dataTaker

Training Note

TR-12RD

DT80 Range Modbus Slave

DT80 range Modbus capabilities explained

Modbus Slave: Firmware version 6.02 or above
Version 1, October 2009.

Contents

Introduction

Modbus is a simple communications protocol which is widely used in SCADA (supervisory control and data acquisition) systems. Modbus provides an efficient and standardized way to transport digital states and data values between a remote terminal unit (RTU) or programmable logic controller (PLC) and a supervisory computer.

In a Modbus-based SCADA system, each RTU/PLC acts as a Modbus server, or slave. These servers (slaves) listen for and reply to requests from a Modbus client, or master system. A Modbus client is typically a computer that provides a mimic display, user interface and various data logging and alarm functions.

Modbus Slave (Server) capability was added to the DT80 at version 6.02 firmware.

Application

The DT80 range slave capabilities open new opportunities including;

- Integration into Industrial SCADA systems.
- Use of Human Machine Interface (HMI) devices.
- Gateway device.
    - SDI-12 to Modbus
    - RS232, RS422, RS485 to Modbus.

Modbus communications methods

Modbus can operate using a broad range of communications media. These fall into two main categories:

- Serial connection, typically RS232, RS422 or RS485
- TCP/IP network, which can use a variety of physical link types e.g. Ethernet, wireless, fibre-optic, serial (PPP)

The DT80 supports both TCP/IP and serial Modbus networks.

Serial Connection

A serial Modbus network has one client (master) system connected to one or more server (slave) devices. Serial networks using the RS485 or RS422 standards support multi-drop, i.e. multiple slaves connected to one master. RS232 or USB can also be used for point-to-point connections (single master and single slave).

Slave devices on a serial Modbus network are identified by an 8-bit slave address (1-247). Every slave device on a particular serial network must have a unique address. The DT80 can be connected to a serial Modbus network using either the serial sensor port, the host RS232 port, or the USB port. The serial sensor port on the DT80 can be configured for RS232, RS422 or RS485 modes.

TCP/IP Connection

The first step in setting up Modbus over TCP/IP is to establish a working TCP/IP connection between the client (Master) system and the DT80. This involves assigning an IP address to the DT80, along with a couple of other settings, depending on whether Ethernet or PPP is used.

By default, the DT80's TCP/IP Modbus server is always enabled. It will listen for connection requests from client systems which are directed to TCP port 502, which is the standard port number for Modbus. Slave addresses are not required on a TCP/IP Modbus network, because the slaves are identified by their IP address.

Modbus register types

The Modbus message consists of;

- Address of the Modbus slave device
  - In the range of 1 to 247 for serial devices
  - TCP/IP address for network devices
- Register type.
  - 0 = Coil (Digital, Read and Write)
  - 1 = Discrete input (Digital, Read only)
  - 3 = Input register (16 bit input registers, Read only)
  - 4 = Output register (16 bit output registers, Read and Write)
- Register number.
  - Either a 5 or 6 digit number of the register. (Device dependant)

Note: The register number will depend on the device. Most devices have more than one register and most registers may be used in more that one data type.

The Modbus convention for representing a resister type has the register type followed by the register number. A Modbus slave device will have a manufacturer supplied mapping for each valid register type and register number.

The DT80 uses the following mapping for its registers:

Modbus slave register range	***DT80 Channel***
0001 to 1000	Channel variable 1 to 1000
4001 to 4053	System variables 1 to 53
8000 to 8009	Digitals 1 to 8 and 1Relay

Table 1 - DT80 range Modbus registers

Putting this into standard Modbus convention we have:

Modbus Register Mapping	***DT80 Channel***	***DT80 Action***
Register type – Coil	**Comment: Coil type is read / write**	
00001 to 01000	Channel variable 1 to 1000	Returns 0 or 1 (If CV >0 then output = 1)
04001 to 04053	System variables 1 to 53	Returns 0 or 1 (If SV >0 then output = 1)
08000 to 08009	Digitals 1 to 8 and 1Relay	Returns digital state
Register type – Discrete	**Comment: Discrete type is read only**	
10001 to 11000	Channel variable 1 to 1000	Returns 0 or 1 (If CV >0 then output = 1)
14001 to 14053	System variables 1 to 53	Returns 0 or 1 (If SV >0 then output = 1)
18000 to 18009	Digitals 1 to 8 and 1Relay	Returns digital state
Register type - Holding Register	**Comment: Holding register is read only**	
30001 to 31000	Channel variable 1 to 1000	Returns current CV value
34001 to 34053	System variables 1 to 53	Returns current SV value
38000 to 38009	Digitals 1 to 8 and 1Relay	Returns current digital value (0 or 1)
Register type - Input Register	**Comment: Input register is read / write**	
40001 to 41000	Channel variable 1 to 1000	Returns current CV value
44001 to 44053	System variables 1 to 53	Returns current SV value
48000 to 48009	Digitals 1 to 8 and 1Relay	Returns current digital value (0 or 1)

