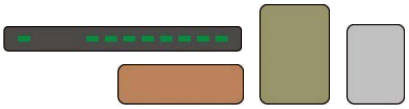
The Modbus Driver



The Modbus driver allows North to interface with a wide range of equipment supporting the Modbus protocol, both over TCP/IP and serial connections. As a client, the driver can request values from Modbus devices and, as a server, provide values from Essential Data to Modbus clients. Available for Commander and ObSys.

This document relates to Modbus driver version 3.0

Please read the *Commander Manual* or *ObSys Manual* alongside this document, available from *www.northbt.com*

Contents

Compatibility with the Modbus System

The Modbus driver allows North to interface with a wide range of equipment supporting the Modbus over TCP/IP, Modbus over serial-line, and JBus protocols. The driver can both request values from Modbus devices, and provide values to a Modbus front-end when requested.

Modbus uses a client-server model. As a client, the driver is capable of both requesting values from Modbus units (i.e. power meters, controllers, etc.), and as a server, providing values from the Essential Data and Extra Data within the North device when requested by a Modbus client (i.e. display or front-end).

The driver connects to an IP network, and can access up to 30 Modbus TCP/IP units on the network (Fig. 1). The driver is also capable of being accessed by two TCP/IP client devices simultaneously.

The driver also connects using the North device's serial port, typically via an RS485 converter, to Modbus over serial-line equipment (Fig. 1). In client mode, a network of Modbus units can be accessed (see *Prerequisites: Driver as Modbus Serial Client* for limitations). In server mode, the driver is capable of being accessed by a single client/master device.

The JBus protocol is fully compatible with Modbus over serial-line.

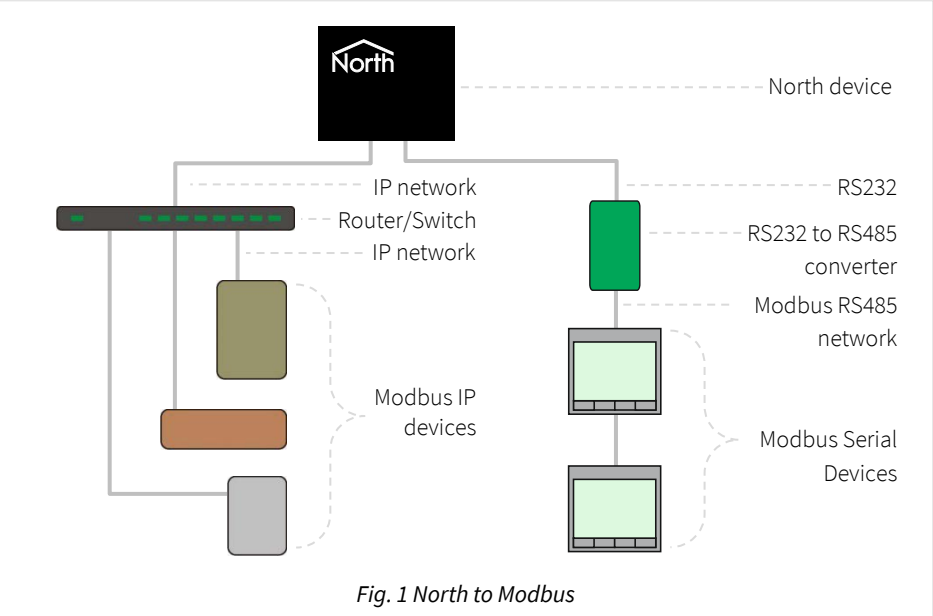


Fig. 1 North to Modbus

Equipment

Many different types of Modbus equipment are compatible with the driver, including:

- Energy meters
- Generators
- Variable speed drives
- PLCs
- AHUs
- Control systems

Equipment is available from many different manufacturers, including ABB, Autometers, Carel, Carlo Gavazzi, Carrier, Ciat, Daikin, Danfoss, Mitsubishi, Schneider, Siemens, Socomec, Stulz, Swegon, Tyco, plus many more.

Application Notes are available for some of these devices, search North product documentation.

Values

The Modbus driver can act as a Modbus client, reading and writing values from a Modbus device elsewhere on the Modbus network. The driver can also act as a Modbus server, allowing other Modbus clients to read and write values within the Essential Data.

Driver as Modbus Client

Depending on the type of Modbus equipment, each Modbus device can have the following primary value types available:

- **Coils** – digital output, e.g. enable command
- **Discrete Inputs** – digital input, e.g. off/on, or alarm states
- **Holding registers** – analogue output, e.g. setpoint values
- **Input registers** – analogue input, e.g. meter readings

Read the Modbus register list, available from the equipment manufacturer, for the values available from a specific device.

