the DT80's isolation blocks most of these as well.

### Other Ground-Loop Solutions (23)

Many ground-loop problems can also be overcome by

- using independent inputs instead of shared-terminal inputs to remove the effects of sensor-to-sensor loops, and/or
- connecting all grounds in a measurement system to a single common point (although this is not always practical).

### Noise Pickup

There are two main ways in which noise can be introduced into signal wiring: by capacitive coupling and by magnetic induction. There are different counter-measures for each.

Shield signal wiring to minimize capacitive noise pick-up. Signal wiring that is close to line voltage cable should always be shielded.

Magnetic induction of noise from current-carrying cables or from electrical machines (especially motors and transformers) is a greater problem. Shielded cable is not an effective counter-measure. The only practical measures are to

- avoid magnetic fields
- use close-twisted conductors for the signal wiring.

Shielding in steel pipe can be effective, but is generally not economic or convenient.

### **Noise Rejection**

The *DT80* is designed to reject mains noise. For best noise rejection, set the *DT80*'s parameter 11 to your local mains frequency, 50Hz or 60Hz — see P11 ([P109](#)).

To force the *DT80* to load this parameter setting every time it restarts by using a **PROFILE** command — see Table 11: *DT80* PROFILE Details ([P115](#)) and the **PARAMETERS** ([P114](#)) section of the Table 9: *DT80* Parameters ([P112](#)) table.

### **Self-Heating of Sensors**

Sensors that need excitation power to be read are heated by power dissipation. This can be particularly acute with temperature sensors and some sensitive bridges. Minimize error by minimising the excitation power



# Part B — Channels

Define channels using channel number, channel type and channel options

## Channel Definitions

A **channel definition** defines a measurement to be taken. It is therefore the fundamental building block that you use when programming the DT80.

Channel definitions are normally enclosed in a **schedule definition**. The schedule definition specifies **when** to take the measurements. The channel definitions specify **what** to measure, on **which** terminals and **how** to sample and process the data value.

A sample schedule definition is shown below

```
RA2S 2DS 3R(4W) 2*V(0.1,GL3V,"Speed~km/h",FF0) 9CV(W)=9CV+1
```

This shows four channel definitions which are part of the "A" schedule. Each time this schedule runs (which will be every 2 seconds), four measurements will be taken:

1. The logic state of digital channel 2 will be sampled
2. A resistance connected to analog channel 3 (4-wire connection) will be measured
3. A voltage connected to analog channel 2 (\* and # terminals) will be measured and displayed as a speed value
4. An internal general purpose variable will be updated (incremented)

Let us now examine the syntax of a channel definition more closely.

A channel definition consists of up to four components

- the **channel type** is a mnemonic code which tells the DT80 what sort of quantity is being measured, or what sort of sensor is attached. In the above example the channel types are **DS** (digital state), **R** (resistance), **V** (voltage) and **CV** (channel variable). A channel definition must always include a channel type.
- a **channel number** prefix is required for most channel types. This specifies which channel to measure. In the above example we are measuring digital channel **2**, analog channel **3**, analog channel **2\*** and internal variable **#9**
- **channel options** are enclosed in round brackets after the channel type and further specify how the channel is to be measured and processed. In the above example, the **3R**, **2\*V** and **9CV** channels have user-specified options, the **2DS** channel does not.
- some channel types are "writable" (eg. internal variables and digital output channels) and therefore allow a value to be assigned using an **expression**. In the above example the **9CV** channel definition contains an expression.

## Channel Numbers

A DT80 **channel number** identifies a particular channel within a certain **class** of channels. The following table lists the various classes of DT80 channels. As can be seen, each class has its own range of channel numbers.

channel class	terminal labels	channel numbers	applicable channel types
analog	1 – 4	1 – 4 plus optional * + - # modifier	V HV I L R BGI BGV AS F T* AD5xx CU NI LMx35 LMxx PT3xx TMPxx Ysxx
digital	1D – 8D	1 – 8	C DB DBO DN DNO DS DSO
counter	1C – 4C	1 – 4 1 – 2	HSC PE (1PE uses terminals 1C-2C, 2PE uses 3C-4C)
relay	RELAY	1	RELAY
LED	Attn	1	WARN
serial		1	SERIAL
channel variable	internal	1 – 500	CV
system variable	internal	1 – 53	SV
string	internal	1 – 10	\$

timer	internal	1 – 4	ST
special	internal	no number	D T DELAY CMRR IBAT R100 REFT VANA VBAT VC VDD VEXT VLITH VREF VRELAY VSYS VZERO

The "applicable channel types" column lists the different ways in which a physical channel can be measured. For example, analog channel 1 can be used to measure a voltage (specified by entering **1V**), or a PT385 RTD (**1PT385**) or a frequency (**1F**). All of these **channel types** fall into the analog class, so when we talk about channel 1 we are talking about **analog** channel 1.

Because each channel type is a member of one class only, there is never any confusion about which of the channel 1s is being referred to. **1C** refers to **digital** input 1 because, from the above table, the **C** (counter) channel type is in the digital class. **1HSC**, on the other hand, refers to **counter** input 1 because the **HSC** (high speed counter) channel type is in the counter class.

An **analog** channel number can be suffixed by a **modifier** character, which identifies the pair of terminals between which to measure, as shown in the following table:

Modifier	Measure voltage between
none	+ and -
*	* and #
+	+ and #
-	- and #
#	# and AGND (normally only used for current measurements)

Thus the channel ID **3V** defines an independent input between the + and – terminals, while **3\*V**, **3+V** and **3-V** define shared-terminal inputs between the \*, + or – terminals (respectively) and the # terminal.

## Channel Number Sequence

A channel ID that contains two channel numbers separated by two decimal points (for example, **1..3**) defines a **continuous sequence** of channels. If the first channel ID indicates a shared channel, the remaining channels in the sequence follow the order \*, +, – then # (where valid for the channel type). For example

Sequence	is equivalent to
<b>1..4V</b>	<b>1V 2V 3V 4V</b>
<b>1+..3-I</b>	<b>1+I 1-I 1#I 2*I 2+I 2-I 2#I 3*I 3+I 3-I</b>
<b>1+..3-R(3W)</b>	<b>1+R(3W) 1-R(3W) 2+R(3W) 2-R(3W) 3+R(3W) 3-R(3W)</b>

# Channel Types

The following table lists all of the channel types supported by the DT80. For each channel type, the table shows:

- the **channel type** mnemonic (eg **HV**). Remember that in most cases this will be prefixed by a channel number. Refer to Channel Numbers ([P25](#)) for details of the allowable range of channel numbers for each channel type.
- whether the channel type is "**writable**" (shown in the Channel Type column). Writable channel types can be assigned a value, eg. **2C=200**.
- the **default channel options** for this channel type. These override the standard default values shown in the channel option table. See also Channel Options ([P32](#)).
- what the **channel factor** does for this channel type
- the **units** in which data will be returned. By default, the indicated units string will be shown on the display and appended to free format returned data, although it can be overridden if required.
- references to typical **wiring configuration diagrams** for the channel type

Category	Signal Sensor Details	Channel Type	Default Channel Options	Channel Factor	Output Units	Wiring config / Comments
Voltage	Voltage input ranges are ±3V, ±300mV & ±30mV	<b>V</b>	<b>(T)</b> <i>Note 1</i>	scaling factor	mV	Figure 54: <b>V1</b> Wiring for shared-terminal voltage input ( <a href="#">P158</a> ) & Figure 55: <b>V2</b> Wiring for independent voltage input. ( <a href="#">P158</a> )
	Higher Voltage input ranges are ±30V, ±3V & ±300mV	<b>HV</b>	<b>(A)</b>	scaling factor	V	
Current	Current	<b>I</b>	<b>(100,T)</b>	current	mA	Figure 56: <b>C1</b> wiring for

			Note 1, 2	shunt $\Omega$		independent current input using external shunt ( <a href="#">P 159</a> ) Figure 57: <b>C2</b> Wiring for Independent current input using internal shunt ( <a href="#">P159</a> ) Figure 58: <b>C3</b> Wiring for shared-terminal current input using external shunt ( <a href="#">P159</a> ) Figure 59: <b>C4</b> Wiring for independent current input using internal shunt and external excitation ( <a href="#">P160</a> ) Internal 100 $\Omega$ shunt is between # and AGND terminals. Other inputs require an external current shunt (typically 100 $\Omega$ , but higher for small currents)
	4-20mA current loop	<b>L</b>	<b>(100, A)</b> Note 2	current shunt $\Omega$	%	
Resistance	Resistance by 2, 3 or 4-wire methods, 10K $\Omega$ maximum.	<b>R</b>	<b>(I, 3W)</b>	offset adjust $\Omega$ Note 4	Ohm	Figure 60: <b>R1</b> Wiring for 4-wire resistance input ( <a href="#">P160</a> ) Figure 61: <b>R2</b> Wiring for 3-wire resistance input ( <a href="#">P160</a> ) Figure 62: <b>R3</b> Wiring for 2-wire resistance input
Bridge See Bridges ( <a href="#">P138</a> )	3 & 4-wire; quarter, half & full bridge; current excitation	<b>BGI</b>	<b>(350, II, 3W)</b>	arm resistance $\Omega$	ppm	Figure 63: <b>B1</b> Wiring for 6-wire bridge using external voltage excitation ( <a href="#">P161</a> ) Figure 64: <b>B2</b> Wiring for 4 wire bridge input using internal excitation ( <a href="#">P162</a> ) Figure 66: <b>B3</b> Wiring for 3 wire bridge input using internal current excitation ( <a href="#">P162</a> ) External completion required for 1/2 & 1/4 bridges. For BGV, must measure reference voltage using another channel with ( <b>BR</b> ) option, otherwise defaults to 5.0 V
	Ratiometric, 4 & 6-wire bridges, voltage excitation	<b>BGV</b>	<b>(V, 4W)</b>	offset adjust ppm Note 4	ppm	
Frequency	Frequency measurement	<b>F</b>	<b>(30, T)</b> Note 1	sample period ms	Hz	Figure 54: <b>V1</b> Wiring for shared-terminal voltage input ( <a href="#">P158</a> ) & Figure 55: <b>V2</b> Wiring for independent voltage input. ( <a href="#">P158</a> ). Threshold is 0V. Use ( <b>2V</b> ) option to set threshold to 2.5V.
Time, Date and System Timers	Time of day	<b>T</b> writable				Specified in current time/date format
	Day or date	<b>D</b> writable				
	System timers See System Timers ( <a href="#">P31</a> ).	<b>1ST</b> <b>2ST</b> <b>3ST</b> <b>4ST</b> writable	<b>(60)</b> <b>(60)</b> <b>(24)</b> <b>(7)</b>	range	Counts	Increment every sec ( <b>1ST</b> ), min ( <b>2ST</b> ), hour ( <b>3ST</b> ), day ( <b>4ST</b> )
Delay	Delays schedule execution for nominated time	<b>DELAY</b> writable			ms	Note 6
System Data	System variable	<b>SV</b> some are writable				Returns See System Variables ( <a href="#">P31</a> ).
Variables	Channel variables: general purpose holders for data, calculation results	<b>CV</b> writable		scaling factor		Use channel options (Variables ( <a href="#">P37</a> )) to assign data to a CV. Read the CV as for a normal channel.
	Integer variables	<b>IV</b>		scaling		

	See Rainflow Cycle Counting <a href="#">(P58)</a> .			factor		
Text	General purpose text for headings, etc. (ten 80-character channels)	<b>\$</b> <i>writable</i>				Assign by sending <b>n\$="my text"</b> See Text <a href="#">(P30)</a> .
Serial Channel	Transmit to and receive from serial device via RS-232, RS422 & RS485	<b>SERIAL</b>	<b>(P53)</b> ie. parameter P53 specifies default timeout	timeout (sec)	State	Figure 55 <a href="#">(P157)</a> See SERIAL CHANNEL <a href="#">(P148)</a> for prompt and scan definition. Use <b>W</b> or <b>NR</b> channel options to prevent state output.
Serial Channel Enable	Enable/disable serial sensor port	<b>SSPORT</b> <i>writable</i>			State	Power supply for transceiver 0=disabled, 1=enabled Automatically enabled if any serial sensor channels or schedule triggers defined
Temperature	Thermocouples Type B, C, D, E, G, J, K, N, R, S and T See Thermocouples <a href="#">(P134)</a> .	<b>TB, TC, TD, TE, TG, TJ, TK, TN, TR, TS, TT</b>	<b>(T)</b> <i>Note 1</i>	scaling factor	degC <i>Note 3</i>	Figure 54: <b>V1</b> Wiring for shared-terminal voltage input <a href="#">(P158)</a> & Figure 55: <b>V2</b> Wiring for independent voltage input. <a href="#">(P158)</a>
	Platinum RTDs ( $\alpha = 0.00385, 0.00392$ ) See RTDs <a href="#">(P137)</a>	<b>PT385</b> <b>PT392</b>	<b>(100, 3W, II)</b>	0°C resistance $\Omega$	degC <i>Note 3</i>	Figure 60: <b>R1</b> Wiring for 4-wire resistance input <a href="#">(P160)</a> Figure 61: <b>R2</b> Wiring for 3-wire resistance input <a href="#">(P160)</a> Figure 62: <b>R3</b> Wiring for 2-wire resistance input
	Nickel RTD ( $\alpha = 0.005001$ ) See RTDs <a href="#">(P137)</a>	<b>NI</b>	<b>(1000, 3W, I)</b>	0°C resistance $\Omega$	degC <i>Note 3</i>	
	Copper RTD ( $\alpha = 0.0039$ ) See RTDs <a href="#">(P137)</a>	<b>CU</b>	<b>(100, 3W, II)</b>	0°C resistance $\Omega$	degC <i>Note 3</i>	
	Thermistors: Yellow Springs 400XX series See Thermistors <a href="#">(P136)</a> .	<b>YS01</b> <b>YS02</b> <b>YS03</b> <b>YS04</b> <b>YS05</b> <b>YS06</b> <b>YS07</b> <b>YS16</b> <b>YS17</b>	<b>(3W, I)</b>	parallel resistor $\Omega$	degC <i>Note 3</i>	
	Semiconductor current source types (Analog Devices)	<b>AD590</b> <b>AD592</b> <b>TMP17</b>	<b>(100, V, U)</b> <i>Note 2</i>	current shunt $\Omega$	degC <i>Note 3</i>	Figure 67: <b>A1</b> Wiring for AD590 series input using internal shunt <a href="#">(P163)</a> Calibrate by variation of shunt value channel factor.
	Semiconductor (zener diode) voltage output types (National Semiconductor Corp.)	<b>LM135</b> <b>LM235</b> <b>LM335</b>	<b>(2, V)</b>	scaling factor	degC <i>Note 3</i>	Figure 70: <b>L3</b> Wiring for LM135 series input <a href="#">(P164)</a> Calibrate using scaling factor relative to 0 Kelvin. Default scaling factor is 2 to suit external voltage divider.
	Semiconductor voltage output types (National Semiconductor Corp., Analog Devices)	<b>LM34</b> <b>LM35</b> <b>LM45</b> <b>LM50</b> <b>LM60</b> <b>TMP35</b> <b>TMP36</b> <b>TMP37</b>	<b>(V)</b>	offset adjust degC <i>Note 4</i>	degC <i>Note 3</i>	Figure 68: <b>L1</b> for LM35 series input – full temperature range <a href="#">(P163)</a> , Figure 69: <b>L2</b> Wiring for LM35 series input – restricted temperature range <a href="#">(P163)</a> Offset adjustment is always in degC
Digital	Digital state input (1 bit)	<b>DS</b>			State	Result is 0 (low) or 1 (high) Max channel number = 8
See Digital Channels <a href="#">(P141)</a> .	Digital nybble input (4 bits)	<b>DN</b>	<b>(15)</b>	bit mask <i>Note 5</i>	Nybble	Result is 0 to 15. Channel number = LSB of nybble. Max channel number = 5

& Figure 71 Digital Input Wiring <a href="#">(P164)</a> & Figure 72 Digital Output Wiring <a href="#">(P165)</a>	Digital byte input (8 bits)	<b>DB</b>	<b>(255)</b>	bit mask <i>Note 5</i>	Byte	Result is 0 to 255. Channel number = LSB of byte. Max channel number = 1.
	Digital state input on an analog channel	<b>AS</b>	<b>(2500)</b>	threshold (mV)	State	Result is 0 (<threshold) or 1 (>threshold)
	Output on a single digital channel.	<b>DSO</b> <i>writable</i>		delay (ms) <i>Note 6</i>	State	<b>nDSO=0</b> : output low <b>nDSO=1</b> : output high
	Nybble output on a group of digital channels	<b>DNO</b> <i>writable</i>	<b>(15)</b>	bit mask <i>Note 5</i>	Nybble	Channel number = LSB of byte. Max channel number = 5.
	Byte output on a group of digital channels	<b>DBO</b> <i>writable</i>	<b>(255)</b>	bit mask <i>Note 5</i>	Byte	Channel number = LSB of byte. Max channel number = 1.
	Pulse count on digital input	<b>C</b> <i>writable</i>		range	Counts	Max count rate 30Hz Not active during sleep Counter resets after <b>range</b> counts, if set
Counter Figure 71 Digital Input Wiring <a href="#">(P164)</a>	High Speed Up Counter	<b>HSC</b> <i>writable</i>		range	Counts	Max count rate 100kHz Active during sleep Counter resets after <b>range</b> counts, if set
Relay Figure 71 Digital Input Wiring <a href="#">(P164)</a> <a href="#">(P144)</a>	Relay Output	<b>RELAY</b> <i>writable</i>		delay (ms) <i>Note 6</i>	State	<b>1RELAY=0</b> : open <b>1RELAY=1</b> : closed Relay is latching type, so it only draws current when changing state.
Attention LED Figure 71 Digital Input Wiring <a href="#">(P164)</a>	Activate <b>Attn</b> LED <a href="#">(P125)</a>	<b>WARN</b> <i>writable</i>		delay (ms) <i>Note 6</i>	State	<b>1WARN=0</b> : LED off <b>1WARN=1</b> : LED on <b>Attn</b> LED may also be used by the DT80 to indicate various warning conditions
Internal Maintenance	Reference temperature of terminal block (the <i>DT80</i> 's body temperature)	<b>REFT</b>			degC	Used for thermocouple reference junction compensation
	Terminal voltage of internal 6V lead acid battery	<b>VBAT</b>			V	Battery flat if below 5.6V.
	Internal Lithium memory-backup battery voltage.	<b>VLITH</b>			V	Replace battery if below 2.8V.
	Internal main battery current	<b>IBAT</b>			mA	Positive if charging, negative if discharging
	Analog 2.5V voltage source reference	<b>VREF</b>			V	
	Analog zero voltage reference	<b>VZERO</b>			mV	
	Internal 100 Ohm Shunt	<b>R100</b>			Ohms	
	Internal analog 3.8V rail voltage	<b>VANA</b>			mV	
	Internal 3.3V rail voltage	<b>VDD</b>			mV	
	Internal system supply rail voltage	<b>VSYS</b>			mV	
	Internal relay supply voltage	<b>VRELAY</b>			mV	
	Raw voltage onto system from external supply	<b>VEXT</b>			V	
	Common-mode rejection ratio at maximum gain	<b>CMRR</b>				dB

Table 1: DT80 Channel Types

## Notes

- Input termination is on by default (**T**) for **independent (differential) inputs only**. For shared inputs the # terminal is

connected to **AGND** via an internal 100 ohm resistor, so the 1M Ohm termination used for differential measurements is not required

2. If the current shunt value is specified (as the channel factor) then that value is used. Otherwise, if the measurement uses the DT80's internal shunt on the # terminal (eg. **3#I**), then the DT80 uses the actual calibrated resistance of its shunt. Otherwise, the external shunt is assumed to be 100.0 ohms.
3. Alternatively, parameter P36 can be set to force all temperatures to be returned in `degF`, `degR` or `K`.
4. Offset corrections are subtracted from the measured value.
5. The bitmask specifies which channels are affected by a multi-bit read or write. Channels where the corresponding bitmask bit is zero are not affected. For example **1DNO(3)=0** will set digital outputs **1D** and **2D** low but the state of outputs **3D** and **4D** will be unchanged.
6. The `delay` channel factor can be used in conjunction with the **R** channel option to generate a fixed width pulse output.

**Note:** use `delay` carefully as it prevents execution of any other schedules, measurements or outputs during the delay.

## Internal Channel Types (in detail)

The *DT80* has its own internal channels, which can be read in exactly the same way as the obvious “external” channels. Use the channel types below.

### Time & Date

The *DT80*'s real-time clock/calendar has a resolution of 122µs, based on a 24-hour clock. Time is read in the same way as any channel, but without a channel number. That is, sending

```
T
returns
Time 11:45:10.213
```

This channel type is writable, so you can set the time by sending:

```
T=12:20:00
```

Time can be in several formats, selected by parameter P39 as follows:

P39=	Format	Example
0 (default)	Hours:minute:seconds.seconds P41 controls the number of sub-second digits between 0 and 6; default is 3 digits	11:45:10.003
1	s.s (decimal seconds) since midnight	42310.003
2	m.m (decimal minutes) since midnight	705.1667
3	h.h (decimal hours) since midnight	11.7528

The current date can also be returned:

```
D
Date 22/05/2005
```

Date can be in several formats, selected by P31 as follows:

P31=	Format	Example
0	Day number	DDDD 86
1 (default)	European	DD/MM/YYYY 28/03/2002
2	North America	MM/DD/YYYY 03/28/2002
3	ISO	YYYY/MM/DD 2002/03/28

System variable **12SV** returns the combined day.time as decimal days. System variable **15SV** returns the day of the current year.

See also [Setting the DT80's Clock/Calendar](#), and [Efficient Storage of Time and Date](#).

### Text

Ten 80-character text channels (**1\$ – 10\$**) are available for labelling, data headings, site identification, *DT80* identification, and so on.

Define the string by sending, for example

```
2$="my text string^M^J"
```

Then, the string is returned (unloaded) whenever **n\$** is included in a channel list.

Text channels can also be set based on data returned via the serial channel. Control String – Input Actions ([P152](#))

Control characters may be included in the text string, eg. **^M** for carriage return.

### Internal Maintenance

There are several internal maintenance channels, which are read in the same way as normal channels. These allow, for



example, the terminal voltage of the DT80's internal batteries to be measured. See the [Internal Maintenance](#) section of the DT80 Channel Types table.

## System Timers

There are four internal reloading system timers, which are read in the same way as channels. The four timers increment at the following rates, and reset to zero when their range (maximum value) is reached:

System Timer	Channel Type	Increments Every	Default range	Provides
1	1ST	1 second	60 (1 minute)	Second of the minute
2	2ST	1 minute	60 (1 hour)	Minute of the hour
3	3ST	1 hour	24 (1 day)	Hour of the day
4	4ST	1 day	7 (1 week)	Day of the week: <b>0</b> Sunday <b>1</b> Monday <b>2</b> Tuesday <b>3</b> Wednesday <b>4</b> Thursday <b>5</b> Friday <b>6</b> Saturday

System timers are normally synchronised to the previous midnight or Sunday, and increment at the beginning of each second, minute, hour or day.

If the DT80's date/time is set, the system timer channels will be updated to match the new time.

The range of a system timer can be set using the channel factor. For example, **2ST(15)** will count from 0 to 14, resetting every quarter hour, on the quarter hour.

If the range is set to 0 then the timer will not reset, except at midnight (1-3ST) or midnight Sunday (4ST)

If a system timer is explicitly set to a value, eg. **1ST=12**, then it will no longer necessarily be synchronised to the actual time. In this example, after being set 1ST will count up from 12 to 60, at which point it will reset back to 0 and start counting again. It will always differ from the time-of-day seconds count by a fixed offset.

If a system timer's range is set, it will automatically be resynchronised to the actual time. Therefore **2ST(60)** can be entered at any time to return 2ST to its default behaviour.

If a system timer is set to a value outside its range, it is immediately adjusted so that it is in range. When you enter **nST=x**, you are actually doing **nST=x mod range**. Thus **2ST=62** will actually set 2ST to 2.

### Examples

Assume the time is now 12:34:56. Then:

**2ST**

**2ST 34.0** (34 minutes past the hour – counter resets on the hour)

**2ST(0)**

**2ST 754.0** (754 minutes since midnight – counter resets at midnight only)

**2ST(22)**

**2ST 6.0** (754 mod 22 – counter resets at midnight and every 22 minutes thereafter)

**2ST=1**

**2ST 1.0** (counter is no longer synchronised to midnight)

**2ST(22)**

**2ST 6.0** (setting range value resynchronises timer to current time)

2ST will now increment every minute, resetting back to 0 each time it reaches 22. When midnight comes around, it will again be reset to 0.

## System Variables

System variables provide various pieces of information about the state of the DT80 and its current job. All system variables are read-only except where indicated as **writable** in the table below.

System Variable	Function	Writable
<b>1SV</b>	Returns kB free in internal memory	
<b>2SV</b>	Returns kB stored in internal memory	
<b>3SV</b>	Returns kB free in USB memory device (0 if no memory device inserted)	
<b>4SV</b>	Returns kB free in USB memory device (0 if no memory device inserted)	
<b>6SV</b>	Returns build number of the DT800's firmware (see also <b>14SV</b> and <a href="#">version number</a> )	
<b>7SV</b>	Returns job presence	= 0 if no current job = 1 if a job is loaded

<b>8SV</b>	Returns current mains frequency setting in Hz (P11)		<input checked="" type="checkbox"/>
<b>9SV</b>	Returns USB memory device presence	= 0 if none = 1 if USB memory device inserted	
<b>10SV</b>	Returns ID of the owning schedule	= 0 for RX schedule = 1 for RA schedule = 2 for RB schedule ↓ = 11 for RK schedule = 12 for immediate schedule	
<b>11SV</b>	Returns 0.0	Can be used as thermocouple reference channel for cases where the thermocouple output is already compensated, eg. <b>RA1S 11SV(TR) 1TT</b>	
<b>12SV</b>	Returns decimal days since base date (1-Jan-1989) Use formatting for more precision — for example, <b>12SV(FF4)</b>		
<b>13SV</b>	Returns DT800's serial number		
<b>14SV</b>	Returns version number (major.minor) of the DT800's firmware (see also <b>6SV</b> and <a href="#">version number</a> ). Use <b>14SV(FF2)</b> to see all decimal places in version number		
<b>15SV</b>	Returns date as day number of the current year (zero =1st of January)		
<b>16SV</b>	Returns Host RS-232 port input handshake line states	Bitmask, 0 to 15 8 = RI, 4 = DCD, 2 = DSR, 1 = CTS	
<b>17SV</b>	Returns Host RS-232 port output handshake line states	Bitmask, 0 to 3 2 = DTR, 1 = RTS	<input checked="" type="checkbox"/>
<b>18SV</b>	Returns Serial Channel input handshake line states (RS232 mode only)	Bitmask, 0 to 1 1 = CTS	
<b>19SV</b>	Returns Serial Channel output handshake line states (RS232 mode only)	Bitmask, 0 to 1 1 = RTS	<input checked="" type="checkbox"/>
<b>25SV</b>	Returns current status of a modem connected to the DT80's Host RS-232 port — see <a href="#">DT80 Modem (Remote) RS-232 Connection</a>	0 =no modem connected (direct connection assumed) 1 =modem connected and no call in progress 2 =modem connected and call in progress	
<b>30SV</b>	Number of logged data records for current job, RX schedule		
<b>31SV</b>	Number of logged alarm records for current job, RX schedule		
<b>32SV</b>	Number of logged data records for current job, RA schedule		
<b>33SV</b>	Number of logged alarm records for current job, RA schedule		
<b>34SV</b>	Number of logged data records for current job, RB schedule		
<b>35SV</b>	Number of logged alarm records for current job, RB schedule		
	↓		
<b>52SV</b>	Number of logged data records for current job, RK schedule		
<b>53SV</b>	Number of logged alarm records for current job, RK schedule		

Table 2: DT80 System Variables

# Channel Options

In brackets, separated by commas, no spaces

## Overview

All channel types can be modified in various ways by **channel options**, which define the way in which the input channel is managed when sampled. There are channel options that specify the type of sensor excitation, the termination of the input channel, scaling and linearization of the input signal, the format and destination of channel data, fixed channel gain values, resistance and bridge wiring methods, statistical operations on the channel data, and so on.

As shown below, channel options are placed in round brackets immediately following the channel ID (channel number and type). If multiple channel options are specified then they should be separated by a comma (no spaces).

**RA2S 1TK 3R(4W) 2\*V(0.1,GL3V,"Speed~km/h",FF0)**

In the above example:



- The first channel, **1TK**, has no channel options specified so it will measure the thermocouple using default settings.
- The second channel (**3R**) includes the **4W** channel option, which specifies that a 4-wire resistance measurement should be taken.
- Finally, the **2\*V** channel is in this case used to read a speed sensor which outputs a voltage that is directly proportional to speed (10mV per km/h). The **0.1** channel option is the **channel factor**, which for a voltage channel is interpreted as a simple scaling factor (mV \* 0.1 = km/h). The **GL3V** (gain lock) option tells the DT80 to select the 3V measurement range (rather than auto-ranging). The last two options concern the presentation of the data on the LCD display and in returned real-time data when in free format (**/h**) mode. In particular, they define the channel name and units, and specify that no decimal places be displayed (**FF0**).

The channel's data will therefore be returned/displayed as:

Speed 72 km/h

instead of the default:

2\*V 721.3 mV

Only certain channel options can be applied to each channel type. If an inappropriate channel option is applied (or an incompatible combination of options), the DT80 notifies by returning an **E3 - Channel option error** message.

The same channel can be put in the list more than once, with the same or different channel options. The DT80 treats each occurrence as a separate measurement.

## A Special Channel Option — Channel Factor

The DT80's **channel factor** channel option is simply a floating point number. This number is interpreted in different ways depending on the channel type, as indicated by the following table.

Channel Type	Channel Factor's function
<b>V, HV, Tx, LMx35, CV</b> (voltage/variable)	Scaling factor – for example, <b>1V(5.5)</b> means multiply the reading by <b>5.5</b>
<b>I, L, AD59x, Tmpl17</b> (current)	Resistance (ohms) of the external current shunt (the DT80 uses this value and the voltage it measures across the shunt to calculate current flow)
<b>BGI</b> (current excited bridge)	Bridge arm resistance (ohms)
<b>BGV</b> (voltage excited bridge)	Offset adjustment (ppm)
<b>PT3xx, NI, CU</b> (RTD)	Resistance of the RTD element at 0°C (ohms)
<b>YSxx</b> (thermistor)	Value of connected parallel resistance (ohms)
<b>R</b> (resistance)	Offset adjustment (ohms)
<b>AS</b> (state)	Logic threshold value (mV)
<b>F</b> (frequency)	Sample Period (ms)
<b>C, HSC, STx</b> (counter, timer)	Count modulo value (reset after every n counts)
<b>DN, DB, DNO, DBO</b> (digital multiple)	Bitmask (only channels with 1 in bitmask are read/output)
<b>DSO, WARN, RELAY, SSPORT</b> (digital output)	Delay time (ms)
<b>SERIAL</b>	Timeout (sec)

For example, the three channel definitions in the schedule command

```
RA30S 1V(10.1) 4PT385(200.0) 2DSO(100,R)=0
```

contain channel factor channel options that instruct the DT80 to do the following:

- scale (multiply) the voltage measured on input channel 1 by **10.1**
- use **200.0Ω** (instead of the default 100.0Ω at 0°C) when calculating the temperature represented by the signal from the RTD on channel 4
- output a **100ms** pulse on digital channel 2.

## Multiple Reports

The DT80 samples each channel in the channel list once every scan. However, by adding additional **channel option sets** (each set enclosed in round brackets) you can generate additional reports. That is, you can report the same data value in different ways.

The first channel option set determines how the channel is sampled, and must include all sampling options required for the channel. These channel options are listed above the **configuration line** in the [Table 3: DT80 Channel Options](#) table. Second and subsequent option sets may only contain reporting options (those below the configuration line).

Multiple reports are particularly useful for statistical reports (see [Statistical Report Schedules](#)) in that several different statistical operations can be performed on the same data set.

For Example:

```
RA1H 3YS04(II,AV)(MX)(TMX)(MN)(TMN)
```

defines five option sets. The first option set specifies one sampling option (**II** – use 2.5mA excitation) and returns the average temperature value, calculated over the period (1 hour in this case) since the last report scan. The remaining option sets will return the maximum reading over the same interval, the time at which it occurred, the minimum and the time of minimum.

Remember that the first option set can contain options from any part of the channel option table, while subsequent option sets can only contain options from below the [Configuration Line](#).

---

## Mutually Exclusive Options

Options grouped by a shaded rectangle in the Mutual Exclusions column of the table below are **mutually exclusive**. If more than one channel option from a mutual exclusion group is placed in a channel list, only the **last** one specified is recognised.

---

## Order of Application

The DT80 applies channel options in a specific order, regardless of the order in which they are specified in a channel definition. The channel option table below lists the channel options more or less in the order of application.

In general terms, the ordering is as follows:

1. First, the raw value is sampled, taking note of **sampling options**, ie. those relating to the physical measurement process. These include options in the input termination (**T**, **U**), input attenuator (**A**, **NA**), resistance/bridge wiring (**3W**, **4W**), gain lock (**GL30V**, **GL3V**, **GL300MV**, **GL30MV**) and excitation (**I**, **II**, **V**, **E**, **N**) categories, along with **NSHUNT**, **2V**, **ES<sub>n</sub>** and **MD<sub>n</sub>**.  
The raw value may then be linearised according to the channel type (e.g. for thermocouples the appropriate polynomial will be applied). The resulting linearised value is then further processed as follows.
2. The **channel factor** is then applied, if specified. For most channel types this is a simple scaling (multiplier) value.
3. A **user specified scaling** option – a span (**S<sub>n</sub>**), polynomial (**Y<sub>n</sub>**), thermistor scaling (**T<sub>n</sub>**) or intrinsic function (**F<sub>n</sub>**) – is then applied.
4. The resulting scaled and linearised value may then be manipulated using a **data manipulation** option – difference (**DF**), time difference (**DT**), rate of change (**RC**), reading per time (**RS**) or integrate (**IB**).
5. A **digital manipulation** option for measuring the timing of signal transitions may then be applied (**TRR**, **TRF**, **TFR**, **TFF**, **TOR** or **TOF**)
6. The **RAINFLOW** option may then be applied.
7. The data value processed up to this point may then be accumulated using one or more **statistical** options (each one in a separate option set). Statistical channel options include **AV**, **SD**, **MX**, **MN**, **TMX**, **TMN**, **DMX**, **DMN**, **IMX**, **IMN**, **INT**, **NUM** and **H** (histogram).
8. Finally, the resultant value after applying the above options (or values if multiple option sets are used) may be stored in a channel variable using **=CV** and **op=CV** options. Return, logging and/or display of the data may be disabled using the **NR**, **NL**, **ND** and **W** options, and output formatting can be specified using **FF<sub>n</sub>**, **FE<sub>n</sub>** and **FM<sub>n</sub>** and **"name~units"**.

---

## Default Channel Options

All channel options have default values. The DT80 follows a 3-step procedure to determine what options to apply:

1. Start with the basic set of default options specified in the channel option table.
2. If the **channel type** specifies any default options then they are applied, overriding any conflicting basic default options. Default options for each channel type are listed in the channel type table refer Table 1: DT80 Channel Types ([P29](#)).
3. Finally, if an option is explicitly specified in the channel definition then that setting is used, overriding any default setting. If more than one **mutually exclusive** option is specified then only the **last** one is used, e.g. **1V(AV, MX)** is interpreted as **1V(MX)**. (If you want to output both the average and the maximum then use two separate option sets, i.e. **1V(AV)(MX)**.)

For example, if you specify:

**1V(GL3V)**

then you are really specifying:

**1V(U, NA, N, ES0, MD10, FF1, T, GL3V)**

In this case the basic default options are (**U, NA, N, ES0, MD10, FF1**). The **V** channel type specifies (**T**) as its default option, which overrides the (**U**) option. Then the user specifies (**GL3V**) which overrides the default gain lock option setting (auto).

## Channel Option Table

Category	Channel Option	Mutual Exclusions	Function	Range of Option (n)	Comment
Input Termination	<b>T</b>		Terminate +, – inputs with 1MΩ to <b>AGND</b> terminal		Provides input bias current path to ground to prevent inputs "floating" – particularly when independent (differential) inputs are used.
	<b>U</b> <i>default</i>		Unterminate +, – inputs		
Input Attenuators	<b>A</b>		Enable ±10 input attenuators		Attenuators default ON for <b>HV</b> , <b>L</b> channel types, OFF for other types. Attenuators cannot be used if the DT80 is supplying excitation.
	<b>NA</b> <i>default</i>		Disable input attenuators		
Resistance and Bridge	<b>3W</b>		3-wire measurement		Specifies the number of wires run between the DT80 and the resistance or bridge. More wires generally mean better accuracy.
	<b>4W</b>		4-wire measurement		
Gain lock	(none) <i>default</i>		Auto-range over 3 gain ranges		Selects between 3V, 300mV, 30mV ranges if input attenuators disabled Selects between 30V, 3V, 300mV ranges if input attenuators enabled
	<b>GL30V</b>		Lock channel gain for ±30V input signal range		Valid only if input attenuators are enabled
	<b>GL3V</b>		Lock channel gain for ±3V input signal range		
	<b>GL300MV</b>		Lock channel gain for ±300mV input signal range		
	<b>GL30MV</b>		Lock channel gain for ±30mV input signal range		Valid only if input attenuators are disabled
Excitation	<b>I</b>		Supply 250µA current excitation on * terminal		Precision current source. Low excitation current minimises self-heating in resistive temperature sensors
	<b>II</b>		Supply 2.5mA current excitation on * terminal		Precision current source. Higher excitation current extends measurement range when measuring large resistances
	<b>V</b>		Supply approx. 4.5V voltage excitation on * terminal		Voltage source is not regulated
	<b>E</b>		Connect external excitation source ( <b>EXT</b> * terminal) to channel's * terminal		
	<b>N</b> <i>default</i>		No excitation by <i>DT80</i> (assumes externally applied excitation)		Excite terminal may be used as a shared-terminal input channel
Internal Shunt	<b>NSHUNT</b>		Disconnect internal 100R shunt between # terminal and <b>AGND</b>		Allows # terminal to be used for shared-input voltage measurements
Reference Offset	<b>2V</b>		Measure relative to 2.5V rather than 0V		Used with <b>F</b> channel type to set threshold to +2.5V (suitable for TTL level input signals) rather than the default of 0V.
Extra Samples	<b>ESn</b> <i>default = 0</i>		Perform <i>n</i> additional samples and average them	0 to 65535	Can reduce noise. Total measurement time is <i>n</i> +1 mains periods.
Measurement Delay	<b>MDn</b> <i>default = 10</i>		After selecting channel, delay for <i>n</i> ms before starting measurement	0 to 65535	Specifies the settling time required before a sensor can be measured. Default is 10ms.
Reset	<b>R</b>		Reset channel after reading		Valid for <b>C</b> , <b>HSC</b> , <b>ST</b> and <b>CV</b> channel types, which are reset to 0 after returning their current value. Also valid for digital output channel types ( <b>DSO</b> , <b>DNO</b> , <b>DBO</b> ) which invert the state of each bit after returning its value.
Channel factor	<b>f . f</b>		Linearise/scale the measured value	depends on chan type	A scale factor or other parameter specific to channel type (see the channel factor column in Table 1: DT80 Channel Types ( <a href="#">P29</a> ))

Scaling	<b>Sn</b>	Apply span <b>n</b>	1 to 50 (poly & span index is shared)	Applies a previously-defined span See Spans (Sn) <a href="#">(P62)</a>																
	<b>Yn</b>	Apply polynomial <b>n</b>		Applies a previously-defined polynomial See Polynomials (Yn) <a href="#">(P62)</a>																
	<b>Fn</b>	Apply intrinsic function <b>n</b> .	1 to 7	<table border="1"> <thead> <tr> <th><b>n</b></th> <th><b>Function</b></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1/x</td> </tr> <tr> <td>2</td> <td><math>\sqrt{x}</math></td> </tr> <tr> <td>3</td> <td>Ln(x)</td> </tr> <tr> <td>4</td> <td>Log(x)</td> </tr> <tr> <td>5</td> <td>Absolute(x)</td> </tr> <tr> <td>6</td> <td>X<sup>2</sup></td> </tr> <tr> <td>7</td> <td>Grey code to binary conversion (32bit)</td> </tr> </tbody> </table> See Intrinsic Functions (Fn) <a href="#">(P61)</a> .	<b>n</b>	<b>Function</b>	1	1/x	2	$\sqrt{x}$	3	Ln(x)	4	Log(x)	5	Absolute(x)	6	X <sup>2</sup>	7	Grey code to binary conversion (32bit)
	<b>n</b>	<b>Function</b>																		
1	1/x																			
2	$\sqrt{x}$																			
3	Ln(x)																			
4	Log(x)																			
5	Absolute(x)																			
6	X <sup>2</sup>																			
7	Grey code to binary conversion (32bit)																			
<b>Tn</b>	Apply thermistor scaling (correction) <b>n</b>	1 to 20	Applies a previously-defined thermistor scaling equation See Thermistor Scaling (Tn) <a href="#">(P63)</a> and Thermistors <a href="#">(P136)</a> ).																	
Data Manipulation	<b>DF</b>	Difference $\Delta x$		Returns the difference (xUnits) between the latest reading and the previous reading																
	<b>DT</b>	Time difference $\Delta t$		Time difference (seconds) between the latest reading and the previous reading																
	<b>RC</b>	Rate of change $\Delta x / \Delta t$		Rate of change (xUnits per second) based on latest and previous readings and their respective times																
	<b>RS</b>	Reading / time difference $x / \Delta t$		Rate of change (xUnits per second). Useful when the sensor reading is already a difference (e.g. resetting counters)																
	<b>IB</b>	“Integrate” $(x - \Delta x / 2) \Delta t$		“Integration” with respect to time (xUnits . seconds) between the latest and the previous readings (area under curve)																
Digital Manipulation	<b>TRR</b>	Time from rising edge to rising edge		Normally used for digital channels. If used on analog channels then channel factor is interpreted as a threshold value.																
	<b>TRF</b>	Time from rising edge to falling edge																		
	<b>TFR</b>	Time from falling edge to rising edge																		
	<b>TFF</b>	Time from falling edge to falling edge																		
	<b>TOR</b>	Time of rising edge																		
	<b>TOF</b>	Time of falling edge																		
Reference Channel	<b>TR</b>	Use this channel's value as thermocouple reference junction temperature		Any non-thermocouple temperature sensor measuring isothermal block temperature. If already compensated use <b>11SV(TR)</b> as reference channel (11SV always returns 0.0). TR channel temperature is used for all subsequent thermocouple measurements in this schedule																
	<b>TZ</b>	Use this channel's value to correct the DT80's electrical zero		This zero would be measured at the isothermal block TZ channel zero is used for all subsequent thermocouple measurements in this schedule																
	<b>BR</b>	Use this channel's value as bridge excitation voltage		BR channel voltage used for all subsequent <b>BGV</b> measurements in this schedule																
Serial Channel Comms Type	<b>RS232</b> <i>default</i>	RS232 levels full duplex		<b>RS232</b> is the default if none of these is specified.																
	<b>RS422</b>	RS422/485 levels full duplex		All instances of <b>1SERIAL</b> in a job use the same comms type.																
	<b>RS485</b>	RS422/485 levels		See Serial Channel.																

		half duplex			
Rainflow Cycle Analysis	<b>RAINFLOW</b>			See Rainflow Cycle Counting ( <a href="#">P58</a> ).	
<b>CONFIGURATION LINE (see Multiple Reports (<a href="#">P33</a>))</b>					
Data tags	<b>DDE</b>	Prefix returned channel ID with DDE tag (&!)		Only for use with DeTransfer version 3.00 or later.	
	<b>OLE</b>	Prefix returned channel ID with OLE tag (\$!)		DDE or OLE tags can also be added to a schedule ID, date or time — see P45 ( <a href="#">P111</a> ) in the Table 9: <i>DT80</i> Parameters ( <a href="#">P112</a> ).	
Statistical  See Channel Options — Statistical ( <a href="#">P56</a> ).	<b>AV</b>	Average of channel readings		These channel options link the channel to the statistical sub-schedule RS. The channel is sampled at times determined by the RS trigger (which defaults to 1S).	
	<b>SD</b>	Standard deviation of channel readings			
	<b>MX</b>	Maximum channel reading		At the report time as determined by the report schedules, the statistical summary is reported. If insufficient samples have been taken before the reporting time, an error is reported (-9e9).	
	<b>MN</b>	Minimum channel reading			
	<b>TMX</b>	Time of maximum channel reading			
	<b>TMN</b>	Time of minimum channel reading			
	<b>DMX</b>	Date of maximum channel reading			
	<b>DMN</b>	Date of minimum channel reading			
	<b>IMX</b>	Instant (time and date) of maximum			
	<b>IMN</b>	Instant (time and date) of minimum			
	<b>INT</b>	Integral for channel (using time in seconds)			
	<b>NUM</b>	Number of samples in statistical calculation			
	<b>Hx:y:m..nCV</b>	Histogram	<b>x, y</b> ±1e18 <b>m, n</b> 1-500		Divide data range <b>x</b> to <b>y</b> into discrete buckets and accumulate in CVs the number of samples in each bucket See Histogram ( <b>Hx:y:m..nCV</b> ) ( <a href="#">P57</a> )
	Variables  See Channel Variables (nCV) ( <a href="#">P63</a> ).	<b>=nCV</b>	Assign channel reading to channel variable. <b>nCV</b> = reading		1 to 500
<b>+nCV</b>		Add channel reading to channel variable. <b>nCV</b> = <b>nCV</b> + reading	1 to 500		
<b>-nCV</b>		Subtract channel reading from channel variable. <b>nCV</b> = <b>nCV</b> – reading	1 to 500		
<b>*nCV</b>		Multiply channel variable by channel reading. <b>nCV</b> = <b>nCV</b> * reading	1 to 500		
<b>/nCV</b>		Divide channel variable by channel reading. <b>nCV</b> = <b>nCV</b> ÷ reading	1 to 500		
Destination	<b>NR</b>	No return		Channels tagged with <b>NR</b> are not returned to the host computer (they may still be logged or displayed).	
	<b>NL</b>	No log		Channels tagged with <b>NL</b> are not logged (they may still be returned or displayed).	
	<b>ND</b>	No display		Channels tagged with <b>ND</b> are not displayed on the LCD (they may still be returned or logged).	
	<b>W</b>	Working channel		Same as ( <b>NR,NL,ND</b> ) Working channels are usually used to hold intermediate values in calculations.	
Output Data Format	<b>FFn</b> default = 1	Fixed-point format <b>n</b> =decimal places	0 to 7	Specifies numeric format for display and free format (/h) real-time data.	
	<b>Fen</b>	Exponential format,	0 to 7	For example, <b>FF2</b> returns 71.46 mV	

		$n$ =decimal places		For example, <b>FE2</b> returns 7.14e1 mV
	<b>FM<math>n</math></b>	Mixed: FF or FE, $n$ =significant digits	0 to 7	<b>FM<math>n</math></b> uses exponential format if exponent is less than -4 or greater than $n$
	<b>BGmin:max</b>			Show value as a bargraph on the display.
Channel name and Units	<b>"name"</b>	User-specified name Default units	ASCII text	Allows channel name and/or units to be overridden for display and free format (/h) real-time data. Max 16 characters for user-specified channel name; 8 characters for units.
	<b>"name~unit"</b>	User-specified name User-specified units		
	<b>"name~"</b>	User-specified name No units		
	<b>"~unit"</b>	No channel name User-specified units		
	<b>"~"</b>	No channel name No units		

Table 3: DT80 Channel Options

# Part C — Schedules

## Schedule Concepts

Tell the *DT80* what to do and when to do it

### What are Schedules?

**Schedules** (full name: **schedule commands**) are the workhorses of the *DT80*. They are the underlying structures that you use to manage the repetitive **processes** of the *DT80* such as

- scanning input channels
- evaluating calculations
- processing alarms
- managing output channels
- returning data to a host computer
- logging data.

The *DT80* supports the following schedules:

- 11 **general purpose** schedules, A-K, which can be triggered by a variety of different events
- a **polled** schedule, X, which is normally triggered by a poll command from the host computer (although most of the other triggers can also be applied to it, making it effectively a 12<sup>th</sup> general purpose schedule)
- the **immediate** schedule, which executes once immediately after being entered
- the **statistical** schedule, which collects and accumulates data to be returned as statistical summaries by the other schedules.

### Schedule Syntax

A typical schedule definition is shown below:

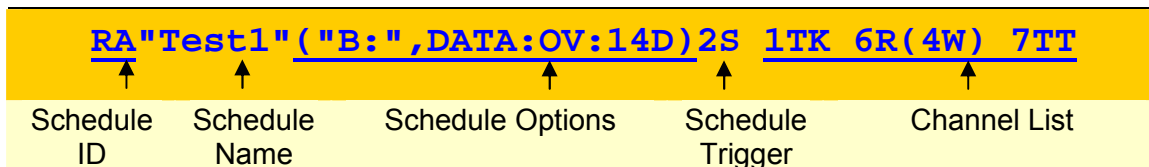
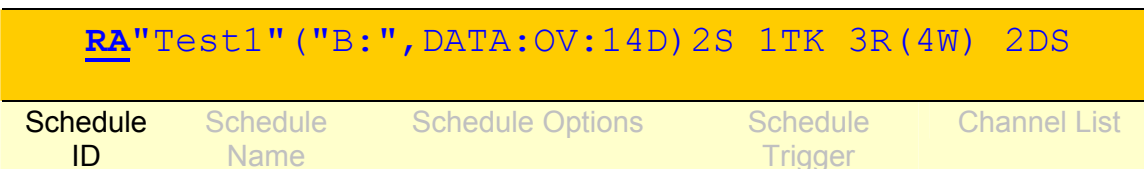


Figure 6: Components of a typical schedule command

A schedule consists of a number of parts. Firstly the Schedule ID, next the schedule options and finally the schedule trigger. There are no spaces between the different parts

#### Schedule ID



The **Schedule ID** consists of the letter **R** ("Report schedule") followed by a letter identifying the schedule. Each schedule has a unique identifier; these are summarized in the following table:

	Schedule ID	Quantity
<b>Report schedules</b>	<b>RA, RB, RC, RD, RE, RF, RG, RH, RI, RJ, RK</b>	11 available
See Types of Schedules		
General-Purpose Report Schedules		
(RA, RB,...RK) <small>(P42)</small>		
<b>Polled schedule</b>	<b>RX</b>	1 available
See Polled Report Schedule (RX) <small>(P45)</small>		



<b>Immediate report schedule</b> See Immediate Report Schedules (P46)	No schedule ID	You may create any number, but only one can be sent to the DT80 at a time.
<b>Statistical sub-schedule</b> See Statistical Report Schedules (P46)	<b>RS</b>	1 available, but is applied to one or more of the other report schedules

## Schedule Name

RA" <u>Test1</u> " ("B:", DATA:OV:14D) 2S 1TK 3R(4W) 2DS				
Schedule ID	Schedule Name	Schedule Options	Schedule Trigger	Channel List

The schedule name is optional, and consists of text (max 8 characters) enclosed in double quotes. This name is used in reports and is normally used to document the purpose of a given schedule.

## Schedule Options

RA" <u>Test1</u> " (" <u>B:</u> ", <u>DATA:OV:14D</u> ) 2S 1TK 3R(4W) 2DS				
Schedule ID	Schedule Name	Schedule Options	Schedule Trigger	Channel List

Schedule options are enclosed in brackets and define where to log the information generated by the schedule. This includes:

- the size of the DATA file and whether data records are to be overwritten when it is full.
- the size of the ALARM file and if the alarm records are to be overwritten when it is full.

The schedule option syntax is as follows:

( <Dest>, DATA: <DataOverwrite>: <DataSize>, ALARMS: <AlarmOverwrite>: <AlarmSize> )

Item	Possible Values	Explanation
<Dest>	"A:"	Logged data and alarm information from this schedule will be stored on the USB memory device.
	"B:"	Logged data and alarm information from this schedule will be stored on the internal flash disk ( <b>default</b> )
<DataOverwrite>	DATA:OV	Data for this schedule will be overwritten when the data store is full ( <b>default</b> )
	DATA:NOV	Data will NOT be overwritten when the data store is full. When the data store fills logging will stop and the 'Attn' LED will flash
<DataSize>	nB	Allocate n bytes for storing data for this schedule
	nKB	Allocate n kilobytes for storing data for this schedule
	nMB	Allocate n megabytes for storing data for this schedule
	nR	Allocate space for n data records for this schedule
	nS <i>Note 1</i>	Allocate space for n seconds worth of data for this schedule
	nM <i>Note 1</i>	Allocate space for n minutes worth of data for this schedule
	nH <i>Note 1</i>	Allocate space for n hours worth of data for this schedule
<AlarmOverwrite>	ALARMS:OV	Alarms for this schedule will be overwritten when the store is full ( <b>default</b> )
	ALARMS:NOV	Alarms will NOT be overwritten when the store is full. When the store fills logging will stop and the 'Attn' LED will flash
<AlarmSize>	nB	Allocate n bytes for storing alarms for this schedule
	nKB	Allocate n kilobytes for storing alarms for this schedule
	nMB	Allocate n megabytes for storing alarms for this schedule
	nR	Allocate space for n alarm records for this schedule

*Note 1:* These are only valid for time-triggered schedules (not for polled or event triggered schedules). Furthermore, if the schedule rate is changed after the job has started running then the storefile may no longer contain data for the indicated time span.

## Default Schedule Options

All schedule options are optional. Default settings are:

- Destination is **B:** (internal flash drive)
- New data and alarms **overwrite** earlier data/alarms once the store file fills



- Space allocated for data is **1MB**
- Space allocated for alarms is **100KB**

## Examples

**RA"Fred" (DATA:NOV:15D)15M**

Schedule A is given the name "Fred". Data and alarms are stored on the internal drive and sufficient space is allocated for 15 days worth of readings (based on a 15 minute scan rate). In this case earlier data is considered more valuable than later data, so no-overwrite mode is selected. If any alarms are defined in this schedule they will use the default storage parameters (100KB, overwrite enabled)

## Schedule Trigger

```
RA"Test1" ("A:", DATA:OV:14D) 2S 1TK 3R(4W) 2DS
```

Schedule ID	Schedule Name	Schedule Options	Schedule Trigger	Channel List
-------------	---------------	------------------	------------------	--------------

All schedules have a **trigger**, which defines when the schedule is to execute the processes assigned to it. Here are the *DT80*'s schedule triggers:

Schedule Trigger	
An interval of time	See Trigger on Time Interval <a href="#">(P42)</a> . Triggers can also be conditional upon an external or internal state
An external event	See Trigger on External Event <a href="#">(P43)</a> . (that is, trigger only while a particular external state or internal state exists) — see Trigger While
An internal event (that is, an event generated within the DT80)	See Trigger on Internal Event <a href="#">(P43)</a> . <a href="#">(P44)</a> .
A poll command from a host computer	See Trigger on Schedule-Specific Poll Command <a href="#">(P44)</a> .

## Channel List

```
RA"Test1" ("A:", DATA:OV:14D) 2S 1TK 3R(4W) 2DS
```

Schedule ID	Schedule Name	Schedule Options	Schedule Trigger	Channel List
-------------	---------------	------------------	------------------	--------------

Most often schedules will be created that instruct the *DT80* to carry out channel-related tasks, such as scanning one or more of its input channels and/or setting one or more of its output channels. When these schedules are created, group the channel details (their IDs and optional instructions) together in a **channel list** within the schedule. *Figure 6* [\(P39\)](#) shows a typical schedule — notice its schedule header and channel list components.

A channel list may contain just one channel entry or many, and each channel in the list must be separated from the next by one or more space characters. Similarly, a schedule's header must be separated from its channel list by one or more space characters.

The *DT80* processes the channels in a channel list from left to right.

### Example — Channel List

The channel list

```
1V 3R 5..7DS 4TK("Boiler Temp") 3DSO=0
```

specifies the following channels (each is separated from the next by a space character):

- **1V** — read analog input channel 1 as a voltage
- **3R** — read analog input channel 3 as a resistance
- **5..7DS** — read the state of digital input channels 5 through 7 (inclusive)
- **4TK("Boiler Temp")** — read analog input channel 4 as a type K thermocouple and assign it the name **Boiler Temp**
- **3DSO=0** — set digital state output channel 3 low

Note that the example above is only a channel list and not a complete schedule. Here's the same channel list used in a schedule (the schedule header **RJ2M** has been added):

```
RJ2M 1V 3R 5..7DS 4TK("Boiler Temp") 3DSO=0
```

The header identifies the schedule as **R**eport schedule **J** that runs every **2** **M**inutes.

## A Simple Schedule

A schedule comprises a schedule ID (schedule identifier), a trigger that determines when the schedule runs, and a list of processes to be carried out every time the schedule runs. For example, the schedule

```
RA10M 1V 3R
```

specifies report schedule A as follows:

- **RA** — schedule ID
- If logging is enabled then data will be stored to the internal flash disk, 1MB will be allocated and old data will be overwritten when full. This schedule does not define any alarms, so no alarm storage will be allocated.
- **10M** — trigger (run the schedule every 10 minutes)
- **1V 3R** — channel list

## Groups of Schedules — Jobs

A *DT80* **job** is essentially a group of one or more schedules (each specifying a set of processes) that performs the overall task.

It's entirely at the user's discretion how the processes of an overall task are allocated between schedules; there are no hard-and-fast rules. For example, choose to differentiate schedules on the basis of function or purpose — some collect primary data, others perform intermediate calculations, others process alarms, and yet others are responsible for returning and logging data; or choose to assign a schedule to a single channel, such as the *DT80*'s Serial Channel.

See also Jobs ([P19](#)).

# Types of Schedules

## General-Purpose Report Schedules (RA, RB,...RK)

The *DT80* supports eleven general-purpose **report schedules**, which you use to carry out the repetitive processes of scanning input channels, evaluating calculations, handling alarms, managing output channels, returning and logging data, and so on.

These report schedules have the identifiers **RA, RB, RC, RD, RE, RF, RG, RH, RI, RJ** and **RK**.

A report schedule executes the processes assigned to it whenever it is triggered. A schedule trigger can be

- an interval of time
- an external event
- an internal event
- a poll request from a host computer.

### Trigger on Time Interval

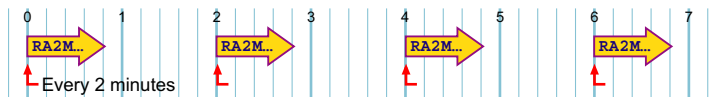


Figure 7: Typical interval-triggered schedule

Report schedules can be triggered at regular intervals of time, determined by the *DT80*'s realtime clock. Intervals can be an integer number of seconds, minutes, hours or days:

Trigger	Run every n	Range
<b>nD</b>	<u>D</u> ays	1<n<65535
<b>nH</b>	<u>H</u> ours	1<n<65535
<b>nM</b>	<u>M</u> inutes	1<n<65535
<b>nS</b>	<u>S</u> econds	1<n<65535
<b>nT</b>	Milliseconds ( <u>T</u> housandths of seconds)	5<n<65535
none	Continuous	

**Note** The schedule first runs on the next multiple of the interval since last midnight (see Time Triggers — Synchronizing to Midnight ([P48](#))), and subsequently runs every multiple of the interval thereafter. If the interval is not an even multiple of 24 hours, the *DT80* inserts a short interval between the last run of the schedule prior to midnight, and the run of the schedule beginning at midnight.

### Examples — Trigger by Time Interval

The schedule header

**RA5S**

instructs the *DT80* to run Report schedule **A** every 5 seconds (**5S**).

The schedule header

**RG**

instructs the *DT80* to run Report schedule **G** continuously (as fast as possible).

## Trigger on External Event

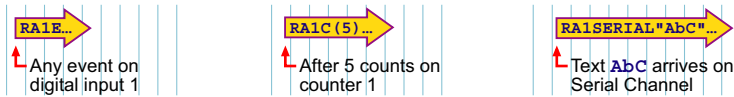


Figure 8: Typical externally-triggered schedules

Report schedules can also be triggered by external events, which are manifested to the logger as state changes on the digital input channels **nDS**, or as pulses on the counter channels **nC**:

Trigger	Action
<b>nE</b>	Trigger on a rising or falling transition of digital input channel <b>n</b>
<b>n+E</b>	Trigger on a rising transition of digital input channel <b>n</b>
<b>n-E</b>	Trigger on a falling transition of digital input channel <b>n</b>
<b>m..nE</b>	Trigger on a rising or falling transition of any of digital input channels <b>m..n</b>
<b>m..n+E</b>	Trigger on a rising transition of any of digital input channels <b>m..n</b>
<b>m..n-E</b>	Trigger on a falling transition of any of digital input channels <b>m..n</b>
<b>nC(c)</b>	Trigger after <b>c</b> counts on a high speed counter channel <b>n</b>
<b>1SERIAL" text "</b>	Trigger on the arrival of characters (from an external serial device) at the DT80's Serial Channel. The trigger can be of the form <b>1SERIAL" "</b> , where <u>any character</u> arriving triggers the schedule (note that there is no space between <b>" "</b> ), or <b>1SERIAL"AbC"</b> , where arrival of the <u>exact string AbC</u> triggers the schedule. See SERIAL CHANNEL (P148).

where

**n** is a digital channel number

**m..n** is a sequence of digital channel numbers (see DIGITAL CHANNELS (P141))

**text** is a string of characters arriving at the DT80's Serial Channel terminals from an external serial device

**Note:** For edge triggering the minimum pulse width is approximately 16ms.

## Triggering on Preset Counters

If a counter is preset to a value greater than its specified trigger count, the schedule is not triggered. For example, a schedule set to trigger after 10 counts on digital counter 2 (**2C(10)**) cannot be triggered if counter 2 is assigned a value of 15.

## Examples — Trigger on Digital Channel Event

The schedule header

**RC1E**

instructs the DT80 to run Report schedule **C** on every transition of digital input 1 (**1E**).

The report schedule trigger

**RA3+E**

instructs the DT80 to run Report schedule **A** whenever digital input channel 3 receives a low to high (positive/rising) transitions.

## Examples — Trigger on Serial Channel Event

The schedule header

**RB1SERIAL"Pasta8zZ"**

instructs the DT80 to run Report schedule **B** on the arrival of the specific character sequence **Pasta8zZ** at the DT80's Serial Channel terminals.

The schedule header

**RG1SERIAL" "**

instructs the DT80 to run Report schedule **G** on the arrival of any character at the DT80's Serial Channel terminals.

## Trigger on Internal Event

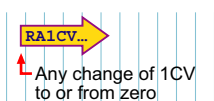


Figure 9: Typical internally-triggered (CV-triggered) schedule

Report schedules can also be triggered by internal events, this must be specified to the DT80 as channel variables (CVs) changing value:

Trigger	Action
<b>nCV</b>	Trigger on any change of <b>nCV</b> to zero or from zero
<b>n+CV</b>	Trigger on any change of <b>nCV</b> from zero
<b>n-CV</b>	Trigger on any change of <b>nCV</b> to zero

where

**n** is the channel variable number

See Channel Variables (*nCV*) ([P63](#)).

### Examples — Trigger on Internal Event

The schedule header

**RK6CV**

instructs the *DT80* to run Report schedule **K** upon any change of channel variable 6 to or from zero. For instance, the schedule **RK**

- will trigger when 6CV changes from 0.0 to 1.0, from 0.06 to 0.0, or from -1.3 to 0.0
- will not trigger when 6CV changes from 0.0 to 0.0, 7.0 to 6.0, or from -112.3 to 0.001.

The schedule header

**RK12+CV**

instructs the *DT80* to run Report schedule **K** whenever the value of channel variable 12 changes from 0 to any value.

### Trigger on Schedule-Specific Poll Command

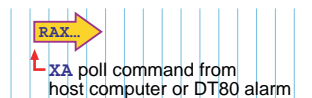


Figure 10: Typical schedule-specific poll-triggered schedule

Instead of a time or event trigger, the poll trigger (**X**) can be applied to a report schedule. Then the schedule can be polled (that is, information requested) at any time by the appropriate schedule-specific poll command **Xa** (where **a** is the schedule letter).

The poll command can be issued

- by a host computer, or
- by an alarm action (see Example — Alarm Action Processes: Using an Alarm to Poll a Schedule ([P83](#))).

See also Using Digital Outputs ([P144](#))

### Example — Trigger on Schedule-Specific Poll Command

The schedule

**RDX 1..3TK**

samples analog channels 1 to 3 as type K thermocouples (**1..3TK**) whenever the *DT80* receives an **XD** poll command (that is, whenever it receives the character sequence **XD**) either from a connected computer, or by means of an alarm action from within the *DT80*.