Table 2 - DT80 range Modbus registers with function numbers

As can be seen, the first digit of the register number indicates the type of register - coil, discrete input, input register or output register. This usage is, however, just a convention. This digit is not part of the actual address transmitted in the Modbus message.

A further potential source of confusion is the fact that the actual transmitted address is zero-based, so register number x0003 is actually transmitted as address 0002.

Note: In some Modbus client applications, register numbers are entered using these raw protocol addresses, while in others you specify register numbers including the initial "register type" digit, as described above. The documentation for the particular package should make clear which convention it uses.

The Modbus protocol then defines a set of messages which allow the client to:

- read the current value of one or more of the slave's coils, discrete inputs, input registers or output registers
- write to one or more of the slave's coils or output registers.

A given type of Modbus slave device will support some quantity of each type of resource - for example a hypothetical device might support 16 coils, 16 discrete inputs, 4 input registers and no output registers.

Furthermore, it is common for the different register arrays to overlap. In the example device mentioned above, the 16 coils and discrete inputs may actually refer to the same physical hardware - in this case 16 bi-directional I/O pins. So for this slave device, if a client wrote a "1" to coil 00007, it would then read the same value back if it did a read from discrete input 10007.

Modbus data types

The Modbus standard only defines the size of the registers as being 16 bit wide. It does not detail how the data is to be stored in the registers.

While the Coil and Discrete register types are only 1 bit, the representation of numbers can vary depending on the type of number and the type of computer system being used.

Note: Whatever method of representing data a system is using, it is important that the Modbus Master be configured to read the matching data format types from the slave device.

Data Type	***Range***
16 bit signed integer	-32768 to 32767
16 bit unsigned integer	0 to 65535
32 bit signed integer	-2,147,483,648 to 2,147,483,647
32 bit floating point number	-3.4028e38 to 3.4028e38

Table 3. Data types and approximate range

To represent decimal numbers with integer data types the slave will take a reading and multiply the number by a scalar then place the scaled number in the register. The master will then read the device register and divide the number by the same scalar.

For example a slave device may read a thermocouple temperature as 23.4 degrees C. If it puts this value in a holding 16 bit signed integer the decimal place will be lost. To overcome this limitation the slave multiples the temperature value by 10 so the register now hold the value 234. When the master reads this value it will divide the register value by 10 to restore the decimal place.
32 bit data types are stored across two consecutive Modbus registers. While the Modbus standard does state that data in a register is in High byte, Low byte format (Big endian) it does not specify the word order (endian) when data is across registers. For this reason you need to be aware of the word order of the slave device when reading 32 bit data.

Planning a Modbus Task

Because of the level of detail required for implementing a Modbus system it is strongly advised that the task is approached systematically.

Before you start:

- Understand the slave device capabilities.
  - Communication method and set up (RS232)
  - Assign unique device address.
  - Understand Modbus register types
  - Understand Modbus register data types
  - Understand data word order
- Design the system
  - Construct a Modbus register map that includes device numbers, register types, data types and details etc.
  - Communications type to be used and settings.
  - Other consideration. Displays, data handling etc.

- Implement and document
  - Program and deploy the system
  - Document the system components, structure, design details, etc

Don't forget to document the project. Remember you may or may not be the person who has to repair or modify the system at a future date. Good documentation will greatly simplify the task.

Note: Data in Channel Variables that are marked as Working Channels can still be read in the Modbus registers. The Modbus master simply does not read those registers.

DT80 Modbus commands

Modbus related profile settings

Serial Sensor Port

PROFILE "SERSEN_PORT" "FUNCTION"="*mode*"
Where *mode* will be Modbus for Modbus slave mode

Used to set serial sensor port operation to Modbus slave mode.