Driver as Modbus Server

The driver presents values from the Essential Data and Extra Data as Modbus values, accessible to any client device on the Modbus network. Essential Data contains 640 values on Commander, and 1280 values on ObSys. If necessary, start the Extra Data driver (which requires an interface licence) for an additional 1024 values. Access to these values can be controlled by configuring privilege levels within the driver.

Prerequisites

If an *Application Note* is not available for your Modbus unit, you will need a Modbus register list from the equipment manufacturer. This should describe the function codes or commands supported, register addresses available, and how register values are stored (16-bit integer, 32-bit integer, IEEE float, multipliers, etc.)

Driver as Modbus TCP Client

All Modbus devices must be configured with a unique IP address on the TCP/IP network. If the TCP port can be configured, then this should typically be set to 502. If you are connecting to Modbus over serial devices via a gateway, each device must be assigned a unique serial address.

If you are connecting to the Modbus IP network via a firewall, then the driver will require outbound access to controllers on TCP port 502 (Port 502 is reserved by IANA for use by Modbus).

Driver as Modbus TCP Server

The driver will respond to requests from a Modbus client with a unit identifier of 255, 0, or 1.

Driver as Modbus Serial Client

We recommend installing only Modbus devices of the same type (manufacturer and model) on a network. Different device types may be incompatible for the following reasons – baud rate, byte format, inter-frame delay, and RS485 isolation.

All Modbus devices must be configured with a unique address on the network, and the following common parameters: baud rate, byte format.

Modbus RTU operating mode is supported.

An RS232-485 adapter is required for RS485 devices. Set the baud rate and data bits to match the devices.

The RS485 standard allows at least 32 devices on a network. However, the maximum number depends on the unit load of each device on the network – typically 64 devices with a 0.5 unit load, or 128 devices with a 0.25 unit load. The Modbus standard and this driver support up to 247 addresses.

Driver as Modbus Serial Server

Configure the driver with a unique address on the Modbus network, and the following common parameters: baud rate, byte format.

Modbus RTU operating mode is supported.

Using the Driver

On ObSys and Commander, the Modbus driver is pre-installed. Using all of these North devices, you can use the driver to create an interface to a Modbus system. Once started, you will need to set up the driver before it can communicate with the Modbus system.

The Modbus driver uses zero licence units.

Starting the Interface

To start an interface using the Modbus driver, follow these steps:

- → **Start Engineering** your North device using ObSys
- → Navigate to **Configuration, Interfaces,** and set an unused **Interface** to 'Modbus' to start the particular interface
- → Navigate to the top-level of your North device and re-scan it

The driver setup object (Mc), labelled **Modbus Setup**, should now be available.

Making the Cable

RS485 Devices

Connect the North device COM port to an RS232 to RS485 adapter.

Using the RS485 cable specification (Fig. 2), connect the RS485 adapter to the Modbus device network.

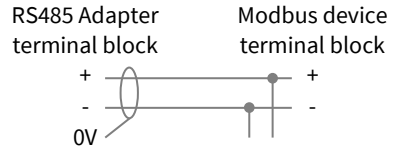


Fig. 2 RS485 adapter to Modbus cable

RS485 adapters are available from North, order code MISC/RS232/485. This adaptor has a unit load of 1.

RS232 Devices

Connect the North device COM port to the Modbus device.

The cable specification will change for each type of Modbus device. Read the *Application Note* or manufacturer's documentation on your specific model for more information.

The maximum RS232 cable length is 15m and should be as short possible.

Setting up the Driver

To set up the driver, follow these steps:

- → Navigate to the **Modbus Setup** object (Mc). For example, if you started interface 1 with the driver earlier, then the object reference will be 'M1'
- → Navigate to **Modbus TCP Client Setup, TCP Unit 1** and set the **Label** and **IP Address** of a Modbus TCP unit that you wish to access values within. If the device is connected via a Modbus gateway, then set the **Serial Address**
- → Follow additional steps on the *Application Note*, if available for the device
- → Repeat for each Modbus TCP unit that your wish to access

- → Navigate to **Modbus Serial Setup** and set the **RS232 Com Port** to the port number of the North device you are connecting to Modbus
- → Set **Modbus Serial Mode** to 'Client' or 'Server' operation
- → Set **Baud Rate** to match the Modbus devices, typically 9600, 19200 or 38400 baud
- → Set the **Byte Format** to the parity and stop bits configured in the Modbus devices
- → If 'Client' mode is enabled, navigate to **Serial Client Setup** and follow additional steps on the *Application Note*, if available for the device
- → If 'Server' mode is enabled, navigate to **Serial Server Setup** and set the **Address on Modbus network** to a unique address on the network.

