dataTaker

dataTaker DT80 Range Code Examples

Updated 13/7/2020

ThermoFisher SCIENTIFIC

Table of Contents

Introduction

Example Programs are working examples produced with dataTaker DT80 series products that can be downloaded and utilized as a starting point for your own projects or applications.

Hardware: DT80 Series 1,2,3 & 4 loggers and CEM 20

For more information visit **thermofisher.com/datataker**

To clear DT80 range logger before sending a new program

Make sure you download all data, alarms, and program code before sending these commands.

Note: Data and alarms must be deleted before the job can be deleted.

Note: You must Halt the current job first, otherwise a discontinuity record will be recorded when attempting to define a new job as it halts the current job at that time. This in turn stops the new job from being defined as there is data in the store.

Using the DeLogger host software, you establish a connection to the data logger and then access the required commands from the 'DataTaker' drop down menu (delete options are found under 'clear'). For users of DeTransfer, the V7 DataTaker Web Interface, or other text-based user interfaces, send the commands as outlined below.

```
H           'Halt the current job
DELDATA*    'Deletes all data from all jobs.
DELALARM*   'Deletes all alarms from all jobs.
DELJOB*     'Deletes all jobs.
```

Version 7 Firmware introduced two new commands to make the clearing the data, alarms, and jobs easier.

```
H
DELALLJOBS  'Halt the current job and deletes all stored jobs and data including ONRESET job.*
DELVIDEOS   'Will delete the training videos to free up internal memory. *
```

Deleting all files and profile settings

If you need to remove all profile settings and reset jobs, then the DT80 range logger can be returned to factory defaults with the following commands. Note: This will not only delete any program and data files but also remove any error and event logs.

```
FACTORYDEFAULTS 'Sets DT80 to factory defaults
DELONRESET      'Deletes any onreset job
DELUSERINI      'Deletes any profile settings
DELALLJOBS      'Deletes all Jobs, data and alarm files.
```

Basic program structure

```
BEGIN”Job Name”

‘Put any initialization commands here.
‘These commands will be run when entered.
‘Put you span and polynomial definitions here.


RA(“B:”,ALARMS:OV:100KB,DATA:OV:1MB)1S
   ‘These store file parameters are the default and can be changed to suit
   ‘Put channel list for schedule A here

RB(“B:”,ALARMS:OV:100KB,DATA:OV:1MB)1S
   ‘Put channel list for schedule B here

RC(“B:”,ALARMS:OV:100KB,DATA:OV:1MB)1S
   ‘Put channel list for schedule C here and like wise for remaining schdules

LOGON
END
```

Low power operation

DT80 range low power

NOTE: The following example is relevant to firmware versions prior to version 6.20. The behavior of the LCD back light was changed in version 6.20 such that it only comes on after user key press, regardless of whether or not external power is applied.

The DT80 range is similar to the DT800 for most low power operations. One exception is the DT80/85 display back light; normally this will be ON when the DT80/85 is externally powered. To reduce power further, the back light and all LED activity can be turned off by the use of parameter P16. This parameter is primarily present for production test purposes but can be exploited to reduce power consumption. Normal setting of P16 is P16=0. This is normal operation of all LEDs and the display back light. Low Power setting would be P16=32. This will turn off all LEDs and back light. Caution is recommended as with P16=32; all LED activity including indication of sampling and error or attention conditions will be suppressed.

Adding appropriate commands to the immediate schedule will ensure that if the logger is reset, low power operation will operate. If the user wishes to manually change out of and into low power mode, the FUNCTION commands will make this possible via the Keypad Function operation.

```
BEGIN”Low_Pwr”
‘The immediate schedule of a DT80 ONRESET program could contain the following commands;
P16=32
FUNCTION1=”Low PWR”{P16=32}
FUNCTION2=”Normal PWR”{P16=0}
‘insert normal DT80 schedules and settings.
END     ‘End of program
```

Serial devices (RS232, RS422, and RS485)

Dimetix DLS–A Distance Laser Sensor

The simplest method to communicate with the DLS is simply to have the DT8x/800 poll the unit for measurement when required.

The DLS is polled to return a distance on request from the DT8x/800

The DLS command is;
sNg

Where s = command header N = Device number (0 for default) = carriage return / line feed combination.

RS232 wiring configuration

- DLS Unit Serial port “”RX””
- DLS Pin Number “”1””
- DT80 Serial Sensor Port “”Tx””

- DLS Unit Serial port “”Gnd””
- DLS Pin Number “”14 & 15””
- DT80 Serial Sensor Port “”Power GND””

- DLS Unit Serial port “”Tx””
- DLS Pin Number “”2””
- DT80 Serial Sensor Port “”Rx””

- DLS Unit Serial port “”Power””
- DLS Pin Number “”7 & 8””
- DT80 Serial Sensor Port “”Power supply Positive””

Notes:

- Pins 14 and 15 on DLS should be linked
- Pins 7 and 8 on DLS should be linked
- D Gnd on DT80, Gnd of DLS and Power supply ground must be connected
- External 9 to 30 VDC power supply required for DLS

The default communications settings on the DSL are

- Baud rate = 19200 • Data bits = 7 • Parity = Even • Stop bits = 1 • Address = 0
- Note: While RS232 is not addressable the address is required to talk to the DLS

DLS Default communications settings

DeTransfer program 1

```
begin”DLS” ‘Start DLS program
‘=====================================================
‘
‘       Program to read a Dimetix Distance Laser Sensor
‘                                20 January 2006
‘                        support@datataker.com.au
‘
‘=====================================================
ps=19200,e,7,1    ‘Set SSP to 19200 baud, Even parity, 7 data bits, 1 Stop bit
ra1s              ‘Report schedule A every 1 second
 1serial(rs232,”\\e{s0g^M^J}g0g%f[1cv]”,w,1) ‘Poll DLS and read result to 1CV
 1cv(“Distance ~mm”)=1cv/10      ‘Divide reading by 10 to give distance in mm
logon             ‘Turn logging on
end               ‘End of program
```

DeTransfer program 2

```
begin"DLS" 'Start DLS program
'=====================================================
'
'       Program to read a Dimetix Distance Laser Sensor
'
'               This example had error trapping added
'
'                               25 January 2006
'
'               For further detail please contact
'
'                       support@datataker.com.au
'
'       Note; Code is for DT8x/800 V5.xx and above firmware
'
'=====================================================
'Set SSP to 19200 baud, Even parity, 7 data bits, 1 Stop bit
ps=19200,e,7,1
'Initialize Channel variables
1..3cv(w)=0
'Report schedule A every 1 second
ra1s
'Poll DLS for measurement and read result
 1serial(rs232,"\\e{s0g^M^J}%4s['g0g+','g0@E',2cv]%f[1cv]",1,=3cv,w)
 alarm(2cv><1,999)and
 alarm1(1cv><204,204.1)"E? Dimension error^M^J"{[1cv=99999]}
 alarm(2cv><1,999)and
 alarm2(1cv><252,252.1)"E? High temperature^M^J"{[1cv=99999]}
 alarm(2cv><1,999)and
 alarm3(1cv><253,253.1)"E? Low temperature^M^J"{[1cv=99999]}
 alarm(2cv><1,999)and
 alarm4(1cv><255,255.1)"E? Signal too weak^M^J"{[1cv=99999]}
 alarm(2cv><1,999)and
 alarm5(1cv><256,256.1)"E? Signal too strong^M^J"{[1cv=99999]}
 alarm(2cv><1,999)and
 alarm6(1cv><257,257.1)"E? Excessive background light^M^J"{[1cv=99999]}
 alarm(2cv><1,999)and
 alarm7(1cv>260)"E? Hardware error^M^J"{[1cv=99999]}
 alarm8(3cv>1)"Hardware timed out^M^J"
'Divide reading by 10 to give distance in mm
 1cv("Distance ~mm")=1cv/10
logon           'Turn logging on
end             'End of program
```

GPS string handling DT80 range

This is an example of code using a Garmin GPS18 5Hz with a DT80 range logger. The default communications rate for this device is 19200 instead of the usual 4800 baud.