### Using Poll Commands with Standard Report Schedules

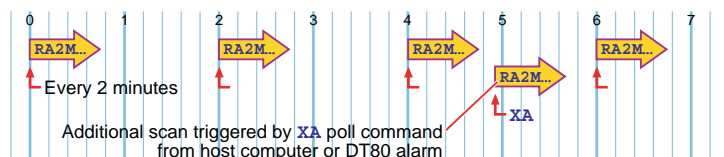


Figure 11: A time-triggered or event-triggered schedule can also be triggered by its poll command

A report schedule defined with a time or event trigger can also be polled by its appropriate poll command at any time. For example, the report schedule

**RC5M 1V 2V 3V**

normally runs every 5 minutes (**5M**), but it can also be run at any time by an **XC** poll command (from the host computer or an alarm).

For schedules that have a long interval, this is useful for checking that a sensor is functioning.

### Trigger While

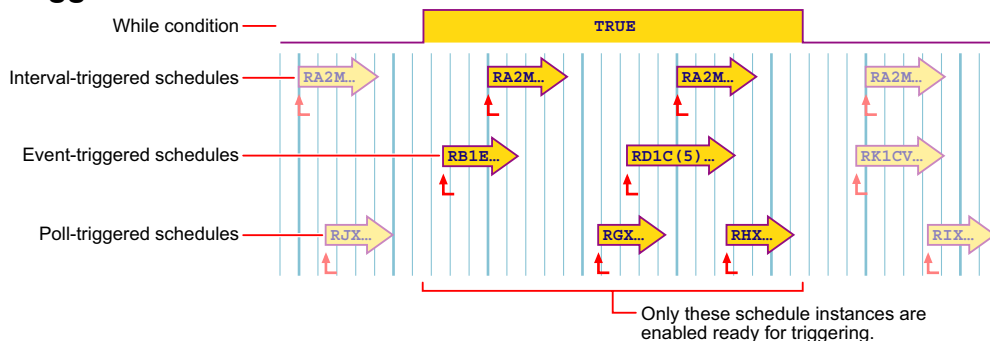


Figure 12: Triggers are enabled/disabled by a While condition

A report schedule's trigger can be enabled or disabled by an external condition (See Figure 13 (P45)). This is called the **While** condition — that is, trigger only While the external or internal condition is true. The While condition can be either

- states on one or more of the DT80's digital input channels (**nDS**), or
- internal conditions specified to the DT80 as states of channel variables:

While clause	Action
<b>:nW</b>	Enable schedule <u>While</u> digital input <b>n</b> is high (true)
<b>:n~W</b>	Enable schedule <u>While</u> digital input <b>n</b> is low (false)
<b>:m..nW</b>	Enable schedule <u>While</u> ANY digital input <b>m</b> to <b>n</b> is high (true)
<b>:m..n~W</b>	Enable schedule <u>While</u> ANY digital input <b>m</b> to <b>n</b> is low (false)
<b>:nCV</b>	Enable schedule <u>While</u> <b>nCV</b> is non-zero
<b>:n~CV</b>	Enable schedule <u>While</u> <b>nCV</b> is zero

Note that the colon (:) is required.

### Examples — While Condition

The schedule header

**RA1E:2W**

instructs the DT80 to run Report schedule **A** on every transition of digital input 1 (**1E**) only while digital input 2 is high (**:2W**).

The schedule header

**RD1S:4~W**

instructs the DT80 to run Report schedule **D** every second (**1S**) while digital input 4 is low (**:4~W**).

The schedule header

**RK2H:9W**

instructs the DT80 to run Report schedule **K** every two hours (**2H**) while digital input 9 is high (**:9W**).

The schedule header

**RC5M:12CV**

instructs the DT80 to run Report schedule **C** every 5 minutes (**5M**) while channel variable 12 is not zero (**:12CV**).

The schedule header

**RF6..8E:5W**

instructs the DT80 to run Report schedule **F** on any transition of digital channels D6, D7 or D8 (**6..8E**) while digital input 5 is high (**:5W**).

### Continuous Report Schedules (No Trigger)



Figure 13: Continuous schedule

Report schedules that run continuously can be created. These schedules start scanning as soon as they are received by the DT80 (they are not activated by a trigger), and run until it is stopped (by sending a halt command or resetting the DT80, for example).

Define a continuous schedule simply by omitting the trigger from a report schedule.

### Example — Continuous Schedule

Sending

**RA 1TK 2R(3W) 3TT**

causes the DT80 to scan channels 1, 6 and 7 continuously. Notice that this schedule has no trigger; for example, no **1S** after **RA**.

## Special-Purpose Report Schedules

### Polled Report Schedule (RX)

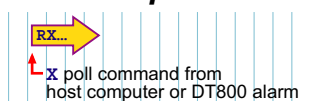


Figure 14: The DT80's solitary polled schedule

The **polled schedule** is a report schedule with a trigger of “request information now” command issued

- by a host computer connected to the DT80 during data acquisition, or
- by an alarm action (see Example — Alarm Action Processes: Using an Alarm to Poll a Schedule (P83) (P83)).

The DT80 supports one polled schedule. It is specified by the **RX** schedule ID, and triggered by an **X** poll command (that is, by an **X** character followed by CR) sent from the host computer or from an alarm.

Channels, calculations and alarms included in a polled schedule are processed, reported and/or logged once every time the DT80 receives an **X** poll command.

Note that underneath, the X schedule is really the same as the general purpose schedules. It can therefore be used as a 12<sup>th</sup> general purpose schedule, except that:

- the continuous trigger is not available. The syntax **RX** is treated the same as **RXX**, ie. it specifies a polled trigger, not a continuous trigger,
- a single **X** character can be used to poll the schedule, which is treated the same as **XX**.

### Example — Polled Report Schedule

The schedule

```
RX 1V 2V
```

runs once every time the DT80 receives an **X** (or **XX**) command.

## Immediate Report Schedules



Figure 15: Typical immediate schedule

Instead of scanning according to time or event triggers, **immediate schedules** run immediately — and once only — when they are received by the DT80.

An immediate schedule is simply a list of input channels, output channels, calculations and/or alarms with no schedule header (that is, no schedule ID and no trigger). The DT80 executes the list (up to the next carriage return) immediately and once only.

**Note** Any data resulting from an immediate schedule is returned to the host computer, but is not logged.

### Example — Immediate Report Schedule

Sending

```
1TK 2R(3W) 3TT
```

causes the DT80 to immediately scan channels 1, 6 and 7 once only and return the data. Notice that this schedule has no schedule ID and no trigger.

### Cautions for Using Immediate Schedules

When programming the DT80, give an immediate schedule time to execute before issuing a following **BEGIN** command, otherwise the immediate schedule's data may not be returned. Using DeTransfer can be helpful, by inserting a **\W** wait command (for example, **\W5**, which pauses program execution for five seconds — between immediate schedule commands and a **BEGIN** command).

If successive immediate schedules are entered too rapidly, then the channels may be appended as if they were part of a single schedule. Setting P22=13 (see P22 ([P110](#)) ([P110](#))) can overcome this by ensuring a return character is placed after each reading.

### Re-Running an Immediate Schedule

The last-entered immediate schedule can be run again by sending the \* (asterisk) command — that is, by sending a \* character.

## Statistical Report Schedules

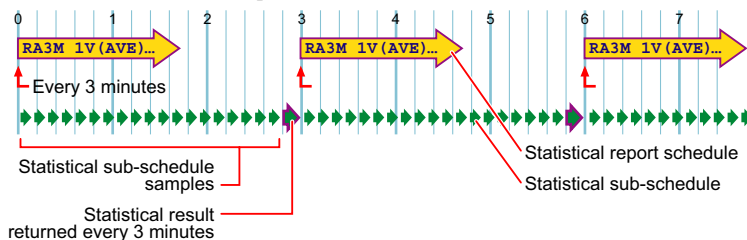


Figure 16: Concepts — statistical report schedule and statistical sub-schedule

A report schedule can instruct the DT80 to return statistical information (average, SD, max., min.,...) for one or more channels. The DT80 does this by

- scanning its input channels and executing calculations at frequent intervals of time, then
- retaining intermediate values to produce a statistical data summary at longer intervals.

Note that there are two schedules involved:

- The primary statistical data is collected at frequent intervals, which are determined by the **statistical sub-schedule RS**.

- The statistical data summary (average, SD,...) is returned and logged at longer intervals, which are determined by the report schedule that is requesting the statistical information — the **statistical report schedule**.

Think of the statistical sub-schedule as a fast schedule (the slave) running within/below its slower statistical report schedule (the master). This is why **RS** is called the statistical sub-schedule.

The statistical sub-schedule has its own interval trigger. The default is one second, but you can change that — see Redefining the Statistical Sub-Schedule's Trigger [\(P47\)](#) [\(P47\)](#) below.

To return statistical data, include — in any report schedule — a statistical channel option for the specific input channels, calculations, and so on where statistically scanned is required. For example

**RA1H 2TT(AV)**

returns, every hour (**RA1H**), the average of one-second readings (because one second is the default scan rate for the statistical sub-schedule) taken from the type T thermocouple connected to channel 2 (**2TT(AV)**).

Note:

- Simply including a statistical channel option (**(AV)** in the example above) invokes the statistical sub-schedule.
- There is no need to include **RS**, the statistical sub-schedule's ID, anywhere (unless you want to alter **RS**'s trigger — see Redefining the Statistical Sub-Schedule's Trigger [\(P47\)](#) [\(P47\)](#) below).

For details of the statistical channel options available, see the Statistical [\(P37\)](#) category of the Table 3: DT80 Channel Options [\(P38\)](#) table, or CHANNEL OPTIONS — STATISTICAL [\(P56\)](#). Triggering and Schedule Order [\(P49\)](#) [\(P49\)](#) is also relevant to the statistical sub-schedule.

### Redefining the Statistical Sub-Schedule's Trigger

The statistical sub-schedule's trigger can be altered from its default of one second.

Define the statistical sub-schedule's trigger in the same way as for report schedules (see [\(P40\)](#)), by using the **RS** schedule ID and sending an **RS...** schedule command to the *DT80*. If you don't specify the **RS** schedule's trigger in this way, it defaults to once per second. Here are some examples:

The schedule header	instructs DT80 to collect specified statistical data
<b>RS10S</b>	every 10 seconds
<b>RS30M</b>	every 30 minutes
<b>RS1-E</b>	on each 1 to 0 transition of digital input 1
<b>RS</b>	continuously

### Statistical Sub-Schedule Halt/Go

The statistical sub-schedule can be halted by sending the **HS** command, and start it again by sending the **GS** command.

**Important** Because statistical sampling of channels stops the moment the **HS** command is sent, be aware that the reported statistical summaries do not include data from this halt period. This is most significant for the integral summary.

See also Halting & Resuming Schedules [\(P49\)](#) [\(P49\)](#).

### Multiple Statistical Information for a Channel

If more than one type of statistical information is required for a channel, then each statistical option must be placed in a separate channel option list (see [\(P33\)](#) [\(P33\)](#)). For example, the channel list

**1TT(AV)(SD)(MX)**

results in periodic average, standard deviation and maximum data for the **1TT** channel.

### Insufficient Statistical Samples

If no statistical data has been scanned before being reported, then the reported data value will be set to **-9.0e9**, a special value that indicates "not yet set". This will also occur if **insufficient** samples have been taken – for example, the standard deviation (**SD**) option requires at least two samples to be able to return a value.

This condition may occur when

- the statistical sub-schedule is event-triggered
- the statistical sub-schedule has been halted
- a statistical sub-schedule scan interval is longer than its statistical report scan interval.

### Example — Statistical Report Schedule

The command

**RS10S RA1H 1TT 2TT(AV)(MX)**

sets the statistical sub-schedule's scan rate to 10 seconds (**RS10S**) and includes statistical sampling. It returns three temperature readings: a spot reading of channel 1 each hour (**RA1H 1TT**), and the average and maximum over the hour from 10-second samplings of channel 2 (**RS10S 2TT(AV)(MX)**).



# Working with Schedules

## Entering Schedules into the DT80 (BEGIN-END)

Report schedules must be entered into (that is, sent to) the *DT80* as a group. Since the schedules and processes that comprise a job or program often extend over more than one line, you embrace them between the keywords **BEGIN** and **END** to designate the beginning and end of the group to the *DT80*. Here's an example:

```
BEGIN
RA10S
  4TT("Oven Temp")
  5TT("Flue Temp")
RB1S
  2C("Water Flow")
END
```

**Note** Additional channels cannot be appended to a schedule once it has been sent to the *DT80*. Instead, it is necessary to re-send the full set of schedules, including the additional channels. (But it is possible to alter an individual schedule's trigger — see Changing a Schedule Trigger [\(P49\)](#) [\(P49\)](#).)

### BEGIN-END Rules

Note the following rules:

- Each line can be up to 254 characters long.
- Processes on lines without a schedule header are included in the schedule immediately above. In the example above, **4TT("Oven Temp")** and **5TT("Flue Temp")** are included in schedule **RA**, and **2C("Water Flow")** is included in schedule **RB**.
- A carriage return must terminate each line.
- Always place the **END** statement on a separate line, by itself. (If a syntax error exists in a line, the *DT80* ignores the remainder of that line. Therefore, if such a line contains an **END** statement, the *DT80* never sees the **END** statement, which can leave the *DT80* in an indeterminate command processing state.)

### How BEGIN-END Works

When the *DT80* receives the **BEGIN** keyword, the *DT80* halts all currently-executing schedules and deletes them, ready to receive the new program... **UNLESS**

- the schedules are locked (see **/F** in Switches [\(P112\)](#) [\(P112\)](#)), or
- the current program contains data, or
- the current program contains alarms, or
- the current program is locked.

The **BEGIN-END** construct can contain blank lines and any other *DT80* commands (these are executed on entry). When **END** is received, the original Halt/Go state is restored.

See Jobs [\(P19\)](#) [\(P19\)](#) for additional important **BEGIN-END** information.

## Using Immediate Schedules in Programs

Immediate schedules are often included in *DT80* programs for tasks such as assigning initial values to channel variables, setting output channels to initial states, taking tare readings for input channels, and so on.

In these cases, enter the immediate schedule processes after the **BEGIN** keyword and before the first report schedule. For example, here's a program containing three immediate schedules (lines 2, 3 and 4):

```
BEGIN
10CV(W)=10
22CV(W)=125
4DSO=1
RA1M
  1TK("Oven Temp",=1CV)
  IF(1CV>22CV){10CV=10CV-1}
  IF(10CV<1){4DSO=0}
END
```

When the above program is sent to the *DT80*, the **CV** and **DSO** assignments (lines 2, 3 and 4) are made exactly as if the three lines were sent as separate immediate schedules.

## Time Triggers — Synchronizing to Midnight

Time triggers for report schedules function in two different ways depending on the setting of the synchronize-to-midnight switch (**/S** or **/S**, see [\(\(P113\)\)](#)).



## Synchronize-To-Midnight Switch Enabled

If the synchronize-to-midnight switch is enabled (`/S`, the *DT80*'s default), the intervals of all schedules with time triggers are synchronized to the previous midnight.

When a time-triggered schedule is entered, the schedules first run on the next multiple of the interval since last midnight, and subsequently run on every multiple of the interval thereafter.

If the interval is not an even multiple of 24 hours, the *DT80* inserts a short interval between the last run of the schedule prior to midnight and the next run of the schedule at midnight.

For example, if you send the schedule

```
RA10H
```

to the *DT80* at 06:00:00, it first runs at 10:00:00 (4 hours since entry, but 10 hours since midnight) and then at 20:00:00 that day; then at 00:00:00, 10:00:00 and 20:00:00 the next day; and so on.

## Synchronize-To-Midnight Switch Disabled

If the synchronize-to-midnight switch is disabled (`/s`), the schedules run at intervals relative to the time that the schedule is entered. For example, if the schedule

```
RA10H
```

is sent to the *DT80* at 09:30:00, it first runs at 19:30:00 that day; then at 05:30:00 and 15:30:00 on the next day; at 01:30:00 and 11:30:00 on the following day; and so on. That is, every 10 hours of elapsed time.

## Retrieving Entered Schedules and Programs

Use the `SHOWPROG` command to return the current program running in the *DT80*, or the `SHOWPROG "JobName"` command to return the program for `JobName` in the logger ([P19](#)) (see Table 4: *DT80* Job Commands ([P55](#))). These commands return everything between `BEGIN` and `END`.

## Triggering and Schedule Order

When different schedules are due to trigger at the same time, the schedules execute in the order of `RA, RB, ...RK`.

When there are statistical channels in a schedule and the statistical sub-schedule is due at the same time as the report schedule, the statistical sub-schedule runs prior to the report schedules. You cannot change this order.

Channels within schedules are sampled in the order of entry (left to right).

## Changing a Schedule Trigger

The schedule's trigger can be changed at any time simply by sending a new schedule ID and trigger without any processes, such as

```
RC10M:2W
```

**Important** If any processes are included, a new schedule is created that replaces all previous schedules UNLESS

- the previous schedules have logged data into memory, or
- logging is enabled by the `LOGON` command (see `LOGON` and `LOGOFF` Commands ([P69](#))), or
- the schedules are locked by the `/F` switch (*(f* ([P112](#))), or
- a `LOCKJOB...` command has been applied (see Table 4: *DT80* Job Commands ([P55](#))), or

## Naming Schedules

A name can be added (maximum eight characters, no spaces) to any of the general-purpose report schedules (`RA, RB, ...RK`). For example

```
RA"Boiler_1"10S 1..5TK
```

Schedule names are returned in the `DIRJOBS` report (see Table 4: *DT80* Job Commands ([P55](#))) and `STATUS14` report (`STATUS14` ([P123](#))).

## Halting & Resuming Schedules

Schedules can be halted individually or as a group:

<b>H</b>	Halt all schedules
<b>HA, HB, ...HK</b>	Halt <code>RA, RB, ...RK</code> schedule
<b>HS</b>	Halt the statistical sub-schedule (see Statistical Sub-Schedule Halt/Go ( <a href="#">P47</a> ))

Schedules can be resumed (`G`Oed) individually or as a group:

<b>G</b>	Resume all schedules
<b>GA, GB, ...GK</b>	Resume <code>RA, RB, ...RK</code> schedule
<b>GS</b>	Resume the statistical sub-schedule (see Statistical Sub-Schedule Halt/Go ( <a href="#">P47</a> ) ( <a href="#">P47</a> ))

## Locking Schedules

Schedules in the *DT80* can be locked by the `/F` switch command (*(F* ([P112](#))) to prevent them from being accidentally changed or deleted.

The entire job that contains the schedules can also be locked — see the `LOCKJOB "JobName"` command in Table 4: DT80 Job Commands (P55).

## Deleting Schedules

The individual schedules cannot be deleted from the *DT80* — the entire job containing the schedules must be deleted.

**Important** A locked job must be unlocked and any stored data and alarms deleted before the job itself can be deleted — see the `UNLOCKJOB`, `DELDATA`, and `DELALARMS` commands in Table 4: DT80 Job Commands (P55).

Once a job has been unlocked, and stored alarms and data has been deleted, use the following commands to delete the job:

Command	Action	
<code>DELJOB</code>	Deletes the current job from the DT80	See the Table 15: DT80
<code>DELJOB "JobName"</code>	Deletes only <i>JobName</i> from the DT80	Retrieval Commands —
<code>DELJOB*</code>	Deletes all jobs from the DT80	Summary (P168) table.

If any schedule has stored alarms or data into memory, or data logging is enabled by `LOGON`, or schedules are locked by `/F`, the job cannot be erased (the *DT80* issues error message E4 or E48 — see ERROR MESSAGES (P174)).

Sending a new job to the *DT80* automatically erases the current job from the runtime environment. However, the job and its data remains in memory and can be run again at a future time.

## Special Commands in Schedules

Here are some special commands that will be useful for controlling the flow of data-processing in schedules. There are two tools for conditional program flow control:

- the `IF...` command
- conditional expressions

and one tool for unconditional program flow control:

- the `DO...` command.

### Conditional Processing — IF... Command

The *DT80*'s `IF...` command allows processes to be performed conditionally — that is, upon other factors being true. The general format of the command is

```
IF(condition)"actionText"{actionProcesses}
```

where

If <i>condition</i> is true	the processes are carried out	See Condition Tests (P50) below.
If <i>condition</i> is false	the processes are not carried out	
<i>actionText</i>	is optional and, if included, is returned to the host computer each time the <code>IF...</code> command tests true. The <i>actionText</i> is not logged.	
<i>actionProcesses</i>	can be reading input channels, setting output channels, performing calculations, setting of any global or system parameters, and so on. If the <i>actionProcesses</i> produce data, this can be returned to the host computer but is not logged. For example <code>IF(1V&gt;100){2V}</code> will sample 2V when 1V is greater than 100mV, however the value <b>will not</b> be returned nor logged. This is because the channel is sampled in the same schedule as the alarm is defined in and as such cannot be returned conditionally as variable length records are not supported by the logger. A technique to solve this problem would involve using an immediate schedule (see immediate schedules). For example <code>RA1S IF(1V&gt;100){XB}</code> <code>RBX 2V</code> Schedule A would run and measure 1V. When 1V exceeds 100mV then schedule B is run. The 2V measurement is made and recorded in the B schedule.	

### Condition Tests

The *condition* works in the same way as the condition for alarms. Normally, channel variables are tested in the condition, and the tests can be as follows:

Operator	Operation	Number of Setpoints
<code>&lt;</code>	Less than setpoint	1
<code>&gt;</code>	Greater than or equal to setpoint	1
<code>&lt;&gt;</code>	Less than first setpoint, OR greater than or equal to second setpoint	2
<code>&gt;&lt;</code>	Greater than or equal to first setpoint AND less than second setpoint	2

The setpoints can be a floating-point constant or a channel variable. The number of setpoints depends on the logical operator.

The `IF...` condition is tested each time that the schedule it belongs to is run, and the text and processes are executed every

time the condition tests true.

### Examples — IF... Command

When run in a schedule, the **IF** command

```
IF(1CV>3.57){2CV=12.6}
```

specifies that if the current value of 1CV exceeds 3.57 (**1CV>3.57**), assign the value 12.6 to 2CV (**2CV=12.6**). If 1CV is less than 3.57 when the condition is tested, no assignment is made.

When run in a schedule, the **IF** command

```
IF(10CV><10,100){5CV(W)=1 RA1M}
```

specifies that if the value of 10CV is between 10 and 100 (**10CV><10,100**), set a flag (**5CV(W)=1**) and change the trigger for the RA schedule to 1 minute (**RA1M**).

Working Channels Hide CV Data ([P64](#)) explains the **W** channel option used in the example above.

### IF...THEN...ELSE

The **IF...THEN...ELSE** command has no ELSE alternative (as in the BASIC language's IF...THEN...ELSE construct). But where an IF...THEN...ELSE test and function is required, this can be achieved by using two **IF...THEN...ELSE** commands.

For example, for any pass of a schedule containing the two IF commands

```
IF(10CV<100){5CV(W)=0}  
IF(10CV>100){5CV(W)=1}
```

only one of the tests will be true, and so the flag will be set appropriately.

See also Conditional Processing — Boolean Expressions ([P51](#)) below.

## Conditional Processing — Boolean Expressions

Boolean expressions can also be used in schedules to return a result that is dependent on a condition being true or false.

For example, the expression

```
2CV=(1CV*2*(1CV<1000))+(1CV*4*(1CV>=1000))
```

sets 2CV to a value of **1CV\*2** if 1CV is less than 1000, or to a value of **1CV\*4** if 1CV is greater than or equal to 1000. (The Boolean expressions **(1CV<1000)** and **(1CV>=1000)** evaluate to a value of **1.0** if true, or **0.0** if false.)

Conditional expressions such as the example above provide an IF...THEN...ELSE function. (See also IF...THEN...ELSE ([P51](#)) above.)

## Unconditional Processing — DO... Command

The **DO...** command is similar to the **IF...** command, except that it has no conditional test. This means that its processes are performed each and every time the report schedule containing the **DO...** command is run.

The general format of the **DO...** command is

```
DO"actionText"{actionProcesses}
```

where

<i>actionText</i>	is optional and, if included, is returned to the host computer each time the <b>DO...</b> command tests true. The <i>actionText</i> is not logged. See <b>DO... actionText</b> ( <a href="#">P51</a> ) below.
<i>actionProcesses</i>	can be commands for reading input channels, setting output channels, performing calculations, setting of any global or system parameters, and so on. <i>actionProcesses</i> is optional. Any potential channel output from the <i>actionProcesses</i> are not returned nor logged. See <b>DO... actionProcesses</b> ( <a href="#">P52</a> ) below.

The **DO...** command can include either **"actionText"** or **{actionProcesses}** or both.

The **DO...** command is particularly useful for executing — within schedules — **DT80** commands that cannot normally be placed in schedules (parameter, switch, unload, alarm, job and delete commands, for example). In other words, the **DO...** command allows direct commands to be placed within runtime commands<sup>6</sup> so that the direct commands are executed during runtime. Even a **RESET** command can be placed within a **DO...** command, but it is not recommend.

**Warning** Use the **DO...** command with great care. Although it's a powerful and flexible command, it does not prevent unreasonable or unrealistic requests being made of the **DT80** during runtime, and some users have found that it can lead to program execution problems or program failure.

### DO... actionText

*actionText* is returned to the host computer whenever the **DO...** command is executed within a report schedule.

*actionText* can be

<sup>6</sup> Direct commands are commands that perform direct tasks within the **DT80** the moment the commands are sent (for example, switch commands and parameter commands). Runtime commands define the runtime tasks (for example, a schedule command).

- any printable text
- **^A** to **^Z** control characters; **^G** (bell), **^M** (carriage return) and **^J** (line feed) are often useful
- the special substitution characters **#** (replaced by the current time when the **text** is returned) and **@** (replaced by the current date when the **text** is returned).

When single quotes are used around the **actionText** instead of the usual double quote, the text is always sent to the Host serial port (see Alarm Action Text [\(P80\)](#))

## DO... actionProcesses

**actionProcesses** can, in theory, be any **DT80** command. However, in practice, some **DT80** commands are likely to cause program runtime errors and even program failure. In particular, do not include report schedules or alarms in **DO...** commands.

The most common use of **DO... actionProcesses** is to change parameter and switch settings to suit requirements of various parts of your program.

**actionProcesses** are executed where they occur in report schedules. Therefore, be aware that their result is in effect when the **DT80** executes the next command of your program.

## Examples — DO... Command

The schedule command

```
RA5M DO"Hello World"
```

instructs the **DT80** to return the message **Hello World** to the host computer when the schedule runs (every five minutes).

In the program

```
BEGIN
  RB1M
  DO"Boiler 99^M^J"
  1..5TK
END
```

the **DO...** command returns the heading text string **Boiler 99** before each record of data (**^M** = carriage return, **^J** = line feed).

The command

```
DO{RUNJOB"Phase_2"}
```

runs the job **Phase\_2** each time it executes.

The schedule command

```
RC2+E DO"AT&F E0 Q1 S0=2 ^M"
```

instructs the **DT80** to output a modem initialization string (**AT&F E0 Q1 S0=2 ^M**) whenever an external event occurs (**2+E**); for example, whenever the modem powers up.

The schedule command

```
RD1M
  DO{P11=500 P47=10}
  1..5TK
  DO{P11=50 P47=5}
  8V 10V
```

instructs the **DT80** to change its ADC setup (**P11** and **P47**) to read channels **1..5TK**, then change it again to read channels **8V** and **10V**.

The schedule command

```
RE1D
  DO{U"Job1"A}
  DO{DELDATA"Job1"}
```

instructs the **DT80** to — every midnight (**1D** trigger) — unload data for schedule A of Job1 (**U"Job1"A**), then delete Job1's data (**DELDATA"Job1"**).

The program

```
BEGIN
  RF5S DO{/h/r} 1..3TK
  RG30S DO{/H/R} 5PT385 6PT385
END
```

instructs the **DT80** to

- every five seconds (**RF5S**) — enter free-format mode (**/h**) and disable the return of real-time data (**/r**) for channels **1..3TK**, and
- every 30 seconds (**RG30S**) — return real-time data (**/R**) in fixed format (**/H**) for the two **PT385** channels.

# Part D — Jobs

## Job Name

A job must always have a name. If there is no name supplied, the *DT80* assigns the default name **UNTITLED**.

To assign a name to a job it needs to be enclosed (maximum of 8 characters, no spaces, not case-sensitive) between straight quotation marks (") immediately after (no space) the **BEGIN** command. (If the quotation marks are omitted "*JobName*" and simply send **BEGIN**, the *DT80* assigns the default name **UNTITLED** to that job.)

A job is created when it is sent

**BEGIN**

or

**BEGIN "JobName"**

to the *DT80*. (Remember, if you omit "*JobName*", the *DT80* assigns the default name **UNTITLED** to the job.)

## Program: a Holder for Jobs

Several jobs can be created, one after the other, in DeTransfer's send window or DeLogger's text window and send them to the *DT80* all at once (with one action). This group of jobs is called a *DT80* **program**.

When the program is sent, all the jobs are stored in the *DT80* for future recall, and the last one sent becomes the *DT80*'s active job. (If the jobs are sent one-by-one, each new job replaces the previous one as the active job, so that there is only ever one active job in the *DT80*.) The *DT80*'s current active job can be changed by sending the **RUNJOB "JobName"** command (see Table 4: *DT80* Job Commands (P55)).

<p><b>Comments</b> can follow the apostrophe character up to a carriage return</p>	<pre> \ Boiler monitoring command file \ Author AVP 12/10/05  RESET \WS  BEGIN"Boiler01" /n/u/S/e P22=44 Y10=4.5,0.312"kPa" S1=0,50,0,100"L/m" RB1M 2..4TT("Temp") RC(DATA:NOV:365D)15M 1V(AV,Y10) 6L(AV,S1)  RK1.0S ALARM1(1V(Y10)&gt;2.25)3DSO ALARM1(4TT&gt;110.0)3DSO,1CV"Over Temp ?"{RB5S}" LOGON END  G                 </pre>	<p>DeTransfer backslash command</p>
<p><b>Pre-Job Commands</b> Not kept in the <i>DT80</i> with the job</p>		<p><b>Job Name</b></p> <p>Switches (P112) determine the system function. Uppercase is <b>ON</b> and lowercase is <b>OFF</b></p> <p>Parameters (P109) are internal system settings that determine system function. Most can be set and all can be read</p>
<p><b>BEGIN</b> This command creates a <i>DT80</i> Job when processed by the <i>DT80</i>. This job's components (name, directory structure, commands and other statements, data and alarms) are all stored on in the <i>DT80</i>.</p>		<p><b>Scaling and Calculations</b> Various methods are available to scale data to engineering units (channel factors, functions, spans, polynomials and calculations).</p> <p><b>Schedules (P133)</b> Eleven general purpose schedules are available (RA, RB, RC, RD, RE, RF, RG, RH, RI, RJ and RK) with an optional statistical sub schedule. Each has a list of channels to be scanned at programmable time intervals or on events. Another schedule (RX) can only be polled by the host computer or by a command from the logger itself.</p> <p><b>Channel List</b> A wide range of channel type provides support for most sensors. Options allow custom configurations. Channel lists sent to the <i>DT80</i> without a schedule header return data immediately (they are not logged)</p>
<p><b>DT80 Job</b></p>		<p><b>Schedule Header</b> The schedule ID (<b>RC</b>) plus file information including the size of the data file (<b>365D</b> i.e 365 Days of data) and the override (<b>OV</b>) or non override (<b>NOV</b>) of memory the schedule trigger (<b>15M</b>)</p> <p><b>Alarms (P133)</b> detect out of range conditions. Can also be used to alter <i>DT80</i> function, reschedule timing, and control <i>DT80</i> outputs and event annunciation</p> <p><b>Logging</b> Global data logging is enabled by the LOGON</p> <p><b>Scan control commands</b> Schedules can be globally or individually started (G, GA, GB...) or halted (H, HA, HB....)</p>
<p><b>Post-Job commands</b> Not kept in the <i>DT80</i> with the job</p>		
<p><b>All commands</b></p> <p>Uppercase and lowercase characters are accepted (except for switches). Must be separated by one or more spaces or carriage returns. Not processed until the carriage return is received</p>		

Figure 17 Anatomy of a sample *DT80* program

## Multiple Jobs in one program

When two or more jobs are sent in a program (that is, all at once), they can work like subroutines. For example, if an out-of-range measurement triggers an alarm in the current job, it can "call" (load and run) another job that has been written to deal with alarm situations. Once the alarm is under control the first job could then be recalled by an appropriate alarm.

## Job Commands

The *DT80* supports these job-related commands:

<b>BEGIN</b> "JobName"	When sent to the DT80, creates a job called <b>JobName</b>
⇓	
<b>END</b>	
<b>BEGIN</b>	When sent to the DT80, creates a job called <b>UNTITLED</b>
⇓	
<b>END</b>	
<b>DIRJOBS</b>	Returns a report of all jobs stored in the DT80 (+ indicates locked job, * indicates current job) See Table 4: DT80 Job Commands ( <a href="#">P55</a> )
<b>DIRJOB</b> "JobName"	Returns a report comprising various <b>JobName</b> details (by schedule)
<b>DIRJOB</b>	Returns a report of the current job and its details (by schedule)
<b>DIRJOB*</b>	Returns a report of all jobs and their details (by schedule)
<b>SHOWPROG</b>	Returns the current job's set of commands file
<b>SHOWPROG</b> "JobName"	Returns <b>JobName</b> 's set of commands file
<b>SHOWPROG*</b>	Returns all currently-defined jobs' sets of command files
<b>LOCKJOB</b>	Protects the current job from deletion, its command file from modification, and its data from deletion
<b>LOCKJOB</b> "JobName"	Protects <b>JobName</b> from deletion, its command file from modification, and its data from deletion
<b>LOCKJOB*</b>	Protects all jobs from deletion, their command file from modification, and their data from deletion
<b>UNLOCKJOB</b>	Allows the current job to be deleted, its command file to be modified, and its data to be deleted
<b>UNLOCKJOB</b> "JobName"	Allows <b>JobName</b> to be deleted, its command file to be modified, and its data to be deleted
<b>UNLOCKJOB*</b>	Allows all jobs to be deleted, their command files to be modified, and their data to be deleted
<b>DELJOB</b>	Deletes the current job from the DT80
<b>DELJOB</b> "JobName"	Deletes only <b>JobName</b> from the DT80
<b>DELJOB*</b>	Deletes all jobs from the DT80
<b>DELDATA</b>	Deletes the current job's data from the DT80
<b>DELDATA</b> "JobName"	Deletes only <b>JobName</b> 's data from the DT80
<b>DELDATA*</b>	Deletes all jobs' data from the DT80
<b>DELALARMS</b>	Deletes the current job's alarms from the DT80
<b>DELALARMS</b> "JobName"	Deletes only <b>JobName</b> 's alarms from the DT80
<b>DELALARMS*</b>	Deletes all jobs' alarms from the DT80
<b>U</b>	See Unload Commands ( <a href="#">P74</a> ).
<b>U</b> (from) (to)	
<b>U</b> [from] [to]	
<b>Ux</b>	
<b>Ux</b> (from) (to)	
<b>Ux</b> [from] [to]	
<b>U</b> "JobName"	
<b>U</b> "JobName" (from) (to)	
<b>U</b> "JobName" [from] [to]	
<b>U</b> "JobName"x	
<b>U</b> "JobName"x (from) (to)	
<b>U</b> "JobName"x [from] [to]	
<b>A</b>	See Retrieving Logged Alarm States ( <a href="#">P86</a> ).
<b>A</b> (from) (to)	
<b>A</b> [from] [to]	
<b>Ax</b>	
<b>Ax</b> (from) (to)	
<b>Ax</b> [from] [to]	
<b>A</b> "JobName"	
<b>UA</b> "JobName" (from) (to)	
<b>A</b> "JobName" [from] [to]	
<b>A</b> "JobName"x	
<b>A</b> "JobName"x (from) (to)	
<b>A</b> "JobName"x [from] [to]	
<b>CURJOB</b>	Returns the current job's name

<b>RUNJOB "JobName"</b>	Makes <b>JobName</b> the <b>DT80's</b> current job. (Of course, <b>JobName</b> must already exist in the <b>DT80</b> — see Table 4: DT80 Job Commands ( <a href="#">P55</a> .) New data created is appended to any existing data for that job.
<b>RUNJOBONINSERT "JobName"</b>	See Startup Job ( <a href="#">P116</a> ).
<b>RUNJOBONRESET "JobName"</b>	

Table 4: DT80 Job Commands

### Pre-Job and Post-Job Commands

See Table 4: DT80 Job Commands ([P55](#)). These are not stored in the **DT80** along with the job. The Pre-Job and Post-Job Commands are run once when the job is sent to the **DT80**, then discarded. Only the job is kept in the **DT80** for re-use later.

### Deleting Jobs

To delete a job from the **DT80**, firstly unlock the job (if it's locked) then delete the job's alarms and data. Therefore, the sequence for deleting a job from the **DT80** is:

- use the appropriate **UNLOCKJOB** command if the job is locked, then
- use the appropriate **DELALARMS** command if the job has logged alarms in the **DT80**, then
- use the appropriate **DELDATA** command if the job has logged data in the **DT80**, then
- use the appropriate **DELJOB** command to finally delete the job.

These commands are presented in Table 4: DT80 Job Commands ([P55](#)) and Table 14: DT80 Delete Commands — Summary ([P166](#)).

DIRJOBS report

DIRJOBS	
UNTITLED	+ indicates locked job
JOB1	* indicates current job
JOB2	
+*JOB3	

DIRJOB "JobName" report

DIRJOB "JOB2"	Job JOB2	S	SchedID	Log	Data Recs	Capacity	First	Last	Alarm Recs	Capacity	First	Last
A				Off	17	20971	08/01/2004 09:26:20	08/01/2004 09:26:36	10	1422	08/01/2004 09:26:20	08/01/2004 09:26:36

DIRJOB report

DIRJOB	Job JOB3 - Locked	S	SchedID	Log	Data Recs	Capacity	First	Last	Alarm Recs	Capacity	First	Last
A				On	63	20971	08/01/2004 09:26:38	08/01/2004 09:27:40	0	0		

DIRJOB\* report

DIRJOB*	Job UNTITLED - no data files	Job JOB1	S	SchedID	Log	Data Recs	Capacity	First	Last	Alarm Recs	Capacity	First	Last
A				Off	31	20971	08/01/2004 09:25:50	08/01/2004 09:26:19	0	0			
DIRJOB*	Job JOB2	S	SchedID	Log	Data Recs	Capacity	First	Last	Alarm Recs	Capacity	First	Last	
A				Off	17	20971	08/01/2004 09:26:20	08/01/2004 09:26:36	10	1422	08/01/2004 09:26:20	08/01/2004 09:26:36	
DIRJOB*	Job JOB3 - Locked	S	SchedID	Log	Data Recs	Capacity	First	Last	Alarm Recs	Capacity	First	Last	
A				On	63	20971	08/01/2004 09:26:38	08/01/2004 09:27:40	0	0			

Figure 18 Sample DIRJOB reports



# Part E — Manipulating Data

## Channel Options — Statistical

### Useful for reducing data

Channels can be sampled frequently and a statistical summary returned at longer intervals (see Statistical Report Schedules [\(P46\)](#)). Statistical channels are sampled for the period between report times, and the statistical summary is generated and returned at report time.

Channels that require statistical sampling must include a channel option to indicate the statistical information to generate. Here's a summary of the statistical channel options — see also the Statistical [\(P37\)](#) category in the [Table 3: DT80 Channel Options \(P38\)](#) table:

Channel Option	Description	Appended to Units
<b>AV</b>	Average	(Ave)
<b>SD</b>	Standard deviation	(SD)
<b>MX</b>	Maximum	(Max)
<b>MN</b>	Minimum	(Min)
<b>TMX</b>	Time of maximum	(Tmx)
<b>TMN</b>	Time of minimum	(Tmn)
<b>DMX</b>	Date of maximum	(Dmx)
<b>DMN</b>	Date of minimum	(Dmn)
<b>IMX</b>	Instant of maximum (combines DMX and TMX)	
<b>IMN</b>	Instant of minimum (combines DMN and TMN)	
<b>INT</b>	Integral	(Int)
<b>Hx:y:m..nCV</b>	Histogram	
<b>NUM</b>	Number of samples	

The statistical option is defined by including it as a channel option in parentheses after the channel type. For example, the schedule

```
RA1M 3TT(AV)
```

returns

```
3TT 103.7 degC (Ave)
```

This is the average ((**AV**)) temperature returned every one minute (**RA1M**) for the type T thermocouple connected to channel 3 (**3TT**). The text (**Ave**) is appended to the units to indicate that the data is an average.

If statistical channels have not been sampled before they are reported, these channels report error E53 (see the Table 18: *DT80* Error Messages [\(P176\)](#) table) and data is returned as 99999.9. This condition is likely to occur when the RS trigger is an event, the statistical sub-schedule has been halted, or a statistical scan interval (**RS**) is longer than the reporting time interval.

### Statistical Channel Options and Multiple Reports

If statistical options are part of a multiple report schedule [\(P33\)](#), each option list must contain a statistical option — for example

```
4PT385(I,500,AV)(MX)(TMX)(MN)(TMN)
```

Note that the first channel option list ((**I,500,AV**)) must include all of the options required for managing and sampling the channel. This rule applies to any options above the configuration line in the [Table 3: DT80 Channel Options \(P38\)](#) table, because the channel is sampled and scaled according to the first option list.

### Statistical Channel Options and Alarms

Statistical results are tested in alarms by first assigning them to channel variables (see Channel Variables (*nCV*) [\(P63\)](#)).

#### Average (**AV**)

The average or mean is the sum of all the channel readings divided by the number of readings. It is very useful in reducing sensor noise. See the Statistical [\(P37\)](#) category.

#### Standard Deviation (**SD**)

Standard deviation is a measure of the variability of the data about the average or mean. The variation may be due to electrical noise or process changes. The units of standard deviation are the same as the channel reading. See the Statistical [\(P37\)](#) category.

#### Maximum and Minimum (**MX** and **MN**)

The maximum and minimum of a set of channel readings can be reported with the **MX** and **MN** channel options.



The time and date of these can be reported with the **TMX**, **TMN**, **DMX**, **DMN**, **IMX** and **IMN** channel options. See the Statistical ([P37](#)) category.

## Integration (**INT**)

The integration channel option (see the Statistical ([P37](#)) category) returns the integral (area under the curve) with respect to time in seconds using a trapezoidal approximation. The units of an integration are those of the original reading multiplied by seconds.

### Example — Integration

When integration is applied to a flow rate sensor as follows

```
S5=0,0.1,0,1000"litres"
3C("Fuel Consumption",S5,INT,R)
```

the volume of the flow is returned:

```
Fuel Consumption 34.54 litres (Int)
```

The flow rate sensor with a counter output (3C) is scaled by a span (S5 — see Spans (S*n*) ([P62](#))) and then integrated. Note that the span units have been declared as litres, which is the result after integration, although the span calibration is actually to litres per second.

## Histogram (**Hx:y:m..nCV**)

The DT80 can be used to generate a histogram (frequency distribution) of channel samples by applying the histogram channel option, which instructs the DT80 to

- divide the measured data range into a number of intervals called **classes** (see *Figure 20* ([P57](#)))
- count the number of readings that occur in each class during the histogram period
- load each class count into a separate channel variable.

Then use another schedule to read, log and clear the channel variables.

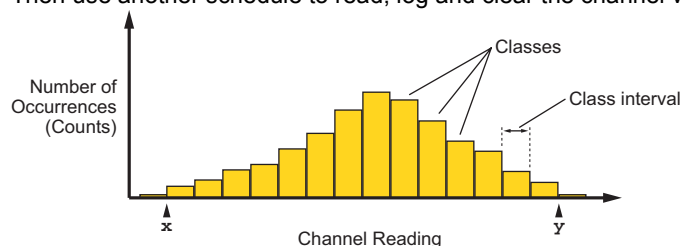


Figure 19: Histogram

In addition, the DT80 automatically counts the number of under-range, over-range and total readings, and stores these in three separate channel variables.

See also the Statistical ([P37](#)) category in the [Table 3: DT80 Channel Options](#) ([P38](#)).

The format of the histogram channel option is

```
Hx:y:m..nCV
```

where

<b>x</b>	is the lowest channel reading of interest
<b>y</b>	is the highest channel reading of interest ( $y > x$ )
<b>m</b>	is the first channel variable ( <b>mCV</b> ) to store counts
<b>n</b>	is the last channel variable ( <b>nCV</b> ) to store counts

Three other counts are also stored by the DT80:

<b>(n-2)CV</b>	Number of readings under range ( $<x$ )
<b>(n-1)CV</b>	Number of readings over range ( $>y$ )
<b>nCV</b>	Total number of readings including those out of range

The histogram channel option does not affect the usual reporting or logging of the channel's readings.

The number of channels that can be histogrammed is limited by the number of channel variables available (the DT80 supports a maximum of 500 CVs).

### Example — Histogram

To create a histogram of a temperature channel over five classes requires eight channel variables:

```
RA1S 1TT(H25.0:35.0:1..8CV)
```

This schedule generates a histogram with five temperature classes and intervals of 2°C:

- 1CV** counts for first class (25.0 to 26.999°C interval)
- 2CV** counts for second class (27.0 to 28.999°C interval)
- 3CV** counts for third class (29.0 to 30.999°C interval)
- 4CV** counts for fourth class (31.0 to 32.999°C interval)
- 5CV** counts for fifth class (33.0 to 34.999°C interval)

It also loads cumulative data into the following three CVs:

- 6CV number of under-range samples (<25°C)
- 7CV number of over-range samples (>35°C)
- 8CV total counts (sum of 1..7CV)

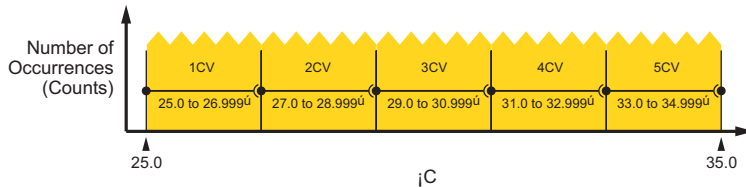


Figure 20: Histogram example

Then return and log the channel variables using another schedule such as

**RB1H 1..8CV(R)**

## Rainflow Cycle Counting

**Rainflow cycle counting** (also called **rainflow analysis**) is an internationally-accepted method of fatigue cycle counting used for monitoring long-term accumulative structural fatigue damage. The process reduces large quantities of cyclic data — collected from sensors attached to the structure over a long period of time — into relatively simple histograms.

As a structure deflects due to repetitive external influences, measurements produce arbitrary peak and valley sequences that form closed loops or cycles. Each loop or cycle has a size (the difference between peak and valley magnitudes), and rainflow analysis accumulates a profile of the number of cycles versus cycle size into a histogram.

A minimum cycle size can be defined that sets a noise rejection level, and cycle sizes below this level are rejected as noise and are not counted.

The *DT80* implements the ASTM E 1049-85 standard: Standard Practices for Cycle Counting in Fatigue Analysis.

Real-time rainflow analysis can be carried out using the *DT80*'s **RAINFLOW...** channel option, which instructs the *DT80* to monitor attached strain gauges at regular intervals and reduce the resulting large quantity of data into simple cycle histograms.

The *DT80* can also produce a formatted report of the accumulated cycle histograms — see Reporting Rainflow Data — The Rainflow Report ([P59](#)).

Although the rainflow cycle counting has been optimized for welded steel structures, it can be used to record arbitrary waveforms from other sources — temperature cycles in a furnace or electrical signals, for example.

### Collecting Rainflow Data — The Rainflow Channel Option

Rainflow analysis is defined by the **RAINFLOW...** channel option. Although this is generally used for channels measuring strain gauge inputs, you can also use it for any type of sensor that is monitoring a process that produces cycles of peaks and valleys with hysteresis.

The overall range of cycle sizes is divided into a number of smaller cycle size **classes** and, as the analysis proceeds, the number of cycles of each size class is counted. These counts are accumulated into the *DT80*'s 32-bit signed Integer Variables (channel type **nIV**).

The **RAINFLOW...** channel option requires a maximum cycle size to be specified, a noise rejection level, and a range of sequential integer variables or channel variables that can be used for accumulating the cycle size counts and other information. It has the form

**(RAINFLOW:a:b:c..dIV)**

where

<b>a</b>	is the maximum cycle size expressed in the channel type units (for example, ppm)
<b>b</b>	is the minimum cycle size for noise rejection in terms of a percentage of <b>a</b>
<b>c</b>	is the first IV in the sequence to be used for accumulating data
<b>d</b>	is the last IV in the sequence to be used for accumulating data

Therefore the range of cycle sizes is from zero to the maximum cycle size defined (**a**), and cycle sizes smaller than **b%** of **a** are rejected and not counted. For example, the channel option

**(RAINFLOW:1000:5:c..dIV)**

sets the cycle size range to 0–1000 units, and cycle sizes less than 5% of 1000 (= 50) units are rejected as noise.

The number of variables allocated for the rainflow analysis must be set to the number of cycle size classes required over the cycle size range, plus seven (7) additional variables for summary data. For example, if you require 10 cycle size classes over the cycle size range then 17 variables will be needed. The variables can begin at any number in their range of 1 to 500 (**c**), and are used sequentially to the last variable number (**d**).

The use of variables in the allocated variable range is summarized in the following table. The first column shows how variables are used within the allocated range, and the last column shows how 20 variables are used. The last 7 variables contain various summary data.

<b>c..dIV</b>	<b>IV Contents</b>	<b>Example:</b> <b>21..40IV</b>
---------------	--------------------	------------------------------------

	<b>c+0</b>	Contains the count of cycles for the first cycle size class	<b>21IV</b>
	<b>c+1</b>	Contains the count of cycles for the second cycle size class	<b>22IV</b>
	<b>c+2</b>	Contains the count of cycles for the third cycle size class	<b>23IV</b>
	↓	↓	↓
	<b>d-7</b>	Contains the count of cycles for the last cycle size class	<b>33IV</b>
	<b>d-6</b>	Contains the count of cycles that over-ranged the maximum cycle size	<b>34IV</b>
	<b>d-5</b>	Contains the count of all cycles	<b>35IV</b>
	<b>d-4</b>	Contains the maximum buffered cycles 0 . . 100 (or 99999 if the buffer has overflowed and buffered half-cycles have been lost)	<b>36IV</b>
Summary data	<b>d-3</b>	Contains the minimum valley encountered	<b>37IV</b>
	<b>d-2</b>	Contains the maximum peak encountered	<b>38IV</b>
	<b>d-1</b>	Contains the total number of good points	<b>39IV</b>
	<b>d-0</b>	Contains the total number of "in error" points (out of range, for example)	<b>40IV</b>

In practice, some cycles do not close immediately and are buffered until a closure is detected. Variable **d-4** contains a count of these unclosed or "half cycles".

**Note:** The rainflow channel option can be used on a maximum of 16 channels.

### Collecting Rainflow Data — The Rainflow Sample Rate

Rainflow cycle data is collected at a rate dependent on the frequency of influences deforming the structure under test. These might be quite slow events (such as waves crashing against a sea wall), or quite fast (such as a high speed boat hull travelling through waves).

Place the channel being sampled for rainflow in a schedule that's triggered fast enough to take sufficient readings during a cycle to adequately characterise the loop closures. For example, the schedule

**RA50T**

**3BGI (RAINFLOW:a:b:c..dIV,W)**

measures the input every 50mSec (**50T**; 20 times/sec), and counts loop closures. The **W** channel option declares this as a working channel (does not return or log the individual samples of strain-stress — see **W** ([P37](#))).

The fastest sampling rate is obtained by the schedule trigger

**RA,FAST (options)**

**3BGI (RAINFLOW:a:b:c..dIV,W)**

See **WORKING WITH SCHEDULES** ([P48](#)).

### Reporting Rainflow Data — The Rainflow Report

Rainflow data is collected over long periods of time using the **RAINFLOW...** channel option. Then, periodically, the rainflow cycle histogram can be retrieved by a computer, using the **RAINFLOW...** command.

To report the rainflow cycle histogram, send the original rainflow channel option exactly as originally defined for the channel, but as a command. That is, send the command

**RAINFLOW:a:b:c..dIV**

to the **DT80**. The **DT80** returns a tabular report as illustrated below, which is normally viewed and saved in the Receive (upper) window of DeTransfer. (DeLogger software does not support the Rainflow Report.)

```
Rainflow (5% rejection)01/01/2000
00:03:43
nIVRangeMeanCycles
=====
110.00.00
223.611.427
337.211.36
4410.812.46
5514.411.96
6618.012.89
7721.612.32
8825.20.00
9928.80.00
101032.40.00
111136.018.01
121239.60.00
131343.20.00
141446.80.00
151550.40.00
161654.00.00
171757.60.00
181861.20.00
191964.80.00
202068.40.00
2121 >=72.00.00
=====
```

---

```
Total cycles58
Peak/Valley mean12.6
Max Peak71
Min Valley-1
Max buffered cycles11
Valid input points %100.00
```

---

The rainflow report provides a complete summary of the rainflow data for the collection period. The cycle size range for each class, the number of cycles in each class, and the mean for each class is shown, as well as the summary data.

Although the rainflow report cannot be logged in the *DT80*, the primary cycle count data used to make up the rainflow report can. For example, the program

```
BEGIN"Rainflow"
  RA50T 2BGI (RAINFLOW:72:5:1..28IV,W)
  RB7D 1..28IV
  LOGONB
END
```

logs the histogram data every 7 days. Reports can be created manually after download of the primary cycle count data.

### Example 1— Rainflow Cycle Counting

Capture raw strain gauge data and perform rainflow cycle analysis using the program

```
BEGIN
  RA50T
  1BGI (RAINFLOW:1000:5:101..127IV,W)
END
```

This instructs the *DT80* to

- collect current-excited bridge data (**1BGI**) every 50ms (**RA50T**) and carry out rainflow analysis over the range of zero to **1000** ppm
- apply a **5%** rejection (that is, cycles smaller than 50ppm are rejected)
- accumulate cycles into histogram variables 101 through 127 (**101..127**); this gives 20 cycle size classes for cycle counts, and 7 others for summary information.

The matching rainflow report command

```
RAINFLOW:1000:5:101..127IV
```

returns rainflow histogram data such as the following:

```
Rainflow (5% rejection)15/03/2000
00:05:22
nIVRRangeMeanCycles
=====
110100.00
21025311.7187805
310310511.5153802
410415811.7128263
510521111.665312
610626311.739342
710731612.325449
810836812.59612
910942113.03950
1011047413.41919
1111152613.6760
1211257913.8354
1311363214.249
1411468414.920
1511573715.13
161167890.00
171178420.00
181188950.00
191199470.00
20120 >=10000.00
=====
Total cycles616640
Peak/Valley mean11.7
Max Peak42
Min Valley-22
Max buffered cycles38
Valid input points %100.00
```

---

### Best Speed for Rainflow.

For best rainflow speed from the *DT80*:

- turn off house keeping (**/k**)

- set the gain lock to just above the maximum expected voltage ([GL30MV](#))
- make the rainflow channels working channels ([W](#))

For example:

```
BEGIN
/k
RA50T
1BGI(RAINFLOW:1000:5:1..20IV,GL30MV,W)
END
```

# Channel Options — Scaling

## Getting sophisticated

The *DT80* provides many different tools for scaling channel readings:

- automatic scaling
- channel factor
- intrinsic functions
- spans
- polynomials
- channel variables

It can also carry out calculations on channel data — see [CALCULATIONS \(EXPRESSIONS\) \(P65\)](#). In addition, scaling and calculation methods can be combined — see [COMBINING METHODS \(P65\)](#).

### Automatic Scaling

The *DT80* automatically scales measurements according to the channel types that have been specified. In other words, whatever the output of a particular sensor, its measurements are always converted (scaled) to engineering units (volts, amps, ohms, °C,...) according to the channel type that was specified in the channel ID in the channel list — this is why it is necessary to specify a channel's channel type.

### Channel Factor (f.f)

Many input channel types support a channel factor (a floating-point number) as a channel option — see the Channel Factor column of the [Table 1: DT80 Channel Types \(P29\)](#) table.

The channel factor channel option usually provides linear scaling. However, for some channel types, the channel factor has a dedicated purpose and therefore cannot be used for scaling. See [\(P35\)](#).

### Example — Channel Factor

In the channel list

```
1V
1V(101.0)
```

the first `1V` returns true millivolts, and `1V(101.0)` includes the channel factor (101.0), which returns the reading multiplied by 101.0 in units of millivolts as follows:

```
1V 2.543 mV
1V 256.84 mV
```

Here, the channel factor could be the attenuation of an input voltage attenuator network.

### Intrinsic Functions (Fn)

The *DT80* has seven inbuilt and mutually exclusive intrinsic functions that are applied as channel options (see [Fn \(P36\)](#)). The intrinsic functions available are

Fn	Description		Text Modifier
F1	1/x	inverse	(Inv)
F2	√x	square root	(Sqrt)
F3	Ln(x)	natural logarithm	(Ln)
F4	Log(x)	base ten logarithm	(Log)
F5	Absolute(x)	absolute value	(Abs)
F6	x * x	square	(Squ)
F7	Grey code conversion (32-bit)		(Gc)

Channels with an intrinsic function applied return data followed by the text modifier listed above. If more than one intrinsic function is placed in a channel's channel option list, only the last is applied.

### Example — Intrinsic Function

The channel list

## 1V(F2)

returns the square root of the reading as follows:

```
1V 455.6 mV (Sqrt)
```

## Spans (Sn)

Spans are used to define calibrations for linear sensors. They are particularly suited to 4–20mA current loop inputs. A total of 50 spans and polynomials can be defined. Spans are applied to a channel as a channel option (see [Sn \(P36\)](#)).

A span is defined as follows:

```
Sn=a,b,c,d"Text"
```

where

<b>n</b>	is the span number ( <b>1</b> to <b>50</b> ) — spans and polynomials must not both use the same number.
<b>a</b> and <b>b</b>	are the physical coordinates of two points on the calibration line — see <a href="#">Figure 26 (P62)</a> .
<b>c</b> and <b>d</b>	are the signal coordinates of the two points on the calibration line — see <a href="#">Figure 26 (P62)</a> . If not specified, <b>c</b> and <b>d</b> default to <b>0</b> and <b>100</b> respectively, which is useful for 4–20mA current-loop channels (channel type <b>L</b> ).
<b>Text</b>	replaces the channel's default units text.

A single span definition may be applied to any number of channels in any schedules or alarms.

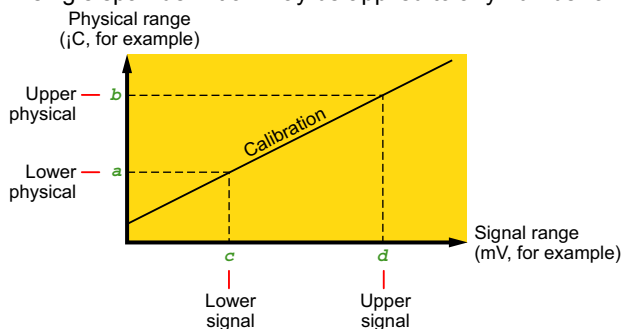


Figure 21: Span coordinates

## Using Spans — Guidelines

- When creating a *DT80* program, define any spans (and polynomials) ahead of the schedules and alarms.
- A span's number is not related to any channel number — they do not have to match. Use any span for any channel.
- Use one span for each type of sensor/transducer.

## Example — Span

The commands

```
S17=0,300,100,1000"kPa"  
RA5M 1V(S17,"Boiler pressure")
```

define the span **S17** then use it in a schedule, which instructs the *dataTaker* to return data in the form

```
Boiler pressure 239.12  
kPa  
Boiler pressure 247.33  
kPa  
↓
```

## Polynomials (Yn)

Polynomials are used to define calibrations for non-linear sensors. A total of 50 spans and polynomials can be defined. Polynomials are applied to channels as a channel option (see [Yn \(P36\)](#)). A maximum of six polynomial coefficients can be entered.

The *DT80* evaluates a polynomial according to the formula

$$y = \sum_{n=0}^5 k_n x^n = k_0 + k_1 x + k_2 x^2 + k_3 x^3 + k_4 x^4 + k_5 x^5$$

where  $x$  is the channel reading, and the  $k$ 's are coefficient terms.

A polynomial is defined as follows:

```
Yn=k0,k1,k2,k3,k4,k5"Text"
```

where

<b>n</b>	is the polynomial number ( <b>1</b> to <b>50</b> ) — polynomials and spans must not both use the same number.
----------	---

$k_0, k_1, \dots$  are the coefficient terms — only the coefficient terms up to the required order need to be entered.

*Text* replaces the channel's default units text.

Simple scale and offset corrections are also possible (internally, the *DT80* treats spans as a first order polynomial). The coefficient terms of a polynomial are evaluated by a least square regression. Various statistical programs are available for this purpose. Some nonlinear sensors are supplied with their calibration polynomial. A single polynomial may be applied to any number of channels in any schedules or alarms.

### Example — Polynomial

The commands

```
Y18=25.5,0.345,0.0452"Deg C"  
RA5M 1V(Y18)
```

define the polynomial **Y18** then use it in a schedule, which instructs the *DT80* to return data in the form

```
1V 44.35 degC  
1V 43.89 degC  
↓
```

### Thermistor Scaling (*T<sub>n</sub>*)

The *DT80* has channel types for many 2-wire YSI thermistors (Yellow Springs Instruments [www.ysi.com](http://www.ysi.com)). For other thermistor types, the *DT80* supports thermistor scaling — the conversion of a resistance reading to a temperature. The *DT80* does the conversion from resistance to temperature using

$$T = \frac{1}{a + b \ln(R) + c \ln(R)^3}$$

To apply thermistor scaling, firstly obtain the constant terms *a*, *b* and *c* from the thermistor manufacturer, then define a thermistor conversion by sending a thermistor command

```
Tn=a,b,c"Temperature units"
```

(*n* is the thermistor conversion number — 1 to 20).

Finally, apply the conversion by including a resistance channel ID with the thermistor channel option

```
mR(Tn)
```

in a schedule (where *m* is the number of the channel to which the thermistor is attached).

### Example — Thermistor Scaling

The commands

```
T1=26.5,1.034,-0.0085"K"  
RA5M 3R(T1,"Solvent temp.")
```

define the thermistor conversion **T1** and its use in a schedule, which instructs the *DT80* to return data in the form

```
Solvent temp. 356.21 K  
Solvent temp. 356.35 K  
↓
```

See also Thermistors ([P136](#)).

### Channel Variables (*nCV*)

Channel variables are memory locations (registers) for holding floating-point data such as channel readings and the result of expressions. The *DT80* has 500 channel variables, identified as **1CV** to **500CV**. They can be used

- to receive
  - the data from input channels at the time of scanning
  - the results of calculations
- to pass input channel data and results of calculations to
  - the host computer
  - the *DT80*'s internal memory and USB memory devices
- to pass input channel data to expressions (see CALCULATIONS (EXPRESSIONS) ([P65](#)))
- as the test data or as the setpoints in alarms
- for temporary storage of date and time, and for time-based calculations such as elapsed time, down time and rates
- to trigger report schedules.

### Assigning Readings to Channel Variables

A channel variable receives (is assigned) the current value of any input channel by including the channel variable in a channel option list — see Variables ([P37](#)) ([Table 3: DT80 Channel Options](#) ([P38](#)) table). For example

```
1V(=2CV)
```

returns the voltage for channel 1 and assigns (overwrites) the voltage value to channel variable **2CV**.

Channel variable assignments are made at the report time of the embracing report schedule. They are not made at the statistical sub-schedule scan time.



Where statistical results are to be tested, then channel variables provide the only means of using statistical results in alarms. For example, the program tests the standard deviation of the temperatures read over each minute.

```
BEGIN
  RS1S
  RA1M
  3TT(SD,=1CV,W)
  ALARM1(1CV>0.1)"Excessive variability"
END
```

A schedule or an immediate scan containing a CV channel option can

- initialize a channel variable automatically
- be sent from the host computer to assign a value to a channel variable.

### Reading a Channel Variable

The *DT80* treats channel variables in the same way as normal input channels — return and log the current value of a channel variable using a normal schedule command.

### Arithmetic Operations

When storing input channel data into channel variables use one of four basic arithmetic operations (**+=**, **-=**, **\*=** and **/=**). For example, the immediate scan command

```
5V(+=1CV)
```

scans channel **5V**, sets **1CV** equal to **1CV+5V** (acts as an accumulator), and reports the value of **5V**.

Similarly, the immediate scan command

```
5V(S1,/=1CV)
```

scans channel **5V**, applies span 1 (**S1**), sets **1CV** equal to **1CV/5V(S1)** and reports the value of **5V(S1)**.

### Statistical Channel Variables

When a channel variable is included as a channel option for a statistically-scanned channel, the statistical result is stored in the channel variable and not the individual readings. For example, the schedule command

```
RS5S RA10M 3V(AV,=1CV)(MX,=2CV)(MN,=3CV)
```

stores the 10-minute average, maximum and minimum into channel variables **1CV**, **2CV** and **3CV** respectively.