Checking Communications

To check Modbus client operation, scanning the **Modbus System** will first respond with all units configured with an IP address in Modbus TCP Client Setup, and then automatically detect any Modbus serial units connected. You can check a unit is communicating by and viewing values within it.

To check Modbus server operation, navigate to **Modbus TCP Server Setup** then **Network** to view interfaces open on the North device and their IP address.

Operation as a Modbus Client

Data Model

A Modbus server device stores data using four primary tables, which can each be accessed by the driver.

Table	Data type	Adjustable	Used for
Coils	1-bit data values	Read-Write	Digital outputs
Discrete Inputs	1-bit data values	Read-only	Digital inputs
Holding Registers	16-bit data values	Read-Write	Analogue outputs: setpoints, calculated values
Input Registers	16-bit data values	Read-only	Analogue inputs: sensor readings, meter values

Each table can contain up to 65536 entries, addressed in the range 0 to 65535 (Fig. 3).

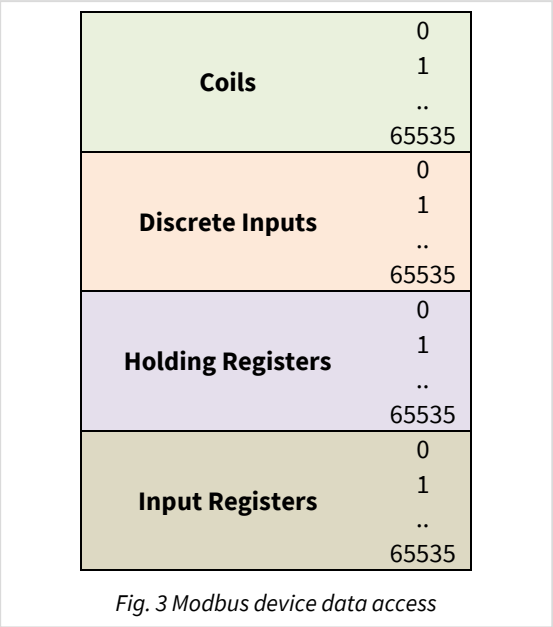


Fig. 3 Modbus device data access

Implementations of the Modbus protocol in a device can vary:

- It is common that these four tables may overlap in a device, so a single address range is used
- Distinctions between inputs and outputs, or 1-bit and 16-bit data values may be blurred
- Multiple consecutive table entries are combined to store a larger data value, e.g. 32-bit or 64-bit data values
- Register addresses are often documented in the range 1 to 65536. To rescale these in the 0 to 65535 range of the Modbus protocol, you will need to subtract 1. E.g. Register 100 becomes 99.

Supported Function Codes

A client can read and write values in the tables using different Modbus function codes. A particular server device may only support some of these codes. The driver function is used as part of the object reference described later.

Function Code (read)	Function Code (write)	Action	Driver Function
01	05	Read/Write Coil	C
01	15	Read/Write Coil	U
02		Read Discrete Input	A
03	06	Read/Write single Holding Register	D
03	16	Read/Write multiple Holding Registers	E to L
04		Read single Input Register	B
04		Read multiple Input Registers	M to T

Value Decoding

The Modbus protocol only describes storing values as either a 1-bit digital or 16-bit register value. All implementations of Modbus have variations from this, including:

- 32-bit and 64-bit integer values (including LSW-MSW order)
- IEEE floating point values (including LSW-MSW order)
- Bit array in register
- Multipliers to change register data from integer
- ASCII string
- Byte-order changed

The driver has several decode types available that are used to translate a raw Modbus register, from a controller, into a value. The decode is used as part of the object reference described later.

Digital State (Decode A)

The value stored within a discrete input or coil entry is always 0 or 1, and always decodes to 0 or 1.

Unsigned Integer (Decode B and 0)

The value stored within the register entry decodes to an unsigned number. For a single register, this will be in the range 0 to 65535.

Either one, two, or four registers can be decoded to a 16-bit, 32-bit, or 64-bit value respectively. In multi-register values, decode B assumes MSW in the first register, and decode O assumes LSW in the first register.

Examples

The single register value of 0x4849 (hex) will decode to 18505 (decimal). The multi-register values of 0x0012 and 0x3456 will decode to 1193046.