- The program assumes that the power to the GPS is controlled by the relay
- A 5 V regulator or other suitable supply is available to power the GPS
- 12 V supply is tied to digital 8. i.e. when 12 V is available the logging starts and when the 12 V is removed the logging stops
- Heading and mps speed measurements are only made when the speed is greater than 42 CV

```
BEGIN”GPSDT8x”
PS=19200,N,8,1,NOFC
FUNCTION1=”Go”{GA 1RELAY(W)=1 }
FUNCTION2=”Stop”{HA 1RELAY(W)=0}
FUNCTION3=”ZERO TRIP”{23CV=0}
FUNCTION4=”Delete Data”{DELDATA*}
SATTN;
42CV(“Spd Lmt”,W)=1
1RELAY(W)=1
‘ Turn off the unnecessary GPS messages
1SERIAL(“{$PGRMO,GPRMC,0\\013\\010}”,W)
1SERIAL(“{$PGRMO,GPGGA,1\\013\\010}”,W)
1SERIAL(“{$PGRMO,GPGSA,0\\013\\010}”,W)
1SERIAL(“{$PGRMO,GPGSV,0\\013\\010}”,W)
1SERIAL(“{$PGRMO,PGRME,0\\013\\010}”,W)
1SERIAL(“{$PGRMO,GPGLL,0\\013\\010}”,W)
1SERIAL(“{$PGRMO,GPVTG,1\\013\\010}”,W)
1SERIAL(“{$PGRMO,PGRMV,0\\013\\010}”,W)
1SERIAL(“{$PGRMO,PGRMF,0\\013\\010}”,W)
1SERIAL(“{$PGRMO,PGRMB,0\\013\\010}”,W)
1SERIAL(“{$PGRMO,PGRMT,0\\013\\010}”,W)
DELAY(W)=1000
RA”Report”(“B:”,ALARMS:OV:100KB,DATA:OV:30MB)200T  LOGONA GA
 1SERIAL(RS232,”$GPGGA,%f[10CV],%2f[11CV]%f[12CV],%1s[‘N’,’S’,43CV=-1],%3f[13CV]%f[14CV],%1s[‘E’,’W’,44CV=-
1],%f[30CV]”,=99CV,.1,W)
 IF(43CV><1,2){11CV(W)=-11CV}
 IF(44CV><1,2){13CV(W)=-13CV}
 ‘ Check 30CV value if you have a Valid GPS signal read the rest of the string
 IF(30CV>1)
{1SERIAL(RS232,”,%f[9CV],%f[16CV],%f[15CV],%14s[1$]$GPVTG,%f[17CV],T,%f[8CV],M,%f[7CV],N,%f[18CV]”,=99CV,.1,W)}
 ‘ If we have a valid GPS signal clear the ATTN led
 ALARM(30CV>1){CATTN}
 ‘ If the GPS signal is not valid set the ATTN led ON and set the CV’s to zero..
 ALARM(30CV><0,1){SATTN 16CV(W)=-1 15CV(W)=0 17CV(W)=0 8CV(W)=0 18CV(W)=0}
 10CV(“GPSTime”)
 11CV(“LatD~Deg”,FF0) 12CV(“LatM~Min”,FF7)
 13CV(“LonD~Deg”,FF0) 14CV(“LonM~Min”,FF7)
 15CV(“Alt~m”)
 16CV(“HErr~m”,FF2)
 30CV(“GPS State”,FF0)
 27CV(“Head~deg”)=(18CV>42CV)*17CV+(18CV<=42CV)*27CV
 18CV(“Spd~kph”)
 20CV(“Spd~mps”,=21CV)=(18CV>42CV)*18CV*1000/3600 ‘ Calc mps from kph
 21CV(“m”,IB,+=23CV,W) ‘ Integrate mps and accumulate reading to 23CV
 23CV(“Trip~km”,.001,FF4)  ‘ Report 23CV in m as km
 1SERIAL(RS232,”\\e”,=99CV,.1,W)
RD”GoHalt”8E LOGOFFD
 IF(8DS<0.5){1RELAY(W)=0 }
 IF(8DS<0.5){HA}
 IF(8DS>0.5){GA 1RELAY(W)=1}
END
```

Updating DT80 range time from GPS

This This code example will read the UTC time from the NMEA string $GPGGA and add the local time zone offset, then update the DT80 internal clock.

Note: Updating the time from GPS can reduce the accuracy of the DT80 clock use with care

```
BEGIN"GPSTime"
'================================================
'
'       Example of updating time from a GPS
'
'       This example read GPS UTC and updates DT80
'       time at 9 am is there is a valid signal
'
'       Notes;
'               Do not set change the time at midnight.
'               Uses V8.06 firmware or above.
'================================================

'Set serial senor port to match GPS comms
Profile SerSen_Port Function=Serial
Profile SerSen_Port Mode=RS232
Profile SerSen_Port BPS=19200
Profile SerSen_Port Flow=None

'Set local time zone in hours +/- UTC
5CV(W)=10                       'Local time zone in hours. (Melbourne is +10, Montreal is -5)
6CV(W)=5CV*3600         'Calculate local Time zone in second

RA"Report"("B:",ALARMS:OV:100KB,DATA:OV:1MB)10S  'LOGONA GA
        1..4cv(w)=0             'Clear variables

        'Read GPS data string
        1SERIAL(RS232,"\e\m[$GPGGA],%2d[1CV]%2d[2CV]%2d[3CV],%*f,%*1s,%*f,%*1s,%f[4cv]",.1,W)

        1CV("Hours")                    'Show Hours
        2CV("Minutes")                  'Show minutes
        3CV("Seconds")                  'Show seconds
        4CV("Valid signal")             'Show signal validity
        7CV("Second since midnight")=1CV*3600+2CV*60+3CV+6CV      'Calculate seconds since midnight

        'If it is 9:00:00 and GPS signal is valid then set time format
        'to seconds since midnight, update time and set time format back to default
        Alarm(3ST><8,9)And
        Alarm(&"Valid signal"==1)"Time updated"{P39=0 T=7CV P39=0}
END
```

AND 3000i Balance

Applies to DT8x and DT800
This program is to poll an AND 3000i top loader balance for data.

The code will alarm for Unstable weight, Overload and RS232 time out (Scale not found)

Note; The scale must be pre-configured for Comms settings and output type.

```
BEGIN
PS=RS232,2400,E,7,1,NOFC    'RS232, 2400 BAUD, EVEN PARITY,7 DATA BITS, 1 STOP BIT, NO FLOW CONTROL
1SERIAL(RS232,"\e")         'Clear the serial buffer
RA5S
    1SERIAL(RS232,"{Q\013\010}",1,=10CV,W)              'Poll the scales for data.
    1SERIAL(RS232,"%2s['ST','US','OL',2CV=10],%f[1CV]\e",1,=10CV,W)    'Get the reurned data
    'ALARM1(2CV><-0.5,0.5)"Good Data^M^J"                'If 2CV=0 Then header was ST
    ALARM2(2CV><0.5,1.5)"Unstable Weight^M^J"            'If 2CV=1 Then header was US
    ALARM3(2CV><1.5,2.5)"Out of range^M^J"               'If 2CV=2 Then header was OL
    ALARM4(10CV><19.5,20.5)"Can't find the scales^M^J"  'If 10CV=20 Then SERIAL TIME OUT
    1CV("Weight ~Grams")
END
```

Mettler Toledo MT-SICS

Mettler Toledo IND226 Load indicator and Mettler Toledo PG-S top loader balance
If the your Mettler Toledo product has the MT-SICS programing interface, then the following code will work. The MT-SICS provides a standard programming interface to access device data over a serial interface.

Program to read Mettler Toledo top loader balance Balance Model PG-S with MT-SICS Programming Language DT800 V4.02.0001 Firmware or above.