### Results of Expressions

Channel variables can also be assigned the results of expressions (see CALCULATIONS (EXPRESSIONS) [\(P65\)](#)). For example, the command

```
3CV=(1+COS(2CV))*1.141
```

evaluates the expression and assigns the result to **3CV**.

### Channel Variables as Triggers

Channel variables can be used as report schedule triggers. See Trigger on Internal Event [\(P43\)](#).

### Using Channel Variables

Channel variables are used in the same way as channels within schedules and alarms. Channel options can be used to modify the function and data format of channel variables. For example, the commands

```
RA1M 3CV(H:1:10:100..122CV)=SQRT(6CV+7CV)
RB1H 100..119CV
```

instruct the *DT80* to evaluate the **SQRT** expression and apply the result to the histogram (**H...**) every minute (**RA1M**), and return the accumulated histogram data to the host every hour (**RB1H**)

Channel variables are not normally returned with units text, however units can be defined using polynomials (Polynomials (*Yn*) [\(P62\)](#)) or "**UserName~UserUnits**" channel options ("**name~unit**" [\(P38\)](#)). For example

```
Y20=0,1.0"kPa"
11CV(Y20)=SQRT(4CV/6CV)
```

Channel variables can be used in alarms, both as the test value and as the setpoint(s). For example

```
ALARM1(4CV<>2CV,3CV)"[5CV=20]"
```

Channel variables are useful when comparing an input channel against several thresholds. For example

```
IF(1V(=1CV)>0.5)"Over 0.5 Volts"
IF(1CV>0.6)"Over 0.6 Volts"
IF(1CV>0.7)"Over 0.7 Volts"
```

where channel **1V** is sampled once (rather than risking different values) and tested against a number of setpoints.

### Working Channels Hide CV Data

When input channels or channel variables are used in intermediate steps of a program, then the **W** channel option (see **W** [\(P37\)](#)) can declare these as **working channels** and prevent data being returned or logged. See examples Example 1 [\(P66\)](#).

During program debugging, the **W** option can be overridden by the **/W** switch to return intermediate data.

### Naming Channel Variables

To name a channel variable use ("**CVname~Units**") For example



## 2CV("WindSpeed~km/h")

CVname may have a maximum of 16 characters; Units may have a maximum of 8 characters.

### Channel Variables Report

Send the command

**NAMEDCVS**

to instruct the DT80 to return a summary of all named channel variables in use, for example

CV	S	CV Name	Value	Units
5	A	Temp	89.1	Deg C
1	A	Speed	23.4	m/s

where

CV	is the channel variable number <i>n</i> ( <i>nCV</i> )
S	is the schedule identifier
CV Name	is the channel variable's name, if any (see Naming Channel Variables (P64))
Value	is the value stored in the channel variable when the NAMEDCVS command was sent
Units	is the units defined for the channel variable

# Calculations (Expressions)

The DT80 has a powerful expression evaluation capability. Results can be assigned to channel variables, output channels, system timers and system variables.

Expressions can ONLY contain channel variables and constants. Data from input channels must first be assigned to channel variables to be used in expressions.

Expressions can contain the following operators:

Arithmetic	+, -, *, /, % (modulus) and ^ (exponent)
Relational	<, <=, =, >=, > (result 1 is true, 0 is false)
Logical	AND, OR, XOR, NOT (>0 is true, result 0 or 1)
Functions	ABS(), LOG(), LN(), SIN(), COS(), TAN(), ASIN(), ACOS(), ATAN(), SQRT(), Sn(), Yn(), Fn()
Other	Round brackets (parentheses) ( )

**Note:** The trigonometric functions require arguments in radians, where 1 radian = 57.296 degrees.

**Note:** The modulus operator (%) converts both operands to integer.

The operator precedence is

First	( )	
2nd	^	
3rd	*, / or %	← These operators have equal precedence
4th	+ or -	← These operators have equal precedence
5th	<, <=, =, >=, >	← These operators have equal precedence
Last	AND, OR, XOR or NOT	← These operators have equal precedence

Expressions evaluate left to right, but parentheses can be used to define a particular order of evaluation. Parentheses can be nested.

Expressions are evaluated at the report time of the embracing schedule, and in the order in which they occur within the schedule.

### Conditional Calculations

Boolean logic within expressions can be used to return a result that is dependent on a condition being true or false. For example,

**2CV=(1CV\*2\*(1CV<1000))+(1CV\*4\*(1CV>=1000))**

returns a value of 2\*1CV if 1CV is less than 1000, or a value of 4\*1CV if 1CV is greater than or equal to 1000.

# Combining Methods

The different scaling and calculation methods can be used together. The following comprehensive examples are the best way to demonstrate.

## Example 1

In this program, a vector average is calculated. The inputs are wind speed and direction.

```
BEGIN"Wind-01"

'Wind speed calibration 0-50m/s = 0-1000mV
S1=0,50,0,1000"m/s"
'Wind direction 0-2Pi radians (0-360deg) = 0-1000mV
S2=0,6.2832,0,1000"radians"
Y3=0,1"m/s" 'Units text for wind speed report
Y4=0,1"Deg" 'Units text for wind direction report

  RA5S 'Schedule to scan every 5 seconds
  1V(S1,=1CV,W) 'Sample wind speed
  2V(S2,=2CV,W) 'Sample wind direction
  3CV(W)=3CV+1CV*COS(2CV) 'Sum x components
  4CV(W)=4CV+1CV*SIN(2CV) 'Sum y components
  5CV(W)=5CV+1.0 'Number of scans
  RB1M 'Calculate, report and log every minute
'Calculate mean magnitude:
  6CV(W)=SQRT((3CV*3CV)+(4CV*4CV))/5CV
  6CV("Mean Wind Magnitude",Y3,FF1)
'Calculate direction
  7CV(W)=ATAN(4CV/3CV)*57.29
'Determine direction quadrant
  7CV(W)=7CV+((3CV>0)AND(4CV<0))*360
  7CV(W)=7CV+((3CV<0)AND(4CV<0))*180
  7CV(W)=7CV+((3CV<0)AND(4CV>0))*180
'If wind speed is zero, return -1.0:
  7CV(W)=7CV-(6CV<=0)*(7CV+1)
  7CV("Mean Wind Direction",Y4,FF0)
  1..5CV(W)=0

END

LOGON G
```

## Example 2

This program scans ten channels and calculates a cross-channel average.

```
BEGIN"Wind-02"
  RA10S
  1CV(W)=0 'Clear 1CV
  1..10V(+=1CV,W) 'Sum 10 voltages into 1CV
  1CV=1CV/10 'Divide by 10 for average
END
```

# Part F — Logging and Retrieving Data

## Format of Returned Data

The DT80 can return the following types of data to the host computer:

- data returned as it is measured — that is, real-time data
- data unloaded from the DT80's internal memory or from a USB memory device — that is, logged data
- data returned by the **TEST** and **STATUS** commands

The format of returned data is controlled globally by the following parameters and switches (see *CCONFIGURING THE DT80* (P109) for full details):

<b>P22</b>	Data delimiter in free-format mode (see P22 (P109))	Default = <b>32</b> (space character)
<b>P24</b>	Scan delimiter in free-format mode (see P24 (P110))	Default = <b>13</b> (carriage return character)
<b>P31</b>	Date format	Default = <b>1</b> (DD/MM/YYYY)
<b>P33</b>	Defines a fixed field width for output data (variable)	Default = <b>0</b> (off)
<b>P38</b>	Decimal point locator character for floating-point numbers	Default = <b>46</b> (full stop character)
<b>P39</b>	Time format — see Time (P30)	Default = <b>0</b> (hh:mm:ss.sss)
<b>P40</b>	Time separator character	Default = <b>58</b> (colon character)
<b>/H</b>	Fixed-format mode (P21)	Default = off
<b>/U</b>	Include <u>Units</u> text appended to the data	Default = on
<b>/N</b>	Include channel <u>N</u> umber and type before data	Default = on
<b>/L</b>	Include <i>dataTaker</i> serial number before scan data	Default = off
<b>/C</b>	Include <u>C</u> hannel type (/C) or number only (/c)	Default = on
<b>/D</b>	Include scan <u>D</u> ate at beginning of returned data	Default = off
<b>/T</b>	Include scan <u>T</u> ime at beginning of returned data	Default = off
<b>/I</b>	Include schedule <u>I</u> D	Default = off

### Character Pairs — Carriage Return + Line Feed

The DT80's default is to automatically add a carriage return character (**CR**, ASCII 13) and a line feed character (**LF**, ASCII 10) to the end of appropriate chunks of information it returns to the host computer. These return-and-line-feed pairs (**CRLF**) make data and other returned information easy to read on the host screen or a printout.

## Two Format Modes for Returned Data

The DT80 has two modes for the format of data and other information it returns to the host computer:

- free-format mode (enabled by **/h**)
- fixed-format mode (enabled by **/H**)

Logged data is always returned to the host computer in fixed format, but real-time data can be returned in either free format or fixed format.

### Free-Format Mode /h

Also known as “unformatted mode”. This is the DT80's default mode for real-time data return (enabled by the **/h** switch command), in which data and other information is returned to the host computer in a verbose (descriptive, conversational) style suitable for on-screen display and printing. For example, when the DT80 is in free-format mode, the schedule command

```
RA5S 1V 3PT385  
1C("Widgets")
```

returns each data item in the form

```
1V 2.490 mV  
3PT385 395.0 degC  
Widgets 3498 Counts
```

to the host computer screen. The switches listed in the table above default to **/U/N/C**, and parameters P22 and P24 are not used as delimiters while units text is enabled (**/U**).

**User-Definable** In free-format mode, you can use format-related parameter and switch commands to alter the format of returned data and other information to suit your requirements.

## Fixed-Format Mode /H

Also known as “formatted mode”. Recommended for those writing drivers to interface host software with the DT80 (that is, advanced users only). Logged data is always returned to the host in fixed-format mode.

In fixed-format mode (enabled by the `/H` switch command), the following parameters and switches are forced to the states shown in order to ensure a predictable, repeatable, comprehensive format for returning data ready to be imported into spreadsheets and other data analysis software:

States Forced by /H	Effect on Returned Data
<code>P22=44</code>	Each measurement is separated by a comma (,).
<code>P24=13</code>	CR (and LF) is applied to the end of each scan's data.
<code>P38=46</code>	Decimal point character = period (.)
<code>/c</code>	Channel type does not appear in returned data.
<code>/u</code>	Units text does not appear in returned data.
<code>/n</code>	Channel number does not appear in returned data.
<code>/e</code>	Echo (automatic return of sent commands) is turned off.
<code>/r</code>	Real-time data return is turned off.

For example, when the DT80 is in fixed-format mode, the schedule command

```
RA5S
3PT385
```

returns data as shown below.

```
D,081044,"JOB1",2005/03/29,10:53:26,0.007568,1;A,0,22.50564;0060;065F
D,081044,"JOB1",2005/03/29,10:53:27,0.000732,1;A,0,22.50643;0060;3BEB
D,081044,"JOB1",2005/03/29,10:53:28,0.001586,1;A,0,22.43500;0060;EFD2
D,081044,"JOB1",2005/03/29,10:53:29,0.080932,1;A,0,22.43676;0060;300D
D,081044,"JOB1",2005/03/29,10:53:30,0.000732,1;A,0,22.43929;0060;7D84
D,081044,"JOB1",2005/03/29,10:53:31,0.000854,1;A,0,22.47816;0060;8DDA
D,081044,"JOB1",2005/03/29,10:53:32,0.006103,1;A,0,22.46851;0060;1BC1
D,081044,"JOB1",2005/03/29,10:53:33,0.000732,1;A,0,22.42886;0060;5617
D,081044,"JOB1",2005/03/29,10:53:34,0.000732,1;A,0,22.43869;0060;DCC9
D,081044,"JOB1",2005/03/29,10:53:35,0.005371,1;A,0,22.45980;0060;F09F
D,081044,"JOB1",2005/03/29,10:53:36,0.000732,1;A,0,22.43405;0060;37F1
D,081044,"JOB1",2005/03/29,10:53:37,0.000732,1;A,0,22.45734;0060;CDC4
D,081044,"JOB1",2005/03/29,10:53:38,0.004638,1;A,0,22.47549;0060;89B3
D,081044,"JOB1",2005/03/29,10:53:39,0.009887,1;A,0,22.43370;0060;E99A
```

Record Type	Record Index
<b>A</b> Alarm record	Alarm number
<b>D</b> Returned Data record	<b>0</b> = real-time data <b>1</b> = logged data <b>2</b> = unused <b>3</b> = end of data return (end of unload for logged data) <b>4</b> = data discontinuity record (caused by sending one of the LOGOFF or Halt commands, or by changing jobs) <b>6</b> = burst complete (applicable to the DT800) <b>7</b> = burst timeout (applicable to the DT800)
<b>E</b> Error record	Error number
<b>I</b> Information record	Information Number
<b>P</b> Parameter record	Parameter number
<b>S</b> Status record	Status message number
<b>T</b> Test record	Test message number
<b>W</b> PassWord query record	Password message number

Figure 22 Typical logged data records returned in fixed-format

See also Figure 25 (P76).

The parameters and switches listed above are restored to their previous values when the DT80 receives `/h` (sets free-format mode).

**Not User-Definable** In fixed-format mode, returned data and other returned information cannot have the format changed: format-related parameter and switch commands have no effect.

### Numeric Format

To set the numeric format of free-format-mode returned data for individual channels by the following channel options (see Output data format (P37) in the Table 3: DT80 Channel Options (P38) table):

**FFn** Fixed-point format, **n** = number of decimal places (0 to 7)

<b>FE<math>n</math></b>	Exponential format, $n$ = number of significant digits (0 to 7)
<b>FM<math>n</math></b>	Mixed FF or FE formats. Uses FE format if exponent is less than -4 or greater than $n$ .

For example:

Default	FF1	FE3	FM1	FM2
23.456	23.5	2.346e1	23.5	23.46
-0.025	-0.0	-2.542e-2	-0.0	-0.03
1034.6	1034.6	1.035e3	1e3	1034.64

The default format depends on the channel type returning the data (see the Table 1: DT80 Channel Types [\(P29\)](#) table, especially the Resolution column). Formatting options are not applied to the 99999.9 error data code (see [ERROR MESSAGES \(P174\)](#)).

**Fixed Field Width** Parameter 33 [\(P110\)](#) allows returned data to be in fixed fields. All data is placed into fields of the same width (defined by P33), by space-padding to the left. If the field width is not sufficient, least significant characters are truncated from the right. Fixed fields are useful when returned data is to be tabulated, or forwarded to software with a simple string parser.

# Logging Data

## Go for quality, not quantity

The DT80 stores data acquired from input channels and calculations into its internal memory or an external USB memory device.

Data is logged in files within the DT80's file system (see [The DT80 File System \(P71\)](#)). The data from each report schedule is logged into separate files. When the logged data is later unloaded, the data is unloaded for each report schedule in turn.

The DT80 stores approximately 90,000 data values per megabyte of memory. Therefore the DT80's 64MB internal memory can hold approximately 5,000,000 data values, and a USB memory device can hold approximately 90,000 data values per Mbyte. See also [Data Storage Capacity — Readings/MB \(P70\)](#).

Logged data is retained in the internal memory of the DT80 when the DT80 is reset and if its main power supply is removed.

## LOGON and LOGOFF Commands

Globally enable data logging by sending the LOGON command, and globally disable data logging by sending the LOGOFF command. Data logging can also be individually controlled for each of the report schedules by the LOGONx and LOGOFFx commands.

Remember: The DT80's default is data logging disabled.

**LOGON** Enables logging (data and alarms) for all schedules.

**LOGOFF** Disables logging (data and alarms) for all schedules.

**LOGONx** Enables logging for schedule x (data and alarms)

**LOGOFFx** Disables logging for schedule x (data and alarms)

## Disabling Data Logging for Specific Channels

Data from all of the input channels and calculations from all of the report schedules RA, RB, RC,...RK schedules is logged after the general LOGON command is sent to the DT80.

Data from all of the input channels and calculations from a particular report schedule is logged if the schedule-specific LOGONx command is sent to the DT80.

Data logging for specific channels can also be disabled and calculations within report schedules by including the NL (No Log) channel option in their channel lists — see [NL \(P37\)](#). Data is still returned to the host computer for channels with the NL channel option.

See also the [Destination \(P37\)](#) (Working) channel option.

## Logging issues

### How media is selected

By default the internal memory will be used.

The user may also specify the storage media to use on a per-schedule basis. An addition to the schedule definition syntax will facilitate this.

### On USB memory device removal

When the USB memory device is removed from the DT80 any schedules that were logging data to the USB device will keep running, but new records will not be logged until a USB memory device with the necessary free space is inserted.

Once a destination medium has been selected for a schedule, it cannot be altered.

## Inserting and Removing a USB memory device

### Appending a Job's Data

#### Data Storage Issues

- If the user indicates that a schedule should be logged to a USB memory device and the device is not present any schedules that are to be logging to USB memory device will have logging switch off. When a USB memory device with the required amount of space is inserted, logging will commence for all schedules set to log to card.
- When a USB memory device is removed logging will be switched off for all schedules for which data is logged. Logging will resume for these schedules when a USB memory device is inserted.
- When there is insufficient space to create a log file of the size requested, or there is insufficient space for only some of the job's scheduled data logging will be turned on for the schedules for which space is available. For the rest, logging will be switched off until storage space becomes available (a new USB memory device is inserted, or data is deleted).
- Also, the user will be informed by error message of the schedules for which logging will be disabled.
- When a different job is entered that has the same name as the one already in the logger that has data records the new job is rejected. The existing data must be removed before the job can be accepted.

#### Storage Status

The amount of data stored and the amount of free space can be checked in the DT80's internal memory and USB memory device with the following commands and system variables:

<b>STATUS</b>	Line 6 — internal memory kB free/stored data Line 7 — USB memory device kB free/stored data (or STATUS6 and STATUS7)	See STATUS Commands ( <a href="#">P122</a> ).
<b>1SV</b>	Internal memory	kB free
<b>2SV</b>		kB stored data
<b>3SV</b>	USB memory device	kB free
<b>4SV</b>		kB stored data

#### Data Storage Capacity — Readings/MB

Each schedule defined in the DT80 requires 185 Bytes of storage. Every data point (measurement) stored requires 16 bytes and each alarm stored requires 255 bytes.

Use the following table to estimate how many readings per megabyte of data storage the DT80 can be expected to log for each schedule it is running:

Internal Store Capacity 64Mbyte = approx 5,000,000 data points.

Removable USB memory device = approx 90,000 data points per megabyte.

See also Storage Capacity ([P125](#)).

#### Halt and Go During Data Logging

While data logging is in progress, data from each report schedule is progressively stored into its respective data log file (DATA\_A.DXD, for example — see Figure 23 ([P72](#)) and Figure 25 ([P73](#))) in the DT80's internal memory (or USB memory device if one is inserted). Whenever a halt command (Halt all report schedules) is issued during a data logging session, a discontinuity record is written into all currently-open data log files. Similarly, whenever a Hx command (halt Report Schedule x) is issued during a data logging session, a discontinuity record is written into the currently open file for that report schedule.

In unloaded data, a discontinuity record looks like a normal data record except that all data items are set to zero and its record index is set to 4:

```
D,080435,"JOB1",2001/05/17,10:32:23,0.118530,4;A,0,0.00000,0.00000,0.00000,0.00000
```

The discontinuity record indicates that a break occurred in the acquisition of data and is included in data subsequently unloaded from memory.

Whenever a G command (Go all report schedules), or a Gx command (Go report schedule x) is issued after a halt command, data logging resumes in the currently-open data log file(s).

#### Deleting Logged Data

To delete logged data from memory at any time use the following commands:

<b>DELDATA</b>	Deletes the current job's data from internal memory and USB memory device	Job names, directory structures, programs and alarms are not erased.
<b>DELDATA "JobName" (datetime)</b>	Deletes only JobName's data from internal memory and USB memory device. Deletes records with time stamps prior to the specified date. Note: The date and time are in the format of the DT80's current date format (P31) and time format (P39) settings	

<code>DELDATA"JobName"[iso datetime]</code>	Deletes only JobName's data from internal memory and USB memory device. Deletes records with time stamps prior to the specified date. Note: The date and time are in the strict format of YYYY/MM/DD,hh:mm:ss,0.ssssss, which overrides the current date format (P31) and time format (P39) settings
<code>DELDATA*</code>	Deletes data for all jobs from internal memory and USB memory device
<code>FORMAT"B:"</code>	Reformats the internal memory, removing all logged data and alarms. See also the Table 12: DT80 Resets ( <a href="#">P119</a> )
<code>FORMAT"A:"</code>	Reformats an inserted USB memory device, removing all logged data and alarms. See also the Table 12: DT80 Resets ( <a href="#">P119</a> ).

See also Summary — Delete Commands ([P166](#)) for a complete list of DT80 delete commands.

When any of the DELDATA commands are used, the appropriate data log files are deleted and logging continues into new files.

Resetting the DT80 by a soft reset or a firm reset (see Resetting the DT80 ([P119](#))) does not delete logged data.

## Moving , Copying and Archive Logged Data

The MOVEDATA and COPYDATA are used to either move or copy data and alarm files to the removable USB memory device. The ARCHIVE command will save in its current directory a version of the file which has been renamed. It also compresses the file by removing unused space in the file.

**COPYDATA<Job(s)>** Data in the specified job is copied to the USB memory device. This includes any archived data files as well.

Where <Job(s)>	*	All store files for the current job are copied
	"JobName"	All store files for the named job are copied.
	"b:\jobs\jobname\a\data_a.dbd"	The store file at the specified path is copied

**MOVEDATA<Job(s)>** Data in the specified job is moved to the USB memory device. This includes any archived data files as well.

Where <Job(s)>	*	All store files for the current job are moved.
	"JobName"	All store files for the named job are moved
	"b:\jobs\jobname\a\data_a.dbd"	The store file at the specified path is moved.

**ARCHIVE[<Schedule>|<Job(s)>][MOVE]or[COPY]** Data in the specified schedule and job is archived to the current directory.

Where <Schedule>	X,A~K	Archive the specified schedule. Otherwise all schedules are archived
Where <Job(s)>	*	All store files for the current job are archived.
	"JobName"	All store files for the named job are archived.
	"b:\jobs\jobname\a\data_a.dbd"	The store file at the specified path is archived.

**[MOVE]or[COPY]** If this option is not specified then the default is for records to be moved. That is, records are deleted from the source store file. Supplying the MOVE argument is redundant but it does allow the user to be explicit about his intent. If the COPY option is supplied then records in the source store file are not deleted when the archive is created.

For Example:

<code>COPYDATA"Fred"</code>	Copies all data files from the job "Fred" including any archived files to the USB memory device.
<code>ARCHIVE[B *][MOVE]</code>	Archives Schedule B from each job in the logger and clear the data file.

## The DT80 File System

The DT80 uses a DOS-style file system for storing logged information (data and alarms) and system information. The file system is transparent to the user, and it's not necessary to know about it for general use of the DT80, including when logging data to USB memory devices and unloading data using the DT80.

If you intend retrieving data from the USB memory devices (see Retrieving Logged Data — USB memory device Transfer ([P73](#))), some knowledge of the DT80 file system will be beneficial.

The internal memory of the DT80 and the memory in USB memory devices is completely managed by the file system. The internal memory is configured as drive B, and the USB memory device is configured as drive A.

To see the directory structure of the DT80's internal memory by executing a DIRTREE"B:" command from either DeLogger's text window or DeTransfer running on a connected computer. Similarly, see the directory structure of an inserted USB memory device by executing a DIRTREE"A:" command.

### Directory Structure of Internal Memory

Volume in drive B has no label.



```

2005/07/06 21:47 69632 <RO> - FAILSAFE
2005/07/06 21:47 <DIR> - EVENTS
2005/07/06 11:05 1309 - EVENT.LOG
2005/07/06 02:50 1407 - ERROR.LOG
2005/07/06 21:52 <DIR> - INI
2005/07/06 11:05 169 - USER.INI
2005/07/06 21:53 <DIR> - JOBS
2005/07/06 21:53 <DIR> - SATURATE
2005/07/06 21:53 74 - PROGRAM.DXC
2005/07/06 21:53 3160 - STATUS14
10 File(s) 1970176 Bytes free

```

Figure 23: Typical structure — DT80 internal data memory (send DIRTREE"B:")

When the logger is in formatted mode (/H see ...) the dirtree returns the following format.

```

L,<serial>,yyyy/mm/dd,hh:mm:ss,0.subsec,0;<is_dir>,<rd_only>,<size>,<date>,<time>,"<name>";<cc>;<cs>
AND
L,<serial>,yyyy/mm/dd,hh:mm:ss,0.subsec,1;<n_lines>,<bytes_free>,"<volume_label>";<cc>;<cs>

```

where

<serial>	is the logger's serial number
<is_dir>	is 1 if the file system entry is a directory, 0 if it is not.
<rd_only>	is 1 if the file system entry is read only, 0 if it is not
<size>	is only present for files. It is the file's size in bytes
<date>	is the file system entry's creation date in yyyy/mm/dd form.
<time>	is the file system entry's creation time in hh:mm:ss form.
<name>	is the file system entry's long name and absolute path.
<n_lines>	is the number of file system entries listed.
<bytes_free>	is the number of bytes free on the medium that was just listed.
<volume_label>	is the volume's (medium's) label.
<cc>	is the character count of the record up to and including the ';' just before <cc>
<cs>	is a 16 CRC of the characters of the record up to and including the ';' just before <cs>.

The data log files are stored in an HFS (hierarchical file structure) as follows:

- within the root directory there is a JOBS subdirectory — see Figure 23 (P72)
- within the JOBS subdirectory there is a subdirectory for each of the jobnames entered into the logger (JOB1, JOB2,...) Figure 23 (P72)
- within each job subdirectory there is a subdirectory for each of the report schedules in the job (A, B, C,...) see Figure 23 (P72)
- within each schedule subdirectory there is
- a logged data and alarms file DATA\_S.DBD, where S is the schedule letter (A, B,...K) and .DBD is the data file extension

Other directories and files are explained in Figure 23 (P72) and Figure 25 (P73).

### Directory Structure of USB memory devices

The same directory structure is used in DT80 USB memory devices, except that it's all contained in a higher order directory (folder) named SNxxxxxx, where xxxxxx is the serial number of the DT80 — see Figure 25 (P73). If the USB memory device has been inserted into more than one DT80, the card contains a serial number directory for each DT80.

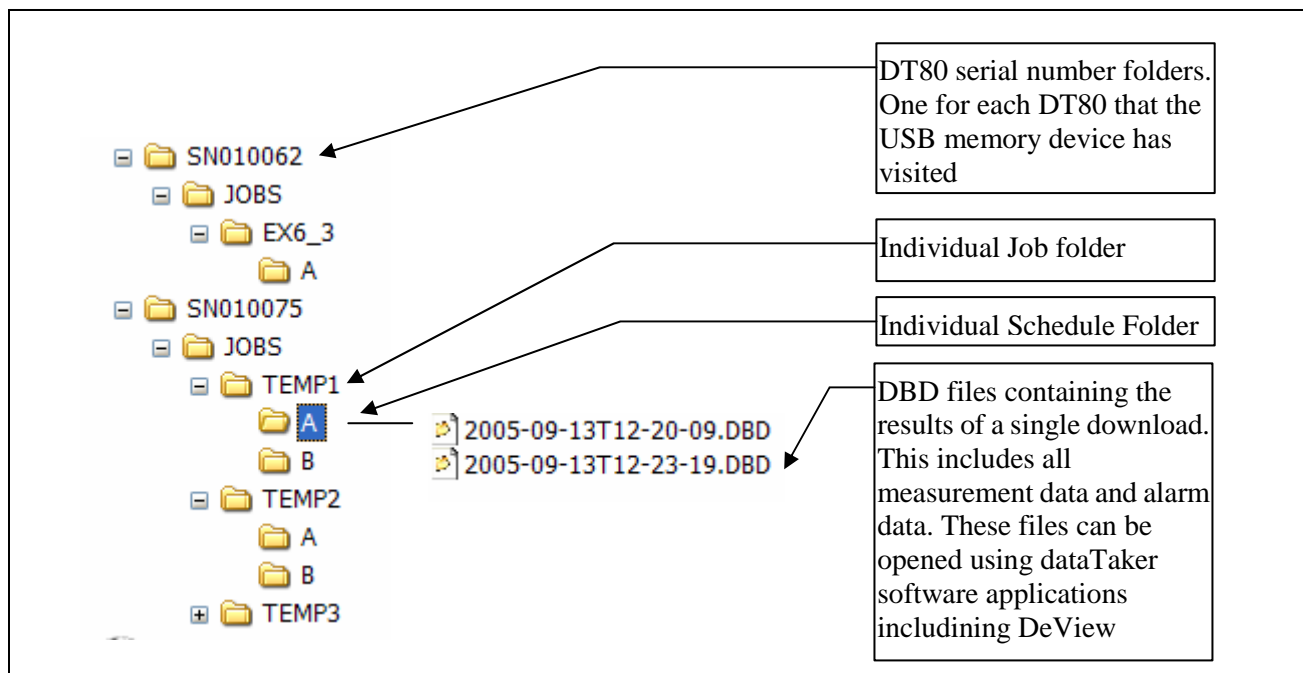


Figure 24: Typical structure — DT80 USB memory device (Explorer-style view)

To retrieve data, USB memory devices can be read directly by a Windows computer with a USB interface.

### Size of Data Files

To see the size of logged data and logged alarms files send the DIRTREE"B:" command (internal memory) or DIRTREE"A:" command (USB memory device) to the DT80, or by inserting the USB memory device into PC and looking at the file size in Windows Explorer.

Alternatively, Storage Status [\(P70\)](#) explains how to get similar information.

## Retrieving Logged Data

Logged data from the *DT80* can be retrieved by

- unloading the data to a host computer through one of the *DT80*'s communications interfaces (this does not remove the data from the internal memory or USB memory device)
- removing an inserted USB memory device from the data logger and reading this in a computer with a USB port
- Inserting a USB memory device into the *DT80* to extract data stored in internal memory, then removing the USB memory device and reading this in a computer (this may remove the data from internal memory).

See also **Logging and Retrieving Alarms** [\(P84\)](#).

---

### Retrieving Logged Data — USB memory device Transfer

When a USB memory device is plugged in to the *DT80* it is immediately detected then the option is given on the display to download data from the internal memory to the external memory device.

If the USB memory device contains a suitable ONINSERT.DXC [\(P116\)](#) file, the ONINSERT.DXC startup job is transferred from the card to the *DT80* and run (becomes the *DT80*'s current job).

---

### Retrieving Logged Data — Comms Unload

Commands for retrieving data from a *DT80*'s internal memory or card to a host computer using one of the *DT80*'s communications interfaces are presented in Unload Commands [\(P74\)](#) next, and summarized in the Summary — Retrieval Commands [\(P166\)](#) table.

During an unload, the **/r** (return), **/e** (echo), **/m** (error messages) and **/z** (alarm messages) switches are disabled to prevent transmissions from these sources being inserted into the unload data stream. The *DT80* automatically sets these switches to their previous state on completion of the unload.

**Note:** Logged data is not cleared from the *DT80* when this command method “comms interface / unload commands” is used for retrieving data. The same logged data can be unload using one of the *DT80*'s communications interfaces as many times as required (until it is deleted on purpose. — see Deleting Logged Data [\(P70\)](#)).

Conversely, logged data is cleared from the *DT80* when the “temporarily insert a USB memory device” method of retrieving data is used. (discussed earlier).

## Always Fixed-Format Mode

When logged data is unloaded, it's always returned to the host computer in fixed-format mode — see Figure 25 (Real-time data can be returned in either free- or fixed-format mode.) [\(P76\)](#)

## Order of Unload

Logged data is always unloaded schedule-by-schedule in the order A, B, C,...K unless you specify a particular schedule — see Unload Commands [\(P74\)](#) below).

## Unload Commands

To retrieve logged data from the *DT80* by means of one of its comms interfaces use the **unload commands** below. There are three formats of unload commands:

<b>U</b> commands	Unload <b>all</b> data from beginning to end of data store	See The U Unload Commands <a href="#">(P74)</a> below.
<b>U( )</b> commands (round brackets / parentheses)	Unload data for a <b>period</b> defined by a beginning and end date and time in the <b>format</b> of the <i>DT80</i> 's current date format (P31) and time format (P39) settings	See The U( ) Unload Commands <a href="#">(P74)</a> below.
<b>U[ ]</b> commands (square brackets)	Unload data for a <b>period</b> defined by a beginning and end date and time in the <b>strict format</b> of YYYY/MM/DD, hh:mm:ss,0.ssssss, which overrides the current date format (P31) and time format (P39) settings	See The U[ ] Unload Commands <a href="#">(P75)</a> .

## The U Unload Commands

The **U** unload commands return all logged data — that is, from the first record to the last record — for a specified job and report schedule.

Here are the *DT80*'s **U** commands:

<b>U</b>	Returns all of the current job's logged data in the order of report schedule A–K
<b>Ux</b>	Returns all of the current job's logged data for report schedule <b>x</b>
<b>U"JobName"</b>	Returns all logged data for <b>JobName</b> in the order of report schedule A–K
<b>U"JobName"x</b>	Returns all logged data for <b>JobName</b> for report schedule <b>x</b>

## Examples — U Commands

The command

**U**

instructs the *DT80* to return all of the current job's data, schedule-by-schedule.

The command

**UC**

instructs the *DT80* to return all of the current job's data for report schedule C only.

## The U( ) Unload Commands

The **U( )** unload commands return logged data for a specified job and report schedule, for a period of time designated by a beginning and end **time** and **date** that must be specified in the *DT80*'s current date format (Parameter 31) and time format (Parameter 39) settings.

Here are the *DT80*'s **U( )** commands:

<b>U</b>	Returns the current job's logged data in the order of report schedule A–K
<b>U( from)</b>	Returns the current job's logged data starting from <b>BEGIN, time</b> or <b>time, date</b>
<b>U( from)( to)</b>	Returns the current job's logged data starting from <b>BEGIN, time</b> or <b>time, date</b> and ending with <b>END, time</b> or <b>time, date</b>
<b>Ux</b>	Returns the current job's logged data for report schedule <b>x</b>
<b>Ux( from)</b>	Returns the current job's logged data for report schedule <b>x</b> starting from <b>BEGIN, time</b> or <b>time, date</b>
<b>Ux( from)( to)</b>	Returns the current job's logged data for report schedule <b>x</b> starting from <b>BEGIN, time</b> or <b>time, date</b> and ending with <b>END, time</b> or <b>time, date</b>
<b>U"JobName"</b>	Returns <b>JobName</b> 's logged data in the order of report schedule A–K
<b>U"JobName"( from)</b>	Returns <b>JobName</b> 's logged data starting from <b>BEGIN, time</b> or <b>time, date</b>
<b>U"JobName"( from)( to)</b>	Returns <b>JobName</b> 's logged data starting from <b>BEGIN, time</b> or <b>time, date</b> and ending with <b>END, time</b> or <b>time, date</b>
<b>U"JobName"x</b>	Returns <b>JobName</b> 's logged data for report schedule <b>x</b>
<b>U"JobName"x( from)</b>	Returns <b>JobName</b> 's logged data for report schedule <b>x</b> starting from <b>BEGIN, time</b> or <b>time, date</b>
<b>U"JobName"x( from)( to)</b>	Returns <b>JobName</b> 's logged data for report schedule <b>x</b> starting from <b>BEGIN, time</b> or <b>time, date</b> and ending with <b>END, time</b> or <b>time, date</b>

where

<i>from</i>	can be <b>BEGIN</b> — start from first data point logged <i>time</i> — start from first data point logged at or after this time today <i>time, date</i> — start from first data point logged at or after this time and date	Both the <i>time</i> and <i>date</i> must be specified in the currently-defined format for date (P31) and time (P39).
<i>to</i>	can be <b>END</b> — finish with last data point logged <i>time</i> — end with last data point logged prior to this time today <i>time, date</i> — end with last data point logged prior to this time and date	

**Note:** **BEGIN** and **END** used here are not the same as the **BEGIN** and **END** keywords used to indicate the start and end of a *DT80* job.

### Examples — U( ) Commands

The command

```
UA(BEGIN)(11:15)
```

instructs the *DT80* to unload schedule **A**'s logged data from the first data point stored (**BEGIN**) until **11:15**.

The command

```
UB(12:00,19/1/2000)(12:05,20/1/2000)
```

instructs the *DT80* to unload schedule **B**'s logged data starting at **12:00** on the **19/1/2000** and ending at **12:05** on the **20/1/2000**.

### Examples — U( ) Shortcuts

The command

```
UC(13:21)(13:21)
```

instructs the *DT80* to return only the current job's data for schedule **C** that was logged during the minute **13:21** (that is, from 13:21:00 to 13:21:59).

The command

```
UD(13)(13)
```

instructs the *DT80* to return only the current job's data for schedule **D** that was logged during the hour **13** (that is, from 13:00:00 to 13:59:59).

### The U[ ] Unload Commands

The **U[ ]** unload commands return logged data for a specified job and report schedule, for a period of time designated by a beginning and end *time* and *date* that must be specified in the strict format **YYYY/MM/DD, hh:mm:ss,0.ssssss** (ISO date format), which overrides the current date format (Parameter 31) and time format (Parameter 39) settings.

Here are the *DT80*'s **U[ ]** commands:

<b>U</b>	Returns the current job's logged data in the order of report schedule <b>A–K</b>
<b>U[ <i>from</i> ]</b>	Returns the current job's logged data starting from <i>time</i> or <i>date, time</i>
<b>U[ <i>from</i> ][ <i>to</i> ]</b>	Returns the current job's logged data starting from <i>time</i> or <i>date, time</i> and ending with <i>time</i> or <i>time, date</i>
<b>Ux</b>	Returns the current job's logged data for report schedule <b>x</b>
<b>Ux[ <i>from</i> ]</b>	Returns the current job's logged data for report schedule <b>x</b> starting from <i>time</i> or <i>date, time</i>
<b>Ux[ <i>from</i> ][ <i>to</i> ]</b>	Returns the current job's data for schedule <b>x</b> starting from <i>time</i> or <i>time, date</i> and ending with <i>time</i> or <i>date, time</i>
<b>U"JobName"</b>	Returns <i>JobName</i> 's logged data in the order of report schedule <b>A–K</b>
<b>U"JobName"[ <i>from</i> ]</b>	Returns <i>JobName</i> 's logged data starting from <i>time</i> or <i>date, time</i>
<b>U"JobName"[ <i>from</i> ][ <i>to</i> ]</b>	Returns <i>JobName</i> 's logged data starting from <i>time</i> or <i>time, date</i> and ending with <i>time</i> or <i>date, time</i>
<b>U"JobName"x</b>	Returns <i>JobName</i> 's logged data for report schedule <b>x</b>
<b>U"JobName"x[ <i>from</i> ]</b>	Returns <i>JobName</i> 's logged data for report schedule <b>x</b> starting from <i>time</i> or <i>time, date</i>
<b>U"JobName"x[ <i>from</i> ][ <i>to</i> ]</b>	Returns <i>JobName</i> 's logged data for report schedule <b>x</b> starting from <i>time</i> or <i>time, date</i> and ending with <i>time</i> or <i>date, time</i>

where

<i>from</i>	can be <i>time</i> — start from first data point logged at or after this time today <i>date, time</i> — start from first data point logged at or after this time and date	Type <i>time</i> and <i>date</i> in fixed-format ISO- style (see examples below).
<i>to</i>	can be <i>time</i> — end with last data point logged prior to this time today <i>date, time</i> — end with last data point logged prior to this time and date	

### Examples — U[ ] Commands

Here's the **U[ *from* ]** command showing valid forms of fixed-format mode date and time:

```
U[2000/07/26,00:00:01,0.250366]
U[2000/07/26,00:00:01]
U[2000/07/26]
```

Here's the **U[ *from* ][ *to* ]** command showing valid forms of fixed-format mode date and time:

```
U[2000/07/26,00:00:01,0.250366][2000/07/26,00:00:01,0.750244]
U[2000/07/26,00:00:01][2000/07/26,00:00:01,0.750244]
U[2000/07/26][2000/07/26,00:00:01,0.750244]
U[2000/07/26,00:00:01,0.250366][2000/07/26,00:00:01]
U[2000/07/26,00:00:01][2000/07/26,00:00:01]
U[2000/07/26][2000/07/26,00:00:01]
U[2000/07/26,00:00:01,0.250366][2000/07/26]
U[2000/07/26,00:00:01][2000/07/26]
U[2000/07/26][2000/07/26]
```

### Labelling the End of Unloaded Data

As *Figure 25* shows, the DT80 automatically includes special records when it returns logged data to the host computer. These records signify

- the end of the unload of an individual schedule's data — **end-of-schedule record**
- the end of a complete unload (may contain one or more schedules) — **end-of-unload record**.

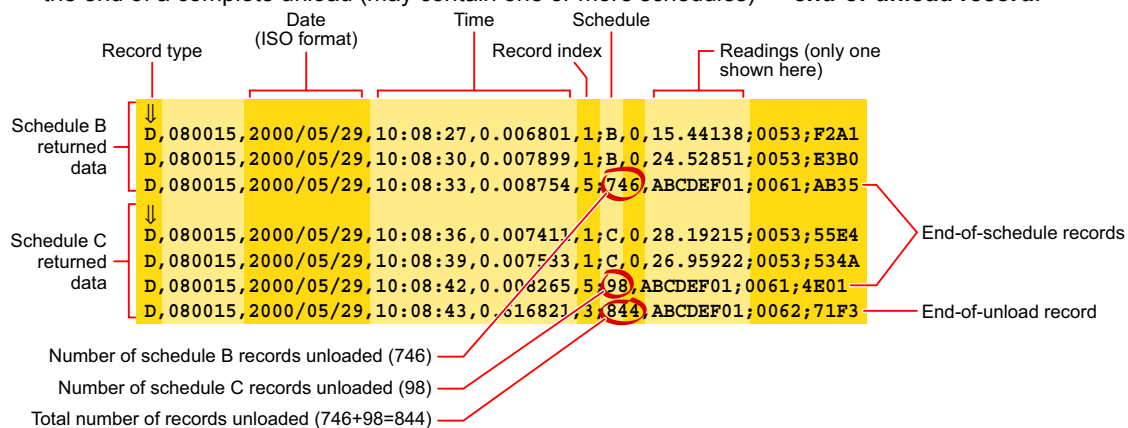


Figure 25 Typical output data

\*always returned in fixed-format mode

The records are added even if the unload process is aborted before completion (by sending the **Q** command — see next topic).

### Quitting an Unload

Stop an unload operation by sending the Quit Unload command

**Q**  
to the DT80.

There may be a short delay between sending the command and the return of data actually stopping. This is because the output buffer of the DT80 is generally full during an unload and these records have still to be returned to the host. (Sending **Q** actually stops the copying of stored data into the DT80's output buffer.)

# Part G — Alarms

## Alarm Concepts

### Limits, tests and actions

**DT80 alarms** allow decisions to be made based on the magnitude of DT80 input channels, channel variables, timers, the clock/calendar, internal channels, system variables and so on. The decision is a **true** or **false** result of an alarm **condition test**. The true/false result is also known as the alarm **state**.

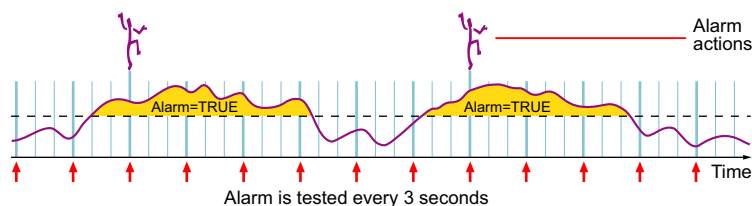
Instruct the DT80 to carry out actions when an alarm tests true. These actions can be setting the DT80's digital state outputs, issuing messages, or executing commands to change the DT80's operation.

There are two types of alarm commands (P77):

- the **ALARM** command — acts once on the transition of the alarm test from false to true (“single-shot” alarm)
- the **ALARMR** command — acts Repeatedly at the schedule interval while the alarm tests true (“repeating” alarm)

Single-shot alarm **RA3S ALARM...**

Alarm actions occur once when the alarm becomes true.



Repeating alarm **RA3S ALARMR...**

Alarm actions occur every 3 seconds while the alarm is true.

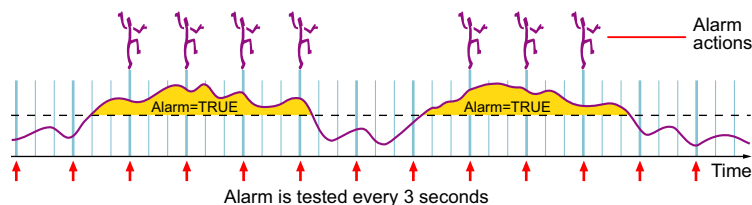


Figure 26: Comparing single-shot and repeating alarms (3-second schedule example)

Alarm commands can be included in any report schedule, and are processed in sequence with other schedule processes such as reading input channels and performing calculations. If an alarm tests true, the alarm's actions are executed before control passes to the next process in the schedule.

The general format of the single-shot alarm command is

```
ALARMn(test)digitalAction"actionText"{actionProcesses}
```

and of the repeating alarm command is

```
ALARMRn(test)digitalAction"actionText"{actionProcesses}
```

where

<i>n</i>	is the alarm number. <i>n</i> is optional. See Alarm Number (P78).
<i>test</i>	is the alarm's true/false test. It is the DT80 input to be tested (see Alarm Input (P78)), followed by the test condition (see Alarm Condition (P79)). An optional delay period can be included in <i>test</i> (see Alarm Delay Period (P79)).
Actions	<i>digitalAction</i> is one or two digital state output channels whose state mimic the alarm state. See Alarm Digital Action Channels (P80). <i>digitalAction</i> is optional. These actions can be replaced by a logical operator (AND, OR or XOR) to combine more than one alarm.

<code>actionText</code>	is a text message that is returned to the host and/or logged when alarm tests true. See Alarm Action Text <a href="#">(P80)</a> . <code>actionText</code> is optional. <b>Note:</b> If single quotes are used instead of the double quotes then the text is sent only to the serial port. This is useful when we need to communicate with modems in command mode.
<code>actionProcesses</code>	is a list of <i>DT80</i> commands that are executed when alarm tests true. See Alarm Action Processes <a href="#">(P82)</a> . <code>actionProcesses</code> is optional.

### Example — Typical Alarm Command

The alarm command

```
ALARM1 (2V>1000) 1DSO"HighVoltage" {5CV=1 HA RB1M}
```

(which you include in any report schedule) contains the following:

<b>ALARM1</b>	is the single-shot alarm command ( <b>ALARM</b> ) and an alarm number ( <b>1</b> ).
<b>(2V&gt;1000)</b>	is the alarm true/false <i>test</i> of the input ( <b>2V</b> ) against the condition ( <b>&gt;1000</b> mV). If, at the moment of testing, the voltage on channel 2 is greater than or equal to 1000mV ( <b>2V&gt;1000</b> ), the alarm is true. Otherwise, the alarm is false.
<b>1DSO</b>	is the <i>digitalAction</i> . It instructs the <i>DT80</i> to set its digital state output channel 1 ON if the alarm is true and OFF if the alarm is false.
<b>"HighVoltage"</b>	is <i>actionText</i> . The <i>DT80</i> returns and/or logs this message when the alarm becomes true.
<b>{5CV=1 HA RB1M}</b>	is the <i>actionProcesses</i> , a list of commands that the <i>DT80</i> executes when the alarm becomes true.

### Alarm Number

Shown as *n* in the alarm command (above); optional.

Each of the two alarm types can be given either

- a number in the range 1 to 32767 — that is, **ALARMn** or **ALARMRn** where *n* is the alarm number (for example, **ALARM37**), or
- no number — that is, **ALARM** or **ALARMR**

The following table compares the functionality of alarms with and without numbers:

Alarms with Alarm Number Examples: <b>ALARM10</b> , <b>ALARMR99</b>	Alarms without Alarm Number <b>ALARM</b> or <b>ALARMR</b>
Do not have to be entered in numeric sequence	Order doesn't matter
The alarm number, entry time and exit time are logged in the <i>DT80</i> 's general data store.	No alarm information is logged
<b>?n</b> (see Polling Alarm Data <a href="#">(P84)</a> ) returns the current value of the tested channel.	<b>?n</b> returns an undefined alarm error.
<b>?x</b> and <b>?ALL</b> (see Polling Alarm Data <a href="#">(P84)</a> ) return all current values.	<b>?x</b> and <b>?ALL</b> have no effect.

### Alarm Input

Part of *test* in the alarm command (**ALARMn(test)digitalAction"actionText"{actionProcesses}** [\(P77\)](#)); not optional.

The **alarm input** is the data item or value that is to be tested by the alarm command. Alarm inputs for alarm commands can be any of the following:

- any analog, digital, counter or serial input channel definition (channel number + channel type + channel options)
- any internal channel
- any channel variable
- time and date
- any system timer
- any system variable

### Examples — Alarm Input

The alarm command

```
ALARMn (2R>50) actions...
```



tests the **2R** analog input channel.

The alarm command

```
ALARMn(REFT<100)actions...
```

tests **REFT**, the *DT80*'s body temperature.

The alarm command

```
ALARMn(T>10:30:00)actions...
```

tests the *DT80*'s internal time channel (**T** channel type — see Time (P30)).

The alarm command

```
ALARMn(2SV>2000)actions...
```

tests **2SV**, the free space in the *DT80*'s internal data store (see System Variables (P31)).

## Alarm Condition

Part of **test** in the alarm command (`ALARMn(test)digitalAction"actionText"{actionProcesses}` (P77)); not optional.

The alarm input is compared with the **alarm condition**, which is either

- one of the logical operators **<** or **>** followed by a single setpoint, or
- one of the logical operators **<>** or **><** followed by a pair of comma-separated setpoints.

Operator	Number of Setpoints	Operation
<b>&lt;</b>	1	Less than the setpoint
<b>&gt;</b>	1	Greater than or equal to the set-point
<b>&lt;&gt;</b>	2	Less than the first set-point, OR greater than or equal to the second set-point
<b>&gt;&lt;</b>	2	Greater than or equal to the first set-point AND less than the second set-point (that is, between the two setpoints)

The setpoints can be floating-point constants or channel variables.

**Recommendation:** When testing digital state inputs for state 0 or 1, it's best to test for state 0 using (`nDS<0.5`) and to test for state 1 using (`nDS>0.5`). This avoids any hunting around the 0 and 1 values.

## Examples — Alarm Condition

The alarm command

```
ALARMn(5TK<100)actions...
```

tests if temperature measured on channel 5 (**5TK**) is less than 100°C (**<100**).

The alarm condition

```
ALARMn(2R>50)actions...
```

tests if resistance on channel 2 (**2R**) is greater than or equal to 50 ohms (**>50**).

The alarm command

```
ALARMn(10CV<>20CV,30CV)actions...
```

tests if the value of **10CV** is less than the value of **20CV**, or greater than or equal to the value of **30CV** — that is, if the value of **10CV** is outside the range of **20CV** to **30CV**.

The alarm command

```
ALARMn(3CV>5.5,8.5)actions...
```

tests if the value of **3CV** is greater than or equal to 5.5 and less than 8.5 — that is, if the value of **3CV** is between 5.5 and 8.5.

The alarm command

```
ALARMn(T><06:30,18:30)actions...
```

tests if the *DT80*'s time is between 06:30:00 and 18:30:00.

## Alarm Delay Period

An addition to **test** in the alarm command

(`ALARMn(test)digitalAction"actionText"{actionProcesses}` (P77)); optional.

The test component of the alarm command can be extended by adding a **delay period**. If this is done, the alarm's test must remain true for the delay period before any of the actions can be performed. The format is

```
/NS Seconds
```

```
/NM Minutes
```

```
/NH Hours
```

```
/ND Days
```

where **N** is an integer in the range 1 to 255.

If the alarm tests false again during the delay period, the delay counter is reset and does not begin counting again until the next true test. The result is a filtering action that ensures input noise does not cause unwanted or rapid output actions.

## Example — Delay Period

The alarm command

`ALARMn(3V>100.0/30S)actions..`

specifies that the voltage on channel 3 (**3V**) must equal or exceed 100.0mV (**>100.0**) for 30 seconds before the actions are performed (**/30S**).

## Alarm Digital Action Channels

Shown as `digitalAction` in the alarm command

(`ALARMn(test)digitalAction"actionText"{actionProcesses}` (P77)); optional.

One — or two (comma-separated) — digital state output channels (`nDSO`) can be declared for each alarm to mimic the state of the alarm. That is, these outputs are set to their default state if the alarm tests false, and are set to their non default state if the alarm tests true:

Alarm	DSO	
False	OFF (default state)	1..4DSO = 1 (DSO is set high) 5..8DSO = 0 (DSO is set low)
True	ON (non default state)	1..4DSO = 1 (DSO is set low) 5..8DSO = 0 (DSO is set high)

Typically, use the digital state outputs to announce the *DT80* alarm by switching devices such as relays, sirens and lights, or to directly control actuators and similar equipment.

In the `ALARM..` command, list the single digital state output, or the pair of comma-separated digital state outputs, immediately after the alarm `test`.

A digital action can also be a channel variable e.g. `ALARM(...)1CV,2DSO` is valid. If the alarm is true then the channel variable (**1CV**) will be set to 1 (true), if the alarm is FALSE then channel variable (**1CV**) will be set to 0 (false).

### Examples — Digital Action

The command

`ALARMn(2V>660.0)4DSO`

tests the voltage on channel 2 (**2V**) and

- turns digital state output 4 ON when the voltage equals or exceeds 660.0mV
- turns digital state output 4 OFF when the voltage drops below 660.0mV.

The command

`ALARMn(2V>660.0)7DSO,8DSO`

tests the voltage on channel 2 (**2V**) and

- turns the two digital state output channels ON when the voltage equals or exceeds 660.0mV
- turns the two digital state output channels OFF when the voltage drops below 660.0mV.

## Alarm Action Text

Shown as `actionText` in the alarm command

(`ALARMn(test)digitalAction"actionText"{actionProcesses}` (P77)); optional.

**Action text** is automatically returned to the host computer and/or logged to alarm memory

- once whenever a single-shot alarm (`ALARM` or `ALARMn`) tests true, or
- repeatedly at the controlling schedule's rate while a repeating alarm (`ALARMR` or `ALARMRn`) remains true.

If the action text is enclosed in single quotes instead of double quotes then the `actionText` is sent exclusively to the RS232 Host Port on the logger. This is useful for communicating with modems when they are in command mode and when the host port is used for other purposes.

Up to 200 characters of action text can be included in each alarm. A total text space of 16384 characters is reserved for all alarms (shared with expressions text).

**Note:** There is no garbage collection in this text space. That is, new text is appended to existing text in the text space, and superseded text is only removed by a system reset.

The action text is listed in quotes " " as follows:

`ALARMn(test)"actionText"`

Setting the alarm message switch to `/z` stops the return of the action text to the host — see the Table 10: *DT80* Switches (P113) table. This is useful when the action text is only required for a display (if fitted).

The format of the action text returned by an alarm differs for the default free-format mode (`/h`) and fixed-format mode (`/H`). See the examples below.

### Substitution Characters

Special substitution characters can be placed into `actionText`. These instruct the *DT80* to dynamically insert the following information when the alarm returns and/or logs its action text:

Characters	Example
!	Substitutes <i>DT80</i> serial number followed by a colon (:) and the alarm number <code>080035:8</code>

?C OR ?c	Substitutes channel ID	2PT385
?N OR ?n	Substitutes user channel name	Boiler
?U OR ?u	Substitutes user channel units	Deg C
?V OR ?v OR ?	Substitutes the data value when the alarm tested true	100.1
@	Substitutes the time that the action text was returned (in P39 and P40 format)	12:13:14.634
#	Substitutes the date that the action text was returned (in P31 format)	11/2/2001
?R OR ?r	Substitutes relation	>50.0
??	Substitutes question mark	?
!!	Substitutes exclamation mark	!
@@	Substitutes @ symbol	@
##	Substitutes # symbol	#
?ncv	Substitutes the current value of the specified channel variable, where <i>ncv</i> is the number of the channel variable (1 to 500). For example, ?3 instructs the DT80 to substitute the contents of 3CV into the alarm action text. You can also specify the format and number of decimal places (see Output Format (P37)). For example: ?3F inserts the value of 3CV in fixed-point format ?3E inserts the value of 3CV in exponential format ?3M1 inserts the value of 3CV in mixed format with 1 decimal place	

Some of these characters are especially useful with SMS messaging.

Any of the ASCII control characters (^A to ^Z) can also be included in action text. Some useful control characters are

^G	Bell
^M	Carriage return
^J	Line feed
^b	Quotation mark ("")

### Example — Action Text in Free-Format Mode (/h)

When the DT80 is in free-format mode (P21) the action text in the alarm command

```
ALARM8(test)"Boiler Pressure is ?V?U at @^M^J"
```

instructs the DT80 to return an alarm record of the form

```
Boiler Pressure is 1.563MPa at
14:32:01.23964
```

on each false-to-true transition of the alarm (includes current value, units text and time that alarm occurred). No action text is issued on the true-to-false transition.

### Example — Action Text in Fixed-Format Mode (/H)

When the DT80 is in fixed-format mode (P21) the action text in the same alarm command

```
ALARM8(test)"Boiler Pressure is ?V?U
at @^M^J"
```

instructs the DT80 to return an alarm record<sup>9</sup> of the form

```
A,080035,"Boiler_1",2001/04/16,14:32:01.25487,8;B,1,"Boiler Pressure is 1.563MPa at
14:32:01.23964^M^J";0102;3D95
```

on each false-to-true transition of the alarm (includes current value, units text and time that alarm occurred)

where

A	signifies that this is an Alarm record
080035	is the DT80's serial number
"Boiler_1"	is the name of the job containing the alarm
2000/04/16,14:32:01.25487	is the date and time of the alarm record
8	is the alarm number (P78)
B	is the name of the schedule containing the alarm
1	is the alarm state: 1=false to true, 2=continuing true, 3=true to false — see Alarm States and Tags (P85)
"Boiler Pressure is 1.563KPa at 14:32:01.23964^M^J"	is the action text

<sup>9</sup> See also [Figure 10](#).

No action text is issued on the true-to-false transition.

## Alarm Action Processes

Shown as `actionProcesses` in the alarm command (ALARMn(test)digitalAction"actionText"{actionProcesses} (P77)); optional.

**Action processes** can be any *DT80* functions to be executed when an alarm is true. These functions can be reading input channels, setting output channels, calculations, setting parameters and switches, and so on.

In addition, action processes are a very powerful programming facility for the *DT80*. Use them to perform a wide range of program-related functions such as re-programming on events, adaptive schedules (see examples below), programmed calibration cycles, management of digital state outputs, and management of the Serial Channel.

Immediate schedules can be included in action processes.

**Note** Action processes cannot include alarm commands or new report schedules.

Place action processes within the bracket sequence { } of an alarm command. Action processes are executed

- once when an **ALARM** or **ALARMn** tests true, or
- repeatedly at the controlling schedule's rate while an **ALARMR** or **ALARMRn** remains true.

Action processes have many uses. Some common ones are demonstrated in the following examples.

### Example — Alarm Action Processes: Switching an Actuator

Probably the most traditional use of action processes is to manage the *DT80*'s digital state output channels, which in turn manage devices connected to them (relays for switching power to lights or sirens to annunciate an alarm condition, to turn pump motors on or off,...).

Every second (1S), the schedule

**RA1S**

```
ALARM(2I<12.54)"Pump ON"{3DSO(W)=1}
```

tests the alarm, whose input is an external-shunt current-type level sensor mounted in a tank and connected to the *DT80*'s analog channel 1 (2I — see I channel type on I (P26)). If the tank level falls below the setpoint (2I<12.54) the alarm tests TRUE, which causes the *DT80* to

- return and/or log the alarm message **Pump ON**
- set its digital state output 3 to 0 (OFF = high/active) to turn a pump on ({3DSO(W)=0} (P144))

This can also be achieved using an alarm digital action (see Alarm Digital Action Channels (P80)) instead of an alarm action process:

```
ALARM(1I<12.54)1DSO
```

### Example — Alarm Action Processes: Controlling a System

Alarm action processes can also be used to control a system or process. Although this method uses simple "bang-bang" actuator control, it is often adequate.

The schedule

**RA1S**

```
ALARM(1TK<74.75)"Heater ON"{1DSO(W)=0}
ALARM(1TK>75.25)"Heater OFF"{1DSO(W)=1}
```

is a simple heater control for a water bath. It assumes that the thermal inertia of the system is sufficient to prevent hunting about the control point. The two alarms work to hold the temperature at 75°C ± 0.25°C.

### Examples — Alarm Action Processes: Adaptive Scheduling

Adaptive scheduling is the dynamic adjustment of the acquisition of data about a system or process as the system or process changes.

As the examples below show, adaptive scheduling can reduce total data volume while giving greater time resolution when required.

The schedule

**RA15M**

```
1V("Wind speed",S1,=1CV)
ALARM1(1CV>5.0){RA2M}
ALARM2(1CV<4.5){RA15M}
```

measures wind speed

- every 2 minutes if wind speed is greater than 5m/s, or
- every 15 minutes if windspeed is less that 5m/s

as follows:

```
1V("Wind speed",S1,=1CV) is the instruction to record the wind speed and assign it to channel variable 1 for testing
```

<code>ALARM1(1CV&gt;5.0){RA2M}</code>	changes the schedule's trigger to every 2 minutes ( <code>{RA2M}</code> ) if windspeed exceeds 5.0m/s ( <code>(1CV&gt;5.0)</code> )
<code>ALARM2(1CV&lt;4.5){RA15M}</code>	changes the schedule's trigger back to every 15 minutes ( <code>{RA15M}</code> ) if windspeed drops below 4.5m/s ( <code>(1CV&lt;4.5)</code> )

Note the deliberate 0.5m/s hysteresis to prevent oscillation around the switchover point.

The program

```
RC30M LOGONC HC
  5TK("Oven Temp")
RD1M
ALARM(5TK>120){GA}
ALARM(5TK<120){HA}
```

continuously monitors the temperature of an oven and logs the temperature whenever it exceeds 120°C:

<code>RC30M LOGONC HC</code>	is a report schedule that logs ( <code>LOGONC</code> ) oven temperature ( <code>5TK</code> ) when active (initially halted by <code>HC</code> )
<code>RD1M</code>	initiates the alarm tests every minute ( <code>1M</code> )
<code>ALARM(5TK&gt;120){GC}</code>	sets schedule C to Going ( <code>{GC}</code> ) if oven temperature exceeds 120°C ( <code>(5TK&gt;120)</code> )
<code>ALARM(5TK&lt;120){HC}</code>	sets schedule C to Halted ( <code>{HC}</code> ) if oven temperature drops below 120°C ( <code>(5TK&lt;120)</code> )

### Example — Alarm Action Processes: Using an Alarm to Poll a Schedule

The program

```
BEGIN
  1CV(W)=0
  5CV(W)=1
  6CV(W)=2^5CV
  RA1S
    1CV(W)=1CV+1
    ALARM(1CV>6CV){XB 5CV(W)=5CV+1 6CV(W)=2^5CV}
  RBX LOGONB
    1..5TK
END
```

logs data at increasing intervals as the experiment proceeds. The program calculates the next log point as an incrementing power of 2 seconds — that is, it logs the temperatures at 1, 2, 4, 8, 16, 32, 64, 128, 256,... seconds as follows:

<code>RA1S</code>	is a simple one-second accumulator
<code>1CV(W)=1CV+1</code>	
<code>ALARM(1CV&gt;6CV)</code>	tests if the next log point has been reached
<code>{</code>	indicates start of <i>actionProcesses</i>
<code>  XB</code>	polls report schedule B
<code>  5CV(W)=5CV+1</code>	increments the exponent
<code>  6CV(W)=2^5CV</code>	calculates the next log point
<code>}</code>	indicates end of <i>actionProcesses</i>
<code>RBX LOGONB</code>	is a report schedule that collects data when polled
<code>1..5TK</code>	

### Combining Alarms

Alarms can be combined together to yield a single result from several alarm tests. They are combined using logical operators, which replace the *digitalAction*, *actionText* and *actionProcesses* of all except the last alarm. The actions associated with the combined test are attached to the last alarm. Any alarm delay period is also associated with the last alarm.

The logical operators are `AND`, `OR` and `XOR`.

The `DT80` evaluates a combined alarm in the order in which its component alarms appear in the schedule command. This means that the alarm numbers do not have to be sequential.

### Examples — Combining Alarms

The combined alarm

```
ALARM4(3TK>100)OR
ALARM2(2TK>100)OR
ALARM7(5TK>100)AND
ALARM3(T>10:00:00)"Temp Error"{1DBO=12}
```

produces a single alarm output based on several temperature tests and a time test. The combined alarm becomes true when any one of `2TK`, `3TK` or `5TK` exceeds 100°C after 10:00:00 am.