Signed Integer (Decode C and P)

The value stored within a register entry decodes to a signed integer number. For a single register, this will be in the range -32768 to 32767.

Either one, two, or four registers can be decoded to a 16-bit, 32-bit, or 64-bit value respectively. In multi-register values, decode C assumes MSW in the first register, and decode P assumes LSW in the first register

Examples

The single register value of 0x4849 (hex) decodes to 18505 (decimal), and the single register value of 0xB7B7 decodes to -18505.

BCD in Lower Nibbles (Decode D)

The value stored within the lower nibbles of the register(s) decodes to a binary decoded decimal value.

One, two, three, or four register values can be decoded.

Example

The two lower nibbles of a single register decodes as Sum(V): 80+8+1 = 89

	Unused nibble				Lower nibble				Unused nibble				Lower nibble			
Bit value (V)					80	40	20	10					8	4	2	1
Register value	0	1	0	0	**1**	0	0	0	0	1	0	0	**1**	0	0	**1**

BCD in Register (Decode E)

The value stored within the register(s) decodes to a binary coded decimal value.

One, two, three, or four register values can be decoded.

Example

A single register value of 0x4849 (hex) decodes as Sum(V): 4000+800+40+8+1 = 4849

Bit value (V)	8000	4000	2000	1000	800	400	200	100	80	40	20	10	8	4	2	1
Register value	0	**1**	0	0	**1**	0	0	0	0	**1**	0	0	**1**	0	0	**1**

ASCII value in LSB (Decode F)

The value stored within the least significant byte of the register(s) decodes to a single ASCII character. Up to 16 registers can be accessed at once, i.e. 16 characters.

Example

The LSB of a single register decodes as a single character: 64+8+1 = ASCII code 73 = 'I'

									ASCII character							
Bit value									128	64	32	16	8	4	2	1
Register value	0	1	0	0	1	0	0	0	0	**1**	0	0	**1**	0	0	**1**

ASCII String (Decode G)

The value stored within each register decodes to two ASCII characters. Up to 16 registers can be accessed at once, i.e. 32 characters.

Example

A single register decodes as two characters: 64+8 = ASCII code 72, and 64+8+1 = ASCII code 73. The full string is 'HI'.

	First ASCII character								Second ASCII character							
Bit value	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
Register value	0	**1**	0	0	**1**	0	0	0	0	**1**	0	0	**1**	0	0	**1**

Unsigned Integer in LSB (Decode H)

The value stored within the least significant byte (LSB) of a single register decodes to an unsigned number, a byte in the range 0 to 255. When writing, the register is first read to preserve the MSB.

Example

The LSB of a single register decodes as Sum(V): 64+8+1 = 73

	Unused MSB								LSB							
Bit value (V)									128	64	32	16	8	4	2	1
Register value	0	1	0	0	1	0	0	0	0	**1**	0	0	**1**	0	0	**1**

Unsigned Integer in MSB (Decode I)

The value stored within the most significant byte (MSB) of a single register decodes to an unsigned number, a byte in the range 0 to 255. When writing, the register is first read to preserve the LSB.

Example

The MSB of a single register decodes using Sum(V): 64+8 = 72

	MSB								Unused LSB							
Bit value (V)	128	64	32	16	8	4	2	1								
Register value	0	**1**	0	0	**1**	0	0	0	0	1	0	0	1	0	0	1

IEEE Float (Decode J and M)

The value stored within two registers decodes to an IEEE floating-point number. A four register value can be decoded to a double-precision floating-point number.

Use decode type J for big-endian values (MSW in first register and LSW in second), or decode type M for little-endian values (LSW in first register and MSW in second).

Decode type J can also decode a single register as a half-precision floating-point number.

Single Bit of Register (Decode K)

Returns the specified bit from a register value. Bits are indexed starting with the most significant byte (MSB) of the first register. This is shown below as bit index 7 to 0 (MSB), followed by 15 to 8 (LSB). Note how this differs from the traditional bit numbering of a 16-bit value (shown 15 to 0 below the register value).

	MSB								LSB							
Bit index (K)	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
Register value																
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Multiple register values are supported. When writing, the register is first read to preserve remaining bits.

Example

If a register has the value 0x4849 (hex). Bit indexes 8, 11, 14, 3, and 6 have the value 1. All other bits have the value 0.

Bit index	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
Register value	0	**1**	0	0	**1**	0	0	0	0	**1**	0	0	**1**	0	0	**1**

Bit Mask (Decode L)

Performs a bitwise AND operation on a register value and mask value. The result indicates which bits of the mask value are also set in the register value.