Scale must be set to the following modes, and then powered off/on.

- Host (peripheral device)
- Auto (data Transfer)
- SICS
- 9600 (Baud)
- 8b-no (8 Bit no parity)
- HS Soft (Software hand shaking)

Use a straight through cable. Scale is DCE and DT800 SSP is DTE Wire connection.

- Scale ---- DT800
- TxD (Pin 2) to T-
- RxD (Pin 3) to R-
- SGnd (Pin 5) to Gd

```
BEGIN”Balance”
PS=9600,N,8,1,SWFC   ‘ Serial Sensor Port to 9600 Baud, No Parity, 8 data bits,
    ‘ 1 Stop bit and software flow control
RA10S       ‘ Read schedule every 10 seconds
   1SERIAL(RS232,”\\e”,W)   ‘ Clear serial I/O buffer
   1SERIAL(RS232,”{S^M^J}”,4,W)   ‘ Request stable reading (requires Charage return line feed)
    ‘ Read return header & Match error messages
   1SERIAL(RS232,”\\w[100]%3s[ ‘S S ‘,’S I ‘,’S + ‘,’S - ‘,2CV]”,4,W,=1CV)
   1SERIAL(RS232,”\\w[100]%f[3CV]”,4,W)  ‘ Read floating point number
    ‘ Error handling Routines
   ALARMR1(1CV><19.5,20.5)”RS232 Error :- Time Out ^M^J”    ‘ Generate RS232 Error message
   ALARMR2(2CV><-0.5,0.5)”Valid Reading ^M^J”    ‘ Valid reading message
   ALARMR3(2CV><0.5,1.5)”Reading Not Stable ^M^J”    ‘ Unstable reading Error
   ALARMR4(2CV><1.5,2.5)”Balance in overload range^M^J”    ‘ Over load Error
   ALARMR5(2CV><2.5,3.5)”Balance in underload range^M^J”    ‘ Under load Error
    ‘ Report weight.
3CV(“Weight ~g”,FF3)
END
```

Egg Buddy

The Egg Buddy from Avitronics measure the heartbeat of a developing embryo inside the egg. The Egg buddy the outputs the Heart rate and heart wave from as a serial string that can then be input in the DT8x via the serial sensor port.

Program 1, Heart Wave Form Monitor

This DT8x data logger program will record one minute’s worth of wave form data every hour.

```
begin”HForm”

‘=================================================
‘
‘       Egg Buddy Heart wave form logging
‘
‘       The heart wave form will be logged for one minute every hour.
‘
‘       Datataker Technical Support (RD)
‘               22 Sept 2006
‘
‘       Egg Buddy output in big endian Hexidecimal format
‘       Format ABDDFVC
‘       Where;
‘               A = 8 bit Header (ASCII No. 170)
‘               B = 8 bit Header (ASCII No. 85  (U))
‘               DD = 16 bit Heart Rate data
‘               F = 8 bit Wave Form data
‘               V = 8 bit Valid Data flag (1 = valid, 0 = invalid)
‘               C = 8 bit check sum.
‘
‘=================================================

‘ Configure Serial sensor port for RS232, 19200 Baud, No parity, 8 data bits, 1 Stop bit
ps=rs232,19200,8,n,1

‘Clear the serial sensor port buffer
1serial(rs232,”\\e”)

ra1h
        alarm1(1st>0)”Start Schedule B”{[ra1s gb]}‘Change schedule A rate to one per nimute and start schedule B
         alarms2(2st>1)”Stop Schedule B”{[ra1h hb]}       ‘At end of minute set Schdule rate to one  per hour and
stop schedule B

‘Trigger shedule on incoming data. (Note: We are using default store file sizes)
rb1serial”” hb


        ‘Read the Egg Buddy output
        ‘       Wait for header match           (\\m[\\170U])
        ‘       Read heart rate data into 1 cv  %2b[1cv]
        ‘       Read wave form data             %1b
        ‘       Read Valid data flag into 2cv   1b[2cv]
        ‘       Discard the rest                \\e”
        1serial(rs232,”\\m[\\170U]%2b[1cv]%1b[2cv]%1b[3cv]%*1b”,w,0.1)

        1cv(“Heart rate”)
        2cv(“WForm”)
        3cv(“Valid”)

logona
end
```

Program 2, Heart Rate Monitor

This DT8x program records the heart rate of the embryo every minute.

```
begin”HRate”

‘=================================================
‘
‘       Egg Buddy Heart rate logging
‘
‘       Datataker Technical Support (RD)
‘               22 Sept 2006
‘
‘       Egg Buddy output in big endian Hexidecimal format
‘       Format ABDDFVC
‘       Where;
‘               A = 8 bit Header (ASCII No. 170)
‘               B = 8 bit Header (ASCII No. 85  (U))
‘               DD = 16 bit Heart Rate data
‘               F = 8 bit Wave Form data
‘               V = 8 bit Valid Data flag (1 = valid, 0 = invalid)
‘               C = 8 bit check sum.
‘
‘=================================================

‘ Configure Serial sensor port for RS232, 19200 Baud, No parity, 8 data bits, 1 Stop bit
ps=rs232,19200,n,8,1

‘Clear the serial sensor port buffer
1serial(rs232,”\\e”)

‘Report schedule A every 1 minute. (Note: We are using default store file sizes)
ra1m
        ‘Put your temperature sensor code in here.

        ‘Read the Egg Buddy output
        ‘       Wait for header match           (\\m[\\170U])
        ‘       Read heart rate data into 1 cv  %2b[1cv]
        ‘       Discard wave form data          %*1b
        ‘       Read Valid data flag into 2cv   1b[2cv]
        ‘       Discard the rest                \\e”
        1serial(rs232,”\\m[\\170U]%2b[1cv]%*1b%1b[2cv]\\e”,w,0.1)

        ‘Log heart rate if valid. If invalid data log 0
        1cv(“Heart rate”)=1cv*2cv

logona
end
```

Canary Systems VW DSP Interface

Canary Systems manufactures a vibrating wire interface that can be used with the DataTaker DT80 range of data loggers.

```
‘ Vibrating Wire example code for use of Canary systems VW DSP Interface with DataTaker DT85
begin”vw”

‘ Configure serial sensor port for RS232 communications at 1200 baud.
ps=RS232,1200

‘ Turn on 12V output to power up VW DSP interface
1SSPWR=1

‘Set the vibrating wire measurement parameters (frequency sweep), send the P command to the VW DSP
‘(arguments to P command are recommended defaults: start freq = 400Hz, end freq = 3500Hz, Number of freq cycles
=500
‘ Sampling duration = 100mS, Width of frequency swath = 100)
1SERIAL(“\\e{P0400 3500 0500 0100 0100^M}”)

‘ Set measurement rate to 10 seconds (ie sample sensors every 10 seconds)
RA10S

‘Read Gauge A (using VA command) and return frequency of gauge
1SERIAL(“\\e{VA^M}VA^MVA%d[1cv] %d[2cv] %d[3cv] %d[4cv] %x[5cv]”,20,W)
6cv(“GaugeA Freq”)=1000000/((((3CV*65536)+4cv)/2cv)*0.1356)

‘ Read Temperature of gauge A (using the TA command) and return resistance of thermistor (in ohms)
‘ to convert to temperature (deg c) consult thermistor specifications.
1serial(“\\e{TA^M}TA^MTA%d[1cv] %d[2cv] %x[3cv]”,20)
6cv(“TempA Vtr”,w)=((((1cv*65536)+2cv)/100)/1023)*2.5
7cv(“TempA Ir”,w)=6cv/6040
8cv(“TempA Vr”,w)=7cv*499
9cv(“TempA R”,ff6)=(2.5-6cv-8cv)/7cv

‘Read Gauge B (using VB command) and return frequency of gauge
1SERIAL(“\\e{VB^M}VB^MVB%d[1cv] %d[2cv] %d[3cv] %d[4cv] %x[5cv]”,20,W)
6cv(“GaugeB Freq”)=1000000/((((3CV*65536)+4cv)/2cv)*0.1356)

‘ Read Temperature of gauge B (using the TB command) and return resistance of thermistor (in ohms)
‘ to convert to temperature (deg c) consult thermistor specifications.
1serial(“\\e{TB^M}TB^MTB%d[1cv] %d[2cv] %x[3cv]”,20)
6cv(“TempB Vtr”,w)=((((1cv*65536)+2cv)/100)/1023)*2.5
7cv(“TempB Ir”,w)=6cv/6040
8cv(“TempB Vr”,w)=7cv*499
9cv(“TempB R”,ff6)=(2.5-6cv-8cv)/7cv

‘ turn on logging
logon

end
```

Vaisala HMP60 (RS485)

This sensor has four wires, three of which are required to be connected to the DataTaker (white->Z, black->Y, blue->DGND). If you wish to power up the sensor with the output of the DataTaker (to save overall power), then you can also connect the brown wire to the 12 V terminal.