The combined alarm

```
ALARM(T<06:00,18:00)AND  
ALARM(1TK>35)"Vents open"{1DSO(W)=1}
```

opens the vents in a glasshouse if the air temperature exceeds a threshold during daylight hours. The combined alarm becomes true if the temperature exceeds 35°C between the hours of 06:00 and 18:00, switching ON (low/active) the *DT80*'s digital state output 1 (`{1DSO(W)=1}`) to activate the vent mechanism, and issuing a `Vents open` message.

## Polling Alarm Data

**Alarm data** (that is, the current value of alarm inputs) can be polled (requested) by the host computer at any time.

There are three commands for polling alarm data:

<code>?n</code>	returns the current input value of alarm <code>n</code>	For numbered alarms only
<code>?x</code>	returns the current input values of all alarms in schedule <code>x</code> , where <code>x = A, B, ...K</code>	For numbered and un-numbered alarms
<code>?ALL</code>	returns the current input values of all alarms in all schedules	

When un-numbered alarms are polled by `?x` and `?ALL`, the alarm number is returned as `A0`.

The format of data returned by an alarm poll command differs for the default free-format mode (`/h`) and fixed-format mode (`/H`). See the following examples.

### Examples — Polling Alarm Data

When the *DT80* is in **free-format mode** ([P21](#)) and the alarm command

```
ALARM5(2R>50)"Warning^M^J"
```

is included in schedule `F`, the alarm poll command

```
?5
```

instructs the *DT80* to return an alarm **data** record of the form

```
A5 2R>50 2R 123.45 Ohm
```

where

<code>A5</code>	is the alarm ID
<code>2R&gt;50</code>	is the alarm <i>test</i>
<code>2R</code>	is the alarm input
<code>123.45</code>	is the current value of the alarm input
<code>Ohm</code>	is the units text for the alarm input

When the *DT80* is in **fixed-format mode** ([P21](#)) and the same alarm command

```
ALARM5(2R>50)"Warning^M^J"
```

is included in schedule `F`, the alarm poll command

```
?5
```

instructs the *DT80* to return an **alarm data** record<sup>10</sup> of the form

```
D,080416,2001/04/12,07:20:40,0.08932,2;F,5,"2R>50",123.45;0069;2614
```

where

<code>D</code>	signifies that this is a Data record
<code>080416</code>	is the <i>DT80</i> 's serial number
<code>2001/04/12,07:20:40</code>	is the date and time of the alarm record
<code>0.08932</code>	
<code>2</code>	signifies "not used"
<code>F</code>	is the name of the schedule containing the alarm command
<code>5</code>	is the alarm number
<code>2R&gt;50</code>	is the alarm test
<code>123.45</code>	is the current value of the alarm input
<code>0069;2614</code>	are communications error checks (record character count;checksum)

# Logging and Retrieving Alarms

The *DT80* stores **alarm state records** for later retrieval and analysis. These records comprise the alarm state, a timestamp, the alarm number and any action text. They can be logged into the *DT80*'s general data store whenever an alarm is tested.

See Logging Alarm States ([P85](#)) below and Retrieving Logged Alarm States ([P86](#)).

<sup>10</sup> See also [Figure 10](#).

---

# Logging Alarm States

## Alarm States and Tags

The *DT80* recognizes these four alarm states/transitions:

- continuing false
- false-to-true
- continuing true
- true-to-false

When the *DT80* tests an alarm and logs the alarm record, it includes the state/transition of the alarm in the form of a numeric tag:

State/Transition	Tag	Comment
Continuing false	0	Not used
False-to-true	1	Universal
Continuing true	2	Only used by <a href="#">ALARM<sub>n</sub></a> (repeating numbered alarms)
True-to-false	3	Universal

For examples of alarm records containing these tags, see the What's Logged column in Logging Alarm States — What's Logged, What's Returned ([P86](#)).

## Which States are Logged?

Choose which alarm states are logged. This is controlled by P9:

P9=	State/Transition Logged
0	None
1	False-to-true only — the <i>DT80</i> 's default
2	True-to-false only
3	Both

The default setting is **P9=1**, which causes the *DT80* to only log the false-to-true transitions.

For a repeating alarm, the alarm state is also logged each time the alarm is tested and is true (that is, at the alarm's report schedule interval).

## Numbered Alarms Only

Alarm states can only be logged for numbered alarms. The alarm number identifies these alarms in retrieved alarm data. See Alarm Number ([P78](#)).

## Where are Alarm States Stored?

Alarm states are logged in the *DT80*'s general data storage memory along with data from other sources such as input channels, calculations, and processes. Although data from these sources each has a separate set of log files, the general data logging commands apply to all.

## Enabling Logging of Alarm States

The actual logging of alarm state is enabled by the general **LOGON** command, or by the **LOGON<sub>x</sub>** command for the report schedules to which the alarms belong, where **x** = **A, B, ...K**. Alarm states are logged by the schedule to which they belong, in the same way that data is logged by the schedule to which it belongs. See LOGON and LOGOFF Commands ([P69](#)).

## Overwrite Mode

Normally alarm state is logged until the *DT80*'s data storage memory is filled, after which further logging ceases (although the alarms continue to be tested and other actions performed). However, as with the logging of data ([P70](#)), the logging of alarm states can also be done in overwrite mode, in which the newest data progressively overwrites (replaces) the oldest data after the data memory is filled. This is enabled by the **OV** schedule option.

## Action Text is Included

If alarms that are being logged have action text, the action text is logged along with the alarm state (in the *DT80*'s general data store). And if the action text includes substitution characters, the substitute text is included in the logged action text. See Alarm Action Text ([P80](#)).

## Example — Logging Alarm State and Action Text

The program

```
BEGIN
P9=3 'Log both transitions
RC1S
ALARM5(3TK>50)"Warning ?v?u"
LOGONC
END
```



instructs the *DT80* to log an alarm record when the alarm occurs, and another alarm record when the alarm recovers (because P9 is set for both transitions). The action text **"Warning ?v?u"** is logged for the false-to-true transition, and **"ALARMn FALSE"** is logged for the true-to-false transition. The two records have the form

```
A,80416,2001/04/12,07:20:40,0.489312,5;C,1,"Warning 53 degC";
0066;15C4
```

and

```
A,80416,2001/04/12,07:23:53,0.216923,5;C,3,"ALARM5 FALSE";
0067;B5DC
```

## Logging Alarm States — What's Logged, What's Returned

For the four types of alarm commands, the following table summarises the alarm information that is included in the logged alarm state records and returned in realtime to the host computer:

Alarm Command	State/Transition	Parameter 9	What's Logged	What's Returned
<b>ALARM( test ) "actionText"</b>	Continuing low	—	—	—
	False to true	—	—	<i>actionText</i>
	Continuing high	—	—	—
	True to false	—	—	—
<b>ALARMn( test ) "actionText"</b>	Continuing low	—	—	—
	False to true	P9=1 or 3	Timestamp,n,1,"actionText"	<i>actionText</i>
	Continuing high	—	—	—
	True to false	P9=2 or 3	Timestamp,n,3,"ALARMn FALSE"	—
<b>ALARMR( test ) "actionText"</b>	Continuing low	—	—	—
	False to true	—	—	<i>actionText</i>
	Continuing high	—	—	<i>actionText</i>
	True to false	—	—	—
<b>ALARMRn( test ) "actionText"</b>	Continuing low	—	—	—
	False to true	P9=1 or 3	Timestamp,n,1,"actionText"	<i>actionText</i>
	Continuing high	—	Timestamp,n,2,"actionText"	<i>actionText</i>
	True to false	P9=2 or 3	Timestamp,n,3,"actionText FALSE"	—

## Retrieving Logged Alarm States

Retrieve (unload) logged alarm state records from the *DT80* to the host computer using alarm unload commands:

- **A** commands (no brackets) — unload all alarm records (from the beginning to the end of the alarm store).
- **A ( )** commands (round brackets / parentheses) — unload alarm records for a user-defined period in the format specified by the *DT80*'s current P31 and P39 settings (date and time formats).
- **A [ ]** commands (square brackets) — unload alarm records for a user-defined period in the *DT80*'s fixed-format style **YYYY/MM/DD, hh:mm:ss,0.ssssss**, which overrides the *DT80*'s current P31 and P39 settings.

**Note** Logged alarm state records are not cleared/deleted from the *DT80* by unload operations. You can unload the same logged records as many times as you want (until you purposefully delete it, of course — see [Deleting Logged Alarm Records \(P88\)](#)).

### The A Unload Commands

The **A** unload commands return all alarm state records (that is, from the beginning to the end of the alarm store) for user-defined jobs and/or report schedules.

<b>A</b>	Returns the current job's alarms in the order of report schedule A to K
<b>Ax</b>	Returns the current job's alarms for report schedule <b>x</b>
<b>A "JobName"</b>	Returns alarms for <b>JobName</b> in the order of report schedule A to K
<b>A "JobName" x</b>	Returns alarms for <b>JobName</b> report schedule <b>x</b>

Table 5: Alarm Unload Commands — A

These commands are also listed in [Summary — Retrieval Commands \(P166\)](#).

### Examples — A Unload Command

The command

**A**

instructs the *DT80* to unload all of the current job's alarm records from beginning to end for all report schedules A to K.

The command

```
A"Boiler"G
```

instructs the *DT80* to unload alarm information for job **Boiler**, report schedule **G**.

## The A() Unload Commands

The **A()** unload commands return subsets of the alarm state records logged in the *DT80*. You define the subset to be returned by specifying combinations of

- jobs and/or report schedules, and
- periods designated by a beginning and end *date* and *time*, which you must specify in the *DT80*'s current date and time formats (as set by P31 and P39).

<b>A</b>	Returns the current job's alarms in the order of report schedule A to K
<b>A(<i>from</i>)</b>	Returns the current job's alarms starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>
<b>A(<i>from</i>)(<i>to</i>)</b>	Returns the current job's alarms starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>
<b>A<i>x</i></b>	Returns the current job's alarms for report schedule <i>x</i>
<b>A<i>x</i>(<i>from</i>)</b>	Returns the current job's alarms for schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>
<b>A<i>x</i>(<i>from</i>)(<i>to</i>)</b>	Returns the current job's alarms for schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>
<b>A"JobName"</b>	Returns alarms for <i>JobName</i> in the order of report schedule A to K
<b>A"JobName"(<i>from</i>)</b>	Returns alarms for <i>JobName</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>
<b>A"JobName"(<i>from</i>)(<i>to</i>)</b>	Returns alarms for <i>JobName</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>
<b>A"JobName"<i>x</i></b>	Returns alarms for <i>JobName</i> report schedule <i>x</i>
<b>A"JobName"<i>x</i>(<i>from</i>)</b>	Returns alarms for <i>JobName</i> report schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>
<b>A"JobName"<i>x</i>(<i>from</i>)(<i>to</i>)</b>	Returns alarms for <i>JobName</i> report schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>

Table 6: Alarm Unload Commands — A()

where

<i>from</i> can be	Both the <i>date</i> and <i>time</i> must be specified in the currently-defined format for date (P31) and time (P39). <b>Note</b> <b>BEGIN</b> and <b>END</b> used here are <u>not</u> the same as the <b>BEGIN</b> and <b>END</b> keywords used to indicate the start and end of a <i>DT80</i> job.
<b>BEGIN</b> — start from first alarm logged	
<i>time</i> — start from first alarm logged at or after this time today	
<i>time, date</i> — start from first alarm logged at or after this time and date	
<i>to</i> can be	
<b>END</b> — end with last alarm logged	
<i>time</i> — end with last alarm logged prior to this time today	
<i>time, date</i> — end with last alarm logged prior to this time and date	

These commands are also listed in Summary — Retrieval Commands ([P166](#)).

### Examples — A() Unload Command

The command

```
A(12:00,19/1/2000)(12:05,20/1/2000)
```

instructs the *DT80* to unload the current job's alarm information for all report schedules in the order A to K, and starting at **12:00** on **19/1/2000** and ending at **12:05** on **20/1/2000**.

The command

```
A"Job1"B(BEGIN)(11:15)
```

instructs the *DT80* to unload the alarms information for report schedule **B** of **Job1**, from **BEGIN** (the first alarm logged) until **11:15** today.

## The A[ ] Unload Commands

The **A[ ]** unload commands return subsets of the alarm state records logged in the *DT80*. You define the subset to be returned by specifying combinations of

- jobs and/or report schedules, and
- periods designated by a beginning and end *date* and *time*, which you must specify in the *DT80*'s fixed-format style **YYYY/MM/DD,hh:mm:ss,0.ssssss** (ISO date format), which overrides the *DT80*'s current P31 and P39 settings.

<b>A</b>	Returns the current job's alarms in the order of report schedule A to K
<b>A[ <i>from</i> ]</b>	Returns the current job's alarms starting from <i>time</i> or <i>date, time</i>
<b>A[ <i>from</i> ][ <i>to</i> ]</b>	Returns the current job's alarms starting from <i>time</i> or <i>date, time</i> and ending with <i>time</i> or <i>date, time</i>
<b>A<i>x</i></b>	Returns the current job's alarms for report schedule <i>x</i>
<b>A<i>x</i>[ <i>from</i> ]</b>	Returns the current job's alarms for schedule <i>x</i> starting from <i>date</i> or <i>date, time</i>
<b>A<i>x</i>[ <i>from</i> ][ <i>to</i> ]</b>	Returns the current job's alarms for schedule <i>x</i> starting from <i>date</i> or <i>date, time</i> and ending with <i>date</i> or <i>date, time</i>
<b>A"JobName"</b>	Returns alarms for <i>JobName</i> in the order of report schedule A to K
<b>A"JobName"[ <i>from</i> ]</b>	Returns alarms for <i>JobName</i> starting from <i>date</i> or <i>date, time</i>
<b>A"JobName"[ <i>from</i> ][ <i>to</i> ]</b>	Returns alarms for <i>JobName</i> starting from <i>date</i> or <i>date, time</i> and ending with <i>date</i> or <i>date, time</i>
<b>A"JobName"<i>x</i></b>	Returns alarms for <i>JobName</i> report schedule <i>x</i>
<b>A"JobName"<i>x</i>[ <i>from</i> ]</b>	Returns alarms for <i>JobName</i> report schedule <i>x</i> starting from <i>date</i> or <i>date, time</i>
<b>A"JobName"<i>x</i>[ <i>from</i> ][ <i>to</i> ]</b>	Returns alarms for <i>JobName</i> report schedule <i>x</i> starting from <i>date</i> or <i>date, time</i> and ending with <i>date</i> or <i>date, time</i>

Table 7: Alarm Unload Commands — A[ ]

where

<b>from</b> can be	<b>Note</b> Type <i>time</i> and <i>date</i> in fixed-format ISO- style (see examples below).
<i>date</i> — start from first alarm logged at or after this date	
<i>date, time</i> — start from first alarm logged at or after this date and time	
<b>to</b> can be	
<i>date</i> — end with last alarm logged on this date	
<i>date, time</i> — end with last alarm logged on this date prior to this time	

These commands are also listed in Summary — Retrieval Commands ([P166](#)).

### Examples — A[ ] Unload Command

Here's the **A[ *from* ]** command showing valid forms of the fixed-format style date and time:

```
A[2000/07/26,12:30:00,0.250366]
A[2000/07/26,12:30:00]
A[2000/07/26]
```

Here's the **A[ *from* ][ *to* ]** command showing valid forms of the fixed-format style date and time:

```
A[2000/07/26,12:30:00,0.250366][2000/07/28,18:30:00,0.750244]
A[2000/07/26,12:30:00][2000/07/28,18:30:00,0.750244]
A[2000/07/26][2000/07/28,18:30:00,0.750244]
A[2000/07/26,12:30:00,0.250366][2000/07/28,18:30:00]
A[2000/07/26,12:30:00][2000/07/28,18:30:00]
A[2000/07/26][2000/07/28,18:30:00]
A[2000/07/26,12:30:00,0.250366][2000/07/28]
A[2000/07/26,12:30:00][2000/07/28]
A[2000/07/26][2000/07/28]
```

## Deleting Logged Alarm Records

Delete logged alarm records from the data memory of the *DT80* using these commands:

---

<b>DELALARMS</b>	Deletes the current job's alarms	Job names, directory
<b>DELALARMS "JobName"</b>	Deletes only <i>JobName</i> 's alarms	structures, programs and
<b>DELALARMS *</b>	Deletes all jobs' alarms	data are not erased.

---

These commands are also listed in Summary — Delete Commands [\(P166\)](#).

# Part H — DT80 Front Panel

The *DT80* front panel has a 2 line by 16 character back-lit liquid crystal display, 6 keys and 3 status indicator lights. The display provides information about *dataTaker* data logger status, channel data, alarms and store operation. In addition the display will indicate conditions that require attention and USB memory device status.

The *DT80* from cannot be programmed from the front panel. However, pre-defined commands can be issued by selecting from the function list via the front panel.



Figure 27 DT80 Display

## Display

The display normally shows the current value for all channels and alarms for the current job. Each channel or alarm is shown one at a time. The actual channel or alarm shown is selected by pressing the up and down directional keys on the front panel. In addition it shows several status values that can also be selected via the up and down directional keys. The channels and alarms are arranged in the same order that they are defined in for the current job.

### Displaying Channels and Alarms

When channel data is displayed, the top line of the display shows the channel identification. The default is the channel number and type. If a channel identification text has been entered as a channel option, then the first 16 characters of that text is displayed.

When alarms are displayed the top line of the display identifies the alarm and the state of the alarm – ON or OFF. If the alarm channel definition includes identification text, then this is displayed when the alarm is not true. If the alarm contains action text, this is displayed when the alarm is true. Alarms must be numbered to be displayed.

The bottom line on the display shows the most recent reading as a numeric value or bar graph. If the channel or alarm has not yet been sampled, the display shows " ---".

For example, assume that the following job has been defined:

```
BEGIN"MYJOB"
RA1M 1TK("Boiler Temp",FF0)
  2LM35
ALARM4(3V>2000)"Over voltage"
  1CV(W)=1CV+1
ALARM7(4TT("Oven OK")>107)"Oven Over Temp"
END
```

The following "screens" will then be available. These can be scrolled through using the up and down arrows on the keypad.


Display screen	Comments
DT80 V5.02 MYJOB	The default "sign-on" screen indicates the DT80's firmware version number and the name of the currently loaded job ( <b>No current job</b> is displayed if there isn't one)
Date: 23/10/2005 Time: 16:44:02	Current date and time (format can be changed using P31 and P39)
Battery: 90% ↓ -290mA 6.2V	Internal battery status. This shows the approximate battery charge as a percentage, a charge (↑) or discharge (↓) indicator, battery current (negative=discharging) and the battery terminal voltage.
Boiler Temp 97 °C	First user channel (user defined channel name)

Channel 2LM35 17.9 °C	Second user channel (default channel name)
Alarm4 OFF 1356.3 mV	Alarm #4 state
Oven Over Temp 117.2 °C	Alarm #7 state (alarm text replaces channel name when alarm is active)

Note that channel **1CV** is not displayed because it is defined as a working (**W**) channel. Working channels are not logged, returned or displayed.

## Bar Graph

The channel value can be shown as a bar graph instead of a numeric value by using the **BG** channel option. The **BG** option allows the values to be set that represent the left and right side of the graph scale. The channel label can be used to set the graph scale labels. For example:

Display screen	Comments
E--Fuel Level--F 	4V("E--Fuel Level--F",BG:10:900) displays zero scale (no bars) if measured voltage < 10mV; displays full scale if voltage > 900mV

## Controlling what is shown on the display

All defined channels and alarms will be shown on the display, **except for**:

- channels which specify the **ND** (no display) channel option
- working channels (**W** channel option)
- un-numbered **ALARM** or **IF** channels
- channels used as the condition in an **ALARM** or **IF**

## Enable/Disable status screens

Status screens can be enabled or disabled for display by P19. Each bit in this parameter value represents a status screen. A "1" enables and "0" disables. The bit mapping is:


Bit Number	Decimal Value	Status Screen
0	1	Sign-on
1	2	Date / Time
2	4	Battery Status
3	8	Reserved
4	16	Reserved
5	32	Reserved
6	64	Reserved
7	128	Reserved

To make screens available set **P19** to the sum of the decimal values following the required screens, e.g. for Battery Condition and Current Job screens only set P19=5 (i.e. 1 + 4). By default P19=255 and all screens are available. If P19=0 and there are no channels or alarms to display then the sign-on screen is displayed.

## Transient Messages

The display may also show temporary status screens, such as.

Display screen	Comments
Reading USB device	When a USB device is inserted the DT80 needs to read certain system information from it before it can be used.
USB device unrecognised	This indicates that the DT80 does not recognise the device as a valid USB mass storage device.

Processing ONINSERT.DXC	If the USB memory device contains a file called <b>ONINSERT.DXC</b> then it will be automatically loaded and run by the DT80
 Copying Data	This indicates progress during a <b>COPYDATA</b> operation.

## Display Backlight

The display backlight stays on when external power is supplied. If the DT80 is running from internal battery then the backlight will only stay on for 30 seconds after the last key press. The actual period that the backlight stays on for after a key press is controlled by P17 in seconds.

## User defined functions

The user can define named macros called functions. These functions can be executed by the user via the LCD and keypad of the DT80.

Functions can be very useful. For example the functions can be used to completely reprogram the DT80, with a different program assigned to each function. The functions can also be assigned by ALARMS.

### The FUNCTION command

Functions are defined with the use of the **FUNCTION** command. The user may define up to 10 functions. A function is deleted when the user provides no label or command text following the = (equal) character. The syntax is as follows:

```
FUNCTIONn="label" {commands}
```

where

Parameter	Meaning
<i>n</i>	This is the id or slot number of the function to be redefined. It must be an integer in the range from 1 to 10 inclusive.
" <i>label</i> "	This is optional. It is a label that effectively names the function. This label will be displayed on the LCD when the user scrolls through the function list. It can be no more than 16 characters long. If this option is not supplied then up to 13 characters of command text will become the label text as displayed on the LCD.
<i>commands</i>	This can be a list of any white space separated DT80 commands, enclosed in braces. These commands will be executed when the user selects the associated function from the function list.

### Examples:

Command	Description
FUNCTION1="Start" {G}	This function would display <b>'Start'</b> and when selected would issue the G or go command to the the logger which would start all schedules
FUNCTION2="Stop" {H}	This function would display <b>'Stop'</b> and when selected would issue the H or halt command to the logger to stop all schedules
FUNCTION3="Init" {1V(S1,=10CV)}	This would display <b>'Init'</b> and when selected would store the current value of 1V scaled to channel variable 10CV
FUNCTION4="Clear" {1..20CV(W)=0}	This function would display <b>'Clear'</b> and would set channel variables 1-20 back to zero.

## Selecting Functions

Pressing the **Cancel/Func** key will cause the function list to be shown on the display. Once function is shown at a time, and only those functions which have been defined are shown. The up and down direction keys can be pressed to scroll through the list of functions. Once the desired function is visible on the display it can be executed by pressing the **OK/Edit** key. If you wish to exit the function list without executing any function then press the **Cancel/Func** key to cancel the function selection process.

After selecting the function to execute the display will indicate that the function selected has been initiated.

## Default Functions

Following reset, the DT80 automatically defines two commonly used functions:

```
FUNCTION9="Remove USB" {REMOVEMEDIA}
FUNCTION10="Copy logged data" {COPYDATA}
```

These can be redefined or removed if desired.



---

## Keypad operation

### Direction Keys



The up and down direction keys allow scrolling through the available channels, alarms and status screens on the display. When the function list is shown, then the up and down direction keys allows scrolling through the list of available of functions.  
The left and right direction keys are not presently used. They will be used in a later version of the firmware that supports editing of values.

### OK (Edit) Key



The **OK/Edit** key is used to select a function to execute when the function list is displayed. The edit function will be used in a later firmware version that supports editing of values.

### Cancel (Function) Key



The **Cancel/Func** key is used to enter the function list display. It can be pressed again to exit the function list without selecting a function.

## Special Key Sequences

### Entering Bootstrap Mode

Holding down the **OK/Edit** key during the logger reset or power-up sequence will force the logger into bootstrap mode. This would only be required if there is a corruption of the firmware in the logger.

---

## Status Indicator Lights

### Sample Indicator

The sample indicator is illuminated whenever any channel in the current job is sampled. This includes all analog, digital and internal channels.

### Disk Indicator

The disk indicator is illuminated whenever the internal disk is reading or writing. For example, the disk indicator will illuminate when writing data to the internal data store or when unloading data from the data store.

### Attn Indicator

This LED is used to:

- warn that an unexpected DT80 reset has occurred (flashing)
- warn that logging has been partially or fully suspended (flashing)
- indicate a warning state under the control of a user program (continuously on)

### Unexpected Reset

One of the following messages may be displayed following DT80 reset, in conjunction with a flashing **Attn** LED. Press any key to clear the message and the flashing LED.

Display	Comments
DT80 restarted Power loss	The DT80 lost power, both external and the internal battery. This message may also be displayed if the hardware reset button (accessed using a paper clip) is pressed.
DT80 restarted Safe mode	A "triple-push" reset was performed (by pressing the hardware reset button three times within 10s), which temporarily restores factory settings

DT80 restarted SW exception	This indicates a possible problem with the DT80 firmware. Contact dataTaker Support if you see this message.
DT80 restarted All mem cleared	The DT80 lost power, and all internal RAM has been cleared, probably due to the internal Lithium memory backup battery being flat. Programs and logged data will not be affected but you will need to reset the DT80's time/date.

### Logging Suspended

If data for one or more schedules cannot be logged for some reason then the DT80 will continue to run the job but it will flash the **Attn** LED and display a message such as the following. Pressing a key will clear the message from the display, but the **Attn** LED will keep flashing until space is made available (eg. by deleting other jobs' data) or the job is stopped.

Display	Comments
Cannot log No space	There is insufficient space on the specified logging drive (internal or USB device) to allow a data file of the requested size to be created.
Cannot log Data full	One or more schedules have ben set to "no-overwrite" mode ( <b>NOV</b> schedule option), and the allocated space is now full
Cannot log No USB device	The " <b>A:</b> " schedule option (log directly to USB device) has been specified, but no USB device is inserted.
DT80 restarted All mem cleared	The DT80 lost power, and all internal RAM has been cleared, probably due to the internal Lithium memory backup battery being flat. Programs and logged data will not be affected but you will need to reset the DT80's time/date.

### User Control

You can also turn the **Attn** LED on or off using the **SATTN** (Set Attention) and **CATTN** (Clear Attention) commands. Alternatively, the **1WARN** channel type (which works in the same way as a digital output channel) may be used.

For example:

```
RA1S ALARM1 (3TT>500) "Meltdown" {SATTN}
```

will cause the LED to come on and stay on if the alarm is triggered, and

```
RA1S 1CV=(1CV+1)%10 IF(1CV<.5){1WARN(R,200)=1}
```

will give a 200ms flash every tenth time the schedule is scanned.

# Part I — Communications

Commands can be sent to and data returned from the DT80 via a number of alternative communications channels:

- Direct RS232 – see *DT80* RS-232 Basics ([P97](#)) *DT80* Direct (Local) RS-232 Connection ([P100](#))
- RS232 via modem – see *DT80* Modem (Remote) RS-232 Connection ([P100](#)).
- USB – see USB COMMUNICATIONS ([P95](#))
- Ethernet (TCP/IP) – see *DT80* ETHERNET COMMUNICATIONS ([P104](#)).
- PPP (TCP/IP) over RS232 – see *DT80* PPP COMMUNICATIONS ([P108](#)).

In addition, the *DT80* includes an FTP (File Transfer Protocol) server so files can be transferred to and from the *DT80* using one of the TCP/IP communications channels (eg Ethernet) – see *DT80* FTP COMMUNICATIONS ([P108](#)).

## Automatic Comms Port Arbitration

The *DT80* can have more than one type of communications link connected at the same time. In this situation, the *DT80* automatically switches between each link as required, responding back through the link from which the most recent communication was received.

A simple CR (carriage return) character or LF (line feed) character received from a link is sufficient to switch the *DT80*'s connection to that link. Before switching, the *DT80* notifies the host on the current link that it is about to change connection.

The only exception to this is where a single quote is used on alarm statements instead of the usual double quote. (see Alarm Action Text ([P80](#))) Alarm text enclosed in single quotes is always sent to the Host serial port.

## Password Protection — Comms Ports

Protect the *DT80*'s communications ports — the **Host RS-232** port, the **Ethernet** port and the **USB** port<sup>13</sup> — with a user-defined password so that communication through these ports is only possible after the password is entered.

### Setting and Removing a Comms Port Password

Set a password by sending

```
PASSWORD="password"
```

to the *DT80*. *password* can be any text string (except command keywords) of up to 10 case-sensitive characters.

Remove a password by sending

```
PASSWORD=" "
```

(two " characters, no space character between them).

**Note** The password is lost ("removed") if the *DT80* performs a power-up reset, triple-push reset or **SINGLEPUSH** reset (see the Table 12: *DT80* Resets ([P119](#))).

### Accessing Password-Protected Comms Ports

To establish communication at anytime, simply send the password followed by a carriage return. If the password is correct, the *DT80* responds with **Accepted** and opens the comms ports.

The ports stay open until you send the **SIGNOFF** command, or while there is comms activity. If there is no communication for a period of time defined by P14 (default is 300 seconds), the ports time out and close.

### Are the Ports Protected?

Simply send the command

```
PASSWORD
```

to determine if a comms port password has been set. If the *DT80* responds with

- **PASSWORD 0** — no password has been set
- **PASSWORD 1** — a password has been set.

Regardless of the password state, the *DT80* responds to the **DEL** character with << **CR LF**. This response can be used to determine the baud rate setting of the *DT80* even when a comms port password is active.

# USB Communications

The *DT80*'s USB interface operates as a "virtual COM port". With the appropriate drivers installed on the PC, the USB link will, when connected, appear to the PC as an additional COM port.

<sup>13</sup> The USB port is not yet operational — see [DT80 USB Communications](#)

## Installing the USB Driver

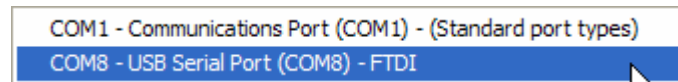
When the DT80 is first connected to a PC via USB, Windows will search for a suitable driver.

If no driver is installed on the PC, the Windows "new hardware wizard" will run. The exact sequence of events that follows varies somewhat depending on the version of Windows.

Normally, Windows will ask where it should look for a driver. Insert the dataTaker resource CD and select the "CDROM drive" as the driver location when prompted.

The New Hardware wizard should then be allowed to run to completion. You should now have an additional COM port available on the PC. We now need to determine what COM port number has been assigned.

When we use DeLogger, the particular COM port in the Connection dialog drop down list can be identified by the **FTDI** manufacturers tag.




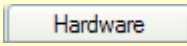
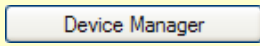
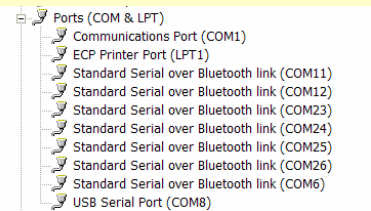



Both DeLogger and DeTransfer only show **active** COM ports in their connection dialogs, so it should not be too difficult to work out which one corresponds to the USB connection.

One possible source of confusion might be if you also use a USB to RS232 converter (such as that supplied with other dataTaker logger models) – it may also be identified as an "FTDI" device. By removing the adapter or the DT80 connection and observing the COM port lists in DeLogger and DeTransfer it should be possible to work out which COM port has been assigned to which device.

**Note** The assigned COM port is associated with the particular PC USB port that you are using. If you subsequently connect the DT80 to a different USB port then it will be assigned a different COM port number.

You can also check the assigned COM port number in the Windows Device Manager, as follows:

Operation	Button
Click Start	
Select the Control Panel	
When the control panel is open select 'System'. This will open the 'System Properties' dialog box	
Select the  Tab. Click  . This will open the 'Device Manager' dialog box	
Click the Ports (COM & LPT). There should be one identified as USB Serial Port.	
Double Click on the most likely candidate. This will open the USB devices properties. We can identify the FTDI manufacturers label here	

## Using the USB Connection

Once the driver has been successfully installed, the USB connection will operate in a very similar way to an RS232 connection, except that

- it will be faster
- you do not need to set baud rate or flow control options

However, note that:

- it is not recommended that the logger be allowed to go to sleep while the USB cable is connected
- modems cannot be used on the USB interface

# RS-232 Communications

## Quick Start

- To connect the *DT80* directly to a local host computer, go to Setting Up a Direct Connection [\(P100\)](#).
- To connect the *DT80* to a distant host computer using a modem communications link, go to Setting Up a Remote Connection [\(P103\)](#).

## DT80 RS-232 Basics

The *DT80* has a 9-pin male connector for RS-232 serial communication to a computer (either directly, or by means of a pair of modems (Figure 29 [\(P97\)](#)). This DTE interface, labelled Host RS-232, is the primary means by which you

- program (configure) and supervise the *DT80* from the host computer
- retrieve stored data from the *DT80* to the host computer.

### Host RS-232 Port Pinout

The pinout of the *DT80*'s Host RS-232 connector is shown in Figure 29. The signal functions are defined in the Table 17: RS-232 Pinouts [\(P171\)](#).

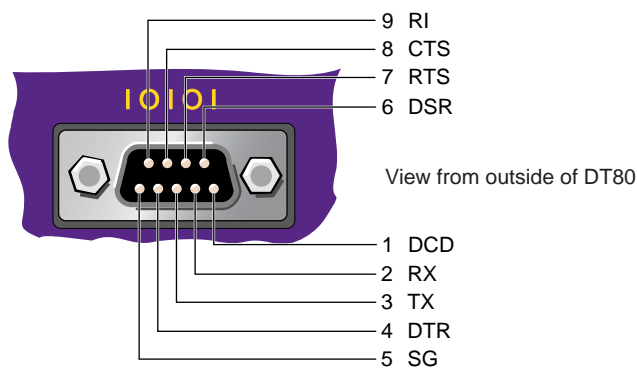


Figure 28: *DT80* Host RS-232 port — connector pinout (DTE)

### Automatic Device Detection

A *DT80* uses the state of its Host RS-232 port's DSR terminal (Figure 29 [\(P97\)](#)) to determine the type of device connected to the port as follows:

If the *DT80*'s DSR terminal is **NOT** held active by the connected device, the *DT80* assumes that it's connected directly to the host computer (Figure 30 [\(P97\)](#)) and operates accordingly.

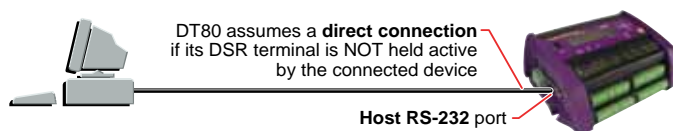


Figure 29: DSR inactive (low)

If the *DT80*'s DSR terminal **IS** held active by the connected device, the *DT80* assumes that it's connected to a modem (Figure 30 [\(P97\)](#)) and operates accordingly, initialising the modem, monitoring other Host RS-232 lines to determine when a modem connection to the host computer has been established, and so on.

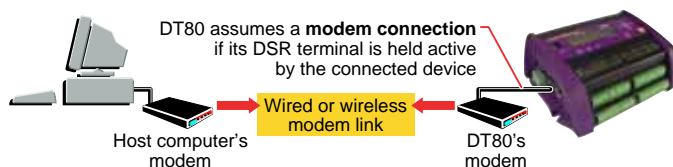


Figure 30: DSR active (high)

This feature greatly simplifies the RS-232 computer-to-*DT80* comms link, which is discussed later in *DT80* *DT80* Direct (Local) RS-232 Connection [\(P100\)](#) and *DT80* *DT80* Modem (Remote) RS-232 Connection [\(P100\)](#).

### Host RS-232 Port Commands

The *DT80* is shipped with its Host RS-232 port set to **57600** baud, **No** parity, **8** data bits, **1** stop bit, and **SoftWare** Flow Control. If you need to change these settings, there are two methods you can use:

- Use one or more **PROFILE...** commands to “permanently” change the settings. The changes are stored in the *DT80*'s USER.INI file and loaded every time the *DT80* is restarted. See the **HOST\_PORT** [\(P114\)](#) section of the Table 11: *DT80* PROFILE Details [\(P115\)](#) table.

- Send a **PH=** command — see PH Configuration Commands ([P98](#)). Doing this changes the settings “temporarily”. That is, the *DT80* uses the new settings only until you restart it, or until you send another **PH=** command.

The following commands are available for configuring the *DT80*'s Host RS-232 port, and for controlling the dial-out process of the modem connected to the *DT80*:

### PH Configuration Commands

The **PH=** commands (**PH** ⇒ **Port Host**) set the comms attributes of the *DT80*'s Host RS-232 port. (Note that **PH=** settings are not remembered through a reset; use **HOST\_PORT PROFILE...** commands for settings that you want to be permanent.)

Command	Action	<i>b</i>	<i>p</i>	<i>d</i>	<i>s</i>	<i>f</i>
PH= <i>b</i>	Sets Host RS-232 port baud rate ( <i>DT80</i> default = 57600)	50, 75, 110, 150, 300,				
PH= <i>b,p</i>	Sets Host RS-232 port baud rate and parity	600, 1200, 2400, 4800,	N (none), O (odd), or			
PH= <i>b,p,d</i>	Sets Host RS-232 port baud rate, parity and databits	9600, 19200,	E (even)	8 or 7		
PH= <i>b,p,d,s</i>	Sets Host RS-232 port baud rate, parity, databits and stopbits	38400, 57600, or 115200			1 or 2	
PH= <i>b,p,d,s,f</i>	Sets Host RS-232 port baud rate, parity, databits, stopbits and flow control					NOFC (no flow control), HWFC (hardware flow control), or SWFC (software flow control) SWHW (both hardware and software flow control)
PH	Returns the current Host RS-232 port comms settings (for example: 57600, N, 8, 1, NOFC)					

Table 8: *DT80* Host RS-232 port — configuration commands

### SETDIALOUTNUMBER Command

Send the command

```
SETDIALOUTNUMBER "digits"
```

to the *DT80* to specify the telephone number to be dialled by the **DIAL** command to establish a connection to the host computer.

### DIAL Command

The **DIAL** command causes the *DT80* to instruct its modem to dial out to the telephone number specified by **SETDIALOUTNUMBER**. If a call cannot be placed for any reason, the command is ignored. This is often used as an alarm action command to cause the *DT80* to dial out when an alarm condition arises (see Alarm Action Processes ([P82](#))).

### HANGUP Command

The **HANGUP** command causes the *DT80* to instruct its modem to hang up (disconnect) the current dial-out or dial-in connection. If there is currently no connection, **HANGUP** is ignored. This can be used in an alarm action command to cause the *DT80* to hangup a call in progress when an alarm condition arises (see Alarm Action Processes ([P82](#))).

### Example — Modem Control Commands

The use of the *DT80*'s modem control commands is demonstrated in the following program:

```
BEGIN
SETDIALOUTNUMBER "12345678"
RA10M
'Read boiler temp
1TK(=1CV,W)
IF(1CV>120){DIAL}
END
```

Every 10 minutes, the program instructs the *DT80* to initiate a dial-out to phone number **12345678** if the boiler temperature is greater than 120°C.

### Flow Control

**Flow control** is the means by which communicating devices (such as the *DT80* and its host computer) control each other's transmission of characters to avoid data loss. Flow control causes the receiver to disable transmissions by the sender if the receiver's input buffer is at risk of overflowing and thereby losing data.

The *DT80* supports all methods of flow control:

- **Software flow control** (SWFC; also known as XON/XOFF flow control, XON/XOFF handshaking, or software handshaking)
- **Hardware flow control** (HWFC; also known as RTS/CTS flow control, RTS/CTS handshaking, or hardware handshaking)
- No flow control (NOFC)



- **SWHW** (both software and hardware flow control)

The *DT80* will often be used set to SWFC, which is the default. But there may be times when there is a need to change the *DT80* to HWFC — for example, when using software on the host computer that doesn't support SWFC, or when using devices in the communications link (such as modems, radios or line drivers) that don't support SWFC.

Figure 30 (P99) summarizes the recommended flow control settings for the two general types of RS-232 connections between a *DT80* and its host computer.

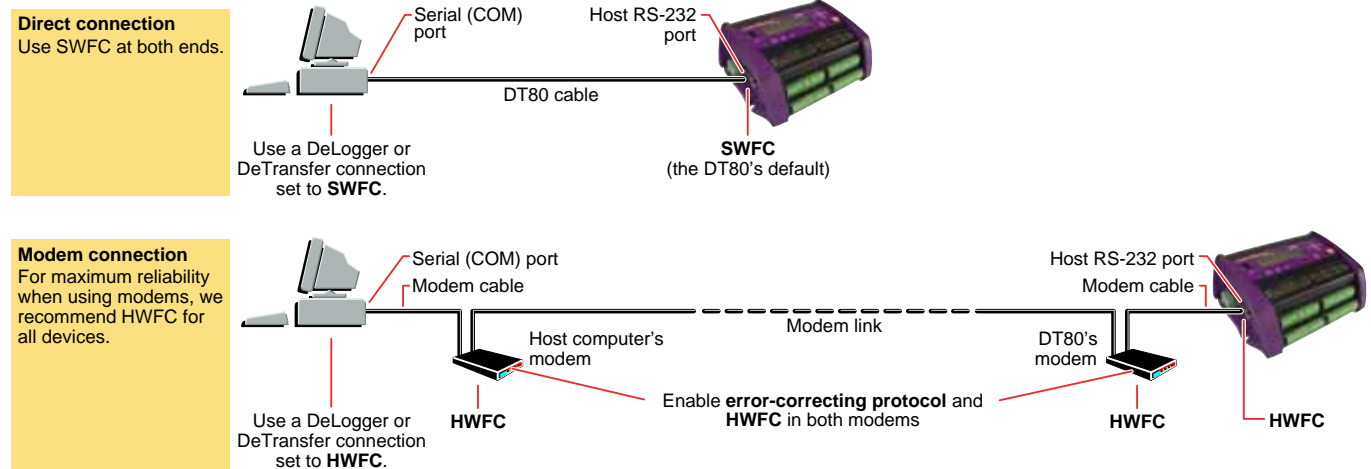


Figure 31: Recommended flow control settings for the two types of connections

### Software Flow Control (SWFC)

In SWFC mode, the receiver controls the flow of characters by transmitting

- the XOFF character (ASCII 19 or Control Q) to stop the sender from sending further characters
- the XON character (ASCII 17 or Control S) to allow the sender to resume sending characters.

During data return to the host computer, if the *DT80* receives an XOFF character from the host, it stops transmission within two character periods. Then when the host has processed the buffered data and is ready to receive again, the host should transmit an XON character to the *DT80*. This is the “resume transmission” signal to the *DT80*. (If the *DT80* is left in the XOFF mode by the host at the end of a transmission session, the *DT80* carries out an auto-XON after 30 seconds — see P26 (P110).)

Similarly, the *DT80* can control the transmission of commands and programs sent to it from the host computer. To do this, the *DT80* issues

- an XOFF character when its input buffer is 50% full (and at 75% full and 90% full)
- an XON character when its input buffer is empty.

### Hardware Flow Control (HWFC)

In HWFC mode, the transmission of characters is managed by the RTS (Ready To Send) and CTS (Clear To Send) lines of the RS-232 serial port of the sender and receiver. The state of these lines determines if transmission by the sender can proceed. The receiver raises the RTS line when it is able to receive characters from the sender, and lowers the RTS line when not able to receive characters. The RTS line of the receiver is connected (by means of the communications cable) to the CTS line of the sender, and the sender only transmits characters when its CTS line is high.

The *DT80* communications cable (product code IBM-6) has the RTS/CTS lines connected in this crossover manner — see Figure 73 (P171).

HWFC is inherently more reliable than SWFC and is therefore preferred, especially if there is any line or other noise on the communications link. SWFC can become confused if the flow control characters are corrupted or lost, whereas HWFC has constant levels that enforce the current flow control state at all times, making it highly resistant to line and other noise.

### No Flow Control (NFC)

The *DT80* can also be set to NFC (No Flow Control), in which case there is no control of the sender by the receiver. Use this setting with care, and only where there is no risk of the receiver being over-run by excess data from the sender, otherwise data loss will occur.

### SWHW (Both)

The *DT80*'s SWHW setting is provided for backwards compatibility. It enables both software flow control and hardware flow control at the same time.

### Echo

By default, the *DT80* **echoes** commands that it receives. That is, it automatically transmits received commands back to the host (see also echo (P181)).

This function can be turned off by sending the echo switch `/e` (P113). Also, it's forced off when the *DT80* is in fixed-format mode for returned data (P21)



## Input Buffer (How the DT80 Receives and Processes a Program)

The *DT80*'s input buffer can hold a maximum of 250 characters at any one time. This means that any single command or line of a program that is sent to the *DT80* must be shorter than 250 characters. Therefore, when writing long command and/or comment lines keep the length below 250 characters.

A command or program line is terminated by a carriage return character (this is how the *DT80* recognizes the end of the command or line), and the *DT80* begins to process the input buffer when each carriage return is received. A full 250 characters of program takes up to 500ms to process if the *DT80* is not scanning, and up to 5 seconds if it is running long schedules and many alarms. Any digital delay periods (such as `1DSO(1000)=0`) or general delay periods (such as `DELAY=1000`) add to this time.

The host must ensure that the *DT80* has sufficient time to process each line of a downloaded program. This is achieved by using either

- software flow control, or
- hardware flow control, or
- time delays between transmissions if no flow control is used.

## Comms Wakes the DT80

If communicating with the Host RS-232 port or the Serial Sensor port of the *DT80* while it's in sleep mode (low-power mode), the first few characters sent wake the *DT80*. To reliably wake a *DT80* that may be in sleep mode, it is recommended that a carriage return is sent or line feed character then wait 500ms before sending any commands. These instructions can be included at the start of your program.

---

## DT80 Direct (Local) RS-232 Connection

A common (and the most simple) way of communicating with the *DT80* (for configuring it, programming it and retrieving data from it) is to connect its Host RS-232 port directly to a serial port on the host computer using the RS-232 comms cable supplied. This is known as a **direct connection** to a **local DT80** (Figure 33([P100](#))).

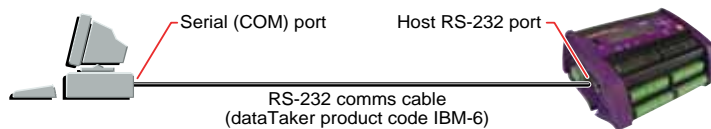


Figure 32: Direct (local) connection

The information in *DT80* RS-232 Basics ([P97](#)) applies to a direct connection.

For recommended flow control settings for this type of connection, see Figure 32. Direct RS-232 Cable

Communications cables for connecting the *DT80* (a DTE device) directly to a 9-pin computer comms port or 25-pin computer comms port (DTE devices) are detailed in Figure 73 ([P171](#)). A suitable 9-pin cable (*dataTaker* product code IBM-6) may be ordered for this purpose.

Since the DSR line is not connected (inactive) in these cables, the *DT80* automatically assumes a direct connection to the host computer — see Automatic Device Detection ([P97](#)) (Figure 30 ([P97](#)) in particular).

### Cable Length

Although the RS-232 standard specifies a cable of not more than 4 metres (15 feet), longer cables can be used. It's possible to use RS-232 cable runs of 100 metres or more, but to achieve reasonably error-free communication these generally need to have heavier wires and a slower baud rate may be necessary.

## Setting Up a Direct Connection

To set up a direct connection:

1. Connect a suitable comms cable between a serial port on the host computer and the *DT80*'s Host RS-232 port as shown in Figure 33([P100](#)). Cables are discussed in Figure 32([P100](#)) above.
2. Run suitable *dataTaker* or terminal software — for example, DeTransfer, DeLogger or HyperTerminal.
3. Configure the software to communicate using the same communications parameters that the *DT80* is set to (it defaults to 57600 baud, 8 data bits, 1 stop bit, no parity and software flow control).
4. Get the software to communicate by means of the computer comms port that is connected to the *DT80*.

---

## DT80 Modem (Remote) RS-232 Connection

Another common way of communicating with the *DT80* is to connect its Host RS-232 port to a wired or wireless modem, which communicates with another modem connected to the host computer at the other end of the comms link ([P101](#)). This way, the *DT80* can be across town or across the world from the host computer, and the link can use PSTN (landline), radio, GSM (cellular) or satellite communication. This is known as a **modem connection** to a **remote DT80**.

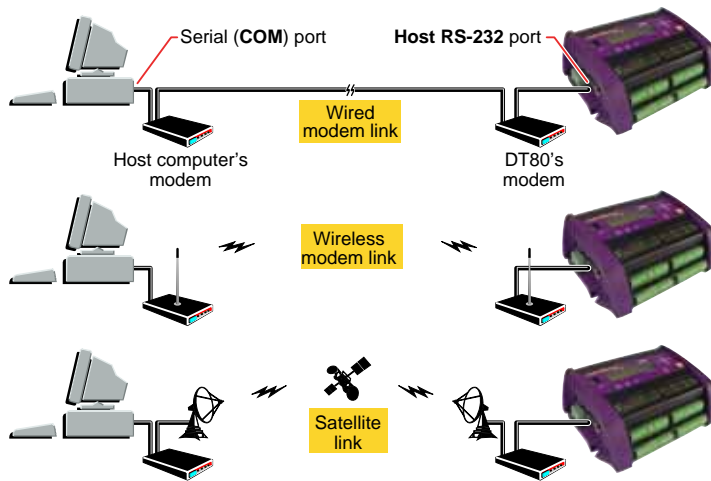


Figure 33: Modem (remote) connections

The information in *DT80 RS-232 Basics* ([P97](#)) applies to a modem connection.

The *DT80* can power the modem directly, or control the modem's power supply. If this facility is used, the *DT80* can automatically reset the modem if it determines that this is necessary. See *Powering the DT80's Modem* ([P102](#)).

### DT80-to-Modem Cable

For the *DT80* to recognise that it's connected to a modem and operates accordingly, the *DT80* must see the signal at the DSR terminal as active (discussed in *Automatic Device Detection* ([P97](#))). There are two ways to ensure this:

- Connect the *DT80*'s Host RS-232 port (a DTE device) to a modem (a DCE device) using a straight-through (full-parallel) comms cable as shown in Figure 74 ([P172](#)).  
When this is done, because the DSR line is connected in these cables and since most modems drive DSR active when on, the *DT80* automatically assumes it is connected by modem to the host computer and operates accordingly. This is the preferred method.
- If a modem is being used that does not drive its DSR line active when turned on, it is recommended that you hardwire DSR to DTR at the *DT80* end of the modem cable.  
This simulates an active DSR terminal, convincing the *DT80* that it's connected to a modem.

### Modem Initialization

The *DT80* monitors its modem and manages it automatically, as follows:

1. When the *DT80* detects that a certain initialization condition exists (see *Modem Initialization Conditions* ([P101](#)))...
2. the *DT80* resets the modem by turning its power off and then on again (must use the **EXT\_POWER\_SWITCH** profile key to specify to the *DT80* how the modem power is controlled — see *Figure 33* ([P101](#)))...
3. then sends initialising commands to the modem (see *Modem Initialization Settings* ([P101](#))).

### Modem Initialization Conditions

The *DT80* automatically attempts to initialise the device attached to its Host RS-232 port whenever it detects any of the following conditions:

- The state of the *DT80*'s DSR terminal changes from inactive to active.  
When this occurs, the *DT80* assumes that a modem has just been connected and therefore needs to be initialised.
- DSR is active at the time the *DT80* powers up or is reset.  
When this occurs, the *DT80* assumes that the attached modem may not have been powered up or reset at the same time as the *DT80* and therefore needs to be initialised.
- DSR is active but CD has been inactive for a specified period of time.  
When this occurs, the *DT80* assumes that the attached modem may be in an error state or locked-up, and therefore needs to be initialised. (Even if the modem is not in this state, the initialization does no harm.)  
Set this period using the **MAX\_CD\_IDLE** profile key — see **MAX\_CD\_IDLE** ([P115](#)) (the *DT80*'s default is 12 hours).

The *DT80* initialises the modem by automatically sending the commands and settings specified in the **HOST\_MODEM** ([P114](#)) section of the Table 11: *DT80* PROFILE Details ([P115](#)) (**INIT** ([P114](#))) to the modem when any of the above situations occur.

### Modem Initialization Settings

The *DT80*'s modem is to be initialised with the following settings:

- Auto-answer incoming calls after a specified number of rings (default is 4 rings).  
The *DT80* does not issue any commands to the modem to answer a call. Therefore, if dial-in functionality is required, the modem must be set to auto-answer incoming calls. The modem command **ATS0=4** instructs the modem to automatically answer after four rings.
- Don't echo commands.

Echo of commands is not required and only serves to confuse the *DT80* if it attempts to interpret the command echo as a command. The modem command **ATE0** turns echo off.

- Don't report results of commands.  
Results codes and strings only serve to confuse the *DT80* as it does not use these codes and may accidentally interpret them as commands. The modem command **ATQ1** enables quiet mode which stops result codes from being returned.
- The modem will keep DSR active at all times.  
This behaviour is ensured by sending the modem command **AT&S0**.
- If DSR goes inactive, the modem must terminate its call (if one has been made).  
This functionality is used by the *DT80* to hang-up its modem. This behaviour is ensured by sending the modem command **AT&D2**.
- The modem must make CD active whenever it establishes a connection with a remote modem, and make CD inactive when the connection is broken.  
The *DT80* interprets an active CD line to mean that it is connected to a remote host via a modem. This behaviour is ensured by sending the modem command **AT&C1**.

All of the above settings are included in the **INIT** profile key (see the **HOST\_MODEM** (P114) section of the Table 11: *DT80* PROFILE Details (P115) — the default value of the **INIT** profile key is **ATE0Q1&D2S0=4&C1&S0**.

The *DT80* issues these commands and their settings to its modem whenever it determines that the modem should be initialised (see Modem Initialization Conditions (P101) above).

### AT Command Set

The *DT80* supervises the modem using standard AT commands and common modem default settings where possible, and therefore supports the majority of modems.

### Modem Automatic Baud Rate Selection

Modems compatible with the AT command set — that is, most modems — automatically set their local baud rate to match that of the host when the first AT command is received. Therefore when the *DT80* sends the initialization string to the modem, it automatically sets the modem's local baud rate to that of the *DT80* (in addition to initializing the modem).

In other words, the modem's local baud rate does not have to be set manually.

### Modem Communications Protocol

It is recommended that an error-correcting protocol (eg. LAPM or MNP2-4) is used between the modems, along with local flow control. To enable an error-correcting protocol ("error control") in DeTransfer or DeLogger, ensure that this option is enabled in the host computer's modem settings. Do this by means of Windows' **Modems** control panel, in the modem's Advanced Connection Settings dialog box..

### Powering the DT80's Modem

When using a modem connection between the *DT80* and its host computer:

- use the *DT80*'s RELAY connection to control an external power supply to the *DT80*'s modem.
- use one of the *DT80*'s digital output channels to control an external power supply to the *DT80*'s modem..

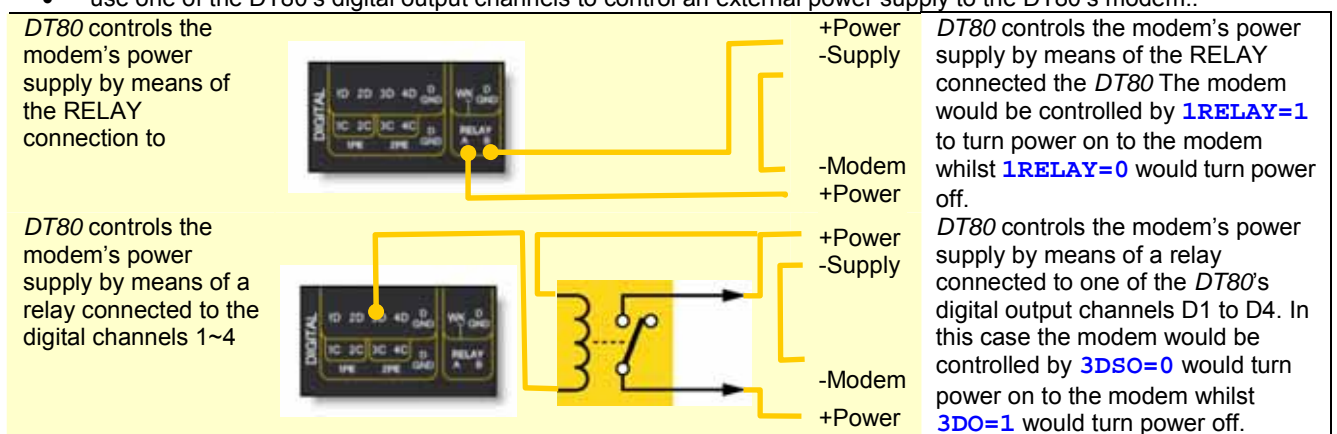


Figure 34: The *DT80* can power a modem directly, or control the modem's supply

### Automatic Modem Power-Down Reset

None of the above modem power arrangements provide for automatic power-down resetting of the *DT80*'s modem if the modem ceases to respond.

To enable this *DT80* feature, send one of the following **PROFILE...** commands (see the **HOST\_MODEM** (P114) section of the Table 11: *DT80* PROFILE Details (P115) table) to the *DT80*:

- If the modem is powered from one of the *DT80*'s digital output channels *n* (where *n* = 1 to 8), send the command **PROFILE"HOST\_MODEM" , "EXT\_POWER\_SWITCH"="n"**

- If the modem is powered via the *DT80*'s relay channel, send the command `PROFILE"HOST_MODEM", "EXT_POWER_SWITCH"="-1"`
- If the modem is not powered by either of the above, send the command `PROFILE"HOST_MODEM", "EXT_POWER_SWITCH"="0"` to disable the feature.

Then carry out a firm reset of the *DT80* (see Resetting the *DT80* [\(P119\)](#)) to load the `PROFILE...` command.

From then on, the *DT80* automatically power-down-resets the modem if it detects it to be unresponsive — see Modem Initialization [\(P101\)](#).

## Modem Communications Operation

### Dialling In

As the modem is initialised to automatically answer incoming calls, the *DT80* does not have to monitor the RI signal at its Host RS-232 port or request the modem to answer the call. But the *DT80* does have to monitor the CD signal to determine when a call has been established.

The *DT80* does not receive commands or transmit data and status information unless it determines that a call has been established between itself and a host. When a modem is attached (DSR active), the *DT80* monitors the CD signal to determine when it can transmit data and status information, and receive commands:

- When CD is active the *DT80* accepts commands, and returns data and status information.
- When CD is inactive the *DT80* ignores any received characters and does not transmit data or status information.

This behaviour ensures that any rubbish characters received outside of a call are ignored, and that the *DT80* does not send characters to the modem that the modem may interpret as commands to switch into a different operating state.

### Dialling Out

The `SETDIALOUTNUMBER` command is used to set the number to dial, and the `DIAL` and `HANGUP` commands are used to initiate and terminate calls by the *DT80*'s modem to the host computer's modem.

These are discussed in Host RS-232 Port Commands [\(P98\)](#).

### Modem Status

The system variable `25SV` gives an indication of the current state of the modem. It can be used with alarms to determine the current state of the modem connection and to behave accordingly.

See `25SV` [\(P32\)](#) in the Table 2: *DT80* System Variables [\(P32\)](#).

## Setting Up a Remote Connection

To set up a remote connection:

1. Configure the following *DT80* `PROFILE...` parameters using a direct connection between a local computer (running DeTransfer, DeLogger or HyperTerminal) and the *DT80* (Figure 35 [\(P103\)](#)):
  - a) Use `PROFILE"HOST_PORT", "BAUD_RATE"...` if the *DT80*'s baud rate is to be set to something other than its default, which is **57600** baud. (When the *DT80* initializes its modem, the modem is automatically set to this baud rate — see Modem Automatic Baud Rate Selection [\(P102\)](#).)
  - b) Use `PROFILE"HOST_PORT", "FLOW_CONTROL"...` to set the *DT80*'s flow control. Hardware flow control is preferred.
  - c) Use `PROFILE"HOST_MODEM", "EXT_POWER_SWITCH"...` if the *DT80* is to power-down reset the modem (see Figure 33 [\(P101\)](#)).
  - d) Reset the *DT80* to enable these `PROFILE...` settings to take effect. Use a firm reset (send the `SINGLEPUSH` command, or carry out a hardware reset or a power-up reset — see Resetting the *DT80* [\(P119\)](#)).

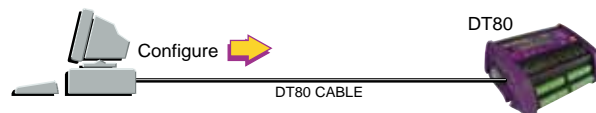


Figure 35: Use a direct connection to configure the *DT80* for a remote connection

Here's an example of commands that may be used:

```
PROFILE"HOST_PORT", "BAUD_RATE"="115200"
PROFILE"HOST_PORT", "FLOW_CONTROL"="HARDWARE"
PROFILE"HOST_MODEM", "EXT_POWER_SWITCH"="3"
SINGLEPUSH
```

2. Provide power to the modem and, if required, control its power as discussed in Powering the *DT80*'s Modem [\(P102\)](#).
3. Connect a suitable comms cable between the serial port on the *DT80*'s modem and the *DT80*'s Host RS-232 port. Suitable cables are discussed in *DT80-to-Modem Cable* [\(P101\)](#).

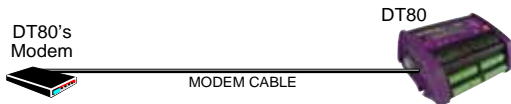


Figure 36: Connect the DT80 to its modem

4. At the remote/host end of the link, connect a suitable comms cable between the serial port on the host computer and the local modem.  
This cable is normally supplied with the modem.
5. On the host computer, launch suitable *dataTaker* or terminal software (for example, DeTransfer, DeLogger or HyperTerminal).
6. Configure the host software's communications parameters as required.  
It is recommended hardware flow control and the use of an error-correcting protocol.
7. Wait a few minutes after connecting the *DT80* to its modem (step 3 above), then attempt to connect to the remote *DT80* from the host computer.  
The few minutes' wait gives the *DT80* time to automatically initialize its modem (usually occurs within a minute of detecting the modem's presence but, for certainty, it is recommended to wait longer).

### Installing the Host Computer's Modem

If running DeTransfer or DeLogger Pro on the computer, install the local modem into Windows using the **Modems** control panel. Then access the modem settings from within DeTransfer's or DeLogger Pro's connection properties.

See [Figure 30 \(P99\)](#) and the appropriate *dataTaker* software manual for instructions on making this connection.

### Using the Modem Connection

Once initialization is complete, dial into the remote DT80/modem site from the host computer and the local modem to program the *DT80*, recover data, change program variables, and so on.

**Note:** The *DT80* can be programmed and configured during the same connection session.

### Visits to Site

If the site is visited where the *DT80* and the modem are installed, the *DT80* can be communicated with directly from a PC/Notebook by unplugging the cable from the modem to the *DT80* at the *DT80* end and then plugging in a direct cable, as supplied with the *DT80*, from your PC/Notebook to the *DT80*.

Differences in the cable wiring allow the *DT80* to determine the type of connection and to respond appropriately.

# DT80 Ethernet Communications

By means of the RJ-45 Ethernet port on the *DT80*'s side, you can communicate with the *DT80* to send commands to it and retrieve data from it over a 10BaseT Ethernet network.

### Ethernet Protocol

TCP/IP is the *DT80*'s default Ethernet protocol. See [TCP/IP \(P186\)](#).

### Configuring a DT80 for Ethernet

To use a *DT80* on a TCP/IP network, you must configure the *DT80* with three numbers:

- an **IP address** number
- an **IP subnet** mask number
- an **IP gateway** number.

There is only one way to do this. You use **PROFILE** commands to include these settings in the *DT80*'s USER.INI file, which the *DT80* automatically reads and applies after a firm reset (but not after a hard reset — see [Resetting the DT80 \(P119\)](#)). This is covered later in this section.

### Ethernet LEDs

The two LEDs on the *DT80*'s Ethernet port [\(P125\)](#) indicate the following:

- Green LED – Link OK; should come on and stay on as soon as you connect the Ethernet cable
- Amber LED – Activity; blinks every time a packet is received

---

## Ethernet Concepts

### IP Address

Every device — a computer or a *DT80*, for example — on an Ethernet network using one of the TCP/IP protocols must have



its own unique address, called its **IP address**. No two devices in the same network can have the same IP address — a device's IP address uniquely identifies it to all other devices on the network.

The IP address you assign to the DT80 is constructed from

- your network's **network number**, and
- an available (unused) **node number** (see later) for your DT80.

The topic DT80 Ethernet Setup ([P106](#)) explains how to construct a suitable (that is, valid and unused) IP address.