Example

If a register has the value 18505 (decimal). To find the value of bits 3 to 6, we first calculate the mask value: Sum(V) = 64+32+16+8 = 120. So, Register AND Mask = 18505 AND 120 = 72.

Bit values (V)	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Register value	0	1	0	0	1	0	0	0	0	**1**	**0**	**0**	**1**	0	0	1
Mask value	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	0	0	0

Bit Mask with Bit Shift (Decode N)

Performs a bitwise AND operation on a register value and mask value, followed by a bit-shifting operation to move all bits to the right by the specified number.

Example

If a register holds a value in bits 3 to 6. First extract this value using the AND Mask value 120 (64+32+16+8 = 120). Then bit shift this value 3 positions to the right (rescaling the value to base 0).

If a register has the value 18505 (decimal): 18505 AND 120 = 72 >> 3 = 9

Bit values	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Mask value	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	0	0	0
Register value	0	1	0	0	1	0	0	0	0	**1**	**0**	**0**	**1**	0	0	1
Bit shift result	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**	**1**

Operation as a Modbus Server

The values from Essential Data may be accessed using any of the supported Modbus function codes. A single address range is used for all functions – 0…639 on Commander, and 0…1279 on ObSys.

Both the Modbus TCP Server Setup and Serial Server Setup objects contain a **Register List** (RL) object detailing the registers available from the North device. Refer also to *Appendix A* for a full list of Essential Data to Modbus register address mapping.

If ExtraData is used, the extra 1024 values appear in registers that follow on from the existing Essential Value registers – i.e. Registers 640…1663 on Commander, or 1280…2303 on ObSys.

Essential Data Value Translation

The North device has only one data table – Essential Data – and so maps all Modbus table requests to Essential Data. The Essential Data value is translated to a Modbus register or state depending which Modbus Table is specified and how the value is configured in Essential Data:

	Essential Data Type			
Modbus Table	**Number**	**Float**	**NoYes or OffOn**	**ENum**
Input Register **Holding Register**	16-bit unsigned register value in the range 0…65535. When reading and the value is negative, the register is converted to a 16-bit signed value in the range -32768…32767	Value scaled 10^{dp} (*dp* is the number of decimal places configured in Essential Data). 16-bit unsigned register value in the range 0…65535. When reading and the value is negative, the register is converted to a 16-bit signed value in the range -32768…32767	'No' and 'Off' states are converted to the value 0 'Yes' and 'On' states are converted to the value 1 16-bit unsigned register value in the range 0…1.	16-bit unsigned register value in the range 0…65535
Discrete Input **Coil**	Binary on/off state. If the value is zero (0) then an 'off' state is returned, any other value returns an 'on' state			

When reading, if the Essential Data object has an Access Security level that does not allow Modbus access to it (see *Security*), a '0' value is returned. When writing, single value writes (Function Codes 05 and 06) will have an error message returned; multi-value writes (Function Codes 15 and 16) will not.

If the Essential Data object has Adjustable set to 'No', and the incoming message attempts to write to it, the write does not occur: single value writes (Function Codes 05 and 06) will be returned an error message, multi-value writes (Function Codes 15 and 16) will not.

Supported Function Codes

The driver supports the following Function Codes. Some Modbus messages can contain a quantity of items, and the table shows the range of quantities supported by the driver for the various codes:

Function Code	Action	Quantity
01	Read Coils	1 to 255
02	Read Discrete Inputs	1 to 255
03	Read Holding Registers	1 to 125
04	Read Input Registers	1 to 125
05	Write Single Coil	n/a
06	Write Single Holding Register	n/a
15	Write Multiple Coils	1 to 255
16	Write Multiple Holding Registers	1 to 123

Object Specifications

Once an interface is started, one or more extra objects become available within the top-level object of the device. As with all North objects, each of these extra objects may contain sub-objects, (and each of these may contain sub-objects, and so on) – the whole object structure being a multi-layer hierarchy. It is possible to navigate around the objects using the ObSys Engineering Software.

Each object is specified below, along with its sub-objects.

Example Object Reference

An example of a reference to an object in the same device: the Modbus System (S1) contains a Unit (U1), which contains an Input register (B2.B). Therefore, the object reference will be 'S1.U1.B2.B'.

An example of a reference to an object in a different device: the IP network object (IP) contains Default Commander object (CDIP), which contains the object above (S1.U1.B2.B) – therefore the complete object reference is 'IP.CDIP.S1.U1.B2.B'.