There are two example programs below.

One Sensor, utilising power savings

```
BEGIN"HMP60LP"
'this code assumes the following:
' - One HMP60 sensor
' - The HMP60 is set to be in poll mode
' - The HMP60 has address number 0
' - Power control is required (low power system)

PROFILE SERSEN_PORT BPS=19200
PROFILE SERSEN_PORT DATA_BITS=8
PROFILE SERSEN_PORT STOP_BITS=1
PROFILE SERSEN_PORT PARITY=NONE
PROFILE SERSEN_PORT FLOW=NONE
PROFILE SERSEN_PORT MODE=RS485
PROFILE SERSEN_PORT FUNCTION=SERIAL

999CV("Address",W)=0 'sensor address
998CV(W)=99999 'preset error value

RA10M LOGONA 'log every 10 minutes
        'the following lines will control the power to the HMP60, wait 3 seconds for it to warm up,
        'send the poll, receive the response and then turn the power back off.
        PWR12V(W)=1;1SERIAL("\\w[3000]\\e{OPEN %d[999CV]\\013}",W,1) 'power up and open the channel to the sensor
        1SERIAL("\\m[\\007]\\e{SEND\\013}",W,1) 'request the data
        1SERIAL("\\m[T=]%f[1CV] '%1s[1$] RH=%f[2CV] %%RH Td=%f[3CV] %*2s",W,1) 'interpret the data
        1SERIAL("{CLOSE %d[999CV]\\013}\\e",W,1);PWR12V(W)=0  'close the channel, power down

        'store the values
        1CV("Temperature")
        2CV("Humidity")
        3CV("Dew Point")
        1$("Temp Units") 'optional
        1..3CV(W)=998CV 'set error values
END
```

Multiple Sensors, Always Powered

```
BEGIN”HMP60MS”
‘this code assumes the following:
‘ - Multiple HMP60 sensor
‘ - The HMP60’s are set to be in poll mode
‘ - The HMP60’s have unique address numbers
‘ - Power control is NOT required (sensors always powered up)

PROFILE SERSEN_PORT BPS=19200
PROFILE SERSEN_PORT DATA_BITS=8
PROFILE SERSEN_PORT STOP_BITS=1
PROFILE SERSEN_PORT PARITY=NONE
PROFILE SERSEN_PORT FLOW=NONE
PROFILE SERSEN_PORT MODE=RS485
PROFILE SERSEN_PORT FUNCTION=SERIAL

998CV(W)=999999 ‘preset error value

RA10M LOGONA ‘log every 10 minutes

        ‘======SENSOR 1 (Address 0)======
        999CV(“Address”,W)=0 ‘this is the HMP60 address
        ‘there is no need to change any of the serial lines for different sensors
        1SERIAL(“\\e{OPEN %d[999CV]\\013}”,W,1) ‘open the channel to the sensor
        1SERIAL(“\\m[\\007]\\e{SEND\\013}”,W,1) ‘request the data
        1SERIAL(“\\m[T=]%f[1CV] ‘%1s[1$] RH=%f[2CV] %%RH Td=%f[3CV] %*2s”,W,1) ‘interpret the data
        1SERIAL(“{CLOSE %d[999CV]\\013}\\e”,W,1) ‘close the channel
        ‘store the values (BE SURE TO GIVE THEM UNIQUE NAMES)
        1CV(“AD0 Temperature”)
        2CV(“AD0 Humidity”)
        3CV(“AD0 Dew Point”)
        1$(“AD0 Temp Units”) ‘optional (delete if not required)
        1..3CV(W)=998CV ‘set error values
        ‘================================

        ‘================================
        ‘to read from another sensor, just copy and paste the below text block
        ‘be sure to change the address number and the names of the stored data
        ‘================================

        ‘======SENSOR 2 (Address 1)======
        999CV(“Address”,W)=1
        1SERIAL(“\\e{OPEN %d[999CV]\\013}”,W,1) ‘open the channel to the sensor
        1SERIAL(“\\m[\\007]\\e{SEND\\013}”,W,1) ‘request the data
        1SERIAL(“\\m[T=]%f[1CV] ‘%1s[1$] RH=%f[2CV] %%RH Td=%f[3CV] %*2s”,W,1) ‘interpret the data
        1SERIAL(“{CLOSE %d[999CV]\\013}\\e”,W,1) ‘close the channel
        ‘store the values (BE SURE TO GIVE THEM UNIQUE NAMES)
        1CV(“AD1 Temperature”)
        2CV(“AD1 Humidity”)
        3CV(“AD1 Dew Point”)
        1$(“AD1 Temp Units”) ‘optional (delete if not required)
        1..3CV(W)=998CV ‘set error values
        ‘================================
END
```

SDI-12

McVan Analite turbidity probe (SDI-12)

This program reads a McVan Analite 390 series Turbidity probe. The probe is connected to the DT80 range Digital channel 5 (for DT81 modify to channel 4). Please refer to the Analite 390 series manual for how to set up the sensor to return statistical data.

```
BEGIN"MCVAN"
RA"Analite"("B:",ALARMS:OV:100KB,DATA:OV:2D)5M LOGONA GA
   5sdi12(ad0,r801,"Wipe",W) 'Activate the wiper
   5sdi12(ad0,r301,"Turbidity~NTU",ff2) 'Return single turbidity reading
   5sdi12(ad0,r201,"Turbidity Mean~NTU",ff2) 'Return Mean Turbidity reading (Refer 390 manual)
   5sdi12(ad0,r202,"Turbidity Min~NTU",ff2) 'Return minimum Turbidity reading (Refer 390 manual)
   5sdi12(ad0,r203,"Turbidity Max~NTU",ff2) 'Return maximum Turbidity reading (Refer 390 manual)
   5sdi12(ad0,r701,"Temperature~degC",ff1) 'Return sensor temperature
END
```

SDI-12 communications error trapping

This code example raises an alarm if the SDI-12 network has failed for two consecutive read attempts.

If the communications with an SDI-12 network fails, the channel will return a value of -9e9.