**Important** Do not connect your DT80 to the network until you've configured the DT80 with a suitable IP address. Connecting a device with an invalid or used IP address may generate incorrect address information in other devices on the network. (The DT80 is pre-set with a "safe" IP address to avoid this happening in the unlikely event that you accidentally connect a powered DT80 to the network without firstly configuring it with a valid, unused IP address.)

An IP address has the general form **n.n.n.n**, where each **n** is a one, two or three-digit number between 0 and 255 (**169.254.97.8**, for example).

## Assigning an IP Address

In general, a device can be assigned an IP address in two ways:

- **Dynamic IP Address**  
The network automatically assigns an IP address to the device whenever it logs on to the network. The address is temporary and may be different each time the device logs on.
- **Static IP Address**  
The user assigns an available IP address to the device. The address is stored in the device and reused every time it logs on to the network.

The DT80 has a Static IP Address— that is, you must assign an IP address to the *dataTaker*.

You do this by means of a **PROFILE...** command, which stores the address in the DT80's initialization file USER.INI. The command has the form

```
PROFILE"ETHERNET", "IP_ADDRESS"="w.x.y.z"
```

For example:

```
PROFILE"ETHERNET", "IP_ADDRESS"="192.9.200.15"
```

## Two-Part Construction

An IP address consists of two parts:

- A **network number** — the part of the IP address that is the same for a group of computers and devices on the same network. The network number identifies the network.
- A **node number** — the part of the IP address that uniquely identifies the node (the network device; a computer or a DT80, for example). No two devices on the network should have the same node number. Often called a **host number**.

There are five **classes** of IP addresses, and the two parts of an IP address vary according to the class of the IP address.

## The Bottom Line

Take great care when obtaining a correct, valid, unused IP address for assigning to your DT80. The topic DT80 Ethernet Setup ([P106](#)) contains step-by-step instructions for doing this.

## IP Subnet Mask, IP Gateway

Small networks are usually a single entity — just one network of nodes (devices) connected together by an Ethernet cable. In these cases, individual nodes on the network only require an IP address.

But larger networks may comprise a number of smaller networks, called **subnets**, connected together. These require an **IP subnet mask** (IPSN), another 32-bit number that determines how each IP address is subdivided into network portion and node portion. For example, network portions (underlined) can be 192.9.200.15 or 192.9.200.15 or 192.9.200.15. In other words, the subnet mask specifies which part or parts of every IP address are to be read as the larger network's address and which parts (the remaining parts) are to be read as the node's address.

Large networks may also be connected together through **gateways** (computers that connect multiple networks together, translating and routing one network's format, protocol or application information for use on another network). If a node on one network needs to communicate with a node on another network through a gateway, each node also requires an **IP Gateway** address. The DT80 is shipped with its IP gateway set to **0.0.0.0** — that is, no gateways.

In summary, all devices on a subnet must have

- the same network number (that is, the portion of the IP address that represents the network number must be the same for all devices on the same network)
- unique node numbers (the node number is what distinguishes one device from another)
- the same subnet mask (determines how the IP address is subdivided into network portion and node portion).

## External or Local Networks

- **External Network** If you want your networked DT80 to communicate with other networks or subnets (that is, across one or more gateways), you must specify the **IP Subnet Mask** for the DT80's subnet, and possibly an **IP Gateway** to access other networks.

Do this by sending the appropriate **PROFILE...** command to the DT80:

```
PROFILE"ETHERNET", "SUBNET_MASK"="s.t.u.v"
```

and/or

**PROFILE "ETHERNET" , "GATEWAY" = "o.p.q.r"**

See Ethernet Commands (P106) below.

- **Local Network** If you only want your networked DT80 to communicate locally (that is, only within the network or subnet to which it is connected), then you don't need to specify an **IP Subnet Mask**. The DT80 automatically applies a default IP subnet mask (**255.255.255.0**) that indicates "no subnets, no gateways".

## Ethernet Settings are Preserved

The Ethernet settings you specify using **PROFILE...** commands (IP address, IP subnet mask, IP gateway and protocol) are preserved through resets and full power-downs because they're stored in the DT80's USER.INI file, which the DT80 maintains in its internal flash memory.

## IP Port Number

The DT80 uses IP port number **8** for TCP/IP protocol communications.

Therefore, when setting up an Ethernet software connection between the DT80 and, say, DeTransfer or DeLogger, be sure to enter the correct IP port number in the software's port number field.

## Network Adapter Address

The DT80's **network adapter address** is the physical hardware address of the DT80's Ethernet port. This address is unique for each DT80 and cannot be changed. If you need a particular DT80's adapter address, use the EAA command in the table below.

Don't confuse the DT80's network adapter address with its IP address — they are not the same.

---

## Ethernet Commands

The current Ethernet parameters can be queried using the following commands

Command	Description
<b>IP</b>	Returns the DT80's current IP address
<b>IPSN</b>	Returns the DT80's current IP subnet mask
<b>IPGW</b>	Returns the DT80's current IP gateway
<b>EAA</b>	Returns the DT80's Ethernet network adapter address      Set at factory (read-only)

---

## DT80 Ethernet Setup

To configure your DT80 for use on an Ethernet network — even for a simple "network" of one computer and one DT80 using Ethernet — carry out the following steps in the order they're presented.

1. Obtain a valid, unused IP address for use on the Ethernet network.  
Ask your network administrator – if this is possible, do it and write the number down, then jump straight to step 2. If this is not possible, continue below.
  - a) Switch on every device on the network to which you're going to connect the DT80. This is vital to ensure automatic detection of every existing static IP address on the network (occurs later in this procedure).
  - b) Using a Windows computer that has TCP/IP installed, configured and operating correctly on the network, do the following:
    - i) From the **Start** menu select the **Control Panel**
    - ii) Select the **Network Connections** icon
    - iii) **Right Click** on the **network icon** which we have connected our logger to and **select status**.
    - iv) Write down the four-part number shown in the **IP Address** field (for example, **192.168.30.3**). The first three parts of this number (**192.168.30** in the example) constitute the network number you'll need for the DT80's IP address.
    - v) If you want the DT80 to communicate beyond the local network to which you're about to connect it, also write down the numbers shown in the **Subnet Mask** and **Default Gateway** fields of the IP Configuration dialog box. You'll use these later.
    - vi) From the **Start** menu, choose **Programs > MS-DOS Prompt**. An MS-DOS window opens. Here you'll use the **PING** command to locate a vacant IP address on the network using the network number found in substep iii above (**192.168.30**) followed by a fourth number between 1 and 254 (the node number) — we recommend you start with **1** as the fourth number.
    - vii) At the flashing DOS prompt, type **PING**, then a space, then the network number found in iii above, then a period (**.**), then the number **1** (**PING 192.168.30.1**, for example), then press the **Enter** key on your keyboard.
      - (1) If a "**Destination host unreachable**" message appears, the IP address **192.168.30.1** is not in use on the network and you may use it for the DT80.
      - (2) If a "**Reply from...**" message appears, the address is currently in use on the network and you can't use it for the DT80. In this case, type the PING command again at the DOS prompt but with a different last digit



(PING 192.168.30.2, for example) and press **Enter**. Repeat this procedure (changing the last digit each time) until the "Destination host unreachable" message appears to indicate that you've found an unused IP address.

viii) When you've identified an unused IP address, write it down and go to step 2.

2. Communicate with the DT80 using RS-232.



Figure 37: Ethernet setup — RS-232 communication first

a) Using the DT80's Host RS-232 port (not Ethernet), connect the DT80 to a computer and establish Host RS-232 communication using, say, DeTransfer software. Do not connect the DT80 to the Ethernet network yet.

3. Add the IP address to the DT80's initialization file.

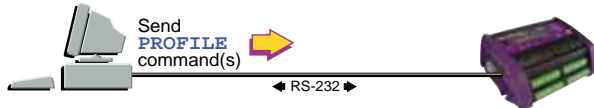


Figure 38: Ethernet setup — use RS-232 comms to add Ethernet settings to your DT80's user startup defaults

a) From DeTransfer, send the command

`PROFILE"ETHERNET", "IP_ADDRESS"="w.x.y.z"`

where **w.x.y.z** is the unused IP address you wrote down in substep 1-b-vii above.

The DT80's initialization file USER.INI now contains the IP address.

b) If you want the DT80 to communicate beyond its local network (that is, with external networks or subnets that are connected to your local network by gateways), go to step 4.

But if you only want the DT80 to communicate on the local Ethernet network, jump to step 5.

4. Add your network's IP subnet mask and IP gateway to the DT80's initialization file — only required if the DT80 is to communicate beyond its local network.

a) From DeTransfer, send

`PROFILE"ETHERNET", "SUBNET_MASK"="s.t.u.v"`

where **s.t.u.v** are the numbers you wrote down from the **Subnet Mask** field in substep 1-b-iv above.

b) From DeTransfer, send

`PROFILE"ETHERNET", "GATEWAY"="o.p.q.r"`

where **o.p.q.r** are the numbers you wrote down from the **Default Gateway** field in substep 1-b-iv above.

The DT80's initialization file USER.INI now contains the local network's IP subnet mask and IP gateway.

5. Reset the DT80.

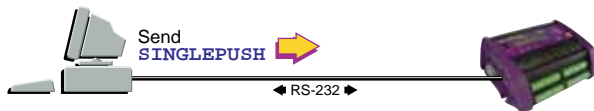


Figure 39: Ethernet setup — reset the DT80 to load the Ethernet PROFILE... setting(s)

a) From DeTransfer, send

`SINGLEPUSH`

or carry out a firm reset (see Resetting the DT80 (P119)) to force the new settings in USER.INI to take effect. The DT80 is now ready for Ethernet operation.

6. Connect the DT80 to the Ethernet network (or directly to a computer's Ethernet port) and test for correct operation.

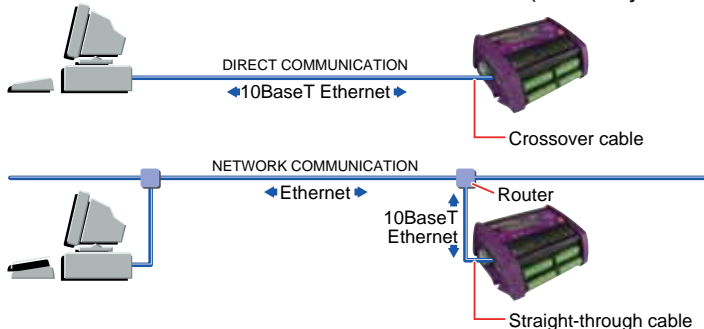


Figure 40: Ethernet connection types — direct connection and network connection

a) Using the correct cable — a standard 10BaseT cable (no crossover) — connect the DT80's Ethernet port to a socket on the Ethernet network (or directly to a computer's Ethernet port using a crossover cable).

b) Ensure that the DT80 is powered.

c) Using *dataTaker* host software — DeTransfer, DeLogger or DeLogger Pro — running on a computer also connected

to the Ethernet network (or, for a direct connection, on the computer to which the DT80 is directly connected), create a software connection for communication with the DT80. The connection must use TCP/IP protocol, the DT80's IP address (from substep 1-b-iv) and port number 8.

Connection Configuration

Connection Name: DT800 Ethernet

Connect Using: TCP/IP

IP Address

Port No: 8

Type a meaningful name for the Connection.

Select TCP/IP.

Type your DT80's IP address (w.x.y.z).

Type 8.

Figure 41: Example software connection for a DT80 on an Ethernet network (DeTransfer software shown)

- d) Send the **TEST** command and check that the DT80 returns a test report (see Test Report (*DT80 Health*) [\(P121\)](#)).
- If there's more than one DT80 on the network, verify that the report is from the correct DT80 by checking that the reported serial number is that of the DT80 you're testing.

## DT80 FTP Communications

The DT80 can function as an FTP server. You can use this mechanism (and third-party FTP software) to transfer data and program files to and from the DT80

The FTP server supports two types of access:

- an anonymous login (username **ANONYMOUS**, password can be anything) provides **read-only** access to the DT80's file system.
- a full login (using the username and password configured in the profile) provides **read/write** access.

Using an FTP client such as Microsoft Internet Explorer, you should be able to browse the DT80's file system by entering the URL:

```
ftp://ip-address/drive:/
```

where *ip-address* is the DT80's IP address and *drive* is the DT80 drive to browse (**A** or **B**).

## DT80 PPP Communications

Point-to-Point Protocol (PPP) allows TCP/IP-based protocols to be run over the Host RS-232 port of the DT80.

To initiate a PPP session with the DT80, clients must send the word **CLIENT** to the DT80 by means of its Host RS-232 port. The DT80 responds with **CLIENTSERVER** and, from then on, expects PPP packets. Note that Microsoft Windows PPP client software issues **CLIENT** and expects the **CLIENTSERVER** response by default.

To close a PPP session with the DT80, clients can simply close the PPP session from their end. Alternately, you can send the command **CLOSEDIRECTPPP** to the DT80 and it will close the PPP session.

# Part J — Configuration

## Configuring the DT80

### Parameters

#### Internal settings

**DT80 parameters** are internal system settings. They are global in their effect, and allow a variety of options to be set. As a general rule, set the parameters that require changing before programming schedules and alarms.

You can

- set or read parameters from the host computer, from a USB memory device program or from alarm actions
- pre-set parameters using a **PROFILE...** command — see the **PARAMETERS** ([P114](#)) row in the Table 11: *DT80* PROFILE Details ([P115](#)) table.

#### Reading Parameters

To read the current setting of a parameter, simply send the parameter's ID. For example, send [P22](#)

to read the current setting of parameter 22.

#### Setting Parameters

Parameters can be set at any time, and new settings generally take effect immediately. For example, send [P22=44](#)

to set parameter 22 to the value **44**.

Parameters are not the same as channels or variables. If you include a parameter in a schedule, it does not become part of the schedule. Instead, it is processed immediately.

In fixed-format mode (**/H**), three parameters are forced: **P22=44**, **P24=13** and **P38=46**. The previous values for these are restored on leaving fixed-format mode (**/h**).

Parameters P11, P46, P57 and P60 are sampled and stored at the time a schedule is defined. You should always set these parameters prior to defining the schedule.

The **DO** command can also be used for for executing parameter commands from within schedules. See Unconditional Processing — **DO...** Command ([P51](#)).

Parameter	Specifies	Units	Default Value	Range of Values	Comment
P3	Minimum sleep period	Seconds	4	1 to 30000	Sleep only if sleep duration can be for at least this period of time ( <i>DT80</i> assesses when next schedule is due to expire)
P4	Sleep-to-wake settling latency	Seconds	3	1 to 30000	Time required by <i>DT80</i> to resume normal operation after leaving sleep mode
P5	Maximum sleep period	Minutes	60	1 to 30000	
P9	Logging of alarm state	Mode	1	0 to 3	<b>P9=</b> <b>Transition Logged</b> <b>0</b> None <b>1</b> False to True only (Default) <b>2</b> True to False only <b>3</b> Both
P10	Logging of <b>TEST</b> results to event log	Mode	0	0 to 1	Send <b>P10=0</b> for <b>disable</b> , <b>P10=1</b> for <b>enable</b>
P11	Mains frequency (Set P11 to local mains frequency for best noise rejection.)	Hz	50	25 to 30000	Sets ADC sample duration to 1/Hz seconds. (Must be set <b>before</b> a schedule is defined)
P12	Minimum measurement period for frequency measurement	ms	20	1 to 30000	See Frequency ( <a href="#">P134</a> ).

P14	Comms ports password protection timeout	Seconds	600	1 to 30000	When a password is defined, the DT80 automatically signs off after this period of inactivity (see Password Protection — Comms Ports (P95)).
P15	Low-power operation	Mode	0	0 to 2	<b>P15= Mode</b> <b>0</b> Auto <b>1</b> Force Sleep <b>2</b> Force Normal Operation See Controlling Sleep (P132).
P17	Delay to low-power mode	Seconds	30	1 to 255	Sets how long the DT80 waits to enter low-power mode after communication or peripheral device activity and so on (see Controlling Sleep (P132)).
P22	Data delimiter character	ASCII	32 (space)	1 to 255	ASCII character (as decimal number) between data points in free-format mode (P21). Forced to 44 (comma) when in fixed-format mode. If using successive immediate schedules, see Cautions for Using Immediate Schedules (P46).
P24	Scan delimiter character	ASCII	13 (CR, which the DT80 automatically follows with LF)	1 to 255	ASCII character (as decimal number) between groups of data points in a scan in free-format mode (P21). <b>Note</b> When P24=13 (the default): the DT80 always automatically follows this carriage return (ASCII 13) with a line feed character (ASCII 10). See FORMAT OF RETURNED DATA (P21).
P26	XOFF timeout before XON	Seconds	30	0 to 255	Timeout before incoming XOFF state is automatically switched to XON state. <b>P26=0</b> disables timeout.
P31	Date format	Mode	1	0 to 5	<b>P31= Date Format</b> <b>0</b> day number <b>1</b> DD/MM/YYYY European <b>2</b> MM/DD/YYYY North American <b>3</b> YYYY/MM/DD ISO <b>4</b> D.D Decimal days since base date <b>5</b> S.S Decimal seconds since base date See Localisation (P13)
P32	Number of significant digits	# digits	7	1 to 9	Sets significant digits of output data. Note: logged data is always stored to 5 digits, so P32>5 is only useful for real-time data.
P33	Field width	# characters	0 (variable)	0 to 80	If P33>0 this defines fixed field width for all output data (right justified, space padded or least significant digits truncated). (P21)
P36	Temperature units	Mode	0	0 to 3	<b>P36= Units</b> <b>0</b> °C Celsius <b>1</b> °F Fahrenheit <b>2</b> K Kelvin <b>3</b> °R Rankin Data is converted before being placed into store and cannot be converted at unload time.

P38	Decimal point character	ASCII	46 (.)	1 to 255	The character used as a decimal point in floating-point numbers (see "Output Format")
P39	Time format	Mode	0	0 to 5	<p><b>P39= Time Format</b></p> <p><b>0</b> Hh:mm:ss.s</p> <p><b>1</b> s.s decimal seconds since midnight</p> <p><b>2</b> m.m decimal minutes since base</p> <p><b>3</b> h.h decimal hours since base</p> <p><b>4</b> D.D decimal days since base</p> <p><b>5</b> s.s decimal seconds since base</p> <p>See Time (<a href="#">P30</a>).</p>
P40	Time separator character	ASCII	58 (:)	1 to 255	ASCII character (as decimal number) separator character for hh:mm:ss time format
P41	Time sub-second digits	Digits	3	0 to 6	Sets number of digits in time outputs
P45	DDE/OLE tag control	Mode	0	0 to 2	<p><b>P45= Mode</b></p> <p><b>0</b> off</p> <p><b>1</b> OLE (\$!)</p> <p><b>2</b> DDE (&amp;!)</p> <p>Whitespace is replaced with underscore</p>
P50	Time instant format (format of instants in time)	Mode	0	0 to 5	<p><b>P50= Mode</b></p> <p><b>0</b> P39P22P31 (time, delimiter, date)</p> <p><b>1</b> P31P22P39 (date, delimiter, time)</p> <p><b>2</b> s.s decimal seconds since base</p> <p><b>3</b> m.m decimal minutes since base</p> <p><b>4</b> h.h decimal hours since base</p> <p><b>5</b> D.D decimal days since base</p> <p>See <b>TOR</b> and <b>TOF</b> in the Digital Manipulation (<a href="#">P36</a>) category in the Table 3: DT80 Channel Options (<a href="#">P38</a>) table.</p>
P51	Time interval format	Mode	0	0 to 5	<p><b>P50= Mode</b></p> <p><b>0</b> P39P22P31 (time, delimiter, days)</p> <p><b>1</b> P31P22P39 (days, delimiter, time)</p> <p><b>2</b> s.s decimal seconds</p> <p><b>3</b> m.m decimal minutes</p> <p><b>4</b> h.h decimal hours since base</p> <p><b>5</b> D.D decimal days since base</p> <p>See <b>TRR</b>, <b>TRF</b>, <b>TFR</b> and <b>TFF</b> in the Digital Manipulation (<a href="#">P36</a>) category in the Table 3: DT80 Channel Options (<a href="#">P38</a>) table.</p>
P53	Serial sensor timeout	Seconds	10	0 to 60	Send <b>P53=0</b> for <b>no timeout</b>
P55	Enable schedule wakeup	Bits	16383	0 to 16383	Bitmap entered as decimal value <b>S K J I H G F E D C B A X *</b> ( <b>S</b> = bit 13, <b>X</b> = bit 1, <b>*</b> = 0)
P56	Serial Channel debugging — see Serial Channel Debugging Tools ( <a href="#">P156</a> )				
P57	Number of frame capture cycles	Count	5	1 to 100	Attempts left to capture complete frame set (must be set <b>before</b> a schedule is defined)

P58	Excitation tolerance	%	10	0 to 100	Sets tolerance of excitation adjustment relative to specified value. Send <b>P58=100</b> for <b>don't adjust excitation</b> .
P59	Maximum gain for automatic gain-ranging	Mode	10	0 to 10	<b>P50= Mode</b> <b>0</b> 20V (Note: maximum analog input voltage must not exceed ±13V) <b>1</b> 10V <b>2</b> 5V <b>3</b> 2V <b>4</b> 1V <b>5</b> 500mV <b>6</b> 200mV <b>7</b> 100mV <b>8</b> 50mV <b>9</b> 20mV <b>10</b> 10mV See (P17).
P60	Maximum sampling frequency	kHz	50	1 to 100	ADC speed (must be set <b>before</b> a schedule is defined)
P61	Internal measurement check interval	Sec	3	0 to 30000	Send <b>P61=0</b> to <b>disable</b>
P63	Gain-set to use		1	0 to 1	Send <b>P63=0</b> for <b>default gain-set</b> , <b>P63=1</b> for <b>characterized gain-set</b>

Table 9: DT80 Parameters

## Switches

UPPERCASE = ON, lowercase = off

**DT80 switches** are analogous to electrical switches, and are turned on by uppercase and off by lowercase. Like parameters, switches are internal system settings and generally global in effect; unlike parameters, switches can only have two values — on or off. Switch commands can be issued at any time, and most take effect immediately. Delay in effect may occur if data is buffered in the *DT80* or in the host computer.

Use a **PROFILE...** command to pre-set switches — see the **SWITCHES** (P114) row in the Table 11: *DT80* PROFILE Details (P115) table.

The **DO** command can also be used for executing switch commands from within schedules. See Unconditional Processing — **DO... Command** (P51).

### Viewing Switch Settings

The **STATUS9** command returns the current switch settings to the host. For example

/B/C/d/E/f/h/i/k/l/M/N/r/S/t/U/v/w/x/Z

Switch Enabled	Switch Disabled	Function	Default	Comment
/B	/b	N/A	/B	Not used.
/C	/c	Include <u>C</u> hannel type	/C	Channel type is included with channel number in returned data — for example <b>5PT392</b> instead of <b>5</b> . See <b>FORMAT OF RETURNED DATA</b> (P21).
/D	/d	Add <u>d</u> ate to returned data	/d	Equivalent to a <b>D</b> at beginning of a schedule's channel list
/E	/e	<u>E</u> cho	/E	Enables echo of commands to host (if not unloading data and not in fixed-format mode). Useful in terminal mode communications with the <i>DT80</i> .
/F	/f	<u>F</u> ix (lock) schedules	/f	Prevents a <i>DT80</i> 's scan schedules (trigger or channel list) being modified. Note that a reset still erases the schedules.
/H	/h	Fixed-format ( <u>H</u> ost) mode	/h	Fixed-format mode of data returned from <i>DT80</i> (P21)
/I	/i	Schedule ID	/i	Returns schedule identifier
/K	/k	Enable default internal housekeeping measurements	/K	Allows <i>DT80</i> to measure battery, charger and other internal quantities — checks these every P61 seconds (see P61 (P112)) if awake

Switch Enabled	Switch Disabled	Function	Default	Comment
/L	/l	dataTaker serial number prefix	/l	Prefixes the DT80's serial number to a schedule's returned data — for example <b>dataTaker DT80 080015 5PT385 232.5</b> indicating the data is from <b>dataTaker DT80</b> number <b>080015</b> .
/M	/m	Messages	/M	Enables error and warning messages to be returned to host (see ERROR MESSAGES (P174))
/N	/n	Channel numbers	/N	Includes channel number (and type if /C switch is on) with returned data. See FORMAT OF RETURNED DATA (P21).
/R	/r	Return data	/R	Allows real-time data to be returned to the host computer. /r can reduce power consumption.
/S	/s	Synchronize to midnight	/S	Synchronizes all schedules' time intervals to midnight — for example, <b>RA1M</b> scans on the minute. Otherwise schedules run from entry time. See Time Triggers — Synchronizing to Midnight (P48).
/T	/t	Add time to returned data	/t	Equivalent to a <b>T</b> at beginning of a schedule's channel list
/U	/u	Units text	/U	Appends measurement units to returned data (see FORMAT OF RETURNED DATA (P21)) and makes error messages verbose (see ERROR MESSAGES (P174))
/W	/w	Intermediate (working) channels	/w	Allows working channels (see <b>W</b> channel option (P37)) to be reported but not logged. See also CALCULATIONS (EXPRESSIONS) (P65).
/X	/x	Progressive maxima and minima	/x	Display the progressive maximum and minimum values for statistical channels on the built-in display only.
/Z	/z	Stops alarm messages	/Z	Enables alarms to issue action text to host computer or printer. See Alarm Action Text (P80).
//	–	Default switches	–	Sets all switches to their default state

Table 10: DT80 Switches

## User Startup Defaults

The DT80 can start up after a firm reset (a **SINGLEPUSH** command, a hardware reset or a power-up reset) pre-configured with the preferred settings and running the preferred job:

- To make the DT80 start with the **preferred configuration settings**, see data logging system (P180) below. Use this method to set startup defaults for time format, date format, mains frequency and other characteristics listed in the Table 11: DT80 PROFILE Details (P115) table below.
- To make the DT80 start running the **preferred job**, see Startup Job (P116).

The defaults can be protected against possible corruption — see Protecting Startup Files (P117).

### User Startup Profile

The DT80 can store the preferred settings and automatically apply them after it is restarted by a firm reset (see Table 12: DT80 Resets (P119)). In this way, the DT80 is automatically returned to the chosen configuration every time it (or, more accurately, its relevant sub-system) is re-started. This saves repeatedly specifying the same settings in every program sent to the DT80.

Configuration settings for DT80 sub-systems such as the Host RS-232 port, host modem, Ethernet interface, FTP server, jobs, switches and parameters can all be restored in this way. In addition, this facility can be used to store information in the DT80.

#### USER.INI (User Initialization File)

The default settings are stored in the DT80's user initialization file USER.INI, which is automatically read and applied upon DT80 startup and sub-system initialization. If USER.INI contains no user default for a particular setting, the DT80 uses its factory default (listed in the table below).

Use **PROFILE...** commands to build the USER.INI file. These commands are described in PROFILE... Commands (P115) and use information from the table below. The table lists the pre-defined DT80 sub-systems — called **sections** — for which can be specified startup defaults.

USER.INI is created the first time a **PROFILE...** command is sent to the DT80. It resides in the DT80's battery-backed SRAM memory and is maintained through all types of reset. It is only lost/deleted if both the internal main battery and the internal memory-backup battery are removed from the DT80, or if a **FORMAT"B: "** command is sent (see the Table 12: DT80 Resets (P119)). In addition, the DT80 automatically stores a backup copy of USER.INI in the DT80's Flash memory — see Protecting Startup Files (P117).

#### When Do User Defaults Take Effect?

After a **PROFILE...** command is sent to the DT80 to set values in the USER.INI file, the DT80's behaviour does not change until the next time the sub-system using that value is re-started. For example:



- If the USER.INI file contains special switch and parameter settings, these are applied (and re-applied) every time a firm reset is carried out<sup>11</sup>. They are not applied until the next reset after the **PROFILE...** command is sent that specifies them.
- **Ethernet Settings** Firstly define a DT80's Ethernet settings using **PROFILE...** commands. Then, for the Ethernet settings to take effect, a firm reset must be carried out (see Resetting the DT80 [\(P119\)](#)).

## Persistent Settings

Settings specified using USER.INI can be considered “persistent” because they are reapplied at every restart. Individual commands can be sent to the DT80 (a parameter command such as **P32=2**, or a baud rate command such as **PH=1200**, for example) that immediately overrides one or more of the DT80's current/permanent settings, but these are only “temporary” in that they will be automatically replaced by the persistent **PROFILE...** settings the next time the related sub-system is restarted.

Section	Initialization Key (Setting Name)	Legal Values	Factory Default <sup>114</sup>	Comment
<b>PARAMETERS</b>	<b>Pn</b>	<i>number</i>	See the Default Value <a href="#">(P109)</a> column in the Table 9: DT80 Parameters <a href="#">(P112)</a>	Any parameter can be pre-set; for example, <b>PROFILE"PARAMETERS", "P11"="60"</b> sets the DT80 for a 60Hz mains frequency environment, <b>PROFILE"PARAMETERS", "P31"="2"</b> sets the DT80 to North American date format.
<b>SWITCHES</b>	<b>A, B, C,...Z</b>	<b>OFF, ON</b>	Varies with switch — see the Default <a href="#">(P112)</a> column in the Table 10: DT80 Switches <a href="#">(P113)</a>	Any switch can be pre-set; for example, <b>PROFILE"SWITCHES", "A"="ON"</b> Switches are checked and set before parameters on every soft or firm reset.
<b>PPP</b>	<b>IP_ADDRESS</b>	<i>nnn.nnn.nnn.nnn</i>	1.2.3.4	
	<b>REMOTE_IP_ADDRESS</b>	<i>nnn.nnn.nnn.nnn</i>	1.2.3.1	
	<b>USER</b>	<i>string</i>	ANONYMOUS	
	<b>PASSWORD</b>	<i>string</i>	PASSWORD	
<b>HOST_PORT</b> (for DT80's Host RS-232 port)	<b>BPS</b>	<b>50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200</b>	57600	Baud rate to use on DT80's Host RS-232 port
	<b>DATA_BITS</b>	<b>7, 8</b>	8	Number of data bits per character
	<b>STOP_BITS</b>	<b>1, 2</b>	1	Number of stop bits per character
	<b>PARITY</b>	<b>NO, EVEN, ODD</b>	NO	Type of parity to include with each character
	<b>FLOW</b>	<b>HARDWARE, SOFTWARE, BOTH, NONE</b>	SOFTWARE	Type of flow control to use on DT80's Host RS-232 port
<b>HOST_MODEM DIAL</b> (for modem connected to DT80's Host RS-232 port) See DT80 Modem (Remote) RS-232 Connection <a href="#">(P100)</a> .	<b>INIT</b>	<i>string</i>	ATD	Issued as prefix to number specified in the <b>SETDIALOUTNUMBER</b> command
			ATE0Q1&D2S0=4&C1&S0	Automatically issued by the DT80 to initialise a connected modem (see Modem Initialization Settings <a href="#">(P101)</a> ). To disable automatic initialization, set to empty string.

<sup>11</sup> A singlepush reset —see [DT80 Resets](#).

Section	Initialization Key (Setting Name)	Legal Values	Factory Default <sup>114</sup>	Comment
	<code>EXT_POWER_SWITCH</code>	<code>0, -1, n</code>	0	<code>PROFILE"HOST_MODEM", "EXT_POWER_SWITCH"=-1</code> instructs the DT80 to use its relay terminals to supply and control power for the modem — see <a href="#">Figure 34 (102)</a> . <code>PROFILE"HOST_MODEM", "EXT_POWER_SWITCH"=n</code> instructs the DT80 to use its digital output channel <i>n</i> ( <i>n</i> = 1 to 8) to control an external power supply to the modem — see <a href="#">Figure 34 (102)</a> . <code>PROFILE"HOST_MODEM", "EXT_POWER_SWITCH"=0</code> disables this function.
	<code>MAX_CD_IDLE</code>	<i>number</i>	43200 seconds (12 hours)	<i>number</i> must be ≥0. The number of seconds to wait while CD is inactive before re-initialising the modem. Set to 0 to disable this function.
	<code>SEND_BANNER_CONNECT</code>	<code>YES, NO</code>	YES	When set to <code>YES</code> a string such as <code>dataTaker 800 Version 4.01</code> is sent whenever CD changes from inactive to active.
	<code>COMMAND_PROCESSING_TIME</code>	<code>1 to 32767</code>	1 second	The number of seconds the DT80 waits after sending a command to the modem (to give the modem time to respond).
	<code>IP_ADDRESS</code>	<i>nnn.nnn.nnn.nnn</i>	0.0.0.0	
	<code>SUBNET_MASK</code>	<i>nnn.nnn.nnn.nnn</i>	255.255.255.0	
	<code>GATEWAY</code>	<i>nnn.nnn.nnn.nnn</i>	0.0.0.0	
<code>FTP_SERVER SUPPORTED</code>		<code>YES, NO</code>	YES	Starts the DT80's FTP server functionality (using <code>NO</code> frees a small amount of DT80 memory)
	<code>USER</code>	<i>string</i>	DATATAKER	FTP username for read/write access
	<code>PASSWORD</code>	<i>string</i>	DATATAKER	FTP password for read/write access

Table 11: DT80 PROFILE Details

**Important:** Characters used within strings ARE case sensitive. Therefore ALL settings should be in case shown (usually uppercase).

### Deleting USER.INI

Delete the working copy of USER.INI from the DT80's internal memory by sending

`DELUSERINI`

A new USER.INI will be created the next time you send a `PROFILE...` command. See also [Deleting the Backup Files from Flash \(P117\)](#) to remove the copy of USER.INI that was automatically created by the DT80.

### PROFILE... Commands

The following `PROFILE...` commands are available for reading and modifying the contents of the USER.INI file:

`PROFILE"section", "key"="string` Sets the value of *section's key* in USER.INI to *string*

<code>PROFILE"section","key"</code>	Returns the current value of <i>section's key</i> in USER.INI	
<code>TYPE"B:\\INI\\USER.INI"</code>	Displays the entire USER.INI file	
<code>PROFILE"section","key"=</code>	Removes <i>section's key</i>	
<code>PROFILE"section","key"=""</code>	Sets the value of <i>section's key</i> to the empty string	
<code>PROFILE</code>	Returns the currently-active profile settings	The currently-active settings may be different from the settings in USER.INI.
<code>PROFILE"section"</code>	Returns <i>section's</i> currently-active settings	

where

<i>section</i>	is the DT80 sub-system	See the Table 11: DT80
<i>key</i>	is the sub-category of the <i>section</i>	PROFILE
<i>string</i>	is the value to which the <i>key</i> is to be set	Details <a href="#">(P115)</a> table.

## No Checking

Note that *section*, *key* and *string* contain free text that is not validated in any way. So, if an entry is misspelled or uses an illegal value, there is no warning. The DT80 simply ignores that line in the USER.INI file and instead starts up using the factory default for that setting. For example, if accidentally a `PROFILE...` command is sent to set the baud rate to `9a00` or `9601` (instead of `9600`), the DT80 ignores the value and uses its factory default instead.

## Examples — PROFILE... Commands

Set Ethernet user defaults:

```
PROFILE"ETHERNET","IP_ADDRESS"="192.168.1.168"
PROFILE"ETHERNET","UDP_SUPPORTED"="YES"
```

Set Host RS-232 port user defaults:

```
PROFILE"HOST_PORT","BPS"="115200"
```

Set modem user defaults:

```
PROFILE"HOST_MODEM","DIAL"="ATDP"
PROFILE"HOST_MODEM","COMMAND_DELAY"="1"
```

## Startup Job

The DT80 can automatically run a user-defined job

- every time it is restarted by a firm reset — see ONRESET.DXC [\(P116\)](#) below, or
- every time a USB memory device containing a special version of the job is inserted into the DT80 — see ONINSERT.DXC [\(P116\)](#) below.

### ONRESET.DXC

To make the DT80 automatically run a job as the DT80's startup job, send the command

```
RUNJOBONRESET"JobName"
```

to the DT80. This places the contents of the job named *JobName* (must already exist in the DT80) into a file named ONRESET.DXC in the root directory of the DT80's B: drive. Then the DT80 automatically runs this file every time it is restarted by a firm reset<sup>12</sup>. (Alternatively, copy an ONRESET.DXC file directly to the DT80's root directory from a USB memory device, by FTP transfer, or over Ethernet.)

The working copy of ONRESET.DXC can be deleted from the DT80's internal memory, which stops this startup job function, by sending

```
DELONRESET
```

To backup the ONRESET.DXC file — see Protecting Startup Files [\(P117\)](#). Note, if the working copy is deleted, the automatic replacement-on-reset function described in Protecting Startup Files [\(P117\)](#) ceases.

The contents of an ONRESET.DXC file cannot be altered. A new file must be created directly in the DT80 using the `RUNJOBONRESET"JobName"` command again.

### ONINSERT.DXC

When a USB memory device is inserted into a DT80, the DT80 first looks for a subdirectory on the card named with its own serial number. If it finds such a subdirectory, it automatically loads and runs any ONINSERT.DXC file it finds in the subdirectory.

If there is no subdirectory named with the DT80's serial number, the DT80 automatically loads and runs any ONINSERT.DXC file it finds in the root directory of the card.

This auto-programming function means that a single USB memory device can be inserted into a number of DT80s, one at a

<sup>12</sup> A singlepush reset —see [DT80 Resets](#).

time, and

- automatically program all the *DT80*s with the same job — if no serial-number-specific subdirectories containing ONINSERT.DXC files exist on the card and an ONINSERT.DXC file exists at the root level, or
- automatically program particular *DT80*s with their own specific job — if serial-number-specific subdirectories containing ONINSERT.DXC files exist on the card, or
- carry out a combination of these two options — *DT80*s that do not find a subdirectory named with their serial number automatically load and run the ONINSERT.DXC file at the root level, and *DT80*s that find their specific subdirectory automatically load and run the ONINSERT.DXC file found there.

An ONINSERT.DXC file for use with a specific *DT80* can be created by sending

```
RUNJOBONINSERT "JobName"
```

while the card is inserted in that *DT80*. This copies *JobName* into the specific *DT80* serial number directory on the card (named, for example, SN080271; creates directory if it does not exist) and calls it ONINSERT.DXC.

To create a generic ONINSERT.DXC file (for use with any *DT80*) by sending

```
RUNJOBONINSERTALL "JobName"
```

while the card is inserted in the *DT80*. This copies *JobName* into the card's root directory and calls it ONINSERT.DXC. Then, when the card is inserted into a *DT80*, if no *DT80*-specific directory exists, the generic ONINSERT.DXC file is run.

## Deleting ONINSERT.DXC

The following delete commands are available for deleting ONINSERT.DXC:

```
DELONINSERT      Deletes ONINSERT.DXC from the DT80's serial number directory on the card
```

```
DELONINSERTALL  Deletes ONINSERT.DXC from the root directory on the card
```

## Protecting Startup Files

The *DT80* has

- an automatic mechanism for protecting its USER.INI (user startup profile) file
- an optional mechanism for protecting its ONRESET.DXC (startup job) file

by making backup copies of these in Flash memory. (The original/working files reside in internal memory drive.) See Figure 42 (P117). Protecting startup files 1 — backups are created. Then, whenever a firm reset is carried out, the *DT80* automatically overwrites the working version (possibly corrupt) in internal memory drive with a copy of the backup version (clean/uncorrupt) from Flash memory and runs the new, clean copy in internal memory. See Figure 28 (P117).

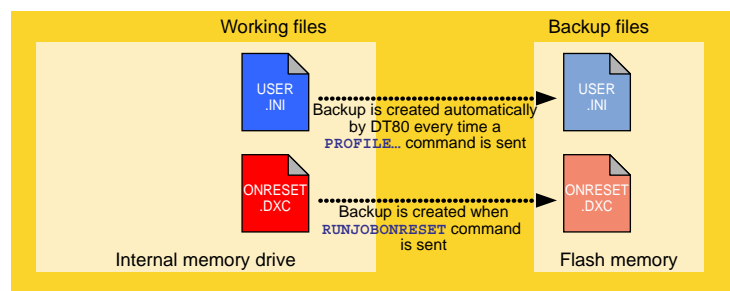


Figure 42: Protecting startup files 2 — backups replace working files on firm reset

## Protecting USER.INI

This process is completely automatic.

The first time a **PROFILE...** command is sent to the *DT80*, USER.INI (the working copy) is created in internal memory, and a backup copy of it is created in Flash memory. Subsequently, every **PROFILE...** command sent

- updates the working version of USER.INI
- causes the backup in Flash to be overwritten with a copy of the updated USER.INI.

In this way, the *DT80* maintains an up-to-date backup of the latest profile settings.

Then, whenever a firm reset occurs, the *DT80* automatically replaces its working USER.INI with a copy of the latest, clean backup version.

## Protecting ONRESET.DXC

A copy of ONRESET.DXC is created in the *DT80*'s Flash memory, and automatically replaces the working copy (in internal memory) every time a reset occurs.

## Deleting the Backup Files from Flash

DELUSERINI command will delete the copy of the USER.INI in flash as well as the one held on the internal memory drive. The RUNJOBONRESET will copy the ONRESET.DXC job to the flash as well as the internal memory drive. The DELJOBONRESET will delete the ONRESET.DXC file from both FLASH and the internal memory drive.

A backup will be re-created the next time you send a **PROFILE...** command.

Without the backup file in Flash, the automatic protection mechanism described above no longer occurs.

---

## Setting the DT80's Clock/Calendar

The DT80's real-time clock/calendar is based on a 24-hour clock that has a resolution of 122µs.

To make the timestamps and datestamps that the DT80 includes with readings meaningful, set the DT80's clock/calendar to local time and date. This is described in the next two topics.

Time and date are maintained in when the logger is switched off or reset. If the logger is switched off and the internal Memory-Backup battery (see Internal Memory-Backup Battery [\(P130\)](#)) is removed or discharged, then the date and time will be set to 1989/01/01 00:00:00

### Setting the DT80's Time (T=)

When setting the *dataTaker* data logger's clock use the time format defined by P39 and P40. For example, if P39=2 (in this case P40 does not matter), then the clock time must be set as a decimal value:

**T=11.7528**

Time is maintained through both software and hardware resets.

Time and date stamps can be added to real-time data and to logged data — see /T and /D in Switches [\(P112\)](#). Time and date are automatically logged whenever data is stored.

### Setting the DT80's Date (D=)

When setting the *dataTaker* data logger's calendar use the date format defined by P31. For example, if P31=2, then the date must be set in North American format:

**D=02/01/2000**

The DT80's date is maintained through both software and hardware resets.

Time and date stamps can be added to real-time data and to logged data — see /T and /D in Switches [\(P112\)](#). Time and date are automatically logged whenever data is stored.

### Setting Date and Time Together (DT=)

Use the DT= command to set the DT80's date and time simultaneously, independent of P31 (date format) and P39 (time format). The command's formats match those of the date and time fields of fixed-format mode records. The command looks like

**DT=[YYYY/MM/DD, hh:mm:ss]**

where

<b>YYYY</b>	is the year	Date
<b>MM</b>	is the month	
<b>DD</b>	is the day	
<b>hh</b>	is the hour	Time
<b>mm</b>	is the minute	
<b>ss</b>	is the second	

Here's an example:

**DT=[2000/11/23,13:09:40]**

# Resetting the DT80

The DT80 provides the following methods for clearing and initializing its sub-systems:119

Command/Action		Effect of Command/Action				
		COMMUNICATIONS ENVIRONMENT		PROGRAM/JOB ENVIRONMENT (Current Job/Program, CVs, IVs, \$\$s, Spans and Polynomials)	Logged Data and Logged Alarms	ONRESET.DXC (Startup Job)
		Current Comms Connections	Comms Parameters			
<b>RESET</b>	Command sent to the DT80	Maintained	Maintained	Cleared	Maintained	Not run
↑ <b>Soft reset</b> — simply clears your current work (the program/job environment).						
<b>SINGLEPUSH</b>	Command sent to the DT80	Disconnected (unless, for Host RS-232 comms only, the DT80's RS-232 connection settings match its factory defaults; TCP/IP is always disconnected)	<b>Uses PROFILE... defaults</b> (see data logging system <a href="#">(P180)</a> )	Cleared	Maintained	Run (see Startup Job <a href="#">(P116)</a> )
Hardware Reset ("hardware singlepush")	<b>One</b> push of the hardware reset button (see Manual Reset Button <a href="#">(P120)</a> below)					
Power-Up Reset	Remove then replace all main power (external supply and internal main battery).				Maintained (unless internal backup battery is also removed)	
↑ <b>Firm resets</b> — like soft reset, but DT80 restarts with user defaults and user startup job. Three methods of achieving the same result.						
Triple-Push Reset	<b>Three</b> pushes of the hardware reset button <b>within 10 seconds</b> (see Manual Reset Button <a href="#">(P120)</a> )	Disconnected (unless, for Host RS-232 comms only, the DT80's RS-232 connection settings match its factory defaults; TCP/IP is always disconnected)	Uses factory defaults (see Factory Defaults <a href="#">(P120)</a> )	Cleared	Maintained	Not run
↑ <b>Hard reset</b> — like firm reset, but DT80 restarts with factory defaults and does not load a new job. Ignores (does not delete) user defaults. For use in case of serious problems (for example, a faulty startup job, or inability to communicate with the DT80 because of invalid user defaults).						
<b>FACTORYDEFAULTS</b>	Command sent to the DT80	Disconnected	Uses factory defaults (see Factory Defaults <a href="#">(P120)</a> )	Cleared	Maintained	Deleted (and not run)
↑ Like triple-push reset but also deletes user defaults.						
<b>FORMAT"B:"</b>	Command sent to the DT80	Maintained	Maintained	All jobs (including the current job) are halted and removed; program/job environment is reset	Cleared (internal memory only; not the USB memory device)	Not run
↑ For low-level system recovery and purging. Reformats internal memory disk and rebuilds standard directory structure.						

Your logged data and alarms are **not** deleted

Table 12: DT80 Resets

See also MEMORY [\(P125\)](#).

## Maintaining Parameters and Switches through a Reset

You can use PROFILE... commands to re-apply parameter and switch settings after every soft and firm reset. See the **PARAMETERS** [\(P114\)](#) and **SWITCHES** [\(P114\)](#) sections in the Table 11: DT80 PROFILE Details [\(P115\)](#) table.

## Wait after RESET

Do not send any commands to the DT80 for five seconds after sending either the **RESET** command or the **SINGLEPUSH** command.

For example, use the **DeTransfer** command `\Wz` to force a pause after a **RESET** command:

```
RESET
\W5
```

## Manual Reset Button

A hardware reset is initiated and a triple-push reset by pressing the *DT80*'s manual reset button. To do this, insert a straightened paper clip (or similar object) through the small hole located between the *DT80*'s Host RS-232 port and the USB port ([P125](#))

## Factory Defaults

Both the triple-push reset and the **FACTORYDEFAULTS** command restart the *DT80* using its factory defaults, which are kept in the *DT80*'s Flash memory. (Some of these settings are listed in the Factory Default ([P114](#)) column of the Table 11: *DT80* PROFILE Details ([P115](#).)

A triple-push reset ignores and does not delete any user defaults (USER.INI and ONRESET.DXC), but the **FACTORYDEFAULTS** command does the following:

- deletes USER.INI (working) from internal memory
- deletes USER.INI (backup) from Flash memory
- deletes ONRESET.DXC (working) from internal memory
- deletes ONRESET.DXC (backup) from Flash memory

To return a *DT80* to its totally original, "as shipped" state, send the **FACTORYDEFAULTS** command, reconnect, then send the **FORMAT"B: "** command.

## LEDs and Messages After a Reset

The *DT80* does the following after it is reset ([P125](#)):

	Command /Action	LED Activity	Message Returned
<b>Soft reset</b>	<b>RESET</b>	Acquire LED flashes immediately, then resumes heartbeat flash . Also, if externally powered, Charge LED off momentarily, then on.	RESET Initializing...Done.
<b>Firm resets</b>	<b>SINGLEPUSH</b>	All LEDs flash rapidly four times Then Attention LED on briefly, Acquire LED resumes heartbeat flash and, if externally powered,	SINGLEPUSH dataTaker 80 Version 4.00 Initializing...Done.
	Hardware reset	Charge LED on.	dataTaker 80 Version 4.00 Initializing...Done.
	Power-up reset		dataTaker 80 Version 4.00 Initializing...Done
<b>Hard reset</b>	Triple-push reset	All LEDs flash rapidly four times, then Acquire LED resumes heartbeat flash	dataTaker 80 Version 4.00 Initializing...Done. === SAFE MODE === Now using factory default settings. Reset will enable user settings.
	<b>FORMAT"B: "</b>	No change	FORMAT"B:" Formatting drive B:. Please wait... Format of drive B: successful.

\* Heartbeat flash: see step [3](#).



# TEST Commands DT80

The **TEST...** commands force the *DT80* to autozero itself, check the functionality of its hardware, and return a test report:

<b>TEST</b>	Returns a full test report
<b>TEST<math>n</math></b>	Returns line $n$ of the test report

Test results that are out of range are flagged with a FAIL message.

A typical test report returned when the *DT80* is in free-format mode is shown in the first column of the following table. (When the *DT80* is in fixed-format mode, less-verbose comma-separated results are returned.)

## Test Report (DT80 Health)

TEST report generated at 2005/09/19,11:49:56  
dataTaker 80 Version 5.02.0040 2005/09/15 15:51:36

```
Serial Number:      082005

VEXT                13.7    V
VBAT (6V)           6.88    V    PASS
IBAT                +19    mA    PASS
VSYS                7.26    V    PASS
VLITH (3.6V)        3.65    V    PASS
VDD (3.3V)          3.25    V    PASS
VANA (3.8V)         3.98    V    PASS
VRELAY (4.5V)       4.65    V    PASS

VREF (2.5V)         2504.04 mV    PASS
Ics I               0.21305 mA    PASS
Ics II              2.5688    mA    PASS
Vos diff            -1.9    uV    PASS
Vos 3W              123.8    uV    PASS
Vos shunt           0.9    uV    PASS
Vos +               -0.3    uV    PASS
Vos -               1.0    uV    PASS
Vos *               -29.4    uV    PASS
Vos #               65.0    uV    PASS
Term. factor        1.00486    PASS
Shunt (100R)        100.112 Ohm  PASS
CMRR                106.7    dB    PASS

DT80 health                PASS
```

Table 13: DT80 TEST Report

# Event Log

## Background record-keeping of critical events

To aid in troubleshooting, the *DT80* automatically logs significant events (power failures, temperature extremes, resets, program failures,...) into an **event log**, which is a file named EVENT.LOG in the EVENTS directory of the *DT80* file system.

The event log may help pinpoint the cause of any unexpected readings or failures, and will be used by *dataTaker* engineers if the *DT80* is returned for service.

The size of the event log file is limited to 500 entries. When EVENT.LOG is full, the *DT80* makes a copy of it (overwriting any existing backup), names the copy EVENT.BAK, and creates a new log file. In this way the *DT80* retains

- the most recent 500 events (in EVENT.LOG), and
- the previous 500 events (in EVENT.BAK).

Results of the **TEST** command can automatically be logged to the event log — see P10 ([P109](#)).

## Unloading the Event Log

Send

**UEVTLOG**

to unload the event log to the host computer. This command instructs the *DT80* to return the entire contents of the event log. For example:

```

EVENT      ,2001/03/29,14:32:26.042234,"Event log created."
EVENT      ,2001/03/29,14:32:33.567891,"FORMAT 2.00.000065"
EVENT      ,2001/03/29,14:33:02.418964,"Reset 2.00.000065"
SELF_DIAGNOSTIC,2001/04/19,01:13:46.228619,"datafilemap.cpp 119"
EVENT      ,2001/04/24,19:23:09.624897,"Main power brownout"
fl

```

The format of the information returned from the event log is the same whether the *DT80* is in free-format mode or fixed-format mode.

## Events — the Whole Log

The event log can only be unloaded in its entirety — it's not possible to unload just a defined timespan of the log. Once an unload of the event log has started, it must go to completion. There is no command to quit an unload part way through.

## Clearing the Event Log

Clear the event log by sending

```
CEVTLOG
```

to the *DT80*. This command clears the EVENT.LOG file and initializes it with an "Event log created" entry such as

```
EVENT,2001/04/23,14:32:26.042234,"Event log created"
```

This entry is the first item unloaded in subsequent unloads of the event log.

# STATUS Commands

## STATUS

The STATUS command returns a report showing the status of the *DT80*'s schedules, channels, alarms, memory and logging to the host computer.

The first line of the report shows the version, creation date and creation time of the *DT80*'s firmware. The last line reports the *DT80*'s current switch settings (see Switches [\(P112\)](#)). Send the `/u` switch to make STATUS results less verbose.

STATUS Report	Line (n)
dataTaker 80 Version 4.09.0001 Flash 2001/02/18 21:28:18	1
none,F Scan Schedules Active,Halted	2
0,0 Alarms/IFs Active,Halted	3
0 Polynomials/Spans Defined	4
none,none Scan Schedules LOGON,LOGOFF	5
13650,0 Internal kB free,used	6
169260,0 External kB free,used	7
/B/C/d/E/f/h/I/K/l/M/N/r/S/t/U/w/x/Z	8

## STATUSn

Each STATUS line can be returned individually. STATUS2 and 4 return extra information. There are also other status levels that are not returned by the general STATUS command (STATUS10, 12 and 14).

### STATUS1

Returns version details of the *DT80*'s current operating system. For example

```
dataTaker 80 Version 4.00.0001 Flash 2001/02/18
21:28:18
```

### STATUS2

Returns information about the *DT80*'s active halted schedules. For example

```
A, none Scan Schedules
Active,Halted
```

### STATUS4

Returns the *DT80*'s current defined polynomials and spans. For example

```
2 Polynomials/Spans Defined
Y1=3.54,1.009"Deg C"
S7=0.0,100,0.0,1.0"kPa"
```

### STATUS5

Returns the data logging status (lists schedules logging and not logging).

## STATUS6

Returns the amount of free and used space in the *DT80*'s internal memory (kB).

## STATUS7

Returns the amount of free and used space in an inserted USB memory device (kB).

## STATUS9

Returns the *DT80*'s current switch settings. For example

---

*/B/C/d/E/f/h/I/K/l/M/N/R/S/t/U/w/x/Z*

---

## STATUS10

Returns additional information about the current program in the *DT80*. For example

---

*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>*

---

This comma-separated list provides details about the *DT80*'s program. In order, they are program ID, base year, time resolution, card status and current \$ string, followed by schedule fields that identify individual channels, their format and their units.

If there is no program the following is returned:

---

*5,2000,1,0,"",<\*>,<X>,<A>,<B>,<C>,<D>,<E>,<F>,<G>,<H>,<I>,<J>,<K>,<S>*

---

## STATUS12

Returns the date/time range of logged data — that is, the time and date of the first and last data points stored in the *DT80*'s internal memory and inserted USB memory device. For example

---

*00:11:33 on 05/11/1992,00:13:00 on 19/01/1993 Data Start,End times*

---

## STATUS14

An extended version of STATUS10.

# Part K — Hardware and Power

## Inputs and Outputs

Terminals, ports, connectors and sockets

### DT80 Front Panel



Figure 43 DT80 Front Panel

2 Line Display see [\(P90\)](#)

Directional Keypad see [\(P93\)](#)

USB Stick Interface see [\(P73\)](#)

Sampling Indicator see [\(P93\)](#)

Internal Disk Indicator see [\(P93\)](#)

Attention Indicator see [\(P93\)](#)

Edit or Confirm Key see [\(P93\)](#)

Func or Reject Key see [\(P93\)](#)

### DT80 Wiring Panel

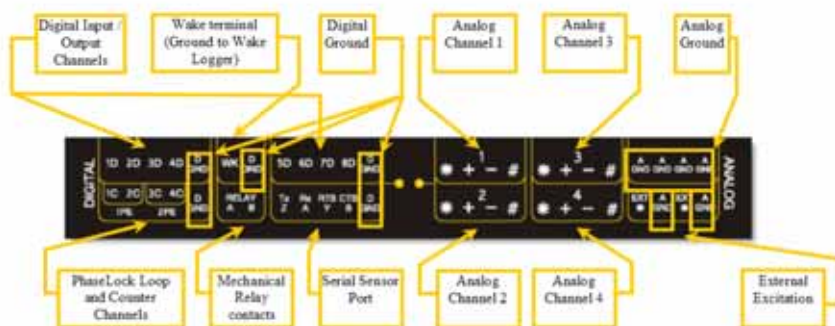


Figure 44 DT80 Wiring Panel

Digital Channels see [\(P141\)](#)

Wake Terminal see LOW-POWER OPERATION [\(P131\)](#)

Digital Ground see DT80 Ground Terminals [\(P141\)](#)

Phase Encoder see [\(P147\)](#)

Counter Channels see High Speed Counter Channels [\(P146\)](#)

Mechanical Relay see Using Digital Outputs [\(P144\)](#)

Serial Sensor Port see [\(P148\)](#)

External Excitation see Sensor Excitation [\(P17\)](#)

Analog Channels see [\(P133\)](#)

## DT80 Side Panel

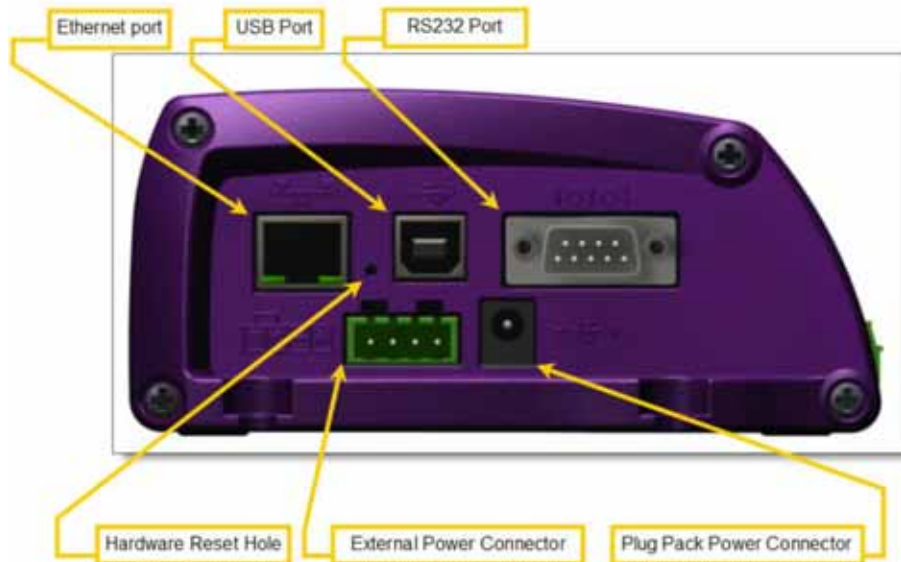


Figure 45 DT80 Side Panel

Ethernet Port see [\(P104\)](#)

USB Port see [\(P95\)](#)

RS232 Port see [\(P96\)](#)

Hardware Reset Hole see Manual Reset Button [\(P120\)](#)

External Power Connector see [\(P130\)](#)

Plug Pack Power Connector see [\(P129\)](#)

# MEMORY

## Storage Capacity

The *DT80* comes equipped with an internal 64Mbyte compact flash card (designated drive B:) and an external customer supplied USB memory device (designated drive A:). The storage format allows around 90,000 readings per megabyte. Therefore the internal compact flash card provides 5 million readings (Note: Time and Date are included as readings in each schedule). Whilst the external will be around 90,000 per Mbyte available. Note each alarm uses 256bytes and consequently reduces data storage accordingly.

See also Data Storage Capacity — Readings/MB [\(P70\)](#).

## USB memory device Commands

### FORMAT "A: "

This command reformats the inserted USB memory device and create the minimum directory and file structure.

**Warning** This command deletes all data from the USB memory device.

# INSIDE THE *DT80*

## Accessing the main battery

1. Remove the power connector



2. Remove the screws from the other end of the logger



3. Remove this end of the logger









4. Pull the purple 'battery tail'





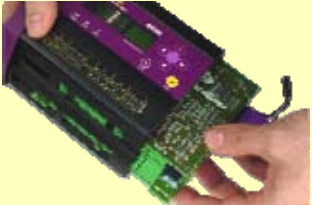

5. Disconnect the battery terminals



## Accessing the lithium memory backup battery

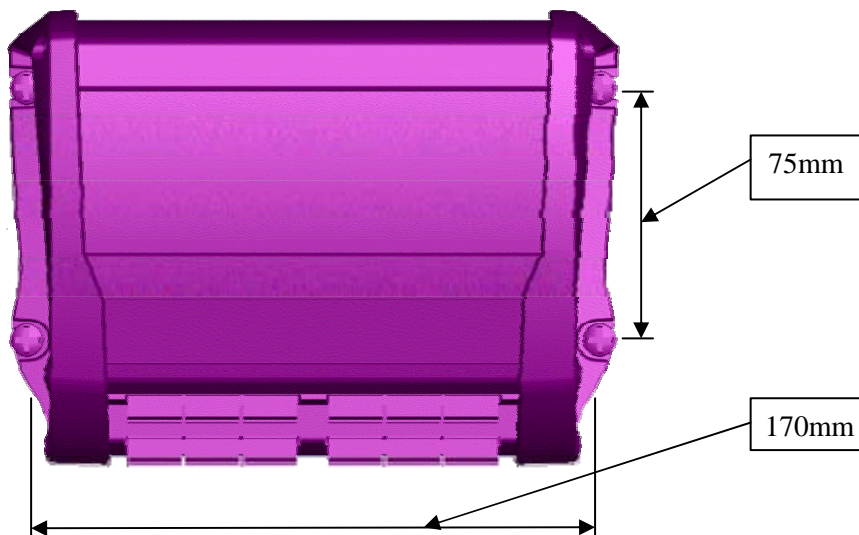
<p>1. Remove the power connector</p>	
<p>2. Remove all the terminal blocks</p>	
<p>3.</p>	
<p>4. Remove the screws from the other end of the logger</p>	
<p>5. Remove this end of the logger</p>	
<p>6. Pull the purple 'battery tail'</p>	



<p>7. Disconnect the battery terminals</p>	
<p>8. Remove battery bracket locking screw (underneath logger) and then remove the battery cage.</p>	
<p>9. Remove the circuit board bundle</p>	
<p>10. Open the boards slightly so the lithium battery can be removed.</p>	

## Mounting the *DT80*

### *Dimensions, Clearances*



# Power

## POWERING THE *DT80*

### Operating Environment

The *DT80* is an electronic instrument. Electronics and water in any form do not mix. Condensation can be a serious problem in the tropics, and in cooler areas where wide temperature variations are possible. Use a sealed case and include sachets of silica gel to avoid problems.

If the *DT80* gets wet, immediately disconnect and remove all power sources (including the main internal battery), and dry the *DT80* in a warm place. If the unit comes into contact with salt water, rinse it thoroughly in fresh water, then in distilled water, then dry it — salt must NOT be allowed to remain on the circuit boards.

The *DT80* operates over a wide temperature range (–45°C to +70°C), but its accuracy can be reduced at extremes. While the electrical zero is stable with temperature, the scale factor can drift slightly. Try to minimize the *DT80*'s exposure to temperature extremes. The lead Acid battery in the logger best operates between -15°C to 20°C. When operating outside this range consideration must be given to an alternative power source for the logger.

Power can be supplied to the *DT80* from internal and external sources:

<b>Internal</b> See Internal Power (Main Battery) <a href="#">(P129)</a> below.	6V lead acid gel cell battery supplied with the <i>dataTaker</i>	
<b>External</b> See Main Battery Life <a href="#">(P129)</a> below.	“Standard” supply <b>11–28Vdc</b> inputs	A DC mains adapter (plug pack) or other “unlimited” DC power source — that is, a source with no special energy conservation or optimization requirements
	“Low-drain” supply <b>Ext Bat</b> terminals (12Vdc)	External batteries, solar panels, vehicle power supplies and other sources for which energy conservation may be critical — that is, applications requiring minimum battery drain by the <i>dataTaker</i> so that the longest possible life is obtained from the power source

### Internal Power (Main Battery)

The *DT80* is fitted with an internal **6V 1.2Ah** sealed lead-acid gel-cell battery. It's known as the *DT80*'s “main” battery to distinguish it from the *DT80*'s other internal battery, the “memory-backup” battery.

The main battery is completely maintenance-free and rechargeable, being automatically charged by the *dataTaker* data logger's inbuilt battery charger whenever an external power supply is connected to either of the *DT80*'s **11–28Vdc** inputs (Figure 45) The battery is fitted with spade terminals.

### Main Battery is Disconnected for Shipping

The *DT80* is shipped with its main battery disconnected. Therefore

- if you intend to use the internal main battery as the data logger's power source, connect the main battery as described in [INSIDE THE \*DT80\* \(P126\)](#) • if an external source is to be used to power the *DT80* there is no need to connect its internal main battery. **HOWEVER...**

**Recommendation** Regardless of how *DT80* is powered, it is recommended internal main battery is connected. By doing this, it guarantees uninterrupted data acquisition and logging because the internal main battery is then always available to continue powering the *dataTaker* data logger if the primary/external supply is accidentally disconnected or fails. The topic [INSIDE THE \*DT80\* \(P126\)](#) explains how to connect the internal main battery.