Device Top-Level Objects

When an interface is started using the Modbus driver, the objects below become available within the top-level object of the device. For example, if Interface 1 is started, then the object with references 'M1' and 'S1' become available.

Description	Reference	Type
Modbus Setup Set up the Modbus driver, started on interface *c* (*c* is the interface number)	M*c*	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30]* On the ObSys platform this will be *[OSM v20\Modbus v30]*
Modbus System Access Modbus systems connected to interface *c* (*c* is the interface number)	S*c*	Variable Container: *[Modbus]*

Modbus Setup

Object Type: *[OSM v20\Modbus v30]*
Object Type: *[CDM v20\Modbus v30]*

The Modbus Setup Module contains the following objects:

Description	Reference	Type
System Label Label displayed when scanning the system	DL	Obj\Text: 20 chars max; Adjustable
Modbus TCP Client Setup Enable and configure Modbus TCP client operation	TC	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Client]* On the ObSys platform this will be *[OSM v20\Modbus v30\Client]*
Modbus TCP Server Setup Enable and configure Modbus TCP server operation	TS	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Server]* On the ObSys platform this will be *[OSM v20\Modbus v30\Server]*
Modbus Serial Setup Enable and configure Modbus serial client or server operation	RS	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Serial]* On the ObSys platform this will be *[OSM v20\Modbus v30\Serial]*
Advanced Settings Set advanced parameters for Modbus operation	A	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Advanced]* On the ObSys platform this will be *[OSM v20\Modbus v30\Advanced]*

Modbus TCP Client Setup

Object Type: *[OSM v20\Modbus v30\Client]*
Object Type: *[CDM v20\Modbus v30\Client]*

The Modbus TCP Client Setup object is used to enable client operation in the driver, and add the details of Modbus TCP devices available on the IP network.

Description	Reference	Type
Enable TCP Client	E	Obj\NoYes; Adjustable
TCP Unit *x* Configure Modbus TCP client device address. Unit number, *x*, is in the range 1…30	U*x*	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Client\Unit]* On the ObSys platform this will be *[OSM v20\Modbus v30\Client\Unit]*

TCP Unit

Object Type: *[OSM v20\Modbus v30\Client\Unit]*
Object Type: *[CDM v20\Modbus v30\Client\Unit]*

The TCP Unit object is used to configure the address of a Modbus TCP device on the network.

Description	Reference	Type
Label Label displayed when scanning the Modbus system	L	Obj\Text: 20 chars; Adjustable
IP Address	IA	Obj\IP; Adjustable
TCP Port	PN	Obj\Num: 1…65535; Adjustable Default: 502
Serial Address If connecting to a single Modbus TCP device, set to the address 255. If connecting via a Modbus gateway, set to the Modbus over serial device address.	A	Obj\Num: 1…255; Adjustable Value 255: Modbus TCP device (default) Value 1…247: Modbus serial address
Device Type Leave blank to view default Modbus objects for a device. Refer to *Application Note* for other types available	DT	Obj\Text: 20 chars; Adjustable

Modbus TCP Server Setup

Object Type: *[OSM v20\Modbus v30\Server]*
Object Type: *[CDM v20\Modbus v30\Server]*

The Modbus TCP Server Setup object is used to enable server operation in the driver, and select which network interface the protocol is accessible on.

Description	Reference	Type
Enable TCP Server	E	Obj\NoYes; Adjustable
Network Configure the local IP address and TCP port for Modbus	*N*	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Network]* On the ObSys platform this will be *[OSM v20\Modbus v30\Network]*
Security Configure privilege levels to control read and adjust access to Essential Data and Extra Data from a Modbus TCP client	S	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Privs]* On the ObSys platform this will be *[OSM v20\Modbus v30\Privs]*
Register List List of registers made available from Essential Data to a connected Modbus TCP client. Useful for documentation.	RL	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\RegList]* On the ObSys platform this will be *[OSM v20\Modbus v30\RegList]*

Network

Object Type: [OSM v20\Modbus v30\Network]
Object Type: [CDM v20\Modbus v30\Network]

Configure the Modbus Network connectivity using this object. By default, all available IP addresses are opened for requests on TCP port 502.

If required, change the TCP port number or restrict access to a single IP address on the North device.