By assigning the channel value to a Channel variable (CV), we can then test the value in an alarm and take action as required.

```
begin
1..2cv=0   'Clear CV's
ra30s
        5sdi12(ad0,r101,=1cv) 'Read SDI-12 channel and assign output to 1CV
        alarmr(1cv<-9e8){2cv=2cv+1}  'Check and count error messages in 2CV
        alarm(1cv>-9e8){2cv=0}       'If no errors clear error counter
        alarm1(2cv>1.5)"SDI-12 Network 5 Communications error @ #^M^J"  'If errors then raise alarm
end
```

General code examples

Comparing two inputs

This DT80 code compares one input to a reference input and turns on a digital output when the first input is within a stated tolerance of the reference input.

```
Begin

RA1S         'Scan every one second
    1V(=1CV)        'Read the reference channel ans assign the reading to a channel variable
    2CV(W)=1CV+0.1  'Calculate the upper tolerance
    3CV(W)=1CV-0.1  'Calculate the lower tolerance.
    2V(=4CV)        'read the channel to be compared
    ALARM1(4CV><2CV,3CV)1DSO"Inside Range"    'Compare input 2 is inside tolerance range and turn on Digital
Output 1.
End
```

“ON” time duration

Records the length of time a light is turned ON for.

```
begin
10CV=0
ra1s
  1v(=1cv)
  if(10CV<0.5)and
  if(1CV>2000)”Light is On ^M^J”{[ T(=2CV) 10CV=1]}
  If(1CV<2000)”Light is Off ^M^J”{[ T(=3cv) 10CV=0]}
  4CV(“On Time ~Sec”)=3CV-2CV

end
```

Calculation of duty cycle

This DT80 script calculates On time, Off time, and duty cycle.

```
BEGIN”DutyCycl”

‘=====================================================
‘
‘       This program calculate;
‘               On Time
‘               Off Time
‘               Number of cycles
‘               Duty cycle
‘
‘       Of a digital input connected to digital input 1
‘
‘       The totals are reported and cleared at midnight
‘       in schedule B.
‘
‘
‘       This script assumes digital 1 is high when process
‘       is running and low when process is stopped.
‘
‘       Datataker Technical Support 22 Oct 2008 (RD)
‘
‘=====================================================

1..6CV=0 ‘Clear channel variables

‘=====================================================
‘
‘       Schedule A triggers on the rising or falling
‘       edges of digital input 1
‘
‘=====================================================

RA”DutyCycl”(“B:”,ALARMS:OV:100KB,DATA:OV:1MB)1E LOGONA
        1DS(W,=1CV)     ‘Read digital input and assign to 1CV.
        2CV(W,=2CV,DT)  ‘Calculate time between readings and assign to 2CV.
        3CV(“On Time ~Sec”)=3CV+(2CV*1CV)       ‘Calculate On time.
        4CV(“Off Time~Sec”)=4CV+(2CV*not(1CV))  ‘Calculate Off time.
        5CV(“Cycle Count”)=5CV+1CV          ‘Count No. times On.
        6CV(“Duty Cycle~%”)=(3CV/4CV)*100   ‘Calculate duty cycle.

‘=====================================================
‘
‘       Schedule B triggers at midnight.
‘       Reports and Resets the couner values
‘
‘=====================================================

RB”Report”(“B:”,ALARMS:OV:100KB,DATA:OV:1MB)1D LOGONB
        3CV(“On Time~Sec”,R)    ‘Report On Time and reset
        4CV(“Off Time~Sec”,R)   ‘Report Off Time and reset
        5CV(“Tot Cycles”,R)     ‘Report Total Cycles and reset
        6CV(“Duty Cycle~%”,R)   ‘Report Duty Cycle and reset

END
```

Putting day, month, & year into channel variables (DT8x code)

Pre V6.18.0002 firmware

The following code calculates the Day of the month, the month of the year, and the year, and places the values into channel variables. These variables could consequently be used to trigger end of month, end of year activity. It will give the correct answer until the year 2100, which is NOT a leap year.

```
begin”leapy”
RA1D
 15sv(=1cv) ‘days since 1-Jan
 d(=2cv) ‘seconds since 1-Jan-1989
 2cv=2cv/86400-5478 ‘days since 1-Jan-2004
 3cv=(2cv-60)/1461 ‘leap days = (days since 1-Mar-2004) / 365.25*4
 4cv=4+(2cv-3cv-1cv)/365 ‘years since 2000
 5cv=(4cv/4-((4cv/4)%1e6))<0.01 ‘1 if this is a leap year
 1cv=1cv-5cv
 7cv=0-5cv 8cv=31-5cv if(1cv><7cv,8cv){6cv=1 9cv=1cv-7cv+1} ‘jan
 7cv=31-5cv 8cv=59    if(1cv><7cv,8cv){6cv=2 9cv=1cv-7cv+1} ‘feb
 if(1cv><59,90){6cv=3 9cv=1cv-58} ‘mar
 if(1cv><90,120){6cv=4 9cv=1cv-89} ‘apr
 if(1cv><120,151){6cv=5 9cv=1cv-119} ‘may
 if(1cv><151,181){6cv=6 9cv=1cv-150} ‘jun
 if(1cv><181,212){6cv=7 9cv=1cv-180} ‘jul
 if(1cv><212,243){6cv=8 9cv=1cv-211} ‘aug
 if(1cv><243,273){6cv=9 9cv=1cv-242} ‘sep
 if(1cv><273,304){6cv=10 9cv=1cv-272} ‘oct
 if(1cv><304,334){6cv=11 9cv=1cv-303} ‘nov
 if(1cv><334,365){6cv=12 9cv=1cv-333} ‘dec
 9cv 6cv 4cv ‘ dd mm yy
end
```

V6.18.0002 and latter firmware

With the introduction of V6.18 firmware, we now have 3 new System Variables to make this task easier.

- 20SV Current day of the Month
- 21SV Current Month of the year
- 22SV Current Year

The code above now becomes:

```
Begin”Date”
RA1D
   20SV(“Day”)
   21SV(“Month”)
   22SV(“Year”)
End
```

Change an alarm value on the DT8x using function keys

If we are checking temperature for example 1TK, the alarm statement that will close the relay when the temperature goes above 30deg looks like this.

```
BEGIN
 RA1S
  ALARM(1TK>30)1RELAY
END
```

If you want to have a settable alarm using 1CV as the set point:

```
BEGIN
 1CV(W)=30
 RA1S
  ALARM(1TK>1CV)1RELAY
END
```

Therefore, if you change 1CV, you can change the threshold. However, if you want to use the function keys to modify the alarm, the following program would work:

```
BEGIN
  Function1=”Alarm Increase”{1CV(W)=1CV+1 XB”
  Function2=”Alarm Decrease”{1CV(W)=1CV-1 XB”
 RA1S
  ALARM(1TK>1CV)1RELAY
 RBX
  1CV(“Alarm Level”)
END
```

That way, you have the current alarm level on display, and in schedule B, a record of when and what the alarm was changed to.

Starting and stopping the dataTaker at a set time

Pre Version 9.80 firmware

The DataTaker has several internal timers and in combination with the alarm functions can be used to start and stop the DataTaker.

This program starts schedule B running at 9 AM and stops schedule B at 10 AM.

Note: 1ST is a seconds timer. 2ST is a minutes timer. 3ST is an hours timer. 4ST is day of week.

```
BEGIN

RA1H
   ALARM1(3ST>9)”Start”{[GB]}
   ALARM2(3ST>10)”Stop”{[HB]}

RB1S
   1V
   LOGON
END
```

Version 9.08 firmware

A CRON like Time based schedule triggering was introduced in V9.08 firmware. This allows the programming of time range greatly simplifying this task. Please refer to the DT80 range manual for further details.

```
Begin

RA[*:*:9-10]
   1V
LogOn
End
```

DT8x gated input

This is an example of the 32798 Hz gated input on a DT8x logger:

- When 1HSC is pulled low counting starts and stops when open
- The counts are reported when 1DS goes high and 1HSC is reset
- The time is then calculated

```
BEGIN
P27=1 ‘Set up 32798 KHz input
RA1+E ‘Report at end of event
 3HSC(R,=1CV,W) ‘Report counts
 1CV(“Time ~Sec”,FF7)=1CV/32767
END
```

Using analog channels as digital inputs

The analog inputs can be used to extend the basic functions of a digital input by using the Analog State (AS) channel type.

The schedule speed sets the speed the inputs are scanned at so if the input switches faster than the scan time, the state will not be recorded correctly.