PROFILE "SERSEN_PORT" "FLOW"="NOFC"

Important note: As Modbus communications is a binary data transfer it must not use software flow control. If Software flow control is set communications may be unreliable.

PROFILE "SERSEN_PORT" "MODE"="*type*"
Where *type* can be

- RS232 (Point to point only)
- RS422 (Multi drop)
- RS482 (Multi drop)

Selects communications interface standard for communications with the Modbus network.

Notes: 1. With RS232, RS422 or RS485 there can only be one Modbus master device per network.
2. Use the other SerSen_Port profile setting to configure the other communications parameters (baud rate etc)

PROFILE "MODBUS_SERVER" "SERSEN_ADDRESS"="*addr*"
Where *addr* is 8 bit Modbus slave address in the range of 1 to 247.

Used to set the slave device unique Modbus network address.

Host Port

PROFILE "HOST_PORT" "FUNCTION"="*mode*"
Where *mode* will be Modbus for Modbus slave mode

Used to set host port operation to Modbus slave mode.

PROFILE "HOST_PORT" "FLOW"="NOFC"

Important note: As Modbus communications is a binary data transfer it must not use software flow control. If Software flow control is set communications may be unreliable.

PROFILE "MODBUS_SERVER" "HOST_ADDRESS"="*addr*"
Where *addr* is 8 bit Modbus slave address in the range of 1 to 247.

Used to set the slave device unique Modbus network address.

Note: 1. As the Host serial port is only RS232 it can only be point to point. That is only one Modbus slave device and one Modbus master.
2. Use the other Host_Port profile setting to configure the other communications parameters (baud rate etc)

USB Port

PROFILE "USB_PORT" "FUNCTION"="*mode*"
Used to set USB port operation to Modbus slave mode.

Used to select USB port operation to be either Modbus master or Modbus slave mode.

PROFILE "MODBUS_SERVER" "USB_ADDRESS"="*addr*"
Where *addr* is 8 bit Modbus slave address in the range of 1 to 247.

Used to set the slave device unique Modbus network address.

Notes: 1. The USB port can only be point to point. That is only one Modbus slave device and one Modbus master.
2. There are no communications parameters to se. e.g. Baud rate etc.

TCP/IP

PROFILE "MODBUS_SERVER" "TCPIP_PORT"="*port*"
Where *port* is the TCP/IP port number Modbus will use to communicate with the DT80.
(Default is port 502)

Notes: 1. The device address is the TCP/IP address or symbolic DNS address.
2. Use the Ethernet profile settings to configure network settings.
3. Use a static TCP/IP address if you don't have a symbolic DNS address for the DT80.

Modbus commands, channel type and parameters

SETMODBUS Command

The DT80 series maps the channel variables to the Modbus register when the DT80 is being configured as a Modbus slave device (Refer to table 1 for details). By default the DT80 will use a signed 16 bit integer. The SetModbus command allows the user to specify the data type the Modbus register will contain.

Syntax: SETMODBUS *Channels Format Scaling*

Where;

Channels Channel Variable or range of channel variables the data format applies. E.g. 1CV or 100..120CV

Format Data format to be applied to the data.

Channel option	***Data Type***
MBI	16 bit signed integer (default)
MBU	16 bit unsigned integer
MBL	32 bit signed integer, straight endian (MSB first)
MBF	32 bit float, straight endian (MSB first)

Table 4: data types.

Scaling Optional floating point scaling factor by which the channel value will be multiplied before being returned. Conversely, when the client writes a value, it will be divided by the scaling factor before being written to the CV.

Examples:

```
SetModbus 1CV MBI          'Set 1CV as 16 bit signed integer. Scaling =1
SetModbus 2CV MBI 10       'Set 2CV as 16 bit signed integer. Scaling =10
SetModbus 3..5CV MBI 100   'Set 3..5CV as 16 bit signed integer. Scaling =100
SetModbus 10CV MBL  'Set 10CV as a 32 bit signed int. straight endian
SetModbus 11..20CV MBF     'Set 11..20CV as a 32 bit floating point, straight endian
```

Notes:

1. *As an integer data type it can not represent decimal points. The reading is multiplied by the scale value number which is passed to the Modbus master. The Modbus master the divides the number by the scale value to restore the decimal places.*

2. *32 bit data types are read across two Modbus registers. This does not affect the DT80 channels variables as the DT80 logically maps the Modbus registers to the Channel variables.*

Troubleshooting

Modbus diagnostics P56=4

Setting P56=4 will turn on the Modbus debug output
When in this mode the Modbus communications between the master and slave devices will be returned.