Description	Reference	Type
Interfaces open Reports the number of interfaces Modbus is available on	C	Obj\Num: 0…8
Force single IP address Force the driver to use only one of the interface IP addresses. By default, all are used when the address is '0.0.0.0'	IA	Obj\IP; Adjustable
TCP Port TCP port number	PN	Obj\Num: 1…65535; Adjustable Default 502
Interface *x* Status information for a network interface available on the North device. The interface number, *x*, is in the range 1…4 on Commander and 1…8 on ObSys	I*x*	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Network\Interface]* On the ObSys platform this will be *[OSM v20\Modbus v30\Network\Interface]*

Network Interface

Object Type: [OSM v20\Modbus v30\Network\Interface]
Object Type: [CDM v20\Modbus v30\Network\Interface]

An Interface represents a physical or virtual network interface on the North device. Use this object to find out the IP address available and if Modbus TCP has opened a port to listen for requests.

Description	Reference	Type
IP address IP address available for the interface	IA	Obj\IP
Port open Indicates if Modbus has opened a port on the interface	S	Obj\NoYes

Modbus Serial Setup

Object Type: *[OSM v20\Modbus v30\Serial]*
Object Type: *[CDM v20\Modbus v30\Serial]*

The Modbus Serial Setup object is used to select client or server operation in the driver, and configure the serial port.

Description	Reference	Type
Modbus Serial Mode Select operating mode for driver on the serial interface. Select 'Client' to request values from other Modbus devices. Select 'Server' if a Modbus master/client will request values from the North device	M	Obj\ENum; Adjustable Values: 0=None, 1=Client, 2=Server
RS232 COM Port	COM	Obj\Num: 1…8; Adjustable
Baud Rate	BR	Obj\Num; Adjustable; Default: 19200 Range: 1200, 2400, 4800, 9600, 19200 or 38400
Byte Format Set the parity and stop bits	BF	Obj\ENum: 0…9; Adjustable; Default: 8 (Even/1) See note 1
Serial Client Setup Configure Modbus serial client operation	C	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Serial\Client]* On the ObSys platform this will be *[OSM v20\Modbus v30\Serial\Client]*
Serial Server Setup Configure Modbus serial server operation	S	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Serial\Server]* On the ObSys platform this will be *[OSM v20\Modbus v30\Serial\Server]*

Notes

1 Modbus RTU uses 8 data bits. Byte format can have the following values:

Value	Parity	Stop bits	Notes	Data Format for RS232-485 converter
0	None	1	2 stop bits are recommended when using no parity	10-bits
1	None	2		11-bits
4	Odd	1		11-bits
5	Odd	2		12-bits
8	Even	1	Default value	11-bits
9	Even	2		12-bits

Modbus Serial Client Setup

Object Type: *[OSM v20\Modbus v30\Serial\Client]*
Object Type: *[CDM v20\Modbus v30\Serial\Client]*

The Modbus Serial Client Setup object is used to optionally set a default device type and add details of Modbus serial devices available on the RS485 network.

Description	Reference	Type
Serial Unit *x* Configure Modbus client device address. Unit number, *x*, is in the range 1…30	U*x*	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Serial\Unit]* On the ObSys platform this will be *[OSM v20\Modbus v30\Serial\Unit]*

Serial Unit

Object Type: *[OSM v20\Modbus v30\Serial\Unit]*
Object Type: *[CDM v20\Modbus v30\Serial\Unit]*

The Serial Unit object is used to optionally configure the address of a Modbus serial device on the network.

Description	Reference	Type
Label Label displayed when scanning the Modbus system	L	Obj\Text: 20 chars; Adjustable
Serial Address Modbus over serial device address	A	Obj\Num: 1…247; Adjustable
Device Type Leave blank to view default Modbus objects for a device. Refer to *Application Note* for other types available	DT	Obj\Text: 20 chars; Adjustable

Modbus Serial Server Setup

Object Type: *[OSM v20\Modbus v30\Serial\Server]*
Object Type: *[CDM v20\Modbus v30\Serial\Server]*

The Modbus Serial Server Setup object is used to configure the address of the North device on the Modbus serial network.

Description	Reference	Type
Address on Modbus network Set to a unique address on the RS485 network	ADDR	Obj\Num; 1…247; Adjustable
Security Configure privilege levels to control read and adjust access to Essential Data and Extra Data from a Modbus serial client	S	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\Privs]* On the ObSys platform this will be *[OSM v20\Modbus v30\Privs]*
Register List List of registers made available from Essential Data to a Modbus serial client. Useful for documentation.	RL	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\RegList]* On the ObSys platform this will be *[OSM v20\Modbus v30\RegList]*

Security

Object Type: *[OSM v20\Modbus v30\Privs]*
Object Type: *[CDM v20\Modbus v30\Privs]*

Security Areas and Levels

Within the North security model, there are eight security areas. Security areas could be actual areas in a building, but are normally functional areas – for example, 'environmental control' and 'North engineering' areas would allow a user to have different privileges in controlling set points and engineering Commanders.