The input must have a Voltage present. If you are using a contact closure, then a Voltage must be supplied.

```
Begin"AS"

RA1S
  1AS("Analog state 1",2500) ' Read analog state between +/- with a threshold of 2.5 VDC (0<2500 mV, 1>2500 mV)
  1AS("Analog state 1",2500) ' Read analog state between */# with a threshold of 2.5 VDC (0<2500 mV, 1>2500 mV)

End
```

DT80 range USB drive applications

Unload to USB drive automatically

This code fragment will implement an automatic transfer of data files from the internal memory to a USB flash drive upon insert. An alarm statement is required and must run at an interval frequent enough to detect the USB drive. The Attn LED (1WARN) is illuminated during data transfer and will be turned off when it is safe to remove the drive, which is particularly useful with the DT81/82.

```
RA1S  ALARM(9SV>0.5)1WARN{CopyD Dest=A: format=CSV; REMOVEMEDIA}
```

Note: For low power operation, we recommend an appropriate 'oninsert.dxc' file be used on the USB Flash Disk, which removes the need for a schedule such as the one listed above. The file would contain the following:

```
CopyD Dest=A: format=CSV; REMOVEMEDIA
```

FTP Client–automatic data transfer

DT80 range loggers can automatically transfer data and alarm files via FTP. Typical use includes daily data transfer during non-working hours so that data is readily available in the morning or to transfer to a database system or website.

Use of FTP Client functions require the logger to be connected to a network that has access to the particular FTP site which you wish to use for data transfer. In addition, appropriate network settings must be set in the logger profile.

Example Profile settings.

```
•       [ETHERNET]
•       IP_ADDRESS = 192.168.1.236
•       SUBNET_MASK = 255.255.255.0
•       DEFAULT_GATEWAY = 192.168.1.6
•       [NETWORK]
•       DNS_SERVER_1 = 203.43.52.195
```

FTP Push–with retry

Version 9.08 firmware and latter

Version 9.08 firmware introduced the concept of communications sessions. Communications sessions introduced automated retires in the case of a session failure. There are now several profile setting controlling the retry process. Please refer to the current DT80 User manual for full details.

Pre Version 9.08 firmware

Code example to scan and log data every 10 minutes (schedule A or RA), then FTP logged data at midnight (schedule C or RC). If the transfer fails or the network is temporarily unavailable, testing the value of 29SV (7.02 firmware and above) with an ALARM function (in schedule A) allows retry every 10 minutes until successful. The ALARM output causes schedule B to run once on each test where the failure condition is TRUE. FTP status information available from 29SV is more thoroughly explained in the user manual. Basic usage is that values below 0 indicate a transfer is in progress or that the last transfer failed.

```
BEGIN"JOB1"
 RA"Scanning"10M LOGONA
  1TK("Ambient")
  ALARM(29SV<0.0000){[XB]}
  29SV("FTP Condition")
 RBX LOGONB GB
  DO{ARCHIVEA "ftp://username:password@host-IP/pathname/"}
 RC1D LOGONC GC
  DO{ARCHIVEA "ftp://username:password@host-IP/pathname/"}
END
```

Custom file–FTP Push with header & footer

Version 9.08 firmware

Version 9.08 firmware introduced the concept of communications sessions. Communications sessions introduced automated retires in the case of a session failure. There are now several profile setting controlling the retry process. Please refer to the current DT80 User manual for full details.

The CopyD command will automatically produce a header file when the Format=CSV option is selected.

Pre version 9.08 firmware

In some instances, generation of custom file formats may be required, for example if you want specific headers, formatting, and footer. ALARMS can be configured to achieve this task. The following are two examples–the first uses a single schedule and the second uses multiple schedules.

- This code example uses a single schedule to read one external and one internal temperature channel. It will create a header and footer in the ALARM file. After 10 samples a file cdata.csv will be transferred to the FTP site. The ALARM data will then be deleted. The file cdata.csv will be constantly updated

```
BEGIN”HDRLine”
 99CV(W)=0
RA1S LOGONA
  ALARM1(99CV<0.5)”Date,Time,Temp(DegC),Internal(DegC)”
  D(=1CV) T(=2CV) 1TK(=3CV,NR) REFT(=4CV,NR)
  ALARMR2(11SV<0.5)”#,@,?3F2,?4F2”
  99CV(W)=99CV+1
  ALARM3(99CV>10)”Footer”
  ALARM(99CV>10){/n/c/u P22=32 P24=13 HA}
  ALARM(99CV>10){AA “ftp://username:password@ftp.address.com/cdata.csv” 99CV=0 delalarms* GA}
END
/z/m
```

- This code example using multiple schedules functions in a similar way to the first example. One difference is that this example always has the last 100 records, so intermediate or overlapping reports can be produced that always have the last 100 records. Note the D schedule which contains the FTP statement MUST include the name of the file i.e. DO{A”HDRLine” “ftp://username:password@ftp.address.com/weather/cdata.csv”} So if you change the name of the program, you MUST change the name here as well

```
BEGIN”HDRLine”
 99CV(W)=0
 XA XC
RA(“B:”,ALARMS:OV:1R,DATA:OV:1R)X LOGONA GA ‘Only one record alarm record will be recorded in this schedule
  ALARMR1(11SV<0.1)”Date,Time,Temp(DegC),Internal(DegC)”
RB(“B:”,ALARMS:OV:100R,DATA:OV:1R)1S LOGONB GB ‘ The last 100 alarm records will be written
  D(=1CV) T(=2CV) 1TK(=3CV,NR) REFT(=4CV,NR)
  ALARMR2(11SV<0.5)”#,@,?3F2,?4F2”
  99CV(W)=99CV+1
  ALARM(99CV>10){XD}
RC(“B:”,ALARMS:OV:1R,DATA:OV:1R)X LOGONC GC ‘Only one alarm record will be written
  ALARMR3(11SV<0.1)”Footer”
RDX
  DO{/n/c/u P22=32 P24=13 HB}
  DO{A”HDRLine” “ftp://username:password@ftp.address.com/weather/cdata.csv”}
  DO{99CV=0 GB}
END
```

FTP HTML files from the logger

The following is an example of using alarm statements to build an HTML file and FTP that file. A couple of features to note:

1. The width of the alarm statement has been changed in schedule A to accommodate the number of characters in the width of the header.
2. The number of records in the ‘Header’ i.e schedule A is equal to the number of lines to be used.
3. The number of lines in the ‘Body’ i.e. schedule B is a multiple of the number of alarm statements.
4. Because the Body (schedule B) is set to overwrite this schedule will always have the last 100 records.
5. The number of alarm statements (lines) in schedule C (the footer) is equal to the number of records as per schedule A.
6. To insert a ? character needs ??
7. To insert a ! character needs !!
8. To insert a “ character needs a \034 (or \\034 in Delogger).
9. To insert a channel variable we use the ?nn format ?nnFm selects the channel variable nn and the number of decimal places m to display.
10. The alarm FTP statement needs the program name, if the program name is changed at the top this line will need to be changed too.
11. This is the deTransfer version of the program replace the \\ characters with \ for use in DeLogger Text window.

```
BEGIN”HDRLine”
 99CV(W)=0
 XA XC
RA(“B:”,ALARMS:OV:13R:W100,DATA:OV:1R)X LOGONA GA ‘Only one record alarm record will be recorded in this schedule
 ALARMR1(11SV<0.1)”< !!DOCTYPE HTML PUBLIC \\034-//W3C//DTD HTML 4.01 Transitional//EN\\034>”
 ALARMR1(11SV<0.1)”< html>”
 ALARMR1(11SV<0.1)”< head>”
 ALARMR1(11SV<0.1)”< title>Frame”
 ALARMR1(11SV<0.1)”< meta http-equiv=\\034Content-Type\\034 content=\\034text/html; charset=iso-8859-1\\034>”
 ALARMR1(11SV<0.1)”< body>”
 ALARMR1(11SV<0.1)”< table border=\\0340\\034>”
 ALARMR1(11SV<0.1)”< tr>”
 ALARMR1(11SV<0.1)”< td width=\\034100\\034>Date”
 ALARMR1(11SV<0.1)”< td width=\\034100\\034>Time”
 ALARMR1(11SV<0.1)”< td width=\\034100\\034>Temp Deg”
 ALARMR1(11SV<0.1)”< td width=\\034100\\034>IntTemp Deg”
 ALARMR1(11SV<0.1)”< /tr>”
RB(“B:”,ALARMS:OV:3600R,DATA:OV:1R)1S LOGONB GB
 D(=1CV) T(=2CV) 1TK(=3CV,NR) REFT(=4CV,NR)
 ALARMR2(11SV<0.5)”< tr>”
 ALARMR2(11SV<0.5)”< td>#”
 ALARMR2(11SV<0.5)”< td>@”
 ALARMR2(11SV<0.5)”< td>?3F2”
 ALARMR2(11SV<0.5)”< td>?4F2”
 ALARMR2(11SV<0.5)”< tr>”
 99CV(W)=99CV+1
 ALARM(99CV>10){XD}
RC(“B:”,ALARMS:OV:3R,DATA:OV:1R)X LOGONC GC ‘Only one alarm record will be written
 ALARMR3(11SV<0.1)”< /table>”
 ALARMR3(11SV<0.1)”< /body>”
 ALARMR3(11SV<0.1)”< /html>”
RDX
 DO{/n/c/u P22=32 P24=13 HB}
 DO{A”HDRLine” “ftp://username:password@ftp.address.com/weather/cdata.csv”}
 ‘do{A}
 DO{99CV=0 GB}
END
```