In addition, the main internal battery is a gel-cell type: if a gel-cell battery remains flat for any length of time, its service life will be significantly reduced.

### Main Battery Life

The life of the *DT80*'s internal main battery depends on

- the scan interval
- the number of analog channels being scanned
- the number of digital channels being scanned
- the number of alarms
- excitation power drawn by sensors
- the complexity of any calculations
- communications activity.

See [Always Trying to Sleep \(P131\)](#) and [Extending Battery Life \(P132\)](#).

---

## External Power

### Solar Charging

The *DT80*'s internal main battery can be charged from a 12V solar panel connected to the *dataTaker* data logger's standard power inputs

The *DT80* provides current and voltage limiting to protect both the panel and battery.

The size of the solar panel required depends on the hours of full sunlight that can be expected. As a general rule, only one day in seven should be regarded as a "charge day", and the charge must be able to fully replenish the batteries on that one day. The solar panel rating is calculated as follows:

$$\text{Panel rating} = \frac{I_w}{T_w \times \eta} \text{ Amps (in full sunlight)}$$

where

---

$I_w$  is the Ah per week consumed by the *dataTaker*

---

$T_w$  is the hours per week of full sunlight

---

$\eta$  is the efficiency; a combination of battery charge absorption and the cosine effect — typical value 0.65

---

Sending **P15=1** ensures that the *DT80* sleeps whenever possible to conserve power.

---

## Internal Memory-Backup Battery

In addition to the internal main battery, the *DT80* contains a small, cylindrical, lithium "memory-backup" battery. See Figure 46: The *DT80*'s memory-backup battery ([P130](#))

The memory-backup battery ensures that

- the data
- the *DT80*'s clock/calendar
- the *DT80*'s primary configuration settings (mains frequency P11, date format P31 and time format P39.

are not lost if power to the *dataTaker* is interrupted. The memory-backup battery can maintain this information for at least 12 months if necessary.

Just like the backup battery in a computer, the *DT80*'s memory-backup battery needs to be replaced every five years or so if the safeguard capability is to be maintained. The topic *INSIDE THE DT80* ([P126](#)) explains how to replace the internal memory-backup battery.

The memory-backup battery is a 1/2AA size 3.6V lithium type (for example, SAFT LS 14250). It's important that 3.6V and not 3.0V types be used (both types are the same physical size).

### Self-Testing the Memory-Backup Battery

Use the *DT80* to test its memory-backup battery by

- sending a **TEST** command — see in TEST Commands *DT80* ([P121](#)) including the **VLITH** channel type in a schedule (or alarm) — see Table 1: *DT80* Channel Types ([P29](#)) table.

### Getting Maximum Life from the Memory-Backup battery

The memory-backup battery is not activated until the main battery is connected to the *dataTaker DT80*. This means that the memory-backup battery is effectively "on-the-shelf" (it has a 10-year shelf life) until the main battery is connected. This is one of the reasons the *dataTaker* data logger is shipped with its main battery disconnected; to prolong the overall life of the memory-backup battery. (The other reason is safety during shipping.)

Once the memory-backup battery is activated, it can be de-activated at a later time, and thereby return it to an on-the-shelf situation of zero current drain, by following the simple procedure described in Internal Memory-Backup Battery During *DT80* Storage ([P131](#)) in the next topic.

---

## Battery Guidelines for Long-Term Storage

To look after your *DT80*'s batteries if it is to be out-of-use for longer than a month or two, do the following:

### Internal Main Battery During *DT80* Storage

Charge the main battery periodically so that it never goes flat. Eight hours charge every three to six months is safe; a battery in good condition need only be charged every twelve months.

The *DT80* can be used to do this by applying power to one of its External Power inputs with the main battery connected inside, or remove the battery from the *dataTaker* data logger (see *INSIDE THE DT80* ([P126](#))) and charge it from a suitable external charger.

## Internal Memory-Backup Battery During DT80 Storage

Remove the memory-backup battery from the *DT80* (described in [INSIDE THE DT80 \(P126\)](#)).

This, of course, returns the memory-backup battery to a zero-drain/shelf-life situation (with an overall shelf life of at least 10 years). Then, as long as the main internal battery is disconnected, the memory-backup battery can be re-fitted in its holder inside the data logger, ready to be automatically reactivated when the main battery is reconnected (refer to [Getting Maximum Life from the Memory-Backup battery \(P130\)](#)).

If an activated memory-backup battery is left in the *DT80*, the memory-backup battery has a life of

- approximately five years if the main battery is present and powering the *DT80*
- approximately one year if no other power is present.

# LOW-POWER OPERATION

## Power

[POWERING THE DT80 \(P129\)](#) discusses the ways to provide power to the *dataTaker* data logger.

For applications where power consumption is critical, the *DT80* has a sleep mode that reduces battery current drain from approximately 400mA (maximum) or 150mA (typical) to just 300µA. The *DT80* automatically wakes from sleep mode when input channels are due to be scanned. Plan your *DT80* program to ensure that the *DT80* does not wake more often than is necessary. This applies particularly to the statistical sub-schedule (Statistical Report Schedules [\(P46\)](#)) and alarms [\(P77\)](#).

When supplied from its internal main 6V battery, the *DT80* has two power-related modes of operation — wake mode and sleep (low-power) mode:

- In wake mode, the *DT80* is fully active and draws 150mA (typical) or 400mA (maximum) from the battery.
- In sleep mode, only the “bare essentials” remain active and current drain is reduced to approximately 350µA.

Once asleep, the *DT80* only wakes, for example, when

- a scheduled scan becomes due
- an immediate scan is sent
- a USB drive is inserted or removed
- the wake terminal is pulled to logic low (see.....)
- communication arrives at the Host RS-232 port (see [Comms Wakes the DT80 \(P100\)](#)).
- communication arrives at the Serial Sensor Port (see [Comms Wakes the DT80 \(P100\)](#)).
- the CT line is asserted on the Serial Sensor Port (see [Comms Wakes the DT80 \(P100\)](#)).

## Exceptions

If the *DT80* is externally-powered or connected to an Ethernet network, it never sleeps. There are also other exceptions. See [No-Sleep Conditions \(P131\)](#) below.

## Always Trying to Sleep

### Sleep Conditions

For maximum life when the *DT80* is powered only by its internal main battery, the *DT80* goes to sleep when ALL of the following conditions exist:

- No external channel measurement is scheduled for the next four seconds<sup>2</sup>.
- No communication has arrived at the *DT80*'s Host RS-232 port within the last 30 seconds<sup>3</sup>.
- The *DT80*'s Host RS-232 port RI (Ring Indicator Pin 9) has not been asserted within the last 30 seconds
- No communication has arrived at the *DT80*'s serial sensor port or CTS changes to active within the last 30 seconds.
- No reset of the *DT80* has occurred within the last 30 seconds
- To override the default operation listed above
- setting P15=2 (see [Controlling Sleep \(P132\)](#) below)
- rapid scanning (see [Extending Battery Life \(P132\)](#) below)
- setting P55 so that specific schedules do not wake the *DT80* (see P55 [\(P111\)](#)).

### No-Sleep Conditions

The *DT80* is designed to not go to sleep when any of the following conditions exist:

<sup>2</sup> Four seconds is the *DT80*'s default — see [P3](#).

<sup>3</sup> 30 seconds is the *DT80*'s default — see [P17](#).

- When the *DT80* is externally-powered. That is, when power is provided to either of the standard power inputs. To override this — see Controlling Sleep ([P132](#)) next. When externally-powered, the *DT80* draws 50mA to 400mA (depends on the state of charge of the internal main battery) from the external supply in addition to the 150mA (typical) or 400mA (maximum) required for normal operation. This occurs even if the *DT80* is forced to sleep by setting **P15=1**.
- When the *DT80*'s Ethernet port is connected to an Ethernet network, or directly to a computer's Ethernet port. (But connecting Ethernet to a sleeping *DT80* does not wake it.)
- When the *DT80* is unloading data.
- When a modem connected to the *DT80*'s Host RS-232 port is in the process of establishing a call or has a call in progress.

## Controlling Sleep

Use parameter 15 (P15 ([P110](#))) to control the *DT80*'s sleep:

P15=	Sleep Entry Condition	
0	Auto-sleep: Sleep when not busy only if powered from internal main battery Never sleep if externally-powered P15=0 is the <i>DT80</i> 's default.	See (P17 ( <a href="#">P110</a> )) and (P55 ( <a href="#">P111</a> )).
1	Force sleep — that is, sleep when not busy regardless of how powered	
2	Force normal operation — that is, keep the <i>DT80</i> awake (never enters sleep mode)	

## Extending Battery Life

Therefore, to extend battery life, do not sample channels more frequently than your data-gathering requires. You can also save power by minimizing comms activity (set the reporting switch to [/r](#)), and using high speed counters.

## Low-Power Programs

### Sleep Program

This framework may be useful when designing low-power programs. Reset the *DT80* before sending this program:

```

BEGIN
P15=1'Sleep if not busy
P17=5'Go to sleep quickly
/u/n'Disable channel ID and units
S1=0,100,0,1.000"%RH"'Define spans, etc. here
RS15M'Scan slowly for statistical schedules
RA1H'Especially for reporting schedules
1V("Humidity",S1,AV)'Define channels
2PT385("Air temp.",4W,AV,=1CV)
IF(1CV>25){LOGON}
IF(1CV<20){LOGOFF}
END

```

# Part L — Sensors and Channels

This part contains physical details (including wiring diagrams) of the sensors and channel types supported by the DT80.

## Analog channels

- 4–20mA Current Loops ([P133](#))
- Frequency ([P134](#))
- Thermocouples ([P134](#))
- Thermistors ([P136](#))
- RTDs ([P137](#))
- IC Temperature Sensors ([P137](#))
- Bridges ([P138](#))
- Humidity Sensors ([P139](#))
- Analog Logic State Inputs ([P140](#))

## Digital Channels

- Digital Inputs ([142](#))
- Digital Outputs ([P144](#))
- High Speed Counters ([P146](#))
- Phase Encoders ([P147](#))

## Serial Channel

- SERIAL CHANNEL ([P148](#))

## Wiring configurations — analog channels

- Voltage Inputs ([P157](#))
- 4–20mA Current Loops ([P158](#))
- Resistance Inputs ([P160](#))
- Bridge Inputs ([P161](#))
- AD590-Series Inputs ([P162](#))
- LM35-Series Inputs ([P163](#))
- LM135-Series Inputs ([P163](#))

## Wiring configurations – digital channels

- Digital Input Wiring configurations ([P164](#))
- Digital output wiring configurations ([P165](#))

# Analog Channels

---

## Analog Sensors and Measurement

### 4–20mA Current Loops

WIRING DIAGRAMS: ([P158](#))

The DT80 supports current loop measurements. The DT80 has an internal 100ohm shunt resistor that can be used with one 4-20mA sensor per DT80 analog channel. External shunt resistors can be used to expand the number of 4-20mA sensors that can be used per DT80 analog channel. — see *Figure 56* ([P159](#)).

The channel type for 4–20mA current loop measurement is

**L([shuntResistance](#))**

where

---

[shuntResistance](#) is the channel factor; the value of the shunt resistor you're using in the loop (default is 100Ω)

---

The DT80 assumes a shunt resistance of 100Ω (the default). If you use a different shunt resistance, you must specify this using the **L** channel type's channel factor (see **I** (P26)) — for example, **3L(50.0)**.

Current-loop measurement is essentially the same as voltage measurement — the DT80 measures the voltage across the internal or external shunt resistor and, knowing the shunt resistance, calculates the loop current.

### Examples — Current Loop Measurement

The schedule

**RA2S 4L**

instructs the DT80 to measure, every 2 seconds (**2S**), the current in the 4–20mA loop sensed by analog channel **4** across a 100Ω shunt resistor (the default).

The schedule

**RA2S**

**4L(50.0)**

instructs the DT80 to measure, every 2 seconds (**2S**), the current in the 4–20mA loop sensed by analog channel **4** across a 50Ω shunt resistor.

### Frequency

WIRING DIAGRAMS: see Voltage Inputs (P157)

The frequency of an analog input signal can be measured using the **F** channel type, which returns a value in Hz.

Useful channel options for **F** channels are:

Channel Option	Description
(channel factor)	sample period (gate time) in ms (default is <b>30ms</b> )
<b>2V</b>	offset input signal by -2.5V. This effectively changes the threshold point from 0V to approx. +2.5V, which is useful for TTL level inputs

The range of frequencies that can be measured depends on the configured sample period (channel factor). For the default setting of 30ms, this range is approximately 25Hz – 20kHz. If the input frequency is too low to be measured, the underrange error value (**-9999.9**) will be returned.

To measure lower frequencies, the sample period should be increased. For example

**3F(1000)**

will measure down to 1Hz (upper limit is still 20kHz), while

**3F(10000)**

will allow frequencies down to 0.1Hz to be resolved.

The drawback to selecting a long sample period is that the measurement will take a long time to complete. This may delay the execution of other schedules.

Note that the default threshold point is 0V, so the input signal must have zero crossings in order to be measured. If this is not the case (eg. for a logic signal), the **2V** channel option can be used to change the threshold point to +2.5V.

### Period Measurement

The period of a signal can be measured by taking the reciprocal of a frequency measurement, eg:

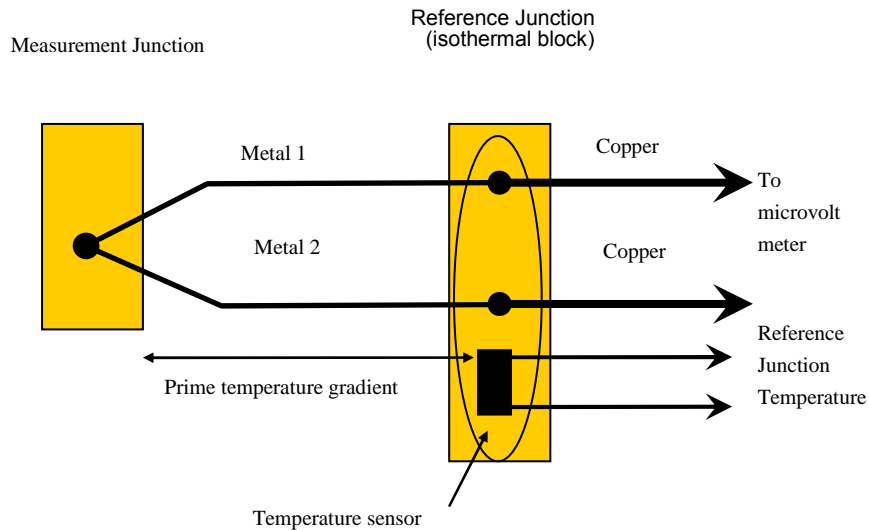
**RA5S 3+F(2V,1000,F1,"Period~s",FF4)**

will return the period, in seconds, of an TTL-level logic signal connected between **3+** and **3#**. Given the 1000ms sample period, the maximum period that can be returned will be approximately 1.0s. The **F1** option applies intrinsic function #1 (1/x).

### Thermocouples

WIRING DIAGRAMS: see Voltage Inputs (P157) Thermocouple Theory





A thermocouple is two wires of dissimilar metals that are

- electrically connected at one end (the measurement junction) and
- thermally connected at the other end (the reference junction).
- A small voltage is produced when the two junctions are at different temperatures. (The voltage is produced by the temperature gradient along the wires, not by the junctions.)

It's important that the purity of the thermocouple wire be maintained where significant temperature gradients occur. Because high purity wire can be expensive, it's common practice to use thermocouple extension wire to cover long distances where temperatures are within the normal environmental range. Such wire can be used for measurement junctions, but only over a restricted temperature range of typically  $-20^{\circ}\text{C}$  to  $120^{\circ}\text{C}$ .

### Making the Measurement Junction

The measurement junction can be made by welding, brazing, soldering or crimping the two wires together. Take care to ensure that the wire material is not contaminated where the temperature gradient is to occur.

The junction can be insulated, or left bare for a more rapid response. If left bare, ensure that the junction does not make intermittent contact with metal objects. This can introduce electrical noise (see Grounded Thermocouples [\(P136\)](#)).

### Using Thermocouples with the DT80

Thermocouples are wired to the DT80 as for any voltage signal — see Figure 55 [\(158\)](#) and Figure 54 [\(158\)](#). The channel type is a **Tt** where **t** is the thermocouple type (**TB, TC, ... TT**).

Using the thermocouple channel type reads the channel as a voltage and automatically applies cold junction compensation and linearization.

### Reference Junction Compensation

Conventionally, the reference junction is held at  $0^{\circ}\text{C}$ , and thermocouple responses are determined with a  $0^{\circ}\text{C}$  reference. This is inconvenient in most situations and so, in practice, the reference junction is allowed to follow to ambient temperature. Then this non-zero reference junction temperature must be compensated for by measuring the reference temperature with another temperature sensor.

The DT80 makes this correction in software. The software approach allows support for any thermocouple type without hardware dependence.

### Isothermal Block

Generally the reference junctions are held at the same temperature by a physical arrangement that ensures good thermal conductivity between the junctions. This structure is called an "isothermal block". It is advisable to insulate the isothermal block from rapid ambient temperature changes.

### Reference Junction Support

By default, the DT80 uses the internal temperature sensor (see Table 1: DT80 Channel Types [\(P29\)](#) REFT channel type) as the reference junction sensor. The internal sensor has an accuracy of  $\pm 0.5^{\circ}\text{C}$ .

### ITS90

In 1990 the definition of the International Temperature Scale changed. The DT80 is calibrated to ITS90.

### Thermocouple Types

The DT80 supports all commonly-recognized thermocouple types:

Type	Positive	Negative	Range $^{\circ}\text{C}$
<b>B</b>	Pt, 30%Rh	Pt, 6%Rh	+300 to 1700

<b>C</b>	W, 5%Re	W, 26% Re	0	to	2320
<b>D</b>	W, 3%Re	W, 25%Re	0	to	2320
<b>E</b>	Ni, 10%Cr	Cu, 45%Ni	-200	to	900
<b>G</b>	W	W, 26% Re	0	to	2320
<b>J</b>	Fe	Cu, 45% Ni	-200	to	750
<b>K</b>	Ni, 10%Cr	Ni, 2%Mn, 2%Al	-200	to	1250
<b>N</b>	Ni, 14%Cr, 1%Si	Ni, 4%Si, 0.1%Mg	-200	to	1350
<b>R</b>	Pt, 13%Rh	Pt	0	to	1450
<b>S</b>	Pt, 10%Rh	Pt	0	to	1450
<b>T</b>	Cu	Cu, 45%Ni	-200	to	350

Each type has characteristics (sensitivity, stability, temperature range, robustness and cost) that make it appropriate for particular applications.

### Grounded Thermocouples

Frequently, thermocouple measurement junctions are electrically connected (by welding, brazing, soldering or by contact) to the object being measured. This is only possible if the object is grounded to the DT80's analog ground terminal **AGND**.

See also Ground Loops ([P22](#)).

### Accuracy — Thermocouple Techniques

The accuracy of temperature measurement with thermocouples depends on

- the reference junction isothermal characteristics
- the reference temperature sensor accuracy
- induced electrical noise
- the quality of the thermocouple wire
- drift in the wire characteristics, especially at high temperatures
- the basic measurement accuracy of the DT80
- the linearization accuracy of the DT80.

The accuracy of temperature measurement with thermocouples is a function of the accuracy of the thermocouple, and not of the DT80.

### Reference Junction Error

The most significant source of error is the reference junction. The DT80 must not be exposed to non-uniform heating because a single reference temperature sensor is used to measure the temperature of the terminals of all channels. If a temperature gradient occurs along the terminals, errors of the magnitude of the temperature difference occur.

The DT80's reference temperature sensor is positioned behind analog channels 2 and 4. Therefore, when precise temperature measurements are required, attach thermocouples here for the least temperature differential from the *dataTaker's* reference temperature.

### Linearization Error

The DT80's linearization errors are much lower (< 0.1°C over the full range) than other error sources.

## Thermistors

WIRING DIAGRAMS: see Resistance Inputs ([P160](#)) Thermistors are devices that change their electrical resistance with temperature. They measure temperatures from -80°C up to 250°C, and are sensitive but highly nonlinear. The DT80 has channel types for many 2-wire YSI<sup>®</sup> thermistors and, for other thermistor types, the DT80 supports thermistor scaling — see Thermistor Scaling (*T<sub>n</sub>*) ([P63](#)).

Channel Type	R (ohms) at 25°C	YSI Thermistor	Max. Temp °C	Min. Temp °C (without Rp)
<b>YS01</b>	100	44001A, 44101A	100	-65
<b>YS02</b>	300	44002A, 44102A		-45
<b>YS03</b>	1000	44003A, 44101A 44035		-20
<b>YS04</b>	2252	44004, 44104 44033 45004, 46004 46033, 46043 44901 44902	150 75 200 90 70	1
<b>YS05</b>	3000	44005, 44105 44030 45005, 46005 46030, 46040	150 75 200	7

<sup>18</sup> Yellow Springs Instruments — YSI Incorporated ([www.ysi.com](http://www.ysi.com))

		44903	90	
		44904	70	
<b>YS07</b>	5000	44007, 44107	150	<b>18</b>
		44034	75	
		45007, 46007	250	
		46034, 46044		
		44905	90	
		44906	70	
<b>YS17</b>	6000	44017	150	<b>22</b>
		45017	250	
		46017	200	
		46037, 46047		
<b>YS16</b>	10k	44016	150	<b>34</b>
		44036	75	
		46036	200	
<b>YS06</b>	10k	44006, 44106	150	<b>35</b>
		44031	75	
		45006	250	
		46006	200	
		46031, 46041		
		44907	90	
		44908	70	

## RTDs

WIRING DIAGRAMS: see Resistance Inputs [\(P160\)](#) Resistance Temperature Detectors are sensors generally made from a pure (or lightly doped) metal whose electrical resistance increases with temperature. Provided that the element is not mechanically stressed and is not contaminated by impurities, the devices are stable, reliable and accurate.

The DT80 supports four RTD types:

Metal	Alpha	Standard
Platinum (PT385)	$\alpha = 0.003850$	DIN43760
Platinum (PT392)	$\alpha = 0.003916$	JIS C1604
Nickel (Ni)	$\alpha = 0.005001$	
Copper (Cu)	$\alpha = 0.00390$	

The alpha is defined by

$$\alpha = \frac{R_{100} - R_0}{100R_0} \quad \Omega / \Omega / ^\circ\text{C}$$

where

$R_0$	is the resistance at 0°C
$R_{100}$	is the resistance at 100°C

The RTD channel types (see [PT385 \(P28\)](#), [PT392 \(P28\)](#), [NI \(P28\)](#) and [CU \(P28\)](#)) are connected as for a resistance. The 0°C resistance is assumed to be 100Ω for platinum, and 1000Ω for nickel types. Other values can be specified as a channel option. The default connection is for a 3-wire measurement, but 4-wire can be specified as a channel option for greater accuracy. For example [PT385\(4W,50.0\)](#)

reads a 4-wire 50Ω (at 0°C) device.

## IC Temperature Sensors

IC (Integrated Circuit) temperature sensors are devices that are constructed on small silicon chips. These are linear, sensitive and available in both voltage and current output configurations. Sometimes called “monolithic” sensors. Their disadvantages are

- limited temperature range; generally –40°C to +150°C (like thermistors)
- self-heating from power dissipation caused by the excitation current needed to read the sensor.

The DT80 supports the following commonly-available IC temperature sensors:

Sensor	Channel Type	Output	WIRING DIAGRAMS
Semiconductor current source types (Analog Devices)	AD590 AD592 TMP17	1μA/K 1μA/K 1μA/K	Figure 70

Semiconductor voltage output types (National Semiconductor Corp.)	LM135 LM235 LM335	10mV/°C 10mV/°C 10mV/K	Figure 70
Semiconductor voltage output types (National Semiconductor Corp., Analog Devices)	LM34 LM35 LM45 LM50 LM60 TMP35 TMP36 TMP37	10mV/°F 10mV/°C 10mV/°C + 500mV 10mV/°C + 500mV 6.25mV/°C + 424mV 10mV/°C 10mV/°C + 500mV 20mV/°C	

For more details, see the Table 1: DT80 Channel Types [\(P29\)](#) table.

## Calibration

IC temperature sensors have different calibration grades. The lowest grades typically have an error of up to  $\pm 2^\circ\text{C}$  at  $25^\circ\text{C}$ . More expensive sensors have an error of  $\pm 0.25^\circ\text{C}$ . This error is a combination of an offset (or zero) error and a slope error.

The DT80 provides a slope (or scale) correction capability on a per sensor basis using the channel factor — see [PT385 \(P28\)](#). Frequently, a slope correction based on a single point calibration point is enough for reasonable accuracy. The pivot point for the slope correction depends on the sensor type.

Sensor	Slope Pivot $T_p$	Channel Factor	Formula
AD590	0.0K ( $-273.15^\circ\text{C}$ )	Series resistor $R$ ( $\Omega$ )	$= R \times C$
LM335	0.0K ( $-273.15^\circ\text{C}$ )	Attenuation factor $A$	$= A \times C$
LM34	$0^\circ\text{F}$ ( $-17.78^\circ\text{C}$ )	Calibration factor	$= C$
LM35	$0^\circ\text{C}$	Calibration factor	$= C$

The calibration factor is calculated from

$$C = 1 - \frac{\Delta T}{T - T_p}$$

where

$\Delta T$	is the temperature error	All temperatures
$T$	is the temperature of the calibration	must be in the same units.
$T_p$	is the pivot temperature	

## Example — AD590

For the AD590 sensor, the channel factor represents the value of the series resistor used to measure the output current (default value is  $100.0\Omega$ ). Without changing the actual resistor, this channel factor can be adjusted as follows.

If the temperature error is determined to be  $1.7^\circ\text{C}$  higher than actual at  $100^\circ\text{C}$ , the channel factor correction is

$$\text{Channel factor} = R \left( 1 - \frac{\Delta T}{T - T_p} \right) = 100 \left( 1 - \frac{1.7}{100 - (-273.15)} \right) = 99.544$$

and is applied as follows:

[5AD590 \(99.544\)](#)

## Bridges

WIRING DIAGRAMS: see Bridge Inputs [\(P161\)](#)

Because of its sensitivity, the Wheatstone bridge circuit is commonly-used for the measurement of small changes in electrical resistance. Applications include load cells, pressure sensors and strain gauges.

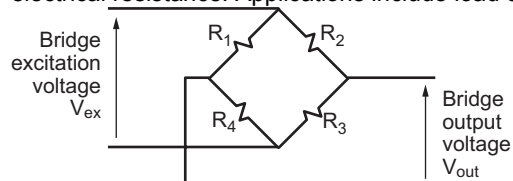


Figure 47: Wheatstone bridge

When one of the four resistors in a bridge is active (that is, sensitive to the quantity being measured) the circuit is called a quarter bridge, and the remaining three resistors are called bridge completion resistors. Similarly, half and full bridges imply two and four active gauges.

### Bridge Excitation (Lead Compensation)

The bridge is a ratiometric circuit where the output sensitivity is proportional to the excitation voltage. Unfortunately, the excitation voltage is reduced by resistive cable and connector voltage drops. There are two ways the DT80 can resolve this problem:

## Voltage Excitation BGV

The DT80 can measure the excitation voltage at the bridge and compensate numerically for the lead voltage loss. This requires a 6-wire connection with the **BGV** channel type (see 6-Wire BGV Inputs [\(P161\)](#)). This is termed **voltage excitation**.

## Constant-Current Excitation BGI

The alternative lead compensation method is to apply a constant-current (default is 2.5mA or 200uA) to the bridge — assuming the bridge resistance is known and constant — and then calculate the excitation voltage  $V_{ex}$ . See Bridge Inputs [\(P162\)](#).

For full and half bridge constant current excitation use the **nBGI (Ra)** channel type, where **Ra** is the bridge arm resistance in ohms. If the arm resistances are not equal, a correction must be applied.

For the full bridge, all four resistors are external to the DT80. One or more of these resistors may be active, and the remainder are completion resistors. Four connection wires are required so that the 4W channel option is required. For example, **nBGI (4W, 120)** defines a 4-wire constant-current bridge with an arm resistance of 120 ohms.

For the half bridge, bridge completion resistors are external to the DT80.

## Scaling

The DT80 scales all bridge channel types to a ratiometric form with units of parts per million (ppm):

$$\text{Reading } B_{out} = \left( \frac{V_{out}}{V_{ex}} \right) 10^6 \text{ ppm}$$

where

$V_{out}$	is measured as a voltage
$V_{ex}$	is measured by a reference channel for voltage excitation, but calculated for constant current excitation

To convert to other engineering units, apply a polynomial, span or use calculations (see Manipulating Data [\(P56\)](#)).

## Strain Gauges

Strain gauges change resistance when stretched or compressed, and are commonly wired in a bridge. The strain-to-resistance relationship is

$$\text{Strain} = \frac{\Delta L}{L} = \frac{1}{G} \cdot \frac{\Delta R}{R}$$

where

$L$	$T$
$\Delta L$	is the length change
$R$	is the initial resistance
$\Delta R$	is the gauge resistance change
$G$	is the gauge factor, a measure of the sensitivity of the gauge (typical foil gauges have a gauge factor of 2.0, which means that if they are stretched by 1% their resistance changes by 2%)

To convert the DT80's ppm bridge readings to strain, use the formula

$$\text{Bridge reading in microStrain} = \left( \frac{4}{G \times N} \right) B_{out}$$

where

$B_{out}$	is the DT80's bridge channel (BGV or BGI) result
$G$	is the gauge factor
$N$	is the number of active gauges in the bridge

The conversion can be done in the DT80 by applying a polynomial as a channel option (see Polynomials (Yn) [\(P62\)](#)):  
**Y1=0,k"uStrain"'Polynomial definition**  
**3BGV(Y1)'Bridge channel**

where

$$k = \frac{4}{G \times N}$$

## Humidity Sensors

Relative humidity is commonly measured by the "wet bulb depression" method. Two temperature sensors are required, one to measure air temperature and the other the cooling effect of a wetted surface. Usually a temperature sensor is encased in

a wick extending into a reservoir of distilled water. The temperature difference between the two sensors is the wet bulb depression.

The choice of temperature sensors is critical if reasonable accuracy is required at high relative humidity where the wet bulb depression is small. If platinum RTDs are used (as in Example — Humidity Measurement [\(P140\)](#) above), they should have good accuracy or matching (0.2°C).

Good accuracy can also be achieved by use of a temperature difference sensor such as a thermocouple or thermopile. Measure the dry bulb with a standard grade temperature sensor and subtract the difference sensor reading to obtain the wet bulb temperature.

The sensors are normally placed within a radiation screen to prevent radiant heat affecting the readings. This is particularly important for outdoor applications.

### Example — Humidity Measurement

The following program reads two RTDs and calculates the relative humidity with an accuracy of a few percent for temperature above 5°C and over most of the relative humidity range (the algorithm assumes that the sensors are ventilated but not aspirated):

```
Y1=6.1,0.44,0.014,2.71E-4,2.73E-6,2.75E-8 'SVP polynomial
BEGIN
  RA5S
  1PT385("Dry bulb",4W,=1CV)
  2PT385("Wet bulb",4W,=2CV)
  3CV(Y1,W)=1CV
  4CV(Y1,W)=2CV
  5CV("RH%",FF1)=(4CV-0.8*(1CV-2CV))/3CV
END
```

### Analog Logic State Inputs

The **nAS** channel type configures analog channel **n** to detect an input voltage relative to a threshold:

- When the input is above the threshold, a 1 is returned.
- When the input is below the threshold, a 0 is returned.

The default threshold is 2500mV, but can be set to any value in mV — see Table 1: DT80 Channel Types [\(P29\)](#) table.

### Example — Analog State Input

The channel list

```
1AS(1750)
```

configures analog channel **1** as an analog state input with a threshold of **1750**mV.

# DT80 Analog Sub-System

Two important sub-sections of the DT80 are completely isolated:

- the DT80 analog section is electrically isolated from the rest of the DT80 (see Figure 49 [\(P141\)](#))
- the DT80 Ethernet interface is transformer-isolated from the outside world.

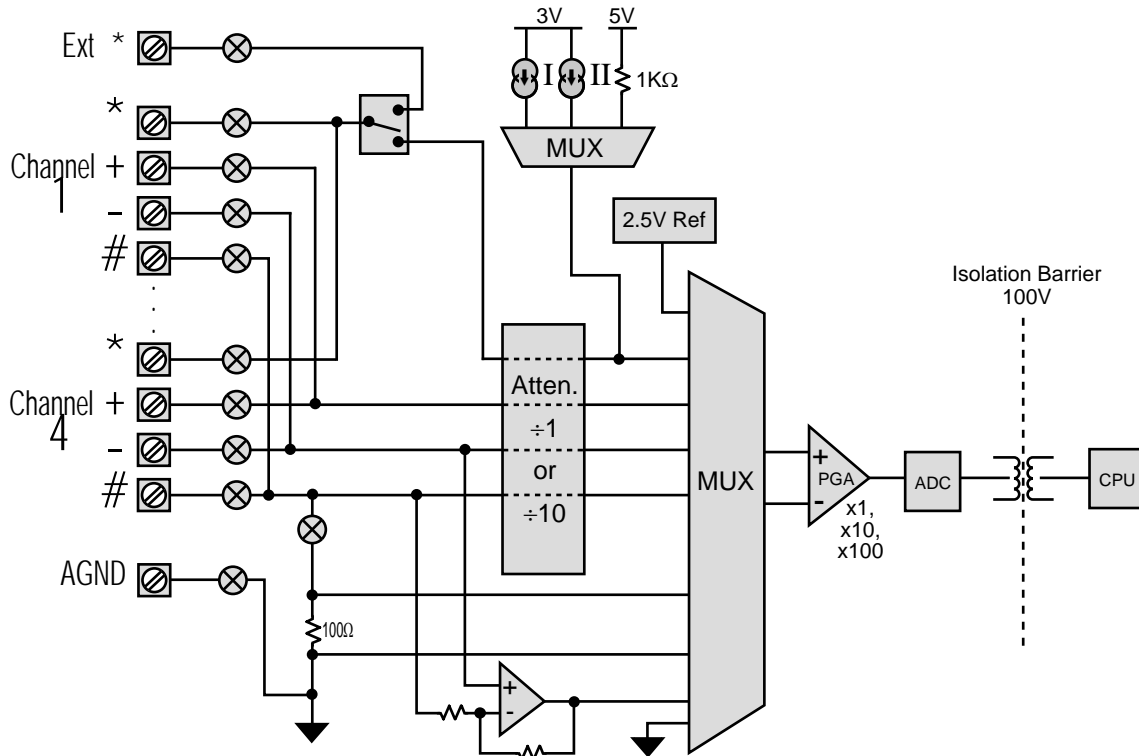


Figure 48: DT80 Analog Sub-System

## Isolated Analog Section

Because the analog section is electrically isolated from the rest of the DT80, sensor-to-equipment ground loops (see *Some of the possible ground-loops in a measurement system*) are unlikely to arise.

## DT80 Ground Terminals

The DT80 has two ground systems — **digital ground** and **analog ground** (Figure 49 [\(P141\)](#)):

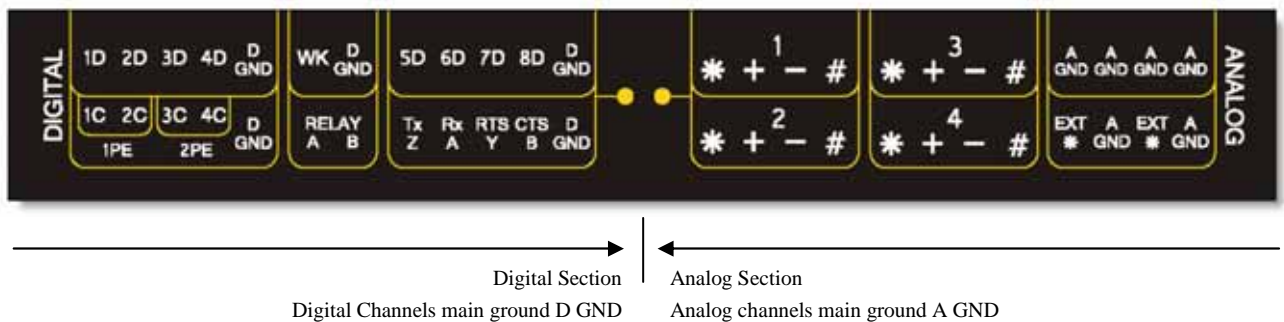


Figure 49: The DT80 has two ground systems

## Analog Ground

The floating analog section of the DT80 has its own ground, “analog ground”, which is brought out to the six AGND terminals. This analog ground is isolated from the DT80’s other ground (the DGND terminals).

# DIGITAL CHANNELS

The DT80 provides:

- 4 bidirectional digital I/O channels (**1D-4D**) with open drain output driver and pull-up resistor;
- 4 bidirectional digital I/O channels (**5D-8D**) with tri-stateable output driver and weak pull-down resistor;
- 1 voltage free latching relay contact output (**RELAY**)



- 1 LED output (**Attn**)
- 4 hardware counter inputs (**1C-4C**) which can be used as independent counter channels or as two quadrature (phase encoder) inputs. Channels 1C and 2C are low threshold capable.

## Bidirectional Digital I/O Channels

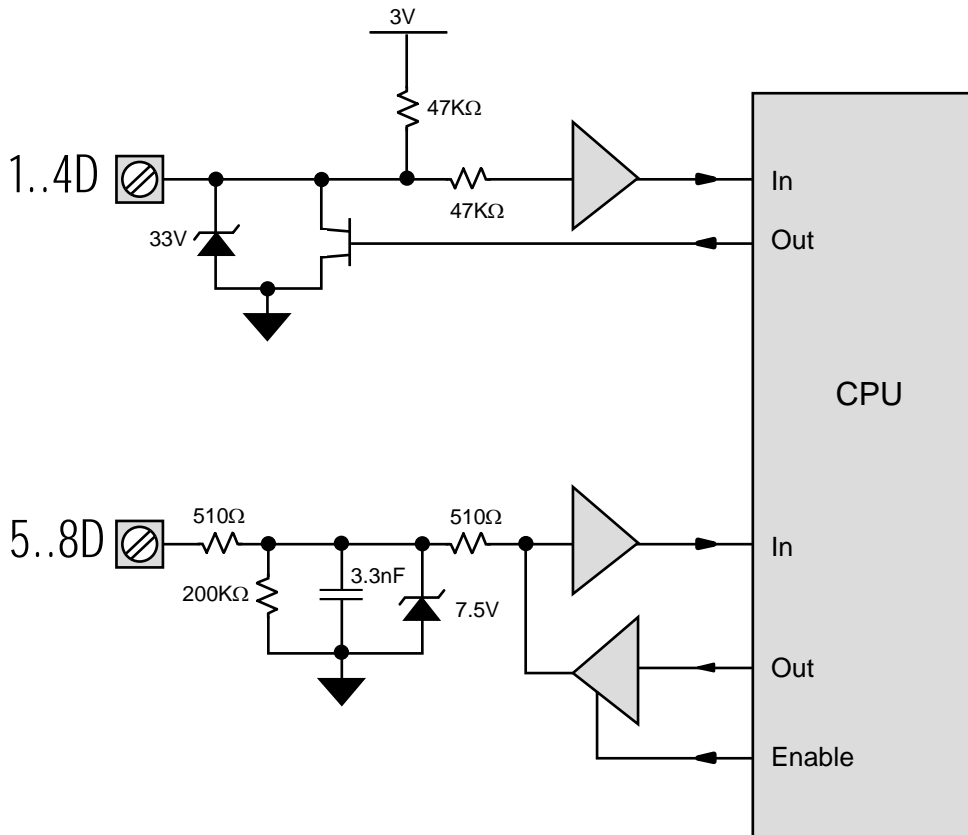


Figure 50 Digital Circuit

Figure 50 shows a simplified circuit diagram for the DT80's eight digital I/O channels. As can be seen, the channels can be divided into two groups, **1D-4D** and **5D-8D**. While these two groups have different hardware characteristics (discussed below), all eight channels are accessed and used in much the same way.

Each of the eight digital channels can be used as *either*:

- a digital input (for monitoring the state of a relay or logic signal), or
- a digital output (for driving a relay or other control device)

**Warning** Beware of conflicts when using the DT80's bi-directional digital channels (**1D** to **8D**). For example, if a device such as a PLC is actively driving one of these channels and you program the DT80 to also drive the same channel as an output (for example, **1DSO=0**), then a conflict exists. This has the potential to damage the digital channel or the driving source. We recommend placing a series resistor between the digital channel and the signal source to limit the current that can be driven into the channel. When choosing the resistor's value and power rating, be sure to consider the source's output voltage, drive current and operating frequency.

## Using Digital Inputs

### Channel Types

Digital inputs can be monitored using the following channel types. In each case *n* represents the channel number (1-8).

<b>nDS</b>	<b>Digital State:</b> returns the state of digital input <i>nD</i> ; 0=low, 1=high
<b>nDN</b>	<b>Digital Nybble:</b> returns the state of four consecutive digital inputs starting at <i>nD</i> as a 4-bit number (0-15). For example, if channel <b>3DN</b> returns the value 13 (binary 1101) then this indicates that input <b>3D</b> is high, <b>4D</b> is low, <b>5D</b> is high and <b>6D</b> is high.
<b>nDB</b>	<b>Digital Byte:</b> returns the state of all eight digital inputs as an 8-bit number (0-255). For this channel type, <i>n</i> must always be 1.
<b>nC</b>	<b>Counter:</b> returns the number of positive-going edges seen on digital input <i>nD</i> . Counter value is a signed 32-bit integer. These counters are low speed polled counters.

### Channel Options

The following channel options are applicable to digital input channel types:

Type	Option	Description
------	--------	-------------

<b>DS</b>	none	
<b>DN, DB</b>	(channel factor)	<b>Bitmask:</b> This specifies which input channels to read. For example <b>2DN ( 7 )</b> (bitmask = 0111 binary) will return the state of inputs <b>2D</b> , <b>3D</b> and <b>4D</b> in bits 0 (lsb), 1 and 2 respectively. For channel <b>5D</b> the mask bit is zero so it is not read and bit 3 of the returned value will always be zero ( <b>5D</b> can then be used as an output if desired). The default values for <b>DN</b> and <b>DB</b> are 15 and 255 respectively (ie. read all bits)
<b>C</b>	(channel factor)	<b>Wrap Value:</b> Counter will reset to 0 (or “wrap around”) when this value is reached. For example, if 8 pulses are received on input <b>4D</b> then channel <b>4C ( 3 )</b> will count in the sequence 1, 2, 0, 1, 2, 0, 1, 2 so after 8 pulses the value 2 will be returned. Default value is 0 (do not reset)
<b>C</b>	<b>R</b>	<b>Reset:</b> counter is cleared to 0 after returning its current value.

## Connecting to Digital Inputs

**Warning** The DT80’s digital inputs are NOT reverse-polarity-protected. Therefore ensure signal polarity is correct — positive to numbered terminals, negative to **DGND** terminals — before connecting signals to the DT80’s digital inputs.

**Warning** Do not apply more than 30Vdc to inputs **1D-4D**, and do not apply more than 20Vdc to inputs **5D-8D**.

Digital input channels **1D-4D** and **5D-8D** have different electrical characteristics. In particular:

- Inputs **1D-4D** include a 47k pull-up resistor. The default state (if nothing is connected) is therefore HIGH. This in turn means that channels **1 . . 4DS** will return 1 if the inputs are not connected.
- Inputs **5D-8D** include a 200k pull-down resistor. Their default state is therefore LOW (0). So if all 8 inputs are disconnected then **1DB** will return 15 (00001111).

Voltage-free relay contact closures can easily be detected on channels **1D-4D** by wiring the relay contacts between the input pin and **DGND**. **nDS=0** indicates contacts closed, **nDS=1** indicates contacts open.

Channels **5D-8D** are less suitable for relay contact inputs, but they can still be used, for example if the contacts are wired between the input pin and an external 3-20V dc supply.

Actively driven logic signals can be directly connected to input channels **1D-8D**, subject to the input voltage level specifications detailed below:

<b>1D-4D</b>	Maximum continuous terminal voltage	30Vdc
	Minimum continuous terminal voltage	-0.6Vdc
	<b>Note</b> Voltages outside this range can permanently damage the channel.	
	Minimum input high voltage (Polled input)	3.0V
	Maximum input low voltage (Polled input)	0.75V
<b>5D-8D</b>	Maximum continuous terminal voltage	30Vdc
	Minimum continuous terminal voltage	-0.6Vdc
	<b>Note</b> Voltages outside this range can permanently damage the channel.	
	Minimum input high voltage (Polled input)	3.0V
	Maximum input low voltage (Polled input)	0.75V

# See **WIRING CONFIGURATION — DIGITAL CHANNELS** [\(P164\)](#) for sample digital input wiring configuration diagrams.

**Important** Although the digital state outputs incorporate transient protection for inductive loads, we recommend that you place a reversed diode across such loads. The output drivers are not current-limited, so avoid shorting a supply line directly to a digital state output.

## Other Considerations

The digital input channels **1D-8D** are scanned at 17ms intervals (60Hz). This means that:

- the minimum input pulse width is **17ms** – shorter pulses may not be recognised.
- the maximum input count frequency, assuming a 50% duty cycle, is **30Hz**.

Use the high-speed counter channels (High Speed Counter Channels [\(P146\)](#)) for higher count frequencies.

Digital inputs are *not* scanned while the DT80 is asleep. Use the high-speed counter channels (High Speed Counter Channels [\(P146\)](#)) if you need the logger to continue to count pulses even while asleep.

Digital input transitions can be used to trigger or enable a report schedule. (see Trigger on External Event [\(P43\)](#)) for more details.

Counter channels can also be configured to trigger a schedule when the wrap value is reached. (see Trigger on External Event [\(P43\)](#)).

A high to low (pull down) digital input transition can be used to wake the DT80 by connecting the digital input in parallel with the **WK** (wake) terminal. The DT80 can then be programmed so that each time an external pulse occurs the DT80 will wake, run an event triggered schedule (see Trigger on External Event [\(P43\)](#)), then go back to sleep.

The count value for a digital input channel can be preset using an expression, eg.

**RA1M 8C=1000 RB2S 8C**

If a 1Hz signal is now applied to input 8D you would expect the values returned every 2s for channel **8C** to follow a sequence similar to:

1000, 1002, 1004 ... 1056, 1058, 1000, 1002 ...

Note that a counter's wrap value (channel factor) is applied when the channel is *defined* (ie. when the job is entered), not when it is *evaluated*. Also, setting the wrap value has the side effect of resetting the count value to zero. This implies that:

- a particular counter's wrap value need only be specified *once* in the job. It does not need to be specified every time the counter is evaluated.
- If querying a counter using the immediate schedule (eg. by periodically typing "**1C**"), do not specify a wrap value each time. Each time you evaluate an immediate channel you are also defining it, so the counter value will always be returned as zero if you specify a wrap value each time.

## Using Digital Outputs

### Channel Types

Digital outputs can be used to control external devices using the following channel types. In each case **n** represents the channel number (1-8); **x** can be either a number, CV or expression.

<b>nDSO=x</b>	Digital State Output: sets the state of digital output <b>nD</b> ; 0=low, 1=high. For example 2DSO=0 sets output 2D low.
<b>nDNO=x</b>	Digital Nybble Output: simultaneously sets the state of four consecutive digital outputs starting at <b>nD</b> . For example, 5DNO=5 (binary 0101) sets 5D high, 6D low, 7D high and 8D low.
<b>nDBO=x</b>	Digital Byte Output: simultaneously sets the state of all eight digital outputs as an 8-bit number (0-255). For this channel type, <b>n</b> must always be 1.
<b>1RELAY=x</b>	Relay Output: sets the state of the latching RELAY output: 0=closed, 1=open
<b>1WARN=x</b>	LED output: sets the state of the Attn LED: 0=off, 1=on

### Channel Options

The following channel options are applicable to digital output channel types:

Type	Option	Description
<b>DSO, RELAY, WARN</b>	(channel factor)	<b>Delay (ms)</b> : The DT80 waits for the specified number of milliseconds after setting the output state. Default is 0, ie. no delay. If the <b>R</b> option is specified then the default and minimum delay setting is 10ms.
<b>DNO, DBO</b>	(channel factor)	<b>Bitmask</b> : This specifies which output channels to set. For example <b>1DNO(14)=1CV*2</b> (bitmask = 1110 binary) will output bits 0 (lsb), 1 and 2 of 1CV on outputs <b>2D, 3D</b> and <b>4D</b> respectively. For digital channel <b>1D</b> the mask bit is 0 so its state will not be affected by this command. The default values for <b>DNO</b> and <b>DBO</b> are 15 and 255 respectively (set all bits)
<b>DSO, DNO, DBO, RELAY, WARN</b>	<b>R</b>	<b>Reset</b> : After setting the output bit(s) to the specified state(s) and waiting for the delay time the output(s) will be set to the opposite state. In other words a pulse will be generated.

### Digital Output Operation

All digital output channels are initialised to their default states on initial power-up, firm reset (**SINGLEPUSH**) or soft reset (**RESET**). Entering a new job does *not* initialise the digital outputs.

The default states are summarised below:

Channel	Default state	Comments
<b>1..4DSO</b>	1	output pulled up (high), controlled load OFF

<b>5..8DSO</b>	0	output driver disabled, pulled down (low)
<b>1RELAY</b>	0	contacts open
<b>1WARN</b>	0	LED off

A digital output command, eg. **1DSO(20,R)=1** is processed as follows:

1. First, the output (or outputs for DNO/DBO) is set to the specified state; if no state is specified then nothing is done.
5. Then the DT80 waits for the specified delay, if any. If a state was specified and the **R** option was also specified then the default delay is 10ms, otherwise 0ms.
6. Then, if **R** is specified, the output(s) is/are inverted.
7. Finally, the output value as at Step 2 is returned.

The current state of any digital output is thus returned when a digital output command is evaluated. For example, typing **2DSO** will return the state to which the output was last set. This will not necessarily reflect the actual state of the **2D** terminal (use **2DS** to read the actual state). And if **2DSO(R)** is entered then the state of **2D** will be inverted and the original state will be returned.

## Connecting to Digital Outputs

As noted above, digital channels **1D-4D** and **5D-8D** have different electrical characteristics. In particular:

- Outputs **1D-4D** use an open-drain FET output driver. This can sink up to 100mA @ 30Vdc so it can drive a low voltage actuator or relay or LED directly. See (wiring diags). A 47k pullup resistor (to +3.3V) is also included, allowing logic devices to be driven.
- Outputs **5D-8D** are *not* suitable for directly driving loads such as relays or LEDs. Logic devices can however be driven. Note that each of these output drivers is tri-stateable.

When outputs **1D-4D** are used to directly drive loads, the load will be ON when the output is in the LOW state. Thus if a load was wired up to output **1D** you would use **1DSO=0** to turn the load ON and **1DSO=1** to turn it OFF.

The default state of the output drivers on channels **5D-8D** is *disabled* (tri-stated). This allows these channels to be used as inputs.

When a digital output command for channels **5D-8D** is evaluated, the output state is set to the required value, then the output driver is enabled. The output will then stay enabled until an input command is evaluated for that channel.

For example,

**6DSO=0 DELAY=10 6DSO(R,5)=1 DELAY=10 6DS(W)**

will drive logic 0 on output **6D** for 10ms, then logic 1 for 5ms, then logic 0 for 10ms, then the output driver will be disabled.

The **RELAY** terminals are voltage-free, normally-open, latching relay contacts. Table 1: DT80 Channel Types ([P29](#)) for voltage and current ratings. Use **1RELAY=1** to close the contacts, **1RELAY=0** to open.

# See WIRING CONFIGURATION — DIGITAL CHANNELS [\(P164\)](#) for sample digital output wiring configuration diagrams.

## Other Considerations

The states of all digital outputs are maintained while the DT80 is asleep. Note also that the **RELAY** output uses a latching relay, so no extra current is required to hold it in the closed state.

One or two digital outputs can be configured to follow the state of an alarm. That is, when the alarm is inactive the output(s) are in their default state (1 for **1..4DSO**, 0 for **5..8DSO**, **1RELAY** and **1WARN**) and when the alarm is active the output(s) will be in their non-default state. Alarm Digital Action Channels ([P80](#)) for more information.

The actual pulse width generated by the Delay option for **DSO** will not necessarily be exactly as specified. For delays of 20ms or less it will be close (within 1ms). For longer delays the resolution is +/- 16ms however it is guaranteed that the duration will be *at least* the specified time.

Note also that, like the **DELAY=** channel type (Table 1: DT80 Channel Types ([P29](#))), high values for the DSO Delay option are not recommended as they can prevent the timely evaluation of other schedules.

The Attention LED may also be flashed by the DT80 to indicate an internal fault or warning condition (Attn Indicator ([P93](#))). This will override the state set using the **1WARN** output channel.

# High Speed Counter Channels

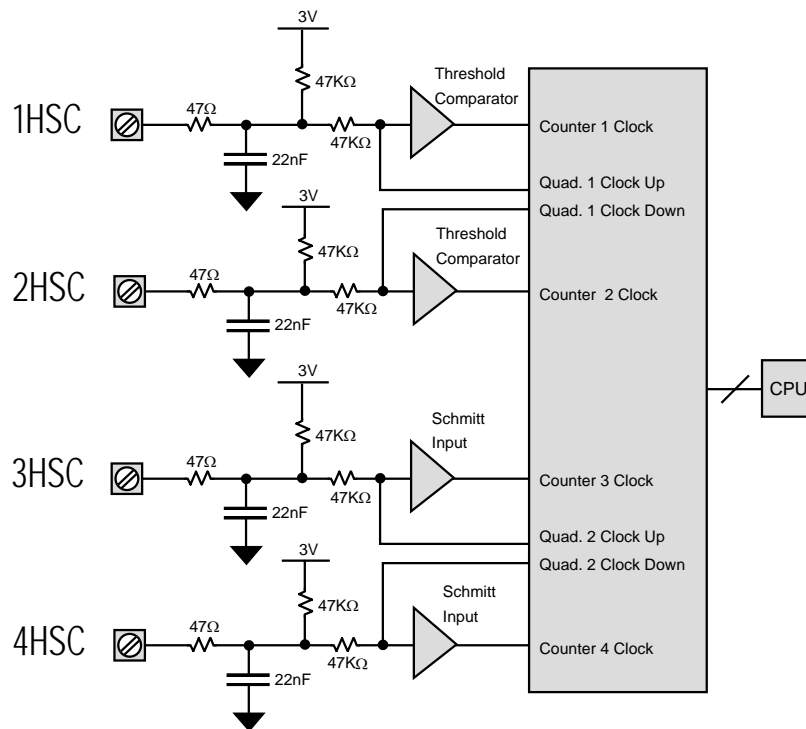


Figure 51 High Speed Counter Channels

Fig x shows a simplified circuit diagram for the DT80's four hardware counter inputs. As can be seen, the channels can be divided into two pairs of inputs, **1C-2C** and **3C-4C**. Each pair can be used as *either*:

- two independent counter inputs, for pulse counting, or
- a single phase encoder (quadrature) input, for use with position sensors that provide phase encoded outputs ("A" and "B")

## Using Counter Inputs

### Channel Types

Counter inputs can be monitored using the following channel types. In each case *n* represents the channel number (1-4 for **HSC**, 1-2 for **PE**).

<b>nHSC</b>	<b>High Speed Counter:</b> returns the number of positive transitions seen on counter input <b>nC</b> . The counter value is a signed 32-bit integer.
<b>nPE</b>	<b>Phase Encoder:</b> returns the current relative position of the phase encoder device connected to input pair <b>nPE</b> . The value returned is in counts and is a signed 32-bit integer, which may be positive or negative depending on the direction of travel.

### Channel Options

The following channel options are applicable to high speed counter input channel types:

Type	Option	Description
<b>HSC, PE</b>	(channel factor)	<b>Wrap Value:</b> Counter will reset to 0 when this value is reached. For example, if 8 pulses are received on input <b>3C</b> then channel <b>4HSC (3)</b> will count in the sequence 1, 2, 0, 1, 2, 0, 1, 2 so after 8 pulses the value 2 will be returned. Default value is 0 (do not reset)
<b>HSC, PE</b>	<b>R</b>	<b>Reset:</b> counter is cleared to 0 after returning its current value.
<b>1..2HSC, 1PE</b>	<b>LT</b>	<b>Low Threshold:</b> select low-level input thresholds (low threshold 2mV and high threshold 7mV). Not applicable to inputs <b>3C-4C</b> .

### Connecting to Counter Inputs

**Warning** The DT80's counter inputs are NOT reverse-polarity-protected. Therefore ensure signal polarity is correct — positive to numbered terminals, negative to **DGND** terminals — before connecting signals to the DT80's counter inputs.

**Warning** Do not apply more than 30V to inputs **1C-4C**.

Counter input channels **1C-2C** and **3C-4C** have different electrical characteristics. In particular:

- Inputs **1C-2C** include selectable TTL or low-level input thresholds. Low thresholds (selected by using the **LT** channel option) allow direct connection to sensors whose output is only a few mV, eg. inductive-pickup flow sensors.
- Inputs **3C-4C** use a standard TTL level Schmitt trigger input.

Voltage-free relay or switch contact closures can be counted on channels **1C-4C** by wiring the relay contacts between the input terminal and **DGND**.

All inputs include low-pass filtering to assist in “debouncing” mechanical switch or relay inputs. For voltage-free contact inputs this limits the maximum count rate to approximately 500Hz. For actively driven inputs, however, the maximum count rate is approximately 100kHz.

## Phase Encoders

A phase encoder is a device for measuring relative angular or linear position. As it moves, it outputs two streams of pulses (“A” and “B”) whose phase relationship (A leading or B leading) indicates the direction of travel.

The DT80’s **PE** channel type decodes these pulses and returns a signed position value in counts.

Note that the “mode” of a counter channel pair (ie. whether it operates as two counters or a single phase encoder channel) is set when the channel is *defined* (ie. when the job is entered), not when it is *evaluated*. This implies that a particular counter input pair *cannot* be read as a phase encoder value at one point in a job, and as a pair of counters at another. In other words, if your job defines a channel **1PE** then it should *not* also define channels **1HSC** or **2HSC**, and vice versa.

## Other Considerations

The high-speed counter inputs continue to function while the DT80 is asleep.

However, it is important to note that each hardware counter is *16 bits* wide. (Count values are maintained and returned as 32-bit values, but the physical hardware counters attached to inputs **1C-4C** are 16-bit.) If more than 65536 pulses occur while the DT80 is sleeping then the hardware counter will overflow, and this will cause an inaccurate count value to be returned when the DT80 wakes.

It is therefore necessary to ensure that the DT80 is programmed to wake often enough to ensure that the hardware counters can be read before they overflow.

For example, if the average counter input frequency is 100Hz then the DT80 must be programmed to wake at least every 65536/100 seconds (about every 10 minutes). This can be done by including a 10-minute schedule (eg **RA10M**) in the job.

Most of the other comments made above regarding digital input counter channels apply equally to the high speed counter channels. For example, HSC channels can be preset to a particular starting value (eg **2HSC=1CV\*10**), HSC channels can trigger a schedule when their wrap value is exceeded, and so on.

## Examples

### Pulse Train Output

The schedule command

```
RA2S 6DS0(500,R)=1
```

produces a pulse train from channel **6D** which is HIGH for 0.5s and LOW for 1.5s.

### Sensor Power Control

In the schedule command

```
RA20M D T 4DSO(1000)=0 1..4V 4DSO=1
```

digital state output 4 controls a relay that switches the power supply to a group of sensors. Every 20 minutes the sensors are powered up, the system waits one second while the sensors settle, the sensors are scanned, and the sensor power supply is turned off again.

### Manual Control

The polled schedule (see Trigger on Schedule-Specific Poll Command [\(P44\)](#)) can also be used to switch digital state output channels. For example, the command

```
RBX 3DSO(5500,R)=0
```

turns a load connected to channel **3D** ON for 5.5 seconds when an **XB** poll command is received.

### Frequency Measurement

The **R** channel option can be used to measure the frequency of an input signal, eg.

```
RA1S 1HSC(R,RS)
```

will return the frequency in Hz of an input signal on channel **1C**, while

```
RA10S 1HSC(R,RS)
```

will do the same thing but resolve down to 0.1Hz.

This technique can also be used for the digital input channels (**1D-8D**), eg.

```
RA1S 7C(R,RS)
```

will return the frequency in Hz of an input signal on channel **7D**, in the range 1-30Hz.

# SERIAL CHANNEL

The DT80's Serial Channel (see Figure 52 (P148)) can be used to connect to serial input and/or output devices such as a serial sensor, GPS terminal, printer, barcode reader, display panel, PLC, or even to another *dataTaker*.

The Serial Channel

- can transmit programmable 'prompt' or 'poll' messages to serial devices and interpret their replies
- can respond to asynchronous incoming serial messages. Incoming data can wake the logger from sleep mode.
- can be configured for either the RS-232, RS-422 or RS-485 comms standard (RS-232 supports a single point-to-point connection; the other standards support multiple devices in a multi-drop configuration)
- has a differential transmitter and receiver that provide for the different serial standards
- has RTS/CTS handshake lines (RS232 only)
- supports baud rates of 50 to 115200 baud

## Connecting to the Serial Channel

The DT80 serial channel terminals have different functions depending upon the configured serial standard (RS232, RS422 or RS485).

Terminal	RS232	RS422	RS485	Wake
<b>Tx Z</b>	Transmit Data	Transmit Data- (A)	Data- (A)	
<b>Rx A</b>	Receive Data	Receive Data+ (B)		<input checked="" type="checkbox"/>
<b>RTS Y</b>	Handshake output	Transmit Data+ (B)	Data+ (B)	
<b>CTS B</b>	Handshake input	Receive Data- (A)		<input checked="" type="checkbox"/>
<b>D GND</b>	Signal Ground	Ground	Ground	

Figure 52: The DT80's Serial Channel terminals (DTE)

Note that:

- The RTS and CTS handshake/control signals are available for RS232 only
- The DGND terminal is the signal return (common) for RS232. RS422/485 use differential signalling – the ground is only used for connection to the cable shield.
- Activity on either of the indicated terminals (ie. **Rx/A** and **CTS/B**) will wake the logger from sleep mode, although the data in the particular message that woke the logger will be lost. Note also that if the Wake feature is required and RS485 is being used then it will be necessary to link the Data terminals (**Tx/Z** and **RTS/Y**) to the wake-enabled terminals (**Rx/A** and **CTS/B**).

## Setting Serial Channel Parameters

The Serial Channel communications parameters are set by the command

**PS=baud,parity,databits,stopbits**

where

Parameter	Settings	Default
<b>baud</b>	is the baud rate at which you want the Serial Channel to operate. Use <b>50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600</b> or <b>115200</b> .	<b>1200</b>
<b>parity</b>	can be <b>N</b> (none), <b>O</b> (odd) or <b>E</b> (even)	<b>N</b>
<b>databits</b>	can be <b>7</b> or <b>8</b>	<b>8</b>
<b>stopbits</b>	can be <b>1</b> or <b>2</b>	<b>1</b>

For example, the command

**PS=9600,N,8,1**

sets the Serial Channel comms quantifiers to 9600 baud, no parity, 8 data bits.

These communications settings are stored in the DT80's SRAM and are therefore retained through a soft reset (**RESET**). After a hard reset (**SINGLEPUSH**), the settings are reset to their defaults (**PS=1200,N,8,1**)



# Serial Channel Commands

## Channel Type

Data flow into and out of the Serial Channel is controlled by the Serial Channel commands. These commands provide for

- formatting and management of output strings and prompts to be sent to the connected serial device.
- interpretation and parsing of input strings received from the connected device into *dataTaker* variables
- general management of the Serial Channel

The general form of a Serial Channel command is

```
nSERIAL("control_string",options)
```

where

<i>n</i>	is the Serial Channel number. For a DT80, this is always <b>1</b> .
<i>control_string</i>	is a string of commands that specify the required output and input actions of the Serial Channel. See The Control String ( <a href="#">P150</a> ).
<i>options</i>	is a list of channel options that can be used to modify general features of the Serial Channel. The options can be placed either before or after the " <i>control_string</i> " in the Serial Channel command.

Note that **SERIAL** is actually a channel type, in the same way that **V** (voltage) is a channel type. It can appear in schedules and it has channel options, like any other channel type. The *control\_string* is a special channel option which applies only to the **SERIAL** channel type.

## Channel Options

As well as *control\_string*, the following channel options are specific to the serial channel

Option	Description
(channel factor)	Maximum time to wait for serial data to be received. Default is 10s. This value is a floating point number, so a value of 0.1 will set the timeout to 100ms.
<b>RS232</b> or <b>RS422</b> or <b>RS485</b>	Specifies the communication standard. Default is <b>RS232</b> . These options are mutually exclusive

- All serial channel commands within a job must select the same communications standard (**RS232**, **RS422** or **RS485**)
- If the standard "*UserName~UserUnits*" channel option (Table 3: DT80 Channel Options ([P38](#))) is specified, it must come after the control string in the list of serial channel options.