P56=4
Modbus RX <192.168.1.60:1168: 00000000000601 0400000008 (12)
Modbus TX >192.168.1.60:1168: 00000000001301 0410000000000000000000000000000000000000 (25)
Modbus RX <192.168.1.60:1168: 00010000000601 0400000008 (12)
Modbus TX >192.168.1.60:1168: 00010000001301 0410000000000000000000000000000000000000 (25)
Modbus RX <HOST: 01 0400000008f1cc (8)
Modbus TX >HOST: 01 0410000000000000000000000000000000552c (21)

This shows a message received from a Modbus client at IP address 192.168.1.60 (port 1168) which has total length of 12 bytes. The data is shown in hexadecimal form. The first part is the TCP/IP-specific header, which comprises: a transaction identifier (0000, used to match up requests and replies), a protocol identifier (0000 indicates Modbus), the number of following bytes (0006) and the slave address (01 – not applicable for TCP/IP).

The actual Modbus frame follows: 04 for "read input registers" function code, starting address 0000 and length 0008. This is therefore a request to read the values of 1CV to 8CV.

The DT80 then replies with the same function code (04), the byte count (10), and 8 x 16- bit data values (all zero in this case). This exchange is then repeated; notice that the client has incremented the transaction identifier, which is now 0001.

A serial Modbus message is then received on the host port, addressed to slave ID 01. Again, this is a request to read 1CV to 8CV. Note that serial Modbus also includes a checksum on the end (f1cc).

Worked Examples

Configuring the DT80 as a Modbus Slave

Example:

A DT80 is being used to monitor the performance of an environmental chamber. The DT80 will read 4 PT100 RTD temperature sensors and display the temperature with one decimal place. The average of the 4 temperature will also be displayed. The average reading will be tested in an alarm for over or under range condition, set point +/- 1 Deg C. The external power state will also be read from 5 SV. Modbus communications will be via Ethernet.

Note: 5SV (System Variable) hold 1 if mains power is connected, 0 if disconnected.

Planning

The four RTD's will be connected to the DT80 channels 1 to 4 and will be stored in Channel Variable 1 to 4. The average temperature reading of the four RDT's will be calculated and stored in channel variable 5. System variable 5 can be directly accessed by the Modbus master.

Modbus Mapping Table

Device details

Name	DT85g
Serial Number	85533
Location	My office
Communications	Modbus TCP/IP
Modbus address	192.168.11.69 on Port 502

Modbus register details

Modbus Register	Register type	Data type	DT80 Channel	Description
30001	3 – holding reg	Signed int	1CV	Temperature 1
30002	3 – holding reg	Signed int	2CV	Temperature 2
30003	3 – holding reg	Signed int	3CV	Temperature 3
30004	3 – holding reg	Signed int	4CV	Temperature 4
30005	3 – holding reg	Signed int	5CV	Average Temp.
04005	1 - Discrete	Digital	5SV	Power On / Off

Comments:
Registers 1 to 5 have been multiplied by 10

All registers are read only

Table 5 - Modbus register map

Implementation

Profile Settings

The DT80 will need to be configured for a static TCP/IP address.

```
Profile "Ethernet" "IP_address"="192.168.11.69"
```

DT80 Code

```
Begin"Slave"

SetModbus      1..5CV MBI 10          'Configure Modbus registers

RA1S           'Sample once per second
        1PT385(=1CV)          'Read RTD 1 and save to CV
        2PT385(=2CV)          'Read RTD 2 and save to CV
        3PT385(=3CV)          'Read RTD 3 and save to CV
        4PT385(=4CV)          'Read RTD 4 and save to CV
        5CV=(1CV+2CV+3CV+4CV)/4          'Cal Average and save to CV
End
```

Appendix 1: Modbus mapping template

Modbus Mapping Table			
Device details			
Name Serial Number Location Communications Modbus address			
Modbus register details			
Modbus Register	Description	Data Type	DT80 Channel
Comments			

Table 6. Modbus register map. Chamber 1