Reading Modbus RTU power meters

The Modbus master capabilities allow the simple interfacing of a number of power meters that are Modbus RTU devices. The code examples below use Modbus block read to simplify and speed up the reading and retry process if polling fails.

Dent Power Scout P3 power meter

```
BEGIN"PS-3"

'=====================================================================
' PS-3 Three Phase Sample - Version 1.0
' Application: Dent PowerScout 3
' dataTaker: DT82I/80/85
' Firmware: 8.08
' Program written by Michael Hampson
' Contact: 02 4739 8400 - sales@lontek.com.au - www.lontek.com.au
' Date: Thursday, 9 December 2010
' Modified for block reads Monday 21 Feb 2011
' By RD. dataTaker technical support
'=====================================================================

'Sets up the DT80 serial sensor port
PROFILE "SERSEN_PORT" "MODE"="RS485"
PROFILE "SERSEN_PORT" "BPS"="9600"
PROFILE "SERSEN_PORT" "DATA_BITS"="8"
PROFILE "SERSEN_PORT" "STOP_BITS"="1"
PROFILE "SERSEN_PORT" "PARITY"="NONE"
PROFILE "SERSEN_PORT" "FLOW"="NONE"
PROFILE "SERSEN_PORT" "FUNCTION"="MODBUS_MASTER"

'PROFILE LOCALE TIME_ZONE=+10H
'PROFILE NTP SERVER=0.datataker.pool.ntp.org
'PROFILE NTP BACKGROUND_ENABLE=YES
'PROFILE NTP BACKGROUND_PERIOD=24H

'Clear "ATTENTION" light on the data logger.
CATTN

1MODBUS(AD1,R4:4600)=50      'Sets the CT Integer value, so 50 would be for a 50 amp CT.
'1MODBUS(AD1,R4:4602)        'Checks the scalar value, see table 2 in the PS manual.

RA("B:",ALARMS:OV:100KB,DATA:OV:5MB)1M LOGONA GA

'Kilowatts
        1MODBUS(AD1,R4:4029,0.01,=1..3CV,RT2,TO1,W)
        1CV("L1 Power_1~Kilowatts")
        2CV("L2 Power_1~Kilowatts")
        3CV("L3 Power_1~Kilowatts")

'KVA
        1MODBUS(AD1,R4:4047,0.01,=4..6CV,RT2,TO1,W)
        4CV("L1 kVA_1~kVA")
        5CV("L2 kVA_1~kVA")
        6CV("L3 kVA_1~kVA")
```

```
'kVAR
        1MODBUS(AD1,R4:4037,0.01,=7..9CV,RT2,TO1,W)
        7CV("L1 kVAR_1~kVAR")
        8CV("L2 kVAR_1~kVAR")
        9CV("L3 kVAR_1~kVAR")

'Current
        1MODBUS(AD1,R4:4056,0.1,=10..12CV,RT2,TO1,W)
        10CV("L1 Current_1~Amps")
        11CV("L2 Current_1~Amps")
        12CV("L3 Current_1~Amps")

'Voltage
        1MODBUS(AD1,R4:4059,0.1,=13..15CV,RT2,TO1,W)
        13CV("L1-N Voltage~Volts")
        14CV("L2-N Voltage~Volts")
        15CV("L3-N Voltage~Volts")

'Apparent Power Factor
        1MODBUS(AD1,R4:4053,0.01,=16..18CV,RT2,TO1,W)
        16CV("L1 Apparent PF~PF")
        17CV("L2 Apparent PF~PF")
        18CV("L3 Apparent PF~PF")

END
```

Cromption Instruments Tyco 1630 power meter

```
Begin'1630a'

'================================================
'
'       DT80 code to read a Tyco 1630 power meter
'       via RS485 Modbus
'
'       3 Phase 4 wire configuration.
'       Reads all Modbus type 3 registers.
'
'       This code used block reads of Modbus registers
'               - Block reads are in groups of 10 for readability
'               - Maximum block read size is 13 registers
'               - Block reading reduces reading times of registers
'
'       Comment out or remove channel variables
'       for items not required.
'
'================================================

' Profile settings for Modbus Master
' Note: Check port settings and set profile to match.
Profile SerSen_Port Mode=RS485
Profile SerSen_Port Function=Modbus_Master
Profile SerSen_Port BPS=9600
Profile SerSen_Port Data_Bits=8
Profile SerSen_Port Stop_Bits=1
```

```
Profile SerSen_Port Parity=None
Profile SerSen_Port Flow=None

RA”PwrMtr”(“B:”,Data:Ov:7D)1M

‘Block read registers 1 to 19 and save to channel variable 1 to 21.
        1Modbus(Ad1,R3:1,MBF,MES,TO1,RT2,=1..10CV,W)
                1CV(“Volts Phase 1~V”,FF6)
                2CV(“Volts Phase 2~V”,FF6)
                3CV(“Volts Phase 3~V”,FF6)
                4CV(“Current Phase 1~A”,FF6)
                5CV(“Current Phase 2~A”,FF6)
                6CV(“Current Phase 3~A”,FF6)
                7CV(“Watts Phase 1~W”,FF6)
                8CV(“Watts Phase 2~W”,FF6)
                9CV(“Watts Phase 3~W”,FF6)
                10CV(“VA Phase 1”,FF6)

‘Block read registers 21 to 40 and save to channel variable 11 to 20.
        1Modbus(Ad1,R3:21,MBF,MES,TO1,RT2,=11..20CV,W)
                11CV(“VA Phase 2”,FF6)
                12CV(“VA Phase 3”,FF6)
                13CV(“var Phase 1”,FF6)
                14CV(“var Phase 2”,FF6)
                15CV(“var Phase 3”,FF6)
                16CV(“Power Factor Phase 1”,FF6)
                17CV(“Power Factor Phase 2”,FF6)
                18CV(“Power Factor Phase 3”,FF6)
                19CV(“Phase Angle Phase 1~Deg”,FF6)
                20CV(“Phase Angle Phase 2~Deg”,FF6)

‘Block read registers 41 to 60 and save to channel variable 21 to 30.
        1Modbus(Ad1,R3:41,MBF,MES,TO1,RT2,=21..30CV,W)
                21CV(“Phase Angle Phase 3~Deg”,FF6)
                22CV(“Volts (Ave)~V”,FF6)
                24CV(“Current (Ave)~A”,FF6)
                25CV(“Current (Sum)~A”,FF6)
                27CV(“Watts (Sum)~W”,FF6)
                29CV(“VA (Sum)”,FF6)

‘Block read registers 61 to 79 and save to channel variable 31 to 40.
        1Modbus(Ad1,R3:61,MBF,MES,TO1,RT2,=31..40CV,W)
                31CV(“var (Sum)”,FF6)
                32CV(“Power Factor (Ave)”,FF6)
                34CV(“Phase Angle (Ave)~ Deg”,FF6)
                36CV(“Freq~Hz”,FF6)
                37CV(“Wh Import”,FF6)
                38CV(“Wh Export”,FF6)
                39CV(“varh Import”,FF6)
                40CV(“varh Export”,FF6)

‘Block read registers 81 to 87 and save to channel variable 41 to 50.
        1Modbus(Ad1,R3:81,MBF,MES,TO1,RT2,=41..50CV,W)
                41CV(“VAh”,FF6)
                43CV(“W Demand Import”,FF6)
                44CV(“W Max. Demand Import”,FF6)
```

```
‘Block read registers 101 to 107 and save to channel variable 41 to 50.
        1Modbus(Ad1,R3:101,MBF,MES,TO1,RT2,=51..60CV,W)
                51CV(“VA Demand”,FF6)
                52CV(“VA Max. Demand”,FF6)
                53CV(“A Demand”,FF6)
                54CV(“A Max. Demand”,FF6)

‘Block read registers 201 to 207 and save to channel variable 101 to 104.
        1Modbus(Ad1,R3:201,MBF,MES,TO1,RT2,=101..104CV,W)
                101CV(“Volts L1 to L2~V”,FF6)
                102CV(“Volts L2 to L3~V”,FF6)
                103CV(“Volts L3 to L1~V”,FF6)
                104CV(“Volts Line to Line (Ave)~V”,FF6)

‘Read register 225
        1Modbus(Ad1,R3:225,MBF,MES,TO1,RT2,”Neutral Current”,FF6)

‘Block read registers 235 to 239 and save to channel variable 118 to 120.
        1Modbus(Ad1,R3:235,MBF,MES,TO1,RT2,=118..120CV,W)
                118CV(“THD Volts 1~V”,FF6)
                119CV(“THD Volts 2~V”,FF6)
                120CV(“THD Volts 3~V”,FF6)

‘Block read registers 241 to 259 and save to channel variable 121 to 130.
        1Modbus(Ad1,R3:241,MBF,MES,TO1,RT2,=121..130CV,W)
                121CV(“THD Current 1~A”,FF6)
                122CV(“THD Current 2~A”,FF6)
                123CV(“THD Current 3~A”,FF6)
                125CV(“THD Voltage (Mean)~V”,FF6)
                126CV(“THD Current (Mean)~A”,FF6)
                127CV(“Run Hours~h”,FF6)
                128CV(“Power Factor (+Ind/-Cap)”,FF6)
                130CV(“Current 1 Demand~A”,FF6)

‘Block read registers 261 to 269 and save to channel variable 131 to 140.
        1Modbus(Ad1,R3:261,MBF,MES,TO1,RT2,=131..140CV,W)
                131CV(“Current 2 Demand~A”,FF6)
                132CV(“Current 3 Demand~A”,FF6)
                133CV(“Current 1 Max. Demand~A”,FF6)
                134CV(“Current 2 Max. Demand~A”,FF6)
                135CV(“Current 3 Max. Demand~A”,FF6)

End
```