## DeTransfer — Use a Double Backslash

**Important** The DeTransfer program, which is often used to supervise the DT80, has a number of commands that have a leading \ (backslash) character (see the DeTransfer Help). The backslash character indicates to DeTransfer that the following command is to be interpreted and executed by DeTransfer. For example:

- \w3 instructs DeTransfer to wait 3 seconds before communicating again with the *dataTaker*.
- \t instructs DeTransfer to read the computer clock and format a current time string.
- \013 instructs DeTransfer to send the single character carriage return to the connected *dataTaker*.

The DeTransfer backslash commands are not sent to the connected *dataTaker*.

However, DeTransfer has an escape mechanism to do this: if a command is preceeded by a double backslash (that is, \\**command**), the command is sent to the connected DT80 as a single backslash command (that is, \b**command**) and is not interpreted and executed by DeTransfer.

Coincidentally, a number of the Serial Channel commands also have a leading \ (backslash) character and, if entered into DeTransfer as such, are not sent to the connected DT80. Therefore, the Serial Channel commands that have a leading \ (backslash) character must be entered into the Send window of DeTransfer with a double backslash. For example:

- To send the Serial Channel command \e, enter \\e into DeTransfer.
- To send the Serial Channel command \r1, enter \\r1 into DeTransfer.
- To send the Serial Channel command \w[1000], enter \\w[1000] into DeTransfer.
- To send the Serial Channel command \013, enter \\013 into DeTransfer.

Note that this rule only applies to DeTransfer. Although other terminal programs can be used to supervise the DT80, most do not interpret the \ (backslash) character, and so the Serial Channel backslash commands must be entered as single backslash commands. Consult the terminal program's documentation for details.

---

# Serial Channel Operation

## The Control String

The "**control\_string**" is always enclosed by quotation marks. It can be broken into two parts:

- **Output actions** — commands, prompts or text strings that are to be sent from the DT80 to the device connected to the serial channel. The various output actions available are detailed in the section. All output actions are enclosed by `{}`.
- **Input actions** — commands to manage the DT80's Serial Channel and to interpret the information coming back from the serial device into the Serial Channel. The various input actions available are detailed in the section Control String – Input Actions ([P152](#)). Input actions are not enclosed by `{}`.

The general form of the "**control\_string**" is

- any combination of output actions enclosed by `{}`, and/or
- any combination of input actions.

There may be any number of blocks of output actions and input actions, as shown in the following example Serial Channel commands:

```
1SERIAL("{output actions}",options)
1SERIAL("input actions",options)
1SERIAL("{output actions}input actions",options)
1SERIAL("{output}input{output}input",options)
```

The "**control\_string**" is always executed in order left to right, giving you complete control over the sequence of actions.

Where a bi-directional dialog occurs between the DT80 and serial device, the output actions and input actions can be included in the same Serial Channel command as shown above, or in separate Serial Channel commands as follows:

```
BEGIN
  RA1M
    1SERIAL("{output actions}",options)
    1SERIAL("input actions",options)
END
```

This latter approach simplifies the appearance of the program steps for supervising the Serial Channel, particularly if there are a number of data points to be prompted and interpreted or parsed in each access. (Note however that each instance of **1SERIAL** uses up one channel table entry (see [178](#)))

## Serial Data Transmission and Reception

If a job contains one or more **1SERIAL** channel definitions then the serial channel is activated. Data may then be received from a connected serial device *at any time* whilst the job is loaded. As data is received, it is stored in an area of memory called the serial channel **receive buffer**.

When a **1SERIAL** channel is evaluated (ie. when the schedule of which it is part executes), the DT80 processes the control string from left to right. *Output actions* involve data being sent from the DT80, so they are performed there and then, as they are encountered in the control string.

When the DT80 finds an *input* action in the control string it will read any previously received data from the receive buffer and attempt to match it against the format specified in the input action. If no data is present in the receive buffer at the time that the input action is processed then the DT80 will **wait** up to 10 seconds (this timeout is configurable) for more data to arrive. Then:

- If the incoming data matches that required by the input action then the DT80 will move on to the next input action in the string. If the end of the control string is reached then the **1SERIAL** channel will return **0** (success).
- If the timeout expires while the DT80 is waiting for more data then evaluation of the **1SERIAL** channel will be terminated and a "receive timeout " error (code **20**) will be returned.
- If the timeout expires while the DT80 is waiting for a particular CTS state (ie. `\c0` or `\c1` input action ([P151](#))) then evaluation of the **1SERIAL** channel will be terminated and a "CTS timeout " error (code **5**) will be returned.
- If data is received which violates the input action specification then evaluation of the **1SERIAL** channel will be terminated and a "Scan Error " (code **29**) will be returned.

Once the **1SERIAL** channel has returned a success or failure code, the DT80 will then move on to evaluating the next channel (if any) in the schedule.

The following sections describe in detail the various output and input actions that can be specified in a control string.

## Control String – Output Actions

The table below lists the ways in which prompts and text strings can be sent from the DT80 to the device connected to the Serial Channel. These commands must be enclosed by `{ }` in the control string.

What to output	Output Action syntax	Description
Text	<code>text</code> eg. <code>abc\009def\013, GETVAL^M^J</code>	A sequence of characters to be <b>sent</b> . Non-printable characters may be specified using <code>\nnn</code> (where <code>nnn</code> is the ASCII code, 0-255). <code>^char</code> notation may also be used for control characters (ASCII 1-31). For example, <code>\013</code> and <code>^M</code> both specify a CR (carriage return) character
Break character	<code>\b[n]</code> or <code>\b[nCV]</code>	Transmit a "break" (set the Tx line to logic-0 state) for <code>n</code> or <code>nCV</code> character periods
Control signal	<code>\r1</code> <code>\r0</code>	Set <b>RTS</b> (to a value $>+3.5V$ ) – RS232 only Clear <b>RTS</b> (to a value $<-3.5V$ ) – RS232 only
(Wait)	<code>\w[n]</code> or <code>\w[nCV]</code>	Delay for <code>n</code> or <code>nCV</code> milliseconds. Actual delay time will be approximately 2ms or 2 character times, whichever is longer.
% character	<code>%</code>	Output % character
CV value	<code>%flag<sub>opt</sub>width<sub>opt</sub>.precision<sub>opt</sub>type[nCV]</code> eg. <code>%d[2CV]</code> or <code>%9.3f[7CV]</code> or <code>%06d[1CV]</code>	Output the value of <code>nCV</code> in the specified numeric format (see below). Note that <code>opt</code> signifies "optional"
string value	<code>%flag<sub>opt</sub>width<sub>opt</sub>.precision<sub>opt</sub>s[n\$]</code> eg. <code>%s[1\$]</code> or <code>%-9.9s[2\$]</code>	Output the value of string variable <code>n\$</code>

### Numeric Formats

This table describes the possible values for `type` – that is, the different ways in which a CV value can be converted into a string of characters.

Type	Description	Example, assumes 1CV = 74.36
<b>f</b>	floating point	<code>1SERIAL({%f[1CV]})</code> → 74.36
<b>e</b>	floating point, exponential format	<code>1SERIAL({%e[1CV]})</code> → 7.436e01
<b>E</b>	floating point, exponential format	<code>1SERIAL({%E[1CV]})</code> → 7.436E01
<b>g</b>	<b>f</b> or <b>e</b> format depending on value	<code>1SERIAL({%g[1CV]})</code> → 74.36
<b>G</b>	<b>f</b> or <b>E</b> format depending on value	<code>1SERIAL({%G[1CV]})</code> → 74.36
<b>d</b>	integer	<code>1SERIAL({%d[1CV]})</code> → 74
<b>x</b>	hexadecimal integer	<code>1SERIAL({%x[1CV]})</code> → 4a
<b>X</b>	hexadecimal integer	<code>1SERIAL({%X[1CV]})</code> → 4A
<b>o</b>	octal integer	<code>1SERIAL({%o[1CV]})</code> → 112
<b>c</b>	single character	<code>1SERIAL({%c[1CV]})</code> → J

- The `%c` conversion outputs the value of `nCV` as a single 8-bit character. Only the lower 8 bits of the integer portion of `nCV` are output. So in the above example the character value 74 (ASCII "J") will be sent.
- The `%g` and `%G` conversions select exponential notation if the exponent is less than  $-4$ , or greater than or equal to the specified

### Width, Precision and Flag

The various conversion types described above can be further qualified using the optional `width`, `precision` and `flag` specifiers. These allow you to control exactly how the transmitted data will be formatted.

The `width` value specifies the **minimum output field width** – that is, the minimum number of characters that will be output. If the converted value requires fewer characters than the specified field width, then space or zero characters are used to pad the field to the specified width. If the converted value results in *more* characters than the specified field width, then all characters will still be output. The `width` parameter is not applicable for the `%c` conversion type.

The `precision` value means different things depending on the conversion type:

Type	<code>precision</code> term specifies:	Default
<b>d, x, X, o</b> (integer)	minimum number of digits to print (leading zeroes will be added if necessary)	no minimum
<b>e, E, f</b> (floating point)	number of digits to the right of decimal point	6 digits
<b>g, G</b> (mixed)	number of <b>significant digits</b> shown	6 digits

<b>c</b> (single character)	not applicable	
<b>s</b> (string)	<b>maximum</b> number of characters from the string to print	no maximum

The *width* and *precision* values are normally specified as numeric constants (eg. `%9.2f`), but they can also be specified as an asterisk (\*), in which case the value of a CV is used instead.

<code>%flag<sub>opt</sub>*.precision<sub>opt</sub>type[nCV,wCV]</code>	output the value of <i>nCV</i> in the specified numeric format, with the <i>width</i> parameter set to the value of <i>wCV</i>
eg. <code>%*d[1CV,4CV]</code> or <code>%-*.2f[1CV,3CV]</code>	
<code>%flag<sub>opt</sub>*.*[nCV,wCV,pCV]</code>	as above, but also set the <i>precision</i> parameter to the value of <i>pCV</i>
eg. <code>%*.*g[1CV,4CV,5CV]</code>	

Finally, the *flag* character allows some further options:

Flag	Applicable conversion types	Description
-	<b>d, x, X, o, e, E, f, g, G, s</b>	left justify (if spaces need to be added to make up the minimum field width, add them <i>after</i> the number rather than before)
+	<b>d, x, X, o, e, E, f, g, G</b>	if the value is positive, prefix it with + character
(space)	<b>d, x, X, o, e, E, f, g, G</b>	if the value is positive, prefix it with space character
0 (zero)	<b>e, E, f, g, G</b>	pad the field with leading zero characters (rather than spaces) if required to make up the minimum field width
#	<b>x, X, o</b>	prefix value with <b>0x, 0X</b> or <b>0</b> , respectively
#	<b>e, E, f</b>	always include a decimal point
#	<b>g, G</b>	do not truncate any trailing zeroes after the decimal point

## Examples

Examples assume 1CV = 12345.67, 1\$ = "pumpkin"	
<code>1SERIAL({%f[1CV]})</code>	→ "12345.67"
<code>1SERIAL({%10f[1CV]})</code>	→ " 12345.67"
<code>1SERIAL({%10.1f[1CV]})</code>	→ " 12345.7"
<code>1SERIAL({%-10.1f[1CV]})</code>	→ "12345.7 "
<code>1SERIAL({%010.1f[1CV]})</code>	→ "00012345.7"
<code>1SERIAL({%10.10d[1CV]})</code>	→ "0000012345"
<code>1SERIAL({%10.4g[1CV]})</code>	→ " 1.235e04"
<code>1SERIAL({%#10.0f[1CV]})</code>	→ " 12346."
<code>1SERIAL({%s[1\$]})</code>	→ "pumpkin"
<code>1SERIAL({%10s[1\$]})</code>	→ " pumpkin"
<code>1SERIAL({%10.4s[1\$]})</code>	→ " pump"

## Control String – Input Actions

The table below lists the commands available to interpret the information coming back into the Serial Channel from the serial device. Input actions are not enclosed by {} in the control string.

Expected data	Input Action syntax	Description
Characters	<i>text</i>	For each character in the input action string, the DT80 will read and discard all incoming characters from the serial device until that particular character is seen. It then discards the matching character and starts looking for the next character in the input action text. For example, if the input action string is <b>abc</b> and the input data from the serial device is <b>3c3aabaAAc123</b> then all characters up to and including the second "c" will match, ie. they will be read and discarded. Non-printable characters may be specified using <code>\nnn</code> or <code>^char</code> notation, as per Output Actions.
Control signal state	<code>\c1[n]</code> or <code>\c1[nCV]</code> <code>\c0[n]</code> or <code>\c0[nCV]</code>	wait up to <i>n</i> or <i>nCV</i> milliseconds for <b>CTS</b> input to be set (high) – RS232 only wait up to <i>n</i> or <i>nCV</i> milliseconds for <b>CTS</b> input to be cleared (low) – RS232 only
(Wait)	<code>\w[n]</code> or	Delay for <i>n</i> or <i>nCV</i> milliseconds. Actual delay time will be

	<code>\w[nCV]</code>	approximately 2ms or 2 character times, whichever is longer.
(Erase receive buffer)	<code>\e</code>	Clear all previously received characters from the receive buffer
(Print & erase receive buffer)	<code>\p</code>	Print the current contents of the receive buffer (for diagnostic purposes) then clear it
Fixed text string	<code>\m[<i>text</i>]</code> or <code>\m[n\$]</code>	Read and discard incoming characters until the exact string <i>text</i> (or the text in <i>n\$</i> ) is seen, then discard the matching string
Numeric data	<code>%width<sub>opt</sub>type[nCV]</code> eg. <code>%d[2CV], %9f[7CV]</code>	Interpret the received data according to the specified numeric format and store the result into <i>nCV</i> . Note that <i>opt</i> signifies "optional"
String data	<code>%width<sub>opt</sub>type[n\$]</code> eg. <code>%6s[5\$]</code>	Interpret the received data according to the specified string format and store the result into <i>n\$</i>
Data to skip	<code>.*width<sub>opt</sub>type</code> eg. <code>.*6s, .*f</code>	Interpret the received data according to the specified numeric/string format but do not store the result. In other words, skip over this data value.
One of a set of strings	<code>%width<sub>opt</sub>type['str1','str2',...,nCV=m<sub>opt</sub>]</code> eg. <code>%9s['goose','moose',23CV=2]</code>	If the incoming string matches <i>str1</i> then set <i>nCV</i> =0 If the incoming string matches <i>str2</i> then set <i>nCV</i> =1 If the incoming string matches <i>str3</i> then set <i>nCV</i> =2 (etc.) If a default value ( <i>=m</i> ) is specified and the incoming string matches none of the strings then set <i>nCV</i> = <i>m</i>

## Numeric and String Formats

These tables describe the possible values for *type* – that is, the different ways in which the incoming string of characters can be interpreted.

Type	Description	Example, assumes input data string is 123.456
<b>f</b>	floating point	<code>1SERIAL(%f[1CV])</code> → 1CV = 123.456 (nothing left in receive buffer)
<b>d</b>	decimal integer	<code>1SERIAL(%d[1CV])</code> → 1CV = 123 (.456 left in receive buffer)
<b>x</b>	hexadecimal integer	<code>1SERIAL(%x[1CV])</code> → 1CV = 291 (.456 left in receive buffer)
<b>o</b>	octal integer	<code>1SERIAL(%o[1CV])</code> → 1CV = 73 (.456 left in receive buffer)
<b>i</b>	decimal/hex/octal integer	<code>1SERIAL(%i[1CV])</code> → 1CV = 123 (.456 left in receive buffer)
<b>c</b>	character	<code>1SERIAL(%c[1CV])</code> → 1CV = 49 (23.456 left in receive buffer)
<b>b</b>	binary (no conversion)	<code>1SERIAL(%b[1CV])</code> → 1CV = 49 (23.456 left in receive buffer)

Type	Description	Example, assumes input data string is aaba cxyab↓
<b>s</b>	string (↓ terminated)	<code>1SERIAL(%s[1\$])</code> → 1\$ = "aaba cxyab" (nothing left in receive buffer)
<b>S</b>	string (whitespace terminated)	<code>1SERIAL(%S[1\$])</code> → 1\$ = "aaba" (cxyab↓ left in receive buffer)
<b>[chars]</b>	string containing only specified chars	<code>1SERIAL(%[abc][1\$])</code> → 1\$ = "aaba c" (xyab↓ left in receive buffer)
<b>[~chars]</b>	string <u>not</u> containing specified chars	<code>1SERIAL(%[~bc][1\$])</code> → 1\$ = "aa" (ba cxyab↓ left in receive buffer)

- The **%i** conversion assumes that the value is hexadecimal if it starts with `0x` or `0X`, octal if it starts with `0` (zero), otherwise decimal.
- The **%f** conversion will accept numbers in standard (eg. `-12.39904`) or exponential (eg. `-1.239904e01`) format.
- The **%c** and **%b** conversions treat the characters as 8-bit binary values. So the character "1" (ASCII 49) will result in the value 49 being stored in the CV.

## Width

The optional *width* value specifies the maximum number of characters to read for conversion. For example, with the above example's input data: `1SERIAL(%2d[1CV])` will result in 1CV = 12 (3.456 left in receive buffer). The default for most of the conversions (except **%c** and **%b**) is to keep reading characters until an invalid character is read. (That's why the integer conversions in the above example stop when the "." character is seen.)

The default *width* value for *%c* and *%b* is 1; with this setting the two conversions behave identically. However, if *width* is specified then:

- for *%c*, only the last character is read; preceding characters are skipped
- for *%b*, the specified number of characters are treated as a multi-byte binary word, in "big-endian" (most significant byte first) format. Note that due to the limited precision of CVs, the maximum practical width value is 3 (24 bits).

For example, with 123.456 input data:

<code>1SERIAL(%1c[1CV])</code>	→ 1CV = 49 (23.456 left in receive buffer)
<code>1SERIAL(%2c[1CV])</code>	→ 1CV = 50 (3.456 left in receive buffer)
<code>1SERIAL(%3c[1CV])</code>	→ 1CV = 51 (.456 left in receive buffer)
<code>1SERIAL(%1b[1CV])</code>	→ 1CV = 49 (23.456 left in receive buffer)
<code>1SERIAL(%2b[1CV])</code>	→ 1CV = 12594 (49*256 + 50) (3.456 left in receive buffer)
<code>1SERIAL(%3b[1CV])</code>	→ 1CV = 3223859 (49*65536 + 50*256 + 51) (.456 left in receive buffer)

**Important** If *width* is not specified then the incoming data must be terminated by a non-matching character, otherwise the serial channel will continue to wait for more characters to be read, eventually returning a timeout.

For example, if the control string is `1SERIAL(%d[1CV])`:

Input data	Result
"abc"	Scan Error (1CV unchanged, abc left in receive buffer)
"123"	Receive Timeout (1CV unchanged, nothing left in receive buffer)
"123 "	1CV = 123 (nothing left in receive buffer)
"123abc"	1CV = 123 (abc left in receive buffer)

## Control String – Example

The control string in the Serial Channel command

```
1SERIAL("\e{WN\013}%d[1CV],%f[2CV]{C\013}\w[2000]")
```

specifies the following output and input actions for supervising electronic weighing scales connected to the serial channel of a DT80 (the scales have a Weigh Now command *WN*, which instructs the scales to perform a weighing operation):

<code>\e</code>	An <b>input</b> action. <code>\e</code> erases any extraneous characters that may have been sent by the scales at some earlier time.
<code>{WN\013}</code>	An <b>output</b> action. Sends the Weigh Now command ( <i>WN</i> ) to the scales. The <i>WN</i> command is terminated by a carriage return ( <code>\013</code> ). (See your serial device's manual for details of its command set.)
<code>%d[1CV],%f[2CV]</code>	An <b>input</b> action. These scales return two comma-separated values: a batch number as an integer, and the weight as a floating-point value, followed by a carriage return. <ul style="list-style-type: none"> <li>• <code>%d[1CV]</code> will interpret the first returned value as an integer batch number, and assign this to 1CV.</li> <li>• Skip the comma in the returned data string (<code>,</code>).</li> <li>• <code>%f[2CV]</code> will interpret the second returned value as a floating-point weight in kilograms, and assign this to 2CV.</li> </ul>
<code>{C\013}</code>	An <b>output</b> action. These scales also have a Clear command ( <i>C</i> ), which instructs the scales to clear ready for the next weighing operation.. This output action sends the Clear command to the scales. The Clear command is terminated by a carriage return ( <code>\013</code> ).
<code>\w[2000]</code>	An <b>input</b> action. These scales do not respond to commands for 2s after a Clear operation. The <code>\w[2000]</code> action ensures that at least this time elapses following a Clear.

## DeTransfer — Use a Double Backslash

**Important** The DeTransfer program, which is often used to supervise the DT80, has a number of special commands that begin with a `\` (backslash) character (see the DeTransfer Help). The backslash character indicates to DeTransfer that the following command is to be interpreted and executed by DeTransfer. For example:



- `\w3` instructs DeTransfer to wait 3 seconds before communicating again with the *dataTaker*.
- `\t` instructs DeTransfer to read the computer clock and format a current time string.
- `\013` instructs DeTransfer to send the single character carriage return to the connected *dataTaker*.

The backslash character and some number of characters following it will therefore *not* be sent to the DT80 if they are typed into DeTransfer.

This is an issue for the serial channel because a number of the Serial Channel input and output actions also begin with a backslash character.

In order to send a backslash character to the DT80, it is necessary to "escape" the backslash, so that DeTransfer treats it as a regular character. To do this, the backslash must be entered twice, which will cause a single backslash to be sent.

Thus if the above example was to be sent to the DT80 using DeTransfer, you would enter it as follows into the Send window in DeTransfer:

```
1SERIAL("\e{WN\013}%d[1CV],%f[2CV]{C\013}\w[2000]")
```

Note that this rule only applies to DeTransfer. Other terminal programs (eg. HyperTerminal) and DeLogger do not interpret the backslash character, so for these packages the Serial Channel backslash commands must be entered using single backslashes.

## Schedules

### Executing Serial Channel Commands in Schedules

Like any other channel type, Serial Channel commands can be placed into scan schedules. For example

```
BEGIN
  RA1M 1SERIAL(RS485,"\e{READ1^M}%6f[1CV]",W)
  RB2-E 1SERIAL(RS485,"\e{READ2^M}%12s[1$]",W)
END
```

will, once a minute, request then read a floating point value from device #1 on the multi-drop RS485 link connected to the serial channel. Also, every time digital input **2D** goes low, the serial channel will request then read a string value from device #2.

Serial commands can also be used in the "immediate" schedule, ie. executed immediately after they are entered. For example, sending

```
1SERIAL("{hello^M^J}")
```

will immediately transmit the indicated string on the serial channel.

### Triggering Schedules

Sometimes the serial device connected to the Serial Channel returns data unsolicited, and so the program must be capable of responding to the device at any time. Any schedule (**Ra**) can be defined to trigger on the receipt of specified characters at the Serial Channel as follows:

```
Ra1SERIAL"text" 1SERIAL("control_string",options)
```

This will trigger the schedule when the specific text string *text* is received. The text string may contain control characters in `^char` or `\nnn` notation.

The text string may also be blank:

```
Ra1SERIAL"" 1SERIAL("control_string",options)
```

in which case any character received into the Serial Channel produces a trigger.

Whenever the Serial Channel produces a trigger by either of these methods, the receive buffer will contain the string that caused the trigger, ready to be processed by a **1SERIAL** command.

## Serial Sensor Power Control

If the current job contains no **1SERIAL** commands then the serial channel interface is automatically switched off to conserve power. If the current job does contain **1SERIAL** commands then serial channel will be continuously powered.

The **1SSPORT** command allows you to turn power to the interface on and off under program control e.g

```
RA1H 1SSPORT=1 1SERIAL("{X}%d[1CV]") 1SSPORT=0
```

will, once an hour, switch on the serial channel, poll and read an integer from a serial device, then switch off the serial channel.

## Serial Channel State

As mentioned above, whenever a Serial Channel command is executed, it will return a state value that indicates success or otherwise of the execution.

If the execution is successful, then **0 State** is returned. If there has been a problem, **n State** is returned, where **n** is the error state code relating to the problem. Here are the DT80's Serial Channel error state codes:

Code	Error
0	OK
5	CTS detect timeout      Time out waiting for CTS while executing <code>\c0</code> or <code>\c1</code> input action



20	Receive timeout	Time out while waiting for data in an input action
29	Scan Error	Received data cannot be converted according to input action

Return of the Serial Channel state can be disabled by the **W** channel option ([P155](#)).

## Serial Channel Debugging Tools

### P56 Debugging

The DT80 has some useful debugging modes available for the Serial Channel. These allow the user to send information about the progressive operation of the Serial Channel from the host communications port, for viewing in DeTransfer or other terminal software.

These debugging modes are activated by setting Parameter56 as follows:

<b>P56=0</b>	No debugging information (default)
<b>P56=1</b>	Show contents of the receive buffer when each character is read as: <code>ReadByte[<i>contents of receive buffer</i>&lt;EndOfBuf&gt;]</code>
<b>P56=2</b>	When <code>\e</code> command is executed, show contents of receive buffer before and after erasure: <code>EraseBefore[<i>contents of receive buffer</i>&lt;EndOfBuf&gt;]</code> <code>EraseAfter[<i>contents of receive buffer</i>&lt;EndOfBuf&gt;]</code>  When <code>\w</code> command is executed, show contents of receive buffer before and after the wait period: <code>WaitBefore[<i>contents of receive buffer</i>&lt;EndOfBuf&gt;]</code> <code>WaitAfter[<i>contents of receive buffer</i>&lt;EndOfBuf&gt;]</code>
<b>P56=4</b>	Show contents of transmit buffer before and after transmission: <code>Transmit:b[<i>contents of transmit buffer</i>&lt;EndOfBuf&gt;]</code> <code>Transmit:a[<i>contents of transmit buffer</i>&lt;EndOfBuf&gt;]</code>
<b>P56=8</b>	Returns type of parsing scan effected each time data is parsed: <code>%d[ ]Scan:Integer[ScanControl]</code> <code>%f[ ]Scan:Real[ScanControl]</code> <code>%s[ ]Scan:String[ScanControl]</code>
<b>P56=16</b>	Indicates where a <code>1SERIAL("control_string")</code> command failed
<b>P56=32</b>	Shows the contents of the receive buffer after a trigger: <code>NullTrig[<i>contents of receive buffer</i>&lt;EndOfBuf&gt;]</code> <code>CharTrig[<i>contents of receive buffer</i>&lt;EndOfBuf&gt;]</code>

Combinations of debug information can be returned by combining those required in the P56 setting. For example, setting **P56=17** shows contents of the receive buffer (P56=1) when each character is read, and where the `1SERIAL("control_string")` was in error if an error occurred (P56=16).

### Error Messages

If the DT80 is in free-format mode (/h), any syntax error message for the serial command string will include a reference to the error's position in the command string.

### Serial Loopback

A useful technique for testing your parsing commands is to implement a serial loopback in the RS-232 mode. Simply connect the **Rx** and **Tx** terminals together, and then send strings out of the Serial Channel by output actions. Because of the loopback, these strings appear in the receive buffer, which can then be parsed by your input actions. The strings you should send should contain data formatted in the same way that the real sensor would. In this way you are simulating the sensor for the purposes of verifying that your program can correctly interpret what it needs.

For example, if a loopback connection is used, the commands

```
1SERIAL("\e{ABCD,1234\013}%4s[1$],%4d[1CV]") 1$ 1CV
```

should store **ABCD** and **1234.0** into **1\$** and **1CV** respectively.

This test allows you to differentiate between a failed sensor and program errors.

### Serial Channel Examples

Control String	Action
<code>1SERIAL(RS232, "xyz ")</code>	Discard the text <b>xyz</b> and one or more whitespace characters; case-sensitive
<code>1SERIAL("%d[1CV] %8f[2CV] %c[3CV]")</code>	Decode integer, float and character with separating whitespaces
<code>1SERIAL("\c1[2000]%d[1CV]")</code>	Wait 2 seconds for CTS to become high, then decode integer
<code>1SERIAL("{1.3TK\013\010}")</code>	Send <b>1.3TK</b> followed by a carriage return command ( <code>\013</code> ) and a line feed command ( <code>\010</code> ) to a device on the Serial Channel
<code>1SERIAL("{%f[1CV] %d[2CV] %c[3CV]}")</code>	Output <b>1CV</b> as float, <b>2CV</b> as integer and <b>3CV</b> as character to a printer or display connected to the Serial Channel
<code>1SERIAL("{\r1\w[1CV]\r0}")</code>	Raise RTS, wait <b>1CV</b> milliseconds, then drop RTS
<code>1SERIAL(RS485, "{D2READ\013}%2.2f[1CV],%2.2f[2CV]")</code>	Send a <b>READ</b> command to the sensor at address <b>D2</b> on multidrop <b>RS485</b> network, decode two returned floats separated by a comma into the two CVs

Schedule	Action
<code>RA1SERIAL"STX" 1SERIAL("STX%d[1CV]") 1CV=1CV+1</code>	Trigger schedule <b>A</b> on arrival of <b>STX</b> , scan the integer into <b>1CV</b> and increment it (barcode scanner)
<code>RA1M 1SERIAL("{Num Temp Press\13\10%3d[1CV] %4.1f[2CV]}")</code>	Send <b>Num Temp Press</b> then <b>CRLF</b> and, on new lines, <b>1CV</b> as integer and <b>2CV</b> as float (report generation)
<code>BEGIN RA1M 1V(=1CV) 2TK(=2CV) 1SERIAL("{Flow %10.1f[1CV]\13\10 Temp %10.1f[2CV]\13\10}") END</code>	Read data from analog channels and report as text to a printer connected to the serial channel, in the same schedule (report generation)
<code>PS=9600,N,8 BEGIN RA1SERIAL"\$GPRMC" 1SERIAL("RS232," ,%f[8CV]%c[9CV],%f[10CV]%c[11CV]\e",2,W) D T 8..11CV(FF2) END</code>	Program to read latitude ( <b>8CV</b> ) and North or South ( <b>9CV</b> ), longitude ( <b>10CV</b> ) and East of West ( <b>11CV</b> ) from the NMEA 183 data stream issued continuously from a GPS unit. Data is logged with date and timestamp. Each read is triggered by the <b>\$GPRMC</b> header at the start of each transmission.

Remember that if DeTransfer is used to send commands then two backslash characters must be sent each time a backslash is required.

### Configuring the Serial Channel

Communications Type	Channel Option	Multidrop	DT80 Terminals			
			Tx Z	Rx A	RST Y	CTS B
RS-232 (default)	<b>RS232</b>	No		Used	Hand-shake	Hand-shake
RS-422	<b>RS422</b>	Yes	Used	Used	Used	Used
RS-485	<b>RS485</b>	Yes	Z	A	Y	B

← That is, Z is connected to A and Y is connected to B.

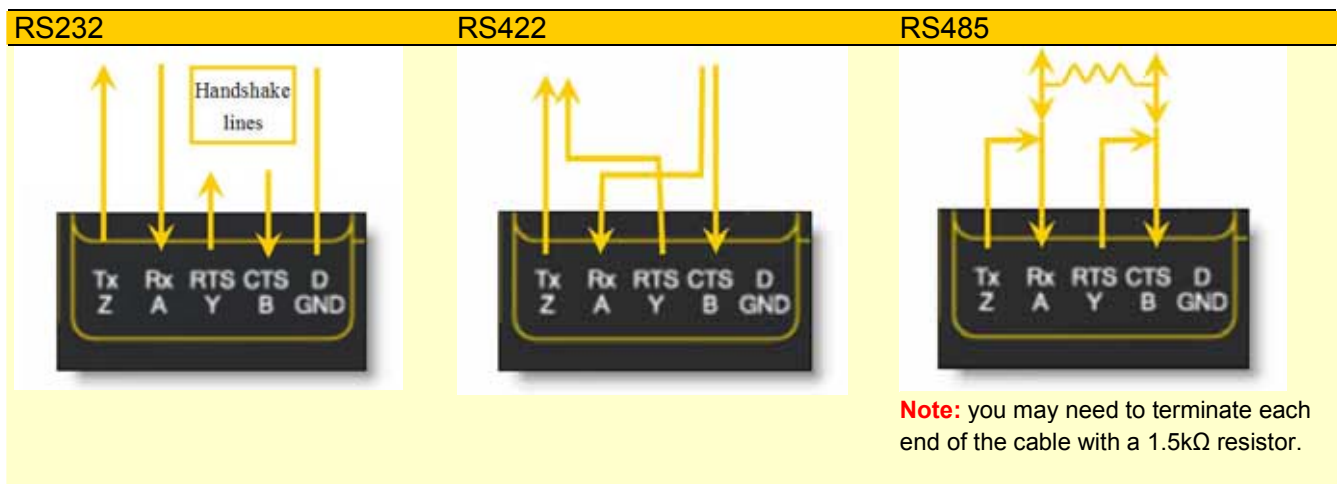


Figure 53: Serial Channel transmit and receive flows

# WIRING CONFIGURATIONS — ANALOG CHANNELS

This section contains configuration diagrams for wiring signals and sensors to the DT80's analog channels (channels 1 to 4 on the right of its front panel).

Analog Channels — Introduction [\(P15\)](#) covers important concepts you need to be familiar with to successfully use the wiring configurations presented here; concepts such as the DT80's terminal designations, independent inputs and shared-terminal inputs, and sensor excitation. Which Analog Input Configuration Should I Use? [\(P17\)](#) also contains useful information.

Note: the dotted line may or may not be required depending upon the noise considerations etc.

## Voltage Inputs

Channel	V	voltage (see V <a href="#">(P26)</a> )	Output in mV
---------	---	--	--------------

<b>TB, TC, TD,</b> <b>TE, TG, TJ,</b> <b>TK, TN, TR,</b> <b>TS, TT</b>	thermocouples (see <b>TB, TC, TD</b> (P28))	Output in Degrees (as set by P36)
<b>AS</b>	analog state input measured as a voltage	Output 0 or 1
<b>F</b>	frequency (see <b>F</b> (P27))	Output in Hz

### Shared-Terminal Voltage Inputs

Shared-Terminal (P16) introduces this wiring configuration.

Up to three of these inputs can share the # terminal on any channel

Maximum number of shared-terminal voltage inputs on a DT80: 3 per channel x 4 channels = 12. Wiring gauge, length and environment is non-critical.

Shared-Terminal Voltage Inputs sharing the # terminal

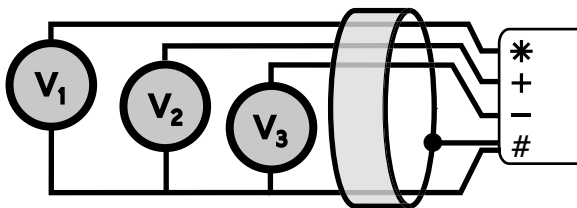


Figure 54: **V1** Wiring for shared-terminal voltage input

Example commands for reading inputs of the type shown in Figure 54 (P158):

1+V	2*V	3-V	4+V
-----	-----	-----	-----

### Independent Voltage Inputs

Independent Analog Inputs (P16) introduces this wiring configuration.

Maximum number of independent voltage inputs to a DT80: 4 (one per analog channel.) Wiring gauge, length and environment is non-critical

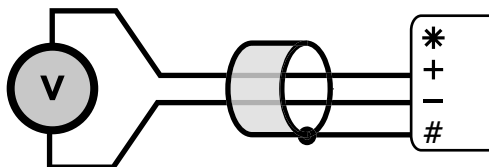


Figure 55: **V2** Wiring for independent voltage input.

Example commands for reading inputs of the type shown in Figure 55 (P158):

1V	2AS	3V	4TK
----	-----	----	-----

### Current Inputs

Channel types:	I current (see <b>I</b> (P26))	Output in mA	Output mA
	L 4–20mA current loop (see <b>L</b> (P27))		Output 0~100% (if not specified as a channel factor then 100Ω Resistor is assumed)

### Independent Current Input with External Shunt

You MUST adhere to the DT80's common-mode voltage limits (default is ±30V relative to analog common) for this configuration to operate correctly.

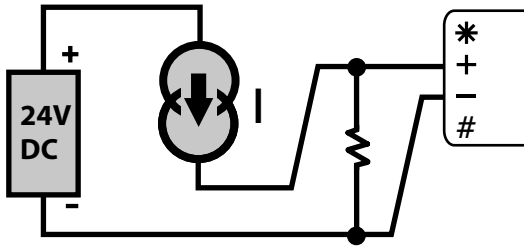


Figure 56: **C1** wiring for independent current input using external shunt

Example commands for reading inputs of the type shown in Figure 56 <sup>(P159)</sup> :			
1I	2I	3L	4I

**Independent Current Input using the internal shunt**

You MUST adhere to the DT80's common-mode voltage limits (default is ±30V relative to analog common) for this configuration to operate correctly.

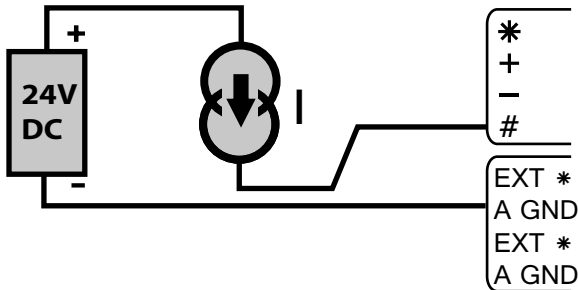


Figure 57: **C2** Wiring for Independent current input using internal shunt

Example commands for reading inputs of the type shown in Figure 56 <sup>(P159)</sup> :			
1#I	2#I	3#L	4#I

**Shared-Terminal Current Inputs with External Shunts**

To avoid cross-channel coupling, connect the bottom of the shunts with the minimum of shared resistance to the sense point.

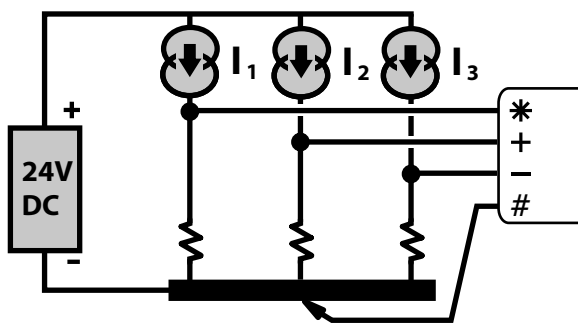


Figure 58: **C3** Wiring for shared-terminal current input using external shunt

Example commands for reading inputs of the type shown in Figure 58 <sup>(P159)</sup> :			
3+I	4+I	1*I	2-I

## Independent current using internal shunt and external excitation

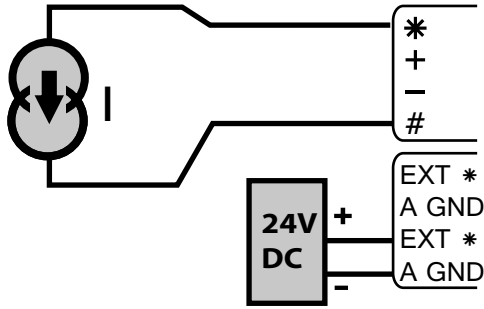


Figure 59: C4 Wiring for independent current using internal shunt and external excitation

Example commands for reading inputs of the type shown in *Figure 59* (P160):

3#I(E)	4#I(E)	1#I(E)	2#I(E)
--------	--------	--------	--------

## Resistance Inputs

Current-excited voltage measurement

Channel type:	R	resistance (see R (P27))	Output in Ohms
	PT385, PT392, NI, CU	RTDs (see PT385 (P28))	Output in Degrees (as set by P36)
	YS01 to YS07, YS16, YS17	thermistors (see YS01 YS02 YS03 YS04 YS05 YS06 YS07 (P28))	Output in Degrees (as set by P36)

### 4-Wire Resistance Inputs

\* and # terminals send an **excitation current** through the unknown resistance while the remaining terminals sense the voltage across it.

4-wire resistance methods are the most accurate because

- the resistances' lead wires are not part of the measurement circuit
- negligible current flows through the sense wires.

Maximum number of independent 4-wire resistance inputs on a DT80: 1 per channel x 4 channels = 4.

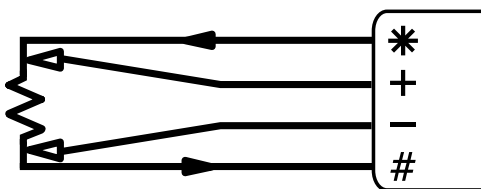


Figure 60: R1 Wiring for 4-wire resistance input

Example commands for reading inputs of the type shown in *Figure 60* (P160):

2R(4W)	1R(4W)	3R(4W,I)	1PT392(4W)
--------	--------	----------	------------

### 3-Wire Resistance Inputs

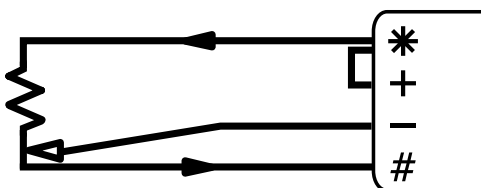


Figure 61: R2 Wiring for 3-wire resistance input

Example commands for reading inputs of the type shown in *Figure 61* (P160):

## 2-Wire Resistance Inputs

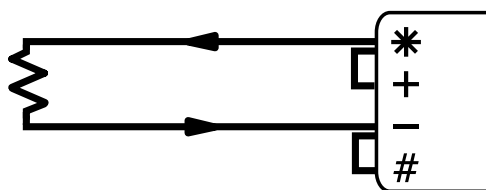


Figure 62: R3 Wiring for 2-wire resistance input

Example commands for reading inputs of the type shown in <i>Figure 62</i> (P161):			
3R	4R	1R	1..4R

## Bridge Inputs

Channel types:	BGV voltage-excited bridge (see BGV (P27))	Output in parts per million (ppm)
	BGI current-excited bridge (see BGV (P27))	Output in parts per million (ppm)

### 6-Wire BGV Inputs

**Important:** a precision external Supply must be used

Related information:

- the Bridge (P27) category in the Table 1: DT80 Channel Types (P29) table
- the Resistance and Bridge (P35) category in the Table 3: DT80 Channel Options (P38) table
- Voltage Excitation BGV (P139)

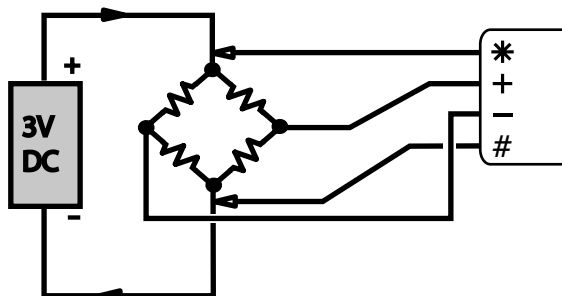


Figure 63: B1 Wiring for 6-wire bridge using external voltage excitation

Example commands for reading inputs of the type shown in <i>Figure 63</i> (P161):			
3BGV(6W)	5BGV(6W)	7BGV(6W)	11BGV(6W)

### 4-Wire BGV Inputs

Use voltage-excited bridge (BGV) configurations only when the bridge is close to the DT80 (BGI configurations are usually preferred).

See also

- the Bridge (P27) category in the Table 1: DT80 Channel Types (P29) table
- the Resistance and Bridge (P35) category in the Table 3: DT80 Channel Options (P38) table
- Voltage Excitation BGV (P139).

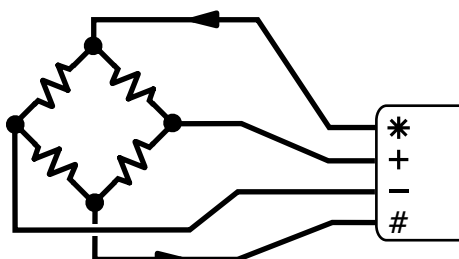


Figure 64: **B2** Wiring for 4 wire bridge input using internal excitation

Example commands for reading inputs of the type shown in Figure 64 (P162):

1BGV(4W)	2BGV(4W)	3BGV(4W)	4BGV(4W)
----------	----------	----------	----------

### 4-Wire BGI Inputs

We recommend the current excited bridge (BGI) method for 4 wire bridge measurement, especially for bridges that are distant from the DT80.

See also

- the Bridge (P27) category in the Table 1: DT80 Channel Types (P29) table
- the Resistance and Bridge (P35) category in the Table 3: DT80 Channel Options (P38) table
- Bridges (P139).

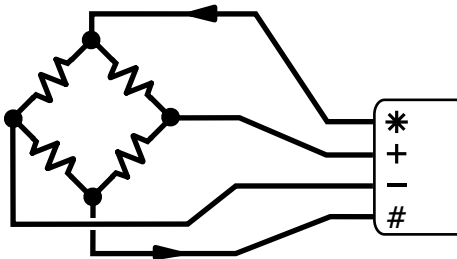


Figure 65: **B2** Wiring for 4 wire bridge input using internal excitation

Example commands for reading inputs of the type shown in Figure 64 (P162):

1BGI(4W)	2BGI(4W)	3BGI(4W)	4BGI(4W)
----------	----------	----------	----------

### 3-Wire BGI Input

Bridge voltage calculated

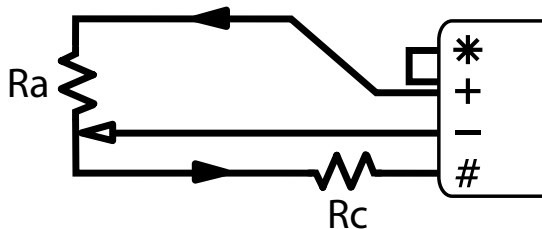


Figure 66: **B3** Wiring for 3 wire bridge input using internal current excitation

Example commands for reading inputs of the type shown in Figure 66 (P162)

3BGI	4BGI	1BGI	2BGI(120)
------	------	------	-----------

## AD590-Series Inputs

IC temperature sensors for long cables (current is proportional to temperature)

Channel types:	AD590, AD592, TMP17	temperature (see AD590 (P28))
----------------	---------------------	-------------------------------



## 2-Wire AD590-Series Inputs

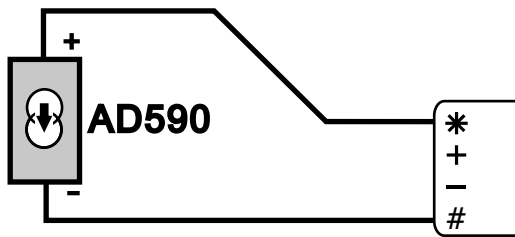


Figure 67: A1 Wiring for AD590 series input using internal shunt

Example commands for reading inputs of the type shown in Figure 64 (P162):

1AD590	2AD590	3AD590	4AD590
--------	--------	--------	--------

## LM35-Series Inputs

IC temperature sensors for long cables (voltage is proportional to temperature)

Channel types:	LM34, LM35, LM45, LM50, LM60,	temperature (see LM34 types: TMP35, TMP36, TMP37 (P28))
----------------	-------------------------------	---

**Note** LM34 and LM35: minimum 0 degrees

### 3 & 4-Wire LM35-Series input - full temperature range

This wiring configuration supports the full sensors operating temperature range. It requires some additional components (two resistors and two diodes).

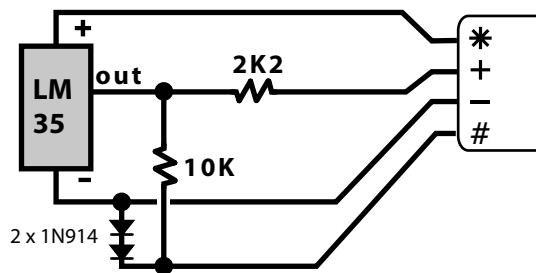


Figure 68: L1 for LM35 series input – full temperature range

### 3 and 4-Wire LM35-Series Inputs – restricted temperature range

This wiring configuration supports a restricted lower operating temperature range for the sensor. The temperature must be above ten degrees (10°C for LM35/45 & TMP35/37, 10°F for LM34). However this configuration does not require any additional components and requires only three wires to the sensor. Accuracy can be improved by replacing the link wire between the '#' and '-' terminals with a wire from the '-' terminal of the logger to the '-' terminal of the sensor.

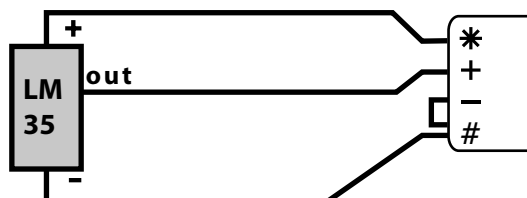


Figure 69: L2 Wiring for LM35 series input – restricted temperature range

## LM135-Series Inputs

IC constant-current supply (voltage is proportional to temperature)

### 4-Wire LM135-Series Inputs

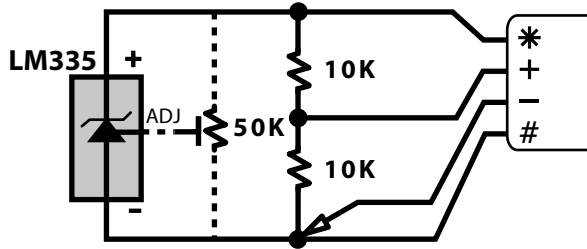


Figure 70: L3 Wiring for LM135 series input

# WIRING CONFIGURATION — DIGITAL CHANNELS

## Digital Input Wiring configurations

Digital Inputs	Notes	Wiring
<b>D1</b> Wiring Voltage Free Contact	1..4D Only 1DS=1 if contact open 1DS=0 if contact closed Channel Types: DB, BN,DB,C,HSC	
<b>D2</b> Wiring for logic level digital state or counter input	1..8D 1DS=1 if input is logic 1 (High) 1DS=0 if input is logic 0 (Low) Channel Types: DB, BN,DB,C,HSC	
<b>D7</b> Wiring Phase Encoder	To read position 1PE 1PE uses 1C & 2C 2PE uses 3C & 4C	

Figure 71 Digital Input Wiring

# Digital output wiring configurations

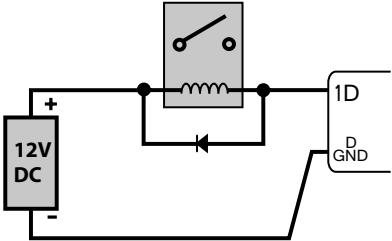
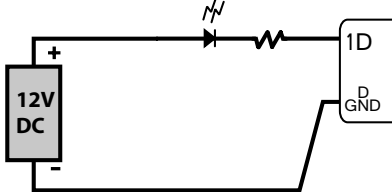
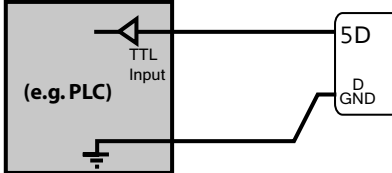
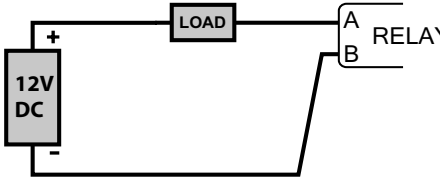
Digital Outputs	Wiring
<p><b>D3</b> Wiring for digital output to drive relay</p> <p>1..4D Only</p> <p>1DSO=0 to turn relay on</p> <p>1DSO=1 to turn relay off</p>	
<p><b>D4</b> Wiring for digital output to drive LED</p> <p>1..4D Only</p> <p>1DSO=0 to turn on LED</p> <p>1DSO=1 to turn off the LED</p>	
<p><b>D5</b> Wiring for logic level digital output</p> <p>1..8D</p> <p>1DSO=1 provides logic 1 (High)</p> <p>1DSO=0 provides logic 0 (Low)</p>	
<p><b>D6</b> Wiring for Relay Output</p> <p>1RELAY=1 to turn load on</p> <p>1RELAY=0 to turn load off</p>	

Figure 72 Digital Output Wiring

# Part M — Reference

## COMMAND SUMMARIES

### Summary — Delete Commands

The following table summarizes all the clear/delete/erase commands supported by the DT80:

	Command	Action	
Job	DELJOB	Deletes the current job from the DT80	When a job is deleted, all traces of it (its name, directory structure, program, data and alarms) are erased from the <i>dataTaker</i> .
	DELJOB" <i>JobName</i> "	Deletes only <i>JobName</i> from the DT80	
	DELJOB*	Deletes all jobs from the DT80	
Data	DELDATA	Deletes the current job's data from the DT80	Job names, directory structures, programs and alarms are not erased.
	DELDATA" <i>JobName</i> "	Deletes only <i>JobName</i> 's data from the DT80	
	DELDATA*	Deletes all jobs' data from the DT80	
Alarms	DELALARMS	Deletes the current job's alarms from the DT80	Job names, directory structures, programs and data are not erased.
	DELALARMS" <i>JobName</i> "	Deletes only <i>JobName</i> 's alarms from the DT80	
	DELALARMS*	Deletes all jobs' alarms from the DT80	
Event Log	CEVTLOG	Clears the event log	
USB memory device	FORMAT"A:"	Formats USB memory device (deletes all data and alarms, and copies the DT80's current internal directory structure to card)	
Attention LED	CATTN	Clears the Attention LED ( <a href="#">P93</a> )	
User Defaults	DELUSERINI	Deletes the working copy of USER.INI from the DT80's internal memory	
	DELONRESET	Deletes the working copy of ONRESET.DXC from the DT80's internal memory	
	DELONINSERT	Deletes ONINSERT.DXC from the DT80-specific directory on the card	
	DELONINSERTALL	Deletes ONINSERT.DXC from the root directory on the card	

See also the [Table 12: DT80 Resets \(P119\)](#).

Table 14: DT80 Delete Commands — Summary

### Summary — Retrieval Commands

The following table summarizes all the information retrieval (unload) commands supported by the DT80 — program, data, alarm and event retrieval.

	Command	Action	
Program	SHOWPROG	Copies the current job's program file to the host port	
	SHOWPROG" <i>JobName</i> "	Copies <i>JobName</i> 's program file to the host port	
	SHOWPROG*	Copies all currently-defined jobs' program files to the host port	
Data U	U	Returns the current job's data in the order of report schedule A to K	
	U <i>x</i>	Returns the current job's data for report schedule <i>x</i>	
	U" <i>JobName</i> "	Returns data for <i>JobName</i> in the order of report schedule A to K	
	U" <i>JobName</i> " <i>x</i>	Returns data for <i>JobName</i> report schedule <i>x</i>	
Data U ( )	U	Returns the current job's data in the order of report schedule A to K	Type <i>date</i> and <i>time</i> in formats that match your DT80's current setup (P31 and P39). If a schedule isn't specified, data is always unloaded schedule-by-schedule. <b>Note</b> BEGIN and END used here
	U( <i>from</i> )	Returns the current job's data starting from BEGIN, <i>time</i> or <i>time, date</i>	
	U( <i>from</i> )( <i>to</i> )	Returns the current job's data starting from BEGIN, <i>time</i> or <i>time, date</i> and ending with END, <i>time</i> or <i>time, date</i>	
	U <i>x</i>	Returns the current job's data for report schedule <i>x</i>	
	U <i>x</i> ( <i>from</i> )	Returns the current job's data for schedule <i>x</i> starting from BEGIN, <i>time</i> or <i>time, date</i>	
	U <i>x</i> ( <i>from</i> )( <i>to</i> )	Returns the current job's data for schedule <i>x</i> starting from BEGIN, <i>time</i> or <i>time, date</i> and ending with	

		END, <i>time</i> or <i>time, date</i>	are <u>not</u> the same as the <b>BEGIN</b> and <b>END</b> keywords used to indicate the start and end of a DT80 job.
	U"JobName"	Returns data for <i>JobName</i> in the order of report schedule A to K	
	U"JobName"(from)	Returns data for <i>JobName</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>	
	U"JobName"(from)(to)	Returns data for <i>JobName</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>	
	U"JobName"x	Returns data for <i>JobName</i> report schedule <i>x</i>	
	U"JobName"x(from)	Returns data for <i>JobName</i> report schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>	
	U"JobName"x(from)(to)	Returns data for <i>JobName</i> report schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>	
Data U[ ]	U	Returns the current job's data in the order of report schedule A to K	Type <i>date</i> and <i>time</i> in the DT80's fixed-format style (ISO format; see Examples — U[ ] Commands (P76)).
	U[from]	Returns the current job's data starting from <i>time</i> or <i>date, time</i>	If a schedule isn't specified, data is always unloaded schedule-by-schedule.
	U[from][to]	Returns the current job's data starting from <i>time</i> or <i>date, time</i> and ending with <i>time</i> or <i>date, time</i>	
	Ux	Returns the current job's data for report schedule <i>x</i>	
	Ux[from]	Returns the current job's data for schedule <i>x</i> starting from <i>time</i> or <i>date, time</i>	
	Ux[from][to]	Returns the current job's data for schedule <i>x</i> starting from <i>time</i> or <i>date, time</i> and ending with <i>time</i> or <i>date, time</i>	
	U"JobName"	Returns data for <i>JobName</i> in the order of report schedule A to K	
	U"JobName"[from]	Returns data for <i>JobName</i> starting from <i>time</i> or <i>time, date</i>	
	U"JobName"[from][to]	Returns data for <i>JobName</i> starting from <i>time</i> or <i>date, time</i> and ending with <i>time</i> or <i>date, time</i>	
	U"JobName"x	Returns data for <i>JobName</i> report schedule <i>x</i>	
	U"JobName"x[from]	Returns data for <i>JobName</i> report schedule <i>x</i> starting from <i>time</i> or <i>date, time</i>	
	U"JobName"x[from][to]	Returns data for <i>JobName</i> report schedule <i>x</i> starting from <i>time</i> or <i>date, time</i> and ending with <i>time</i> or <i>date, time</i>	
Alarms A	A	Returns the current job's alarms in the order of report schedule A to K	
	Ax	Returns the current job's alarms for report schedule <i>x</i>	
	A"JobName"	Returns alarms for <i>JobName</i> in the order of report schedule A to K	
	A"JobName"x	Returns alarms for <i>JobName</i> report schedule <i>x</i>	
Alarms A( )	A	Returns the current job's alarms in the order of report schedule A to K	Type <i>date</i> and <i>time</i> in formats that match your DT80's current setup (P31 and P39).
	A(from)	Returns the current job's alarms starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>	If a schedule isn't specified, alarms are always unloaded schedule-by-schedule.
	A(from)(to)	Returns the current job's alarms starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>	<b>Note BEGIN</b> and <b>END</b> used here are <u>not</u> the same as the <b>BEGIN</b> and <b>END</b> keywords used to indicate the start and end of a DT80 job.
	Ax	Returns the current job's alarms for report schedule <i>x</i>	
	Ax(from)	Returns the current job's alarms for schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>	
	Ax(from)(to)	Returns the current job's alarms for schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>	
	A"JobName"	Returns alarms for <i>JobName</i> in the order of report schedule A to K	
	A"JobName"(from)	Returns alarms for <i>JobName</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>	
	A"JobName"(from)(to)	Returns alarms for <i>JobName</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>	
	A"JobName"x	Returns alarms for <i>JobName</i> report schedule <i>x</i>	

	A"JobName"x(from)	Returns alarms for <i>JobName</i> report schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i>	
	A"JobName"x(from)(to)	Returns alarms for <i>JobName</i> report schedule <i>x</i> starting from <b>BEGIN</b> , <i>time</i> or <i>time, date</i> and ending with <b>END</b> , <i>time</i> or <i>time, date</i>	
Alarms A[ ]	A	Returns the current job's alarms in the order of report schedule A to K	Type <i>date</i> and <i>time</i> in the
	A[from]	Returns the current job's alarms starting from <i>time</i> or <i>date, time</i>	DT80's
	A[from][to]	Returns the current job's alarms starting from <i>time</i> or <i>date, time</i> and ending with <i>time</i> or <i>date, time</i>	fixed-format style
	Ax	Returns the current job's alarms for report schedule <i>x</i>	(ISO format; see
	Ax[from]	Returns the current job's alarms for schedule <i>x</i> starting from <i>date</i> or <i>date, time</i>	Examples — A[ ]
	Ax[from][to]	Returns the current job's alarms for schedule <i>x</i> starting from <i>date</i> or <i>date, time</i> and ending with <i>date</i> or <i>date, time</i>	Unload Command
	A"JobName"	Returns alarms for <i>JobName</i> in the order of report schedule A to K	(P88).
	A"JobName"[from]	Returns alarms for <i>JobName</i> starting from <i>date</i> or <i>date, time</i>	If a schedule isn't
	A"JobName"[from][to]	Returns alarms for <i>JobName</i> starting from <i>date</i> or <i>date, time</i> and ending with <i>date</i> or <i>date, time</i>	specified, alarms
	A"JobName"x	Returns alarms for <i>JobName</i> report schedule <i>x</i>	are always
	A"JobName"x[from]	Returns alarms for <i>JobName</i> report schedule <i>x</i> starting from <i>date</i> or <i>date, time</i>	unloaded
	A"JobName"x[from][to]	Returns alarms for <i>JobName</i> report schedule <i>x</i> starting from <i>date</i> or <i>date, time</i> and ending with <i>date</i> or <i>date, time</i>	schedule-by-sche- dule.
	Event Log	UEVTLOG	Returns the contents of the event log

Table 15: DT80 Retrieval Commands — Summary

# GETTING OPTIMAL SPEED FROM YOUR DT80

## Speed Factor — ADC Settings

The current settings of the DT80's ADC (calibration, settling time, sampling time,...) can be adjusted using various channel options, parameters and switches. See Table 9: DT80 Parameters (P112) and Table 10: DT80 Switches (P113). The default ADC settings are suitable for most sensors and we recommend that you don't alter these unless you're an experienced DT80 user.

## Speed Factor — Number of Channels

The more channels you define, the slower each channel is sampled.

## Speed Factor — Data Storage and Return Options

Formatting data for storage or return consumes processor time and consequently reduces sampling speeds.

## Speed Factor — Data Formatting

Minimizing the number of significant digits in the stored and returned data frees processor time, which results in increased sampling speed.

## Best Speed

You can adjust one or more of the settings in the table below to increase the DT80's normal-mode sampling speed. However, be aware that such optimization generally compromises the accuracy of the DT80. The settings are listed in the order in which we recommend you apply them.

<b>P11</b>	Mains frequency	Increase P11 to increase speed
<b>/r</b>	Return data	Turn data returns off
<b>GLn</b>	Gain lock	Use gain lock <b>GLx</b> to prevent autoranging
<b>P10</b>	Sets recording of Test	The default is zero which is the

	results on or off. The Default is off	fastest option
<b>MDn</b>	Measurement delay	Sets the delay after setting up an analog channel, before the actual measurement is taken. Should avoid setting this less than 5mS if changing input channels.



# ASCII-DECIMAL TABLE

Decimal	ASCII	Control	Description	Decimal	ASCII	Description	Decimal	ASCII	Description	Decimal	ASCII	Description
0	NUL		null	32	space		64	@		96	`	backquote
1	SOH	^A		33	!		65	A		97	a	
2	STX	^B		34	"	dbl quote	66	B		98	b	
3	ETX	^C		35	#		67	C		99	c	
4	EOT	^D		36	\$		68	D		100	d	
5	ENQ	^E		37	%		69	E		101	e	
6	ACK	^F	acknowledge	38	&		70	F		102	f	
7	BEL	^G	bell	39	'	quote	71	G		103	g	
8	BS	^H	backspace	40	(		72	H		104	h	
9	HT	^I	tab	41	)		73	I		105	i	
10	LF	^J	line feed	42	*		74	J		106	j	
11	VT	^K	vertical tab	43	+		75	K		107	k	
12	FF	^L	form feed	44	,	comma	76	L		108	l	
13	CR	^M	carriage return	45	-	dash	77	M		109	m	
14	SO	^N		46	.	period	78	N		110	n	
15	SI	^O		47	/	slash	79	O		111	o	
16	DLE	^P		48	0		80	P		112	p	
17	DC1	^Q	XON	49	1		81	Q		113	q	
18	DC2	^R		50	2		82	R		114	r	
19	DC3	^S	XOFF	51	3		83	S		115	s	
20	DC4	^T		52	4		84	T		116	t	
21	NAK	^U	negative acknowledge	53	5		85	U		117	u	
22	SYN	^V		54	6		86	V		118	v	
23	ETB	^W		55	7		87	W		119	w	
24	CAN	^X		56	8		88	X		120	x	
25	EM	^Y		57	9		89	Y		121	y	
26	SUB	^Z		58	:	colon	90	Z		122	z	
27	ESC		escape	59	;	semicolon	91	[		123	{	
28	FS			60	<		92	\	backslash	124		
29	GS			61	=		93	]		125	}	
30	RS			62	>		94	^	caret	126	~	tilde
31	US			63	?		95	_	underline	127	DEL	delete

Table 16: ASCII Characters

# RS-232 STANDARD

The following table lists the standard RS-232 pinouts for both 9-pin (DB-9) and 25-pin (DB-25) DCE and DTE interfaces as used in *Figure 73 (P171)* and *Figure 74 (P172)*. Normally the DTE (DT80, computer) device's connector is male and the DCE (modem) device's connector is female.