Schneider Electric PowerLogic ION6200 power meter

```
Begin”Ion6200”

‘=======================================
‘ DT80 code to read Schneider Electric
‘       PowerLogic ION620 Power and Energy meter.
‘
‘       This code used block reads of Modbus registers
‘               - Block reads are in groups of 15 for readability
‘               - Block reading reduces sample times of registers
‘
‘       Comment out or remove channel variables
‘       for items not required.
‘
‘=======================================

Profile SerSen_Port BPS=9600
Profile SerSen_Port Flow=none
Profile SerSen_Port Mode=RS485
Profile SerSen_Port Function=Modbus_Master


RA30S
        ‘Block read registers 40100 to 40115 and save to CV’s
        1Modbus(ad132,R4:100,MBU,0.1,=1..15CV,RT2,TO1,)
                1CV(“VIn a~V”)
                2CV(“VIn b~V”)
                3CV(“VIn c~V”)
                4CV(“VIn avg~V”)
                5CV(“VII ab~V”)
                6CV(“VII bc~V”)
                7CV(“VII ca~V”)
                8CV(“VII avg~V”)
                9CV(“I a~A”)
                10CV(“I b~A”)
                11CV(“I c~A”)
                12CV(“I avg~A”)
                13CV(“I Demand~A”)
                14CV(“I Peak Demand~A”)
                15CV(“I4~A”)

        1Modbus(ad132,R4:115,MBU,0.01,=16..20CV,RT2,TO1,)
                16CV(“Frequency~Hz”)
                17CV(“PF sign Total”)
                18CV(“PF sign a”)
                19CV(“PF sign b”)
                20CV(“PF sign c”)

        1Modbus(ad132,R4:119,MBI,1,=21..38CV,RT2,TO1,)
                21CV(“kW total”)
                22CV(“kVAR total”)
                23CV(“kVA total”)
                24CV(“kW a”)
                25CV(“kW b”)
                26CV(“kW c”)
                27CV(“kVAR a”)
```

```
                28CV("kVAR b")
                29CV("kVAR c")
                30CV("kVA a")
                31CV("kVA b")
                32CV("kVA c")
                33CV("kW demand")
                34CV("kW peak demand")
                35CV("kVAR demand")
                36CV("kVA demand")
                37CV("kVAR peak demand")
                38CV("kVA peak demand")

        1Modbus(ad132,R4:138,MBL,1,=39..43CV,RT2,TO1,)
                39CV("kWh del")
                40CV("kWh rec")
                41CV("kVARh del")
                42CV("kVARh rec")
                43CV("kVARh del+rec")

        1Modbus(ad132,R4:148,MBU,0.1,=44..55CV,RT2,TO1,)
                44CV("V1 THD")
                45CV("V2 THD")
                46CV("V3 THD")
                47CV("I1 THD")
                48CV("I2 THD")
                49CV("I3 THD")
                50CV("I a demand")
                51CV("I b demand")
                52CV("I c demand")
                53CV("I a peak demand")
                54CV("I b peak demand")
                55CV("I c peak demand")

'=======================================
'Registers 40160 to 40188 are the same contect
' as 41138 to 41188 and 32 bit unsigned integer data type
' but in Low word / High word order.
'
' 41138 to 41188 are High word / Low word order
' which is the dataTaker default.
'
'Note The DT80 does not have a Modbus 32 bit unsigned integer
' The range of an unsigned int is 0 to 4,294,967,295
' while an signed is - 2,147,483,648 to 2,147,483,647
' For most applications 2,147,483,647 kWh will never be reached.
' If it is the number will jump from 2,147,483,647 to - 2,147,483,648
' and then count down to zero.
'=======================================
        1Modbus(ad132,R4:1138,MBL,1,=71..80CV,RT2,TO1,)
                71CV("kWh del")
                72CV("kWh rec")
                73CV("kVARh del")
                74cv("kVARh rec")
                75CV("kVAh del+rec")
                76CV("kWh a del")
                77CV("kWh b del")
                78CV("kWh c del")
```

```
                79CV(“kWh a rec”)
                80CV(“kWh b rec”)

        1Modbus(ad132,R4:1170,MBL,1,=81..90CV,RT2,TO1,)
                81CV(“kWh c rec”)
                82cv(“kVARh a del”)
                83CV(“kVARh b del”)
                84CV(“kVARh c del”)
                85CV(“kVARh a rec”)
                86CV(“kVARh b rec”)
                87CV(“kVARh c rec”)
                88CV(“kVAh a”)
                89CV(“kVAh b”)
                90CV(“kVAh c”)

End
```

In Australia:
For customer service, call 1300-735-292
To email an order, ordersau@thermofisher.com

In New Zealand:
For customer service, call 0800-933-966
To email an order, ordersnz@thermofisher.com

Find out more at **thermofisher.com/datataker**

ThermoFisher SCIENTIFIC