Signal	Function	Direction	DB-9 Pin	DB-25 Pin
DCD	Data Carrier Detect	DTE ← DCE	1	8
RXD	Receive Data	DTE ← DCE	2	3
TXD	Transmit Data	DTE → DCE	3	2
DTR	Data Terminal Ready	DTE → DCE	4	20
GND	Signal Ground		5	7
DSR	Data Set Ready	DTE ← DCE	6	6
RTS	Request To Send	DTE → DCE	7	4
CTS	Clear To Send	DTE ← DCE	8	5
RI	Ring Indicator	DTE ← DCE	9	22

Table 17: RS-232 Pinouts

For applications where a DTE is connected to another DTE (eg. a DT80 is connected to a host computer):

- the RXD and TXD signals must be "crossed over" so that one device's TXD is connected to the other device's RXD.
- The "Request To Send" output changes its meaning to "Clear To Send Output" (ie. a device sets it active when it is able to receive data). This allows hardware flow control to operate in both directions.
- DCD, DTR, DSR and RI are not normally used.

## CABLE DETAILS

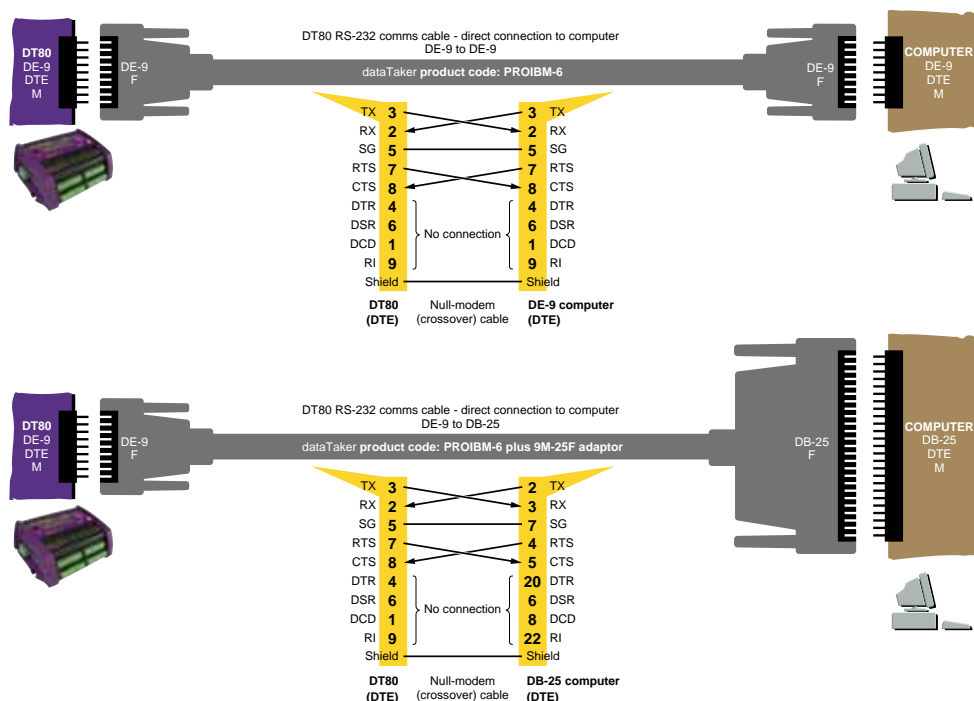


Figure 73: DT80-to-computer RS-232 comms cable — DE-9 computer (upper diagram) and DB-25 computer (lower diagram)

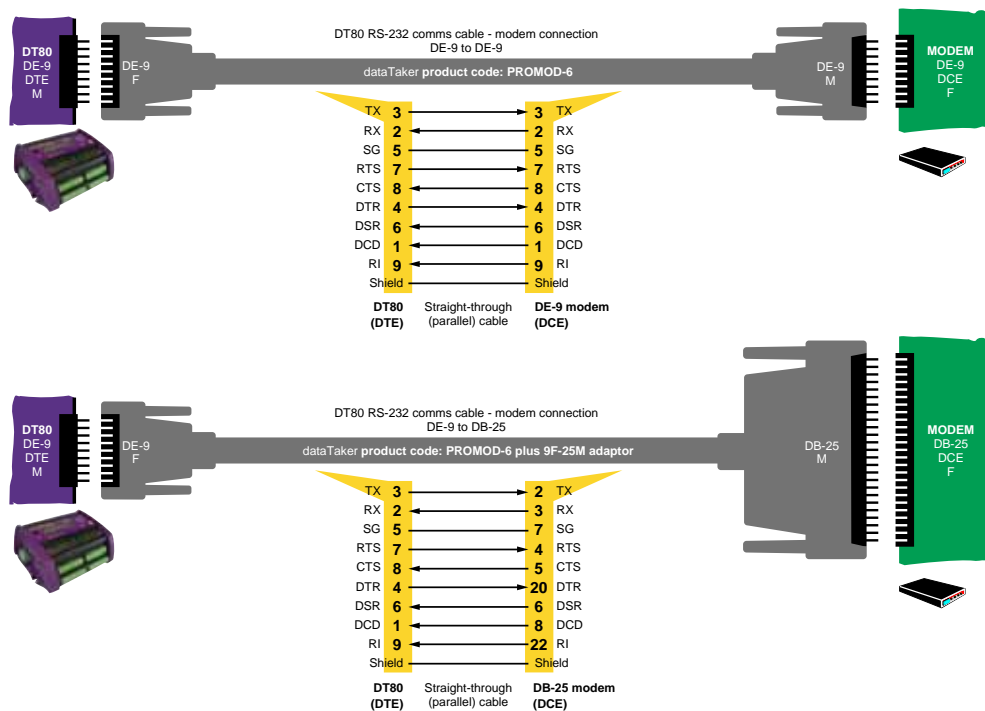


Figure 74: DT80-to-modem RS-232 comms cable — DE-9 modem (upper diagram) and DB-25 modem (lower diagram)

# UPGRADING DT80 FIRMWARE

## Operating system upgrade

The DT80's operating system is stored as "firmware ([P181](#))" in the DT80's Flash memory (see MEMORY ([P125](#))). This means that you can easily upgrade your DT80's operating system from a host computer running DeLogger (or DeTransfer).

An upgrade takes between 2 and 15 minutes (depends on the method used) and is completely safe. (In case of a problem occurring during an upgrade, the DT80 automatically protects its firmware upgrade "bootstrap" file, which always allows you to restart the process.)

**Important** Always check the release notes distributed with the new firmware version for any changes to the upgrade procedure documented here.

### Recommended Preparation

We recommend that you carry out the following procedure before upgrading the DT80's firmware to ensure that no compatibility problems arise.

This procedure returns the DT80 to a completely unprogrammed state. Once the upgrade is complete you will have to restore any settings and programs.

1. Connect to the DT80 and perform the following operations, most of which can be done from the text window interface in DeLogger, or from DeTransfer or other terminal software.

**Note** if you are using DeLogger, it may ask if you want to upgrade the DT80 when you connect to the DT80. If this occurs, answer **No** so that you can perform the following steps before the upgrade occurs.

8. Save any previously logged data stored in the DT80's internal memory by unloading it to the host computer or copying to a USB memory device.
9. In DeLogger, select the Profile... option from the dataTaker menu and note any important profile settings, such as Ethernet IP Address. You must re-enter these once the upgrade is complete.

If you are using DeTransfer or other terminal software, you can issue the **PROFILE** command to return the current profile settings.

10. Delete any ONRESET job stored in flash memory using the **DELONRESET** command.
11. Delete any PROFILE settings stored in flash memory using the **DELUSERINI** command.

12. Format the internal disk using the **FORMAT "B: "** command.

**Note** This will erase all jobs and data stored in the DT80.

13. Carry out the firmware upgrade as described below.

When the upgrade is done, you'll need to reconnect to the DT80 and set up your required settings and programs.

**Recommendation — Power** Before carrying out a firmware upgrade, we recommend that you charge the DT80's main internal battery for 12 hours. Furthermore, if at all possible, power the DT80 from an external source as well during the upgrade. These two precautions minimise the possibility of a power failure to the DT80 during the upgrade.

There are also other ways you can upgrade your DT80's firmware (remotely by FTP transfer, for example). If you have a special requirement such as this contact your *dataTaker* representative.

## **Firmware Upgrade — Host USB or RS232 Port**

Here's the procedure for upgrading the firmware of a DT80 by using DeLogger (version 2 revision 16 or later) on a computer directly connected to the DT80's Host RS-232 port. You can also use DeTransfer Version 3.18 or later.

**Note** If you attempt to make a connection to a DT80, DeLogger checks the firmware version of the DT80 against the latest version that it can see in the **C:\Program Files\dataTaker\DeLogger\Firmware\dt80** directory and offers to initiate a firmware upgrade if it finds that a later version is available. You should ensure you have properly prepared the DT80 (as described in Recommended Preparation (P172) above) before allowing either this automatic upgrade to occur, or when using the **Upgrade Firmware...** menu option to initiate the upgrade process, as described in the following steps:

1. Obtain the appropriate firmware ("Flashware") upgrade zip file from **www.dataTaker.com** or your *dataTaker* representative. Unzip the files in the zip file and place them where the host computer can access them on the computer's hard disk or a network drive.  
  
Placing them in the directory **C:\Program Files\dataTaker\DeLogger\Firmware\dt80** is a good idea because DeLogger looks in this directory by default.  
  
The firmware upgrade file is named according to its firmware version number and has a .DXF extension — for example, **DT80-5080002.dxf**. The size of a typical .DXF file is 1 to 2MB.  
  
If you have a choice of upgrade files, always use the latest (the one with the highest number) unless you've been directed to do otherwise.
14. Power the DT80 as described in Recommendation — Power above.
15. Connect the host computer to the DT80 using USB or RS232.
16. Start DeLogger and check that you're using version 2 revision 16 or later (from DeLogger's Help menu, choose About DeLogger...).  
  
Alternatively, start DeTransfer and check that you have version 3.18 or later.
17. In DeLogger, choose Upgrade Firmware from the dataTaker menu.  
  
In DeTransfer, choose Upgrade Firmware(DT80/800) from the File menu.
18. In the dialog box that opens, select the serial port that will be used to access the DT80 during the upgrade and then click OK.
19. In the navigation dialog box that opens, locate the firmware upgrade file (**DT80-5080002.dxf**, for example), highlight it (one click) and click Open.  
  
DeLogger looks in the directory **C:\Program Files\dataTaker\DeLogger\Firmware\dt80** by default. You can select files in other directories using this dialog box if necessary.
20. The upgrade process will now proceed automatically. The DT80's LEDs will flash and the LCD display should indicate that the upgrade is progressing. You will notice that the upgrade proceeds through three phases:
  - a) A special "loader" program is downloaded (**Attn** LED flashes and display shows: **DT80 Bootstrap**)
  - b) The main firmware is downloaded (**Sample** LED flashes and display shows: **DT80 Loader**)
  - c) A firm reset is performed (display shows the normal "sign-on" screen)
- Important** During the upgrade, do not remove any cables, or reset or power-down the DT80.
21. Once the upgrade is complete, check that the version number displayed on the sign-on screen is correct. For example if the file **DT80-5080002.dxf** was loaded then the display should indicate **DT80 V5.08**.
22. Connect to the DT80 and configure it with your preferred settings and programs.  
  
The upgraded DT80 is now ready for use.

**Note** The above procedure can also be used to revert back to an earlier version of the firmware, should that be required.

## In Case of a Failed Upgrade

In the unlikely event that something goes wrong during an upgrade (eg. power to the DT80 or host computer is lost, or the firmware file is corrupted), use the following recovery procedure.

1. Reset the DT80 by inserting a paper clip or similar into the reset hole (Figure 45 DT80 Side Panel [\(P125\)](#))
23. If the old firmware starts correctly, simply repeat the above upgrade procedure.
24. If the firmware does not start correctly (ie. the normal sign-on display is not shown and the DT80 does not respond to commands) then *hold down* the **Edit/OK** key and reset the DT80. The **Attn** LED should now be flashing slowly (5s on, 5s off) and the display should show **DT80 Bootstrap**. Release the **Edit/OK** key.
25. At this point you should now be able to repeat the above upgrade procedure.

# ERROR MESSAGES

The DT80 returns a message when it detects an error in a command, or an operational difficulty. The form of the error report is controlled by the `/U` switch. The default is the verbose form shown in the table below. If the switch is set to `/u` the error message is reduced to an error number (e.g. `E3`). (Note this Switch also reduces the verbosity of other returned data).

Error messages can be switched off by the `/m` switch. The default is for errors to be reported (`/M`). During a data retrieve operation, error reporting is disabled until the unload is complete [\(P73\)](#)

If an error is detected during job entry (ie. between **BEGIN** and **END**) then the remainder of the job is ignored.

Errors that are a result of reading a channel cause a special “error” value (eg. `99999.9`, see also `x`) to be returned or logged as the reading.

The table below lists all of the DT80 errors, along with an explanation of their likely cause and/or correction.

Error Number and Description Cause/Action	Error Category				
	Syntax	Operation	Memory	Reading	Hardware
<b>E1 - Time set error</b> <ul style="list-style-type: none"> <li>• Must be in format defined by P39 and P40</li> <li>• Illegal separator or non-digits entered</li> </ul>	☑				
<b>E2 - Command line too long</b> <ul style="list-style-type: none"> <li>• Command too long (maximum 250 characters)</li> </ul>		☑			
<b>E3 - Channel option error</b> <ul style="list-style-type: none"> <li>• Illegal channel option used</li> <li>• Incompatible options used</li> <li>• Option invalid for channel type</li> </ul>	☑				
<b>E7 - Day set error</b> <ul style="list-style-type: none"> <li>• Illegal day number entered</li> </ul>	☑				
<b>E8 - Parameter read/set error</b> <ul style="list-style-type: none"> <li>• Parameter index out of range</li> <li>• Parameter value out of range</li> </ul>	☑				
<b>E9 - Switch error</b> <ul style="list-style-type: none"> <li>• Illegal switch command character</li> </ul>	☑				
<b>E10 - Command error</b> <ul style="list-style-type: none"> <li>• Unrecognised keyword</li> </ul>	☑				
<b>E12 - Channel list error</b> <ul style="list-style-type: none"> <li>• Channel number outside the legal range</li> <li>• Incomplete channel sequence</li> <li>• Invalid channel type</li> <li>• Polynomials or spans index out of range</li> </ul>	☑				
<b>E18 - STATUS command error</b> <ul style="list-style-type: none"> <li>• STATUSn outside the range 1 to 14</li> </ul>	☑				

Error Number and Description Cause/Action	Error Category				
	Syntax	Operation	Memory	Reading	Hardware
E23 - Scan schedule error <ul style="list-style-type: none"> <li>Schedule ID not A-K, X or S</li> <li>Scan time interval too large</li> <li>Scan interval type invalid</li> <li>Event or counter channels invalid</li> </ul>	<input checked="" type="checkbox"/>				
E24 - Unload command error <ul style="list-style-type: none"> <li>Schedule ID is not one of A-K or X</li> </ul>	<input checked="" type="checkbox"/>				
E25 - Channel table full <ul style="list-style-type: none"> <li>Internal acquisition and alarm table filled (maximum 800 entries). Additional channels cannot be declared</li> </ul>			<input checked="" type="checkbox"/>		
E29 - Poly/span declaration error <ul style="list-style-type: none"> <li>Polynomial or span index out of range (1 to 50)</li> <li>Individual terms not separated by a comma</li> <li>Range of terms outside 1.0e-18 to 1.0e18</li> </ul>	<input checked="" type="checkbox"/>				
E30 - Analog measurement fault <ul style="list-style-type: none"> <li>Faulty circuit board, circuit board connector, or circuit board power supply</li> <li>If fault persists after a hard reset contact your <i>dataTaker</i> representative.</li> </ul>					<input checked="" type="checkbox"/>
E32 - Job not found <ul style="list-style-type: none"> <li>The named <i>job</i> cannot be found.</li> </ul>	<input checked="" type="checkbox"/>				
E37 - No current job <ul style="list-style-type: none"> <li>A command was entered that operates on the current <i>job</i>, but there is no <i>job</i> currently loaded.</li> </ul>	<input checked="" type="checkbox"/>				
E39 - Channel list fixed /F <ul style="list-style-type: none"> <li>Changes to program are not allowed</li> </ul>		<input checked="" type="checkbox"/>			
E42 - USB device not ready <ul style="list-style-type: none"> <li>No USB memory device inserted</li> <li>DT80 has not yet read the required system information from the device (wait a few seconds)</li> <li>USB device is faulty or not a memory device</li> </ul>		<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
E50 - Job locked <ul style="list-style-type: none"> <li>A <i>job</i> that has been locked cannot be modified</li> </ul>		<input checked="" type="checkbox"/>			
E51 - ALARM/IF command error <ul style="list-style-type: none"> <li>Setpoint character &lt;, &gt;, &lt;&gt; or &gt;&lt; missing</li> <li>AND, OR, XOR incorrectly entered</li> <li>Setpoint not specified or too large</li> <li>Delay incorrectly specified</li> </ul>	<input checked="" type="checkbox"/>				
E52 - Alarm text memory full <ul style="list-style-type: none"> <li>Memory for storage of alarms text is full (total is 16384 characters, shared with expressions text)</li> </ul>			<input checked="" type="checkbox"/>		
E54 - Expression error <ul style="list-style-type: none"> <li>Syntax error</li> <li>Expression too complex</li> </ul>	<input checked="" type="checkbox"/>				
E55 - Expression memory full <ul style="list-style-type: none"> <li>Memory for storage of expressions text is full (total is 16384 characters, shared with alarms text)</li> </ul>			<input checked="" type="checkbox"/>		
E65 - ALARM undefined <ul style="list-style-type: none"> <li>Alarm <i>n</i> does not exist</li> </ul>	<input checked="" type="checkbox"/>				
E74 - Serial sensor CTS detect timeout <ul style="list-style-type: none"> <li>Serial sensor: CTS did not go to the required state</li> </ul>				<input checked="" type="checkbox"/>	

Error Number and Description Cause/Action	Error Category				
	Syntax	Operation	Memory	Reading	Hardware
within timeout period					
E89 - Serial sensor receive time out <ul style="list-style-type: none"> <li>Serial sensor: expected characters were not received within timeout period</li> </ul>				<input checked="" type="checkbox"/>	
E90 - Serial sensor scan Error <ul style="list-style-type: none"> <li>Serial sensor: could not convert the received text as specified in the control string</li> </ul>				<input checked="" type="checkbox"/>	
E98 - Flash writing error <ul style="list-style-type: none"> <li>ONRESET job too large (max 16k)</li> <li>Flash memory faulty</li> </ul>			<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>
E104 - Drive format failed <ul style="list-style-type: none"> <li>Unable to format the specified drive. It may be write-protected or damaged.</li> </ul>		<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
E106 - Pn(s) in USER.INI out of range <ul style="list-style-type: none"> <li>An out-of-range parameter has been specified in the PROFILE. It must be changed for correct operation.</li> </ul>	<input checked="" type="checkbox"/>				
E107 - Counter is already used as a trigger <ul style="list-style-type: none"> <li>The specified counter is already used as a schedule trigger and cannot be used again.</li> </ul>		<input checked="" type="checkbox"/>			
E109 - File IO error: <i>detailed description</i> <ul style="list-style-type: none"> <li>An error occurred while reading or writing to a file on one of the drives (A: or B:). The detailed description will contain exact details of the error type. For example, a write-protected file cannot be written to.</li> </ul>		<input checked="" type="checkbox"/>			
E113 - Schedule option error <ul style="list-style-type: none"> <li>Invalid schedule option specified</li> </ul>	<input checked="" type="checkbox"/>				
E114 - Command parameter error <ul style="list-style-type: none"> <li>Invalid parameter specified for a command</li> </ul>	<input checked="" type="checkbox"/>				
E115 - Serial sensor string error <ul style="list-style-type: none"> <li>Invalid syntax within serial sensor control string</li> </ul>	<input checked="" type="checkbox"/>				
E116 - Cannot log: <i>detailed description</i> <ul style="list-style-type: none"> <li>The DT80 cannot log data for one or more schedules for the indicated reason</li> <li>If the problem was that an existing job of the same name had logged data/alarms then you need to give the new job a different name, or delete the old job's data using DELDATA/DELALARMS</li> </ul>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>
E117 - Incompatible schedule store units and trigger <ul style="list-style-type: none"> <li>You can only specify storefile size by time (eg 12 hours data) if the schedule trigger is time based.</li> </ul>	<input checked="" type="checkbox"/>				
E118 - Error accessing drive x: <i>detailed description</i> <ul style="list-style-type: none"> <li>The indicated drive (A: or B:) could not be accessed</li> <li>Media may be absent, not formatted or faulty.</li> </ul>		<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>

Table 18: DT80 Error Messages



# Glossary

## 4–20mA loop

A common industrial measurement standard. A transmitter controls a current in the range of 4 to 20mA as a function of a measurement parameter. Any receiver(s) or indicator(s) placed in series can output a reading of the parameter. Main advantage is 2-wire connection and high immunity to noise pick-up. Usually powered from a 24V supply.

## 50/60Hz rejection

The most common source of noise is that induced by AC power cables. This noise is periodic at the line frequency. DT80s are able to reject most of this noise by integrating the input for exactly one line cycle period (20.0 or 16.7ms).

## Ω

ohm, a unit of resistance

## μA

microamp,  $10^{-6}$  A

## μs

microsecond,  $10^{-6}$  s

## μStrain

microstrain, strain expressed in parts per million (ppm). Strain is a measure of the stress-induced change in length of a body.

## μV

microvolt,  $10^{-6}$  V

## A

Ampere or amp, a unit of current

## actuator

A device that converts a voltage or current input into a mechanical output.

## ADC

Analog-to-Digital Converter. Part of the DT80's input circuitry that converts an analog input voltage to a digital number (in other words, it converts a smoothly-varying signal to a quantised digital value). The DT80 is a digital instrument, and therefore requires an ADC to convert analog sensor signals into digital form prior to processing. Important characteristics of an ADC are its linearity, resolution, noise rejection and speed.

## ADC settling time

See channel settling time ([P178](#)).

## Ah

Ampere-hour, a unit of electrical charge, often used when referring to battery capacity

## analog

a quantity that can vary continuously through a potentially infinite number of values — for example, the time swept out by the hands of a clock, or the output of a thermocouple. Compare with digital.

## append

To add a new record or data to the end of a file, database, string or list.

## ASCII

American Standard Code for Information Interchange. A coding system designed for standardising data transmission to achieve hardware and software compatibility. It assigns a 7-bit code to each of the 128 standard characters: 96 visible characters — letters, numbers and punctuation marks (including the space character); 32 hardware control characters — sounding a bell, advancing a printer page, carriage return, line feed and so on. See Table 16 ([P169](#)).

## asynchronous

Not synchronised, not occurring at pre-determined or regular intervals. A telephone conversation is asynchronous because both parties can talk whenever they like. In an asynchronous communications channel, data is transmitted intermittently rather than in a constant stream.

## autoranging

The process of changing amplifier gain automatically so that the signal is amplified as much as is possible without exceeding output limits.

## autozeroing

A stabilization method for removing errors due to a drift in the input offset of a measuring system. Also called **zero correction**.

## base date and time

The DT80's base date is 0:00:00 on 01/01/1989. All timestamps are stored as offsets from this point in time.

## bit

The smallest unit of information in a computer. A bit has a single value: either 0 or 1. Computers generally store information and execute instructions in bit-multiples called **bytes**.

## brackets and braces

Delimiters:

( )	Round brackets (parentheses)	Used to identify channel options, eg. <code>1V("Flow Rate",Y1)</code> group terms within expressions, eg. <code>1CV=3.14*(2CV+3CV)</code>
[ ]	Square brackets	Used to enclose channel variable to be used within a serial prompt or parse command, eg. <code>1SERIAL("%f[17CV]")</code>
{ }	Braces	Used to enclose channels and commands to be conditionally executed within <b>ALARM</b> and <b>IF</b> statements, eg. <code>ALARM1(2CV&gt;3){1CV=1CV+1 HB}</code> signify output actions in the serial sensor command, eg. <code>1SERIAL("{MD004}%f[1CV]")</code>

## bps

bits per second, a measure of data transfer rate

## bridge

A sensitive and stable means of measuring small changes in resistances. They are particularly useful when applied to strain gauges (as found in pressure sensors and load cells). See Bridges ([P138](#)).

## buffer

An area of memory where data is held temporarily until the system is ready for it, or in case it is needed in the future.

## byte

A unit of information that is eight bits long

## carriage return

Also known as a **return**, usually abbreviated to **CR**. The DT80's default is to suffix each scan's data with a carriage return (see P24 ([P110](#))). Symbol: ↵

In the ASCII character set, a carriage return has a decimal value of 13.

## channel definition

A channel's ID followed by any channel options (in round brackets). See *Figure 6* ([P39](#)).

## channel ID

A channel's number and type (eg. **3TK**). See *Components of a typical schedule command* ([P39](#)).

## channel list

A list of channel definitions within one report schedule.

## channel settling time

The time allowed for the input signal to the ADC to stabilise before it is measured. This can be controlled using the measurement delay (MDn) channel option.

## channel table

An internal DT80 data structure that stores details of all defined channels. The channel table is limited to a maximum of 800 entries.

A channel table entry is used each time a channel is referenced in the current job. For example, the job `RA10S T 4V 1CV(W)=1CV+1 ALARM2(1CV>10) "boo"{1DSO=0}` uses 5 channel table entries (for `T`, `4V`, `1CV`, `1CV` and `1DSO`).

### clock

The DT80 a real-time clock/calendar, which you can set to your actual time

### CMRR

Common-Mode Rejection Ratio. A measure of the influence of common-mode voltage (unwanted) on the output of the DT80's instrumentation amplifier (see common-mode voltage [\(P179\)](#) below).

More precisely, CMRR is the ratio of the common-mode voltage at the amplifier's input to the common-mode voltage at the amplifier's output, expressed in dB. It indicates the quality of a measuring system's input electronics. Relevant to basic (differential) inputs only.

$$CMRR = 20 \log \left( \frac{V_{CM}}{V_{out} \times A_V} \right)$$

where

$V_{CM}$	is an applied common-mode voltage
$V_{out}$	is the resulting output voltage
$A_V$	is the amplifier's voltage gain

### command line

One or more DT80 commands typed one after the other, separated by tab or space characters, and ending with a return character. Limited to a maximum of 250 characters (including spaces, tabs, underscores,...). For example

`RA10S T 4V 5TK.`

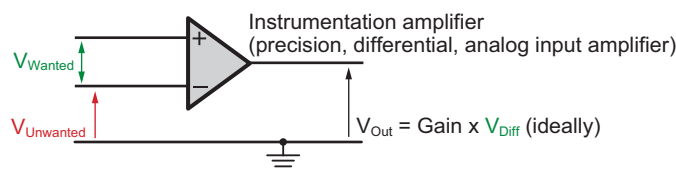
is a command line made up of four DT80 commands (separated by spaces).

### common-mode voltage

An unwanted AC and/or DC voltage that offsets both inputs to the DT80's instrumentation amplifier (with respect to amplifier ground). It is unwanted because it usually originates from nuisance sources such as electrical noise, DC offset voltages caused by the sensors or the equipment being measured, or from ground loops.

Typically in industrial measurement, the sensor signals you apply to the DT80's input terminals consist of

- the small component you want to measure (a few mV to a few tens of mV), PLUS
- a large unwanted component (a few V to a few tens of V) — the **common-mode voltage**.



$$V_{Unwanted} = V_{Common-mode} (V_{CM})$$

$$V_{Wanted} = V_{Differential} (V_{Diff})$$

Figure 75: Common-mode voltage  $V_{CM}$  and Differential voltage ( $V_{Diff}$ ) - 1

When the DT80 makes a measurement, both of these components are applied to the inputs of the its instrumentation amplifier. Then, when configured for basic (differential) use, the amplifier does two things:

- It rejects most of the common-mode voltage (the unwanted signal). How well the amplifier does this is indicated by its common-mode rejection ratio — see CMRR [\(P179\)](#).
- It amplifies the difference between the signals on its two inputs. This is the wanted signal and is called the **differential voltage** — see differential voltage [\(P180\)](#).

Common-mode voltage is calculated as the average of the voltages between the measurement system's ground and the two input terminals:

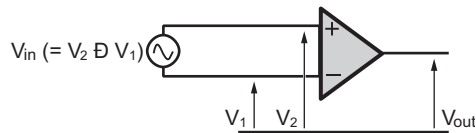


Figure 76: Common-mode voltage VCM and Differential voltage (VDiff) — 2

## CR

See carriage return ([P178](#)).

## crest factor

The peak-to-RMS voltage ratio of an AC signal (Crest factor ([P180](#))).

A pure sine wave has a crest factor of 1.414. If the crest factor is less than 1.4, the waveform is flattened; if the crest factor is greater than 1.4, the waveform is peaked.

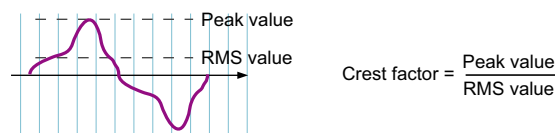


Figure 77: Crest factor

## DAC

Digital-to-Analog-Converter

## data acquisition system

A measurement system that scans a range of analog and digital channels, converts the readings to digital format, and forwards the data to a host. The host does any storage or data manipulation required. See also logging ([P182](#)).

## data logging system

A data acquisition system with its own on-board data storage and manipulation facilities. See also logging ([P182](#)).

## dataTaker

The name of the family of stand-alone data logging, acquisition and associated equipment manufactured by *dataTaker* (Aust.) Pty Ltd.

*dataTaker* releases:

1983 *dataTaker* DT100

1987 *dataTaker* DT200

1990 *dataTaker* DT500 series, DT600 series, and the DT50

2000 *dataTaker* DT800

2005 *dataTaker* DT80

## DCE

Data Communications Equipment. A DCE device (a modem, for example) enables a DTE device (such as a computer or a DT80) to communicate over phone lines or data circuits. A DCE device connects to the RS-232 interface of a DTE device. See DTE ([P181](#)).

## default

An attribute, value or option that is assumed if none is explicitly specified. A state or group of operating conditions (determined by the manufacturer and factory-set) to which the DT80 automatically reverts after a reset.

## differential input

An analog input where the difference between two voltages is measured, without reference to ground or any other common point. For example the **1V** command measures the differential voltage between the 1+ and 1- terminals.

## differential voltage

The difference between the voltages on the two inputs of the DT80's instrumentation amplifier (the *dataTaker's* precision, differential, analog input amplifier). See common-mode voltage ([P179](#)).

## digital

a quantity that is represented by a number that has a finite number of possible values. The number of bits used to store a digital value determines the resolution, ie how close two values can be and still be resolved (distinguished). Some quantities are inherently digital, eg. a logic signal or switch (whose state can be represented by 1 bit)

## direct commands

Commands that perform direct tasks within the DT80 the moment they are sent (for example, switch, parameter, unload, alarm, job and delete commands).

## directory

an area on a data storage device used to store related files. Also known as a *folder*.

## DTE

Data Terminal Equipment. The information source and/or destination in an RS-232 communications link. The DT80's Host RS-232 port and Serial Channel are DTE devices, as is a PC's RS-232 port (serial port).

The RS232 standard was originally designed for connecting a DTE to a DCE (eg a modem). However, a DTEs can also be directly connected to another DTE by means of a null-modem cable

## echo

A communications option for commands you send to the DT80. When echo is turned on see Table 10: DT80 Switches [\(P113\)](#), commands you send to the DT80 are automatically returned to the host computer screen.

Echo is useful for troubleshooting: when the echo is on, you can see by the returned commands that the DT80 is actually receiving them. (Once you're confident that it is receiving, you can turn the echo off.) Also, any error message appears right under the echo of the erroneous command, making the error obvious.

## EEPROM

Electrically-Erasable Programmable Read-Only Memory. A special type of PROM that can be erased by exposing it to an electrical charge. Requires data to be written or erased one byte at a time (compare with Flash [\(P181\)](#) below). Retains its contents even when power is unavailable. See also MEMORY [\(P125\)](#).

## enable

Turn on or activate

## Ethernet

A standard method for connecting a network of computers so that they can share information. The DT80 supports "10 Base-T" Ethernet, that is it operates at a data rate of 10Mbps and uses Twisted-pair cable. See DT80 ETHERNET COMMUNICATIONS [\(P125\)](#).

## firmware

The "operating system" software stored inside the DT80. The DT80's firmware is semi-permanent, and you can upgrade it with a simple file transfer.

## Flash

A special type of EEPROM that can be erased and reprogrammed in blocks (instead of one byte at a time — compare with EEPROM [\(P181\)](#) above). Flash memory is therefore much faster to erase and re-write. Retains its contents even when power is unavailable. The DT80's firmware is stored in Flash memory. See also MEMORY [\(P125\)](#) and UPGRADING DT80 FIRMWARE [\(P172\)](#).

## flow control

The process of controlling the flow of information between communications devices. For example, if data is being sent too quickly from a DT80 to its host computer, the computer tells the DT80 to temporarily stop sending data; then when the computer has caught up, it tells the DT80 to resume sending data. See Flow Control [\(P98\)](#). Hardware handshaking (hardware flow control; RTS/CTS) and software handshaking (software flow control; XON/XOFF) are alternative mechanisms of flow control.

## folder

Another name for **directory**

## format

A specific way of organising related information. For example, the DT80's internal data memory is formatted as a DOS/Windows compatible file system.

## FTP

File Transfer Protocol. A TCP/IP protocol for copying files from one computer to another.

## ground

A common return path that is the zero voltage reference level for the equipment or system. It may not necessarily be connected to earth.

## ground loops

More often than not, grounds in a measurement system are not at the same electrical potential — differences may be from microvolts to many volts. Then, if signal wires happen to connect different grounds together, currents can flow and result in unpredictable measurement errors. These unintended conduction paths are referred to as **ground loops**. The DT80 has been designed for maximum immunity to ground loops — see DT80 Analog Sub-System [\(P141\)](#).

## guard

An actively-driven shield around input signal conductors that is maintained at the common-mode voltage of the input signal. Signal guarding is used when a sensor has a high output impedance and the cable's capacitance and insulation leakage are significant.

## host computer

The computer you use for supervising the DT80

## host software

The software you run on the host computer to supervise the DT80. See DT80-Friendly Software [\(P11\)](#).

## hunting

An undesirable oscillation

## HWFC

Hardware flow control (RTS/CTS). Also known as **hardware handshaking**. See flow control [\(P181\)](#).

A device using hardware flow control monitors its Clear To Send (CTS) input and will not send data until the signal is active. Conversely, a device indicates that it can receive data by driving its RTS output active (which is then connected to the other device's CTS input)

## Hz

Hertz, a unit of frequency

## instrumentation amplifier

A precision differential amplifier for amplifying the DT80's analog input signals (wanted) and rejecting any common-mode voltage (unwanted). See Figure 47 [\(P141\)](#).

## IP address

A device's address on a TCP/IP Ethernet network. Every device connected to an Ethernet network must be assigned its own unique IP Number. An IP address is written as four decimal numbers eg. 192.168.1.209

## ISO

International Organization for Standardisation

## job

A logical "hold-all" for a group of schedules and other commands, and related data and alarms. Each job has a name and a directory structure that organizes this information. See Jobs [\(P19\)](#).


## kB

kilobyte, 1024 bytes

## kbps

kilobits per second, 1024 bps

## kelvin sense point

A particular point in a measurement circuit where a measurement should be made to ensure the best possible accuracy by ensuring that unwanted voltage drops due to current flows are minimized. Symbol 

## LED indicator

Light-emitting diode indicator. The DT80 has three LEDs on the front panel, which light to indicate Sampling, Internal Disk Activity, and Attention Required. See [\(P90\)](#) for more details.

## logging

Recording or storing data. The DT80 logs data to its internal memory and/or an external USB memory device. Logging is a separate, user-configurable operation that the DT80 performs in addition to its basic function of data acquisition (taking measurements from sensors connected to its inputs). See also data logging system [\(P180\)](#) and data acquisition system [\(P180\)](#).

**lsb**

least significant bit (within a byte)

**LSB**

Least Significant Byte (within a multi-byte word)

**mΩ**

milliohm,  $10^{-3} \Omega$

**mA**

milliamp,  $10^{-3} \text{ A}$

**MB**

megabyte, 1048576 bytes

**Mbps**

Megabits per second

**monolithic sensors**

Also called **IC** (Integrated Circuit) sensors. Sensors that are constructed on a single piece of silicon using integrated circuit fabrication techniques. Available sensors include those for measuring temperature (see IC Temperature Sensors [\(P137\)](#)), pressure, acceleration and concentration of various compounds in gases and liquids.

**ms**

millisecond,  $10^{-3} \text{ s}$

**msb**

most significant bit (within a byte)

**MSB**

Most Significant Byte (within a multi-byte word)

**multidrop**

In communications, a multidrop configuration allows multiple devices to be connected in parallel by means of a single twisted-pair cable. This requires that each device switch off (tri-state) its transmitter when it is not actively transmitting.

**multiplexer**

A “many-in, one-out” switching network that allows many input signals to time-share one analog input circuit. It sequentially routes multiple channels to a single signal processing system.

**noise**

Unwanted voltage or current (generally with an AC component) superimposed on the wanted signal.

**null-modem cable**

A communications cable for connecting two DTE devices together (for example, a PC to another PC, or a DT80 to a PC) [\(P171\)](#). Also known as a **crossover cable**.

**nybble**

Half a byte (four bits)

**parse**

To identify components of a command string

**PC**

A personal computer of the IBM or IBM-compatible type. (Although the Macintosh is technically a PC, it is not referred to as such.)

**PCB**

Printed Circuit Board

**peak-to-peak**

The value of an alternating quantity measured from its negative peak to its positive peak.



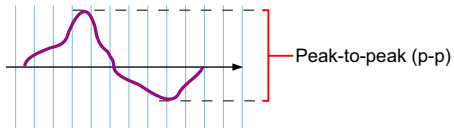


Figure 78: Peak-to-peak measurement

### period

The time taken for a cyclic event to repeat itself. Reciprocal of frequency:

$$Period = \frac{1}{Frequency}$$

### PID

Proportional, Integral, Derivative. A three-mode control algorithm commonly used in industrial control. A PID control loop automatically adjusts its response according to how close the measured value is to the target value. Deals with system hysteresis more effectively than simple on/off controls.

A PID loop with two-state output can be programmed on the DT80 using the difference, integration and calculation facilities.

### PLC

Programmable Logic Controller. Used to automate monitoring and control of industrial equipment.

### plug-and-play

A device whose characteristics are automatically determined when it is plugged in. All USB devices are plug-and-play.

### polling

Requesting information

### port

A plug, socket or interface that enables connection to another device for information transfer. For example, the DT80 has three ports for communicating with a host computer: Ethernet, USB and RS232.

### PPP

Point-to-Point Protocol. A low-level protocol that allows TCP/IP based protocols to be used over an RS232 connection.

### process list

The part of a schedule command that follows the schedule header and trigger, and lists the processes you want the schedule to carry out. It may include, for example, a channel list and an **IF** command.

### program

A DT80 program is a group of one or more jobs or commands that you send to the DT80.

### protocol

The language (or set of rules) that devices use to communicate over a network. For the Information Superhighway, think of protocols as the "rules of the road". All devices on a network must use the same protocol to communicate with each other. See TCP/IP ([P186](#)).

### RAM

Random Access Memory. Memory that allows the storage locations within it to be accessed (written to or read from) directly (non-sequentially). This characteristic makes RAM very fast. Often simply called **memory**. See MEMORY ([P125](#)).

### RAM disk

An area of RAM configured by a software program to emulate a disk drive.

### real-time

As it happens. The DT80 can return data directly to the host computer in real time — that is, as each scan is made, its resulting data is returned to the host computer straight away and displayed on-screen immediately.

### resolution

The smallest detectable increment of measurement — that is, the smallest change in input that produces a detectable change in output. In the field of data acquisition, resolution is the number of bits that the ADC uses to represent the analog signal — the greater the resolution, the smaller the changes in input signal that can be resolved/detected.

### retrieve

To unload or return data and other information from the DT80 to the host computer, either by:

- Unloading through one of the DT80's communications interfaces
  - Unloading by temporarily inserting a USB memory device into the DT80
- See [RETRIEVING LOGGED DATA \(P73\)](#).

## return

See carriage return [\(P178\)](#).

## ROM

Read Only Memory. Memory that can be randomly read from but not normally written to. The DT80 uses flash ROM.

## RS-232

A common communications and interface standard for connecting serial devices.

RS232 uses a negative voltage (typically  $-5V$ ) to represent a logic "0" and a positive voltage (typically  $+5V$ ) to represent a logic "1". These signals are with respect to a common ground terminal, hence RS232 is said to use *single-ended* signalling.

## RS-422

Another communications interface standard for connecting serial devices. RS-422 uses *differential* signalling (a pair of wires for each signal, no signal-ground connection) which provides improved noise immunity and allows operation over longer distances than RS-232.

## RS-485

Yet another communications interface standard. Like RS-422, RS-485 uses differential signalling. RS-485 is designed for multi-drop operation over a single shared pair of wires.

## RTD

Resistance Temperature Detector. A resistive sensor that changes resistance with changes in temperature. See RTDs [\(P137\)](#).

## sampling speed

The maximum rate at which analog-to-digital conversions can be done. This includes any channel selection time, settling time (for the signal to stabilise) and processing time (if required).

## SCADA

Supervisory Control and Data Acquisition. SCADA systems are used to monitor and control plant status and provide data logging facilities.

## schedule

Full name: **scan schedule command**. A scan that automatically triggers whenever specified condition(s) and/or event(s) occur. For example, whenever 5 seconds have elapsed (repeating every 5 seconds), whenever a door closes (scan on digital event), or whenever an alarm occurs. This is the command you'll send to the DT80 most often. There are several flavours of schedule command.

## schedule header

The schedule's ID and trigger, eg. **RA1S** — see [Figure 6 \(P39\)](#).

## serial

One by one. In serial data transfer, data is sent in a single stream of bits, one bit at a time, one after the other. The opposite of serial is parallel. In parallel data transfer, several streams of bits are sent concurrently.

## settling time

The time allowed for an input signal to stabilise after the DT80 selects the channel, selects the gain, and applies excitation (if required). See channel settling time [\(P178\)](#).

## shared-terminal inputs

Analog inputs where a common reference is used. Also called single-ended inputs. For example, the **1\*V**, **1+V** and **1-V** commands all measure single-ended voltages relative to a common point (the 1# terminal)

See Shared-Terminal [\(P16\)](#).

## shield

A conductor surrounding input signal wires that is generally connected to a data *dataTaker's* ground. The purpose is to shield the input signal from capacitively-coupled electrical noise. Such a shield provides little protection from magnetically-induced noise.

## SRAM

Static Random Access Memory. An extremely fast and reliable type of RAM. "Static" derives from the fact that it doesn't need to be refreshed like other types of RAM. See [\(P125\)](#).

## S-Record

A printable ASCII format consisting of strings of hexadecimal digits; used for transferring binary data between computers.

## stand-alone

Not connected to a host computer. The DT80 is designed to operate in stand-alone mode: once programmed, you can disconnect the *dataTaker* from the host computer leaving the *dataTaker* operating totally independently. Later, to download data or reprogram the *dataTaker*, you reconnect the host computer.

## state

Of an alarm: the true/false result of an alarm test. The actual states or transitions recognized by the DT80 are

- continuing false
- false-to-true
- continuing true
- true-to-false.

See Alarm States and Tags [\(P85\)](#).

## SWFC

Software flow control (XON/XOFF). Also known as **software handshaking**. See flow control [\(P181\)](#).

A device using software flow control will stop transmitting if an XOFF character is received and will resume when an XON character is received.

## switch

Full name: **switch command**. A software control. A two-state (ON or OFF) command that changes a DT80 internal setting. For example, sending the switch command **/R** to the DT80 turns ON the return-of-data-to-the-host-computer switch, and sending **/r** turns it OFF. See Switches [\(P112\)](#) for a complete listing.

## syntax error

An error in the order, arrangement or spelling of the components of a command.

## TCP/IP

Transmission Control Protocol / Internet Protocol. A commonly-used family of communication protocols. TCP/IP protocols are used on the DT80's Ethernet interface, and can also be used on an RS232 link if PPP is enabled.

All TCP/IP protocols allow data to be transported across a local area network or the Internet.

## TCP

Transmission Control Protocol. TCP is the default TCP/IP protocol used by the DT80 to communicate over an Ethernet or PPP link.

TCP provides:

- flow control (prevents data being sent faster than it can be received)
- reliable data transfer (errors are detected and data is automatically re-sent)
- support for application protocols, such as e-mail and FTP

## thermocouple

A temperature-sensing device constructed from dissimilar metals. See Thermocouples [\(P134\)](#).

## transducer

A device that converts a physical parameter (temperature, for example) into an electrical voltage or current. It is usually a sensor with additional electronics for signal conditioning and scaling.

## UART

Universal Aynchronous Receiver/Transmitter. A hardware component that provides an RS232 serial interface. The DT80 uses two UARTs – one for the host RS232 port, one for the serial sensor.

## UDP

User Datagram Protocol. A component of the TCP/IP suite of protocols. UDP is a simple "connectionless" protocol that operates in a similar way to an RS232 link, except that the link can be across a LAN or the Internet.

Unlike TCP, UDP does *not* guarantee that all data will be delivered.

### unshared input

a differential input.

## USB

Universal Serial Bus. A standard method of connecting peripheral devices to a host computer. The DT80 operates both as a USB *device* (when talking to a host computer) and as a USB *host* (when talking to a USB memory device).

See USB COMMUNICATIONS ([P95](#))

### USB Memory Device

A memory device designed to be connected to USB. These devices can either be hard disk drives or flash memory devices. They are generally powered from the USB port.

## V

volt, a unit of electrical potential (voltage)

### version number

The version number of the DT80's firmware consists of a major number, a minor number and a build number — see Figure 79 ([P187](#)).

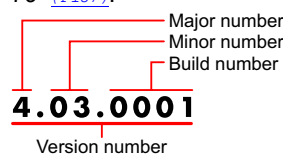


Figure 79: Version number components

## XON/XOFF

Transmitter on / transmitter off. Control characters used for software flow control (SWFC), instructing a device to start transmission (XON) and end transmission (XOFF).

## YSI

Yellow Springs Instruments — YSI Incorporated, 1725 Brannum Lane, Yellow Springs, Ohio 45387 (800 765-4974, 937 767-7241, Fax 937 767-9353, [www.ysi.com](http://www.ysi.com))

### zero correction

See autozeroing

# Index

/F	<a href="#">112</a>	4-wire BGV input	<a href="#">162</a>	Boolean Expressions	<a href="#">51</a>
/f	<a href="#">112</a>	4-wire LM135-series inputs	<a href="#">164</a>	brackets and braces	<a href="#">178</a>
/K	<a href="#">112</a>	6-wire BGV input	<a href="#">161</a>	Bridges	<a href="#">27, 35, 133, 133, 138, 138, 161, 161, 178</a>
/R	<a href="#">20, 113, 113</a>	6-Wire BGV Inputs	<a href="#">139, 161</a>	Buffer	<a href="#">178</a>
/S	<a href="#">113</a>	A[ ] Unload Command	<a href="#">88, 168</a>	Byte	<a href="#">178</a>
/Z	<a href="#">113</a>	Adaptive Scheduling	<a href="#">82</a>	Calculations (Expressions)	<a href="#">61, 63, 64, 113, 65</a>
25SV	<a href="#">32, 103</a>	Alarm Logging States	<a href="#">85</a>	Cautions	<a href="#">46, 110</a>
2SV	<a href="#">31</a>	Alarm Unload	<a href="#">88</a>	Changing a Schedule Trigger	<a href="#">48, 49</a>
3-wire resistance input	<a href="#">160</a>	Alarms	<a href="#">77, 77, 84, 84, 86, 86</a>	Channel Factor (f.f)	<a href="#">61</a>
4 Wire	<a href="#">35</a>	Analog Channels	<a href="#">15, 16, 157</a>	Channel Lists	<a href="#">41</a>
4–20mA Current Loops	<a href="#">133, 133</a>	Analog Sub-System	<a href="#">141, 182</a>		
4W	<a href="#">35</a>	ASCII	<a href="#">2, 170</a>		
4-Wire BGI Input	<a href="#">139, 162</a>	Bit	<a href="#">178</a>		

Channel Name	<a href="#">38</a>	Delete Commands	<a href="#">2, 71, 89, 166</a>	Firmware Upgrade	<a href="#">173</a>
Channel Numbers	<a href="#">25</a>	Deleting Logged Alarm Records	<a href="#">88</a>	Fixed-Format Mode (/H)	<a href="#">86, 81</a>
Channel Options	<a href="#">2, 17, 18, 32, 34, 38, 47, 56, 57, 61, 63, 111, 111, 161, 161</a>	Deleting Logged Data	<a href="#">19, 70</a>	Flash	<a href="#">181</a>
Channel settling time	<a href="#">177, 178, 185</a>	Deleting the Backup Files from Flash	<a href="#">115, 117</a>	Flow Control	<a href="#">98, 99, 181, 181, 182, 186</a>
Channel Types	<a href="#">2, 18, 29, 61, 130, 138, 140, 161, 161</a>	Differential input	<a href="#">180</a>	Fn	<a href="#">36, 61</a>
Channel Units	<a href="#">38</a>	Differential voltage	<a href="#">179, 180</a>	Format of Returned Data	<a href="#">21, 67, 110, 112</a>
Channel Variables	<a href="#">37, 44, 56, 63, 64, 64</a>	Digital Channels	<a href="#">17, 28, 43, 141</a>	Free-Format Mode (/h)	<a href="#">86, 81</a>
Clock/Calendar	<a href="#">118</a>	Digital Manipulation	<a href="#">36, 111, 111</a>	Frequency	<a href="#">27, 109, 133, 134</a>
CMRR	<a href="#">179</a>	Digital output	<a href="#">144</a>	Front Panel	<a href="#">90</a>
Combining Methods	<a href="#">61, 65</a>	Digital state input (1 bit/input)	<a href="#">28</a>	FTP Communications	<a href="#">108</a>
Common-Mode Voltage	<a href="#">179, 180</a>	Direct (local) connection	<a href="#">100</a>	FTP_SERVER	<a href="#">115</a>
Comms Wakes the DT80	<a href="#">100, 131, 131</a>	Direct (Local) RS-232 Connection	<a href="#">100</a>	Glossary	<a href="#">177</a>
Condition Tests	<a href="#">50</a>	Direct Connection	<a href="#">97, 100</a>	Ground Loops	<a href="#">22, 23, 136, 141</a>
Conditional Calculations	<a href="#">65</a>	Directory Structure	<a href="#">20</a>	Ground systems	<a href="#">141</a>
Conditional Processing	<a href="#">19, 50</a>	Directory Structure of USB memory devices	<a href="#">72</a>	Ground Terminals	<a href="#">141</a>
Configuration Line	<a href="#">37</a>	Disabling Data Logging	<a href="#">19</a>	Guard	<a href="#">182</a>
Constant-Current Excitation BGI	<a href="#">139</a>	Display	<a href="#">90</a>	Halting & Resuming Schedules	<a href="#">47, 49</a>
Continuous Report Schedules (No Trigger)	<a href="#">45</a>	DO... actionText	<a href="#">51</a>	Hardware Reset	<a href="#">119</a>
Control String – Input Actions	<a href="#">152</a>	DO... Command	<a href="#">109, 112</a>	Histogram	<a href="#">57</a>
Control String – Output Actions	<a href="#">151</a>	Download	<a href="#">2, 20</a>	Host Port	<a href="#">97, 97</a>
Controlling Sleep	<a href="#">110, 131</a>	DSR active (high)	<a href="#">97</a>	Host RS-232 Port Commands	<a href="#">103</a>
Crest factor	<a href="#">180</a>	DSR inactive (low)	<a href="#">97</a>	HOST_MODEM	<a href="#">101, 102, 102, 114</a>
data acquisition system	<a href="#">180, 182</a>	DTE	<a href="#">180, 181</a>	HOST_PORT	<a href="#">114</a>
data logging system	<a href="#">180, 182</a>	Echo	<a href="#">99, 181</a>	Hot Plugging	<a href="#">182</a>
Data storage	<a href="#">125</a>	EEPROM	<a href="#">181</a>	Humidity Measurement	<a href="#">140</a>
Data Storage Capacity — Readings/MB	<a href="#">69, 70</a>	Error Messages	<a href="#">2, 50, 56, 113, 174, 176</a>	Humidity Sensors	<a href="#">133, 139</a>
Date	<a href="#">37</a>	Ethernet	<a href="#">108, 181</a>	IC Temperature Sensors	<a href="#">133, 137, 183</a>
DCE	<a href="#">180</a>	Ethernet Commands	<a href="#">106</a>	IF	<a href="#">19, 50</a>
Default	<a href="#">114</a>	Ethernet Communications	<a href="#">104</a>	IF...THEN...ELSE	<a href="#">51, 51</a>
Default Value	<a href="#">109, 114</a>	Ethernet Setup	<a href="#">105, 106</a>	Immediate Report Schedules	<a href="#">46</a>
Delay	<a href="#">27, 27</a>	Excitation	<a href="#">35</a>	Immediate Schedule	<a href="#">40</a>
		EXT_POWER_SWIT CH	<a href="#">115</a>	Independent 2-wire AD590-series inputs	<a href="#">163</a>
		Extending Battery Life	<a href="#">129, 131</a>	Independent Analog Inputs	<a href="#">16, 158</a>
		Factory Defaults	<a href="#">114, 119</a>	Independent Current input	<a href="#">159</a>
		File Structure	<a href="#">72</a>	Independent Voltage Inputs	<a href="#">158</a>
		File System	<a href="#">69, 71</a>	INIT	<a href="#">101, 114</a>
		Firmware	<a href="#">172, 181</a>		

Input Actions	<a href="#">150</a>	Modem	<a href="#">102</a>	<a href="#">137, 138,</a>
Internal Maintenance	<a href="#">29, 30</a>	Communications		<a href="#">160</a>
Internal Memory	<a href="#">31</a>	Protocol		PT392
Internal	<a href="#">118, 130</a>	Modem Initialization	<a href="#">101, 102,</a>	<a href="#">137</a>
Memory-Backup			<a href="#">103, 114</a>	Rainflow Cycle
Battery		Modem power	<a href="#">102</a>	Counting
Internal Power (Main	<a href="#">129</a>	Modem Status	<a href="#">32</a>	Rainflow Data
Battery)		Mounting the DT80	<a href="#">128</a>	<a href="#">58, 59</a>
Intrinsic Functions	<a href="#">36, 61</a>	Multiple Reports	<a href="#">33, , 47, 56</a>	Real Time
(Fn)		Naming Channel	<a href="#">65</a>	<a href="#">184</a>
Isolation	<a href="#">23</a>	Variables		Repeating alarms
Issues	<a href="#">69</a>	NI	<a href="#">137</a>	<a href="#">77</a>
Job Commands	<a href="#">50</a>	No Logging	<a href="#">37, 69</a>	Resets
Jobs	<a href="#">19, 42, 48,</a>	No-Sleep Conditions	<a href="#">131</a>	<a href="#">2, 95, 166</a>
	<a href="#">182</a>	null-modem cable	<a href="#">183</a>	Resetting the DT80
Keys	<a href="#">90</a>	ONINSERT.DXC	<a href="#">73, 116</a>	<a href="#">71, 71,</a>
Labelling the End of	<a href="#">76</a>	ONRESET.DXC	<a href="#">116, 117</a>	<a href="#">103, 103,</a>
Unloaded Data		Optimal Speed	<a href="#">168</a>	<a href="#">104, 107,</a>
LED	<a href="#">182</a>	Output Data Format	<a href="#">37</a>	<a href="#">113, 113,</a>
LED indicator	<a href="#">182</a>	Parameters	<a href="#">2, 24, 37,</a>	<a href="#">119, 119</a>
LEDs and Messages	<a href="#">120</a>		<a href="#">53, 109,</a>	Resistance
After a Reset			<a href="#">112, 114,</a>	<a href="#">35</a>
LM135	<a href="#">28</a>		<a href="#">119, 168</a>	Resistance and
LM137	<a href="#">133</a>	Password Protection	<a href="#">95, 110</a>	Bridge
LM138	<a href="#">163</a>	— Comms Ports		Resistance Inputs
LM35-series	<a href="#">133, 163</a>	PH Configuration	<a href="#">98, 98</a>	<a href="#">160, 136,</a>
Locking Schedules	<a href="#">49</a>	Commands		<a href="#">137</a>
Logging	<a href="#">20, 69, 69,</a>	PLC	<a href="#">184</a>	Retrieval Commands
	<a href="#">180, 182</a>	Plug and Play	<a href="#">184</a>	<a href="#">73, 86, 166</a>
Logging Alarm States	<a href="#">85, 86</a>	Polled Report	<a href="#">45</a>	Retrieval Commands
Logging and	<a href="#">73, 84</a>	Schedule (RX)		— Summary
Retrieving Alarms		Polled Schedule	<a href="#">39</a>	Retrieving Logged
LOGON and	<a href="#">19, 49, 69,</a>	Polling Alarm Data	<a href="#">78, 78, 84</a>	Alarm States
LOGOFF Commands	<a href="#">85</a>	Polynomials (Yn)	<a href="#">36, 64,</a>	Retrieving Logged
Main Battery	<a href="#">129</a>		<a href="#">139, 62</a>	Data
Manual Reset Button	<a href="#">119</a>	Powering the DT80	<a href="#">129</a>	Returned Data
MAX_CD_IDLE	<a href="#">101, 115</a>	Powering the DT80's	<a href="#">101, 102,</a>	Format
Maximum	<a href="#">37</a>	Modem	<a href="#">103</a>	RISC
Memory	<a href="#">119, 125,</a>	PPP	<a href="#">108, 109,</a>	<a href="#">185</a>
	<a href="#">172, 181,</a>	Communications	<a href="#">114</a>	RS232
	<a href="#">184</a>	Profile	<a href="#">2, 24, 97,</a>	<a href="#">97, 97,</a>
Memory-backup	<a href="#">131, 131</a>		<a href="#">101, 102,</a>	<a href="#">100, 171</a>
battery			<a href="#">102, 109,</a>	RS-232 Pinouts
Modem	<a href="#">114, 172</a>		<a href="#">112, 113,</a>	<a href="#">2, 97, 171</a>
Modem (remote)	<a href="#">101</a>		<a href="#">113, 115,</a>	RTD
connections			<a href="#">115, 116,</a>	<a href="#">28, 133,</a>
Modem (Remote)	<a href="#">100</a>		<a href="#">119, 120</a>	<a href="#">137, 138,</a>
RS-232 Connection		program	<a href="#">184</a>	<a href="#">160, 185</a>
Modem and logger	<a href="#">103</a>	Programming	<a href="#">18</a>	SCADA
configuration		Protecting Startup	<a href="#">113, 113,</a>	<a href="#">185</a>
Modem Automatic	<a href="#">102, 103</a>	Files	<a href="#">116, 116</a>	Scaling
Baud Rate Selection		protocol	<a href="#">184</a>	<a href="#">19, 61</a>
Modem Cable	<a href="#">101, 103</a>	PT385	<a href="#">28, 28,</a>	Schedule
				<a href="#">185</a>
				Schedule Name
				<a href="#">40</a>
				Schedules
				<a href="#">19, 20, 39,</a>
				<a href="#">42, 178</a>
				Serial Channel
				<a href="#">43, 148,</a>
				<a href="#">157</a>
				Serial Channel
				Commands
				<a href="#">149</a>
				Serial Channel
				Commands
				<a href="#">155</a>
				Serial Channel
				Commands in
				Schedules
				Serial Channel
				Debugging Tools
				<a href="#">111, 156</a>
				Serial Channel
				Enable
				<a href="#">28</a>
				Serial Channel
				Examples
				<a href="#">156</a>

Serial Channel terminals (DTE)	<a href="#">148</a>	Characters		Trigger While	<a href="#">19, 41, 44</a>
SETDIALOUTNUMBER Command	<a href="#">98</a>	Switches	<a href="#">2, 48, 53, 80, 112, 113, 114, 118, 119, 122, 186, 186</a>	Triggering and Schedule Order	<a href="#">47, 49</a>
Setting the DT80's Time (T=)	<a href="#">118</a>	Synchronizing to Midnight	<a href="#">48, 113</a>	Triggers	<a href="#">45, 47, 177</a>
Setting Up a Direct Connection	<a href="#">100</a>	System Timers	<a href="#">27, 31</a>	U[ ] Commands	<a href="#">167</a>
Setting Up a Remote Connection	<a href="#">97, 103</a>	System Variables	<a href="#">2, 27, 31, 32, 70, 79, 103</a>	Unload	<a href="#">87</a>
Shared-Terminal	<a href="#">16, 158, 185</a>	TCP/IP	<a href="#">104, 184, 186</a>	Unconditional Processing — DO... Command	<a href="#">51</a>
Shared-terminal current inputs u	<a href="#">159</a>	Terminal Labels	<a href="#">15</a>	Unload Commands	<a href="#">2, 50, 73</a>
Shared-terminal voltage inputs	<a href="#">16</a>	Test	<a href="#">2</a>	Upgrading DT80 Firmware	<a href="#">20, 172</a>
Shared-Terminal Voltage Inputs	<a href="#">158, 158</a>	TEST	<a href="#">121</a>	USB	<a href="#">187, 187</a>
Single Quote	<a href="#">80, 99</a>	TEST Commands	<a href="#">121, 130</a>	USB Communications	<a href="#">95</a>
Single-shot alarms	<a href="#">77</a>	Test Report (DT80 Health)	<a href="#">108, 121</a>	USB Memory Device	<a href="#">20, 20, 69, 70, 71, 73</a>
Sleep	<a href="#">100, 110, 131, 131, 132</a>	Text	<a href="#">30</a>	USB memory device	<a href="#">73</a>
Sn	<a href="#">36, 62</a>	The Control String	<a href="#">149, 150</a>	USB Memory Device	<a href="#">125</a>
Span coordinates	<a href="#">62</a>	The U Unload Commands	<a href="#">74</a>	USB memory device Commands	<a href="#">125</a>
Spans (Sn)	<a href="#">36, 57, 62</a>	The U( ) Unload Commands	<a href="#">74, 74</a>	User Name	<a href="#">64</a>
SRAM	<a href="#">186</a>	The U[ ] Unload Commands	<a href="#">74, 75</a>	User Units	<a href="#">64</a>
S-Record	<a href="#">186</a>	Thermistor Scaling (Tn)	<a href="#">36, 63, 136</a>	USER.INI	<a href="#">117</a>
ST	<a href="#">27</a>	Thermistors	<a href="#">28, 63, 133, 136, 160</a>	Using an Alarm to Poll a Schedule	<a href="#">83</a>
Stand Alone	<a href="#">186</a>	Thermocouples	<a href="#">28, 133, 134, 135, 136, 158, 186</a>	Using Digital Outputs	<a href="#">44</a>
Startup Defaults	<a href="#">113</a>	Time	<a href="#">30, 37, 79, 111</a>	Variables	<a href="#">27, 37, 63</a>
Startup Files	<a href="#">113, 116, 117</a>	Time Triggers	<a href="#">48, 113</a>	VCM	<a href="#">180</a>
Startup Job	<a href="#">13, 20, 113, 116, 119</a>	Time Triggers — Synchronizing to Midnight	<a href="#">42</a>	Vdiff	<a href="#">180</a>
Statistical	<a href="#">47, 56</a>	Trigger on External Event	<a href="#">41, 43</a>	Version number components	<a href="#">187</a>
Statistical Report Schedules	<a href="#">46, 56</a>	Trigger on Internal Event	<a href="#">41, 43, 64</a>	Visits to Site	<a href="#">104</a>
Statistical Schedule	<a href="#">40</a>	Trigger on Schedule-Specific Poll Command	<a href="#">41, 44</a>	Voltage	<a href="#">26</a>
Statistical Sub-Schedule Halt/Go	<a href="#">47, 49</a>	Trigger on Time Interval	<a href="#">41, 42</a>	Voltage Excitation BGV	<a href="#">139, 161, 161</a>
Statistics	<a href="#">19, 37</a>			Voltage Inputs	<a href="#">, 133, 134, 157</a>
STATUS Commands	<a href="#">70, 122</a>			While condition	<a href="#">45</a>
STATUS14	<a href="#">49, 123</a>			Working Channels Hide CV Data	<a href="#">51, 64</a>
STATUS2	<a href="#">122</a>			Working with Schedules	<a href="#">48, 59</a>
Storage Capacity	<a href="#">70, 125</a>			Yn	<a href="#">36, 62</a>
Storage Status	<a href="#">70, 73</a>			YS01 to YS0730	<a href="#">28</a>
Strain Gauges	<a href="#">139</a>				
Substitution	<a href="#">80</a>				