Typically, a user is assigned a privilege level in each of the eight areas. The level is in the range zero to seven, seven being the most powerful. When a user wishes to pass a door, his/her privilege level in the door's area is checked against the minimum required for that area – and then either allowed to pass, or rejected.

The engineer must decide the use of the eight areas. The engineer must also decide the power of the privilege levels. Most systems use only a few levels per area: 0=None, 1=Guest, 2=User, 7=Administrator.

As an example, imagine a page of values in Essential Data. The page needs a user to have a minimum privilege level of 2 in area 1 before it can be viewed. The page is available in a Web browser that checks users with a security database. User A has privilege level 7 in area 1 – she can view the page. User B has privilege level 5 in area 1 – he can also view the page. User C has privilege level 1 in area 1 – she cannot view the page.

The example continues: within this page of values in Essential Data is a temperature set point object. Users need a minimum privilege level of 6 in area 1 to adjust it – therefore User A can adjust the set point, but User B cannot.

Specifying Access Security

Essential Data and Extra Data have Access Security objects to control who can view a page, and who can adjust an adjustable object.

Each Access Security object has a two-digit value. Each controls the access to a particular feature - such as viewing the page or adjusting the value. The two-digit value is made up of the area digit (1-8), followed by the minimum privilege level (1-7) – for example, if the minimum privilege level is 6 in area 2, then the two-digit value is 26. If the value is 00, then no security checks are made.

Modbus Driver

The Security object contains a privilege level for each of the eight security areas, representing a virtual user. The Modbus driver uses these to control access to Essential Data and Extra Data when reading or adjusting a value.

Description	Reference	Type
Privilege Level in Area *x* The area, *x*, can be in the range 1…8	P*x*	Obj\Num; Adjustable; Range: 0…7

Register List

Object Type: *[CDM v20\Modbus v30\RegList]*
Object Type: *[OSM v20\Modbus v30\RegList]*

The Register List object contains the list of available Modbus registers presented from the North device's Essential Data. This list is provided for documentation and fault-finding purposes.

Description	Reference	Type
Registers Available Count of maximum objects available from Essential Data and Extra Data	EDC	Obj\Num
Register *x* The register address, *x*, can be in the range 0…640 on Commander, and 0…1024 on ObSys. If Extra Data is used, the register address range is extended to 1663 on Commander, and 2303 on ObSys.	A*x*	Fixed Container: On the Commander platform this will be *[CDM v20\Modbus v30\RegList\Addr]* On the ObSys platform this will be *[OSM v20\Modbus v30\RegList\Addr]*

Register

Object Type: *[CDM v20\Modbus v30\RegList\Addr]*
Object Type: *[OSM v20\Modbus v30\RegList\Addr]*

A Register contains a single register address, holding a value from Essential Data. Use this object to view the value (V) held by Essential Data and the 16-bit Modbus register value (RV). To assist with documentation, the object includes label (L), multiplier factor (MF) and adjustability (A) objects.

Description	Reference	Type
Label Label from Essential Data	L	Obj\Text
Value Value from Essential Data	V	Obj\Text
Register Value Value converted to a 16-bit Modbus value	RV	Obj\Num: 0…65535
Multiplier Factor Multiplication applied to the value to obtain Register Value	MF	Obj\Num: 1…10000
Adjustable Indicates if the register can be written to using Modbus function codes 05 or 06	A	Obj\NoYes

Advanced Settings

Object Type: *[OSM v20\Modbus v30\Advanced]*
Object Type: *[CDM v20\Modbus v30\Advanced]*

The Advanced Settings object contains the following advanced configuration objects:

Description	Reference	Type
Maximum Requests Maximum number of simultaneous requests from the interface to connected Modbus client devices	MR	Obj\Num: 1…3; Adjustable Default: 3
Reply Timeout (ms) Maximum time to wait from sending a request to receiving a reply from a Modbus device	TO	Obj\Num: 250…2000; Adjustable Default: 2000ms
Inter-Frame Delay (ms) On Modbus serial connections, time to wait between message frames. As a minimum, this value must be the time taken to send 3.5 characters.	T3	Obj\Num: 5…1000; Adjustable Default: 50ms
Register Byte Order The Modbus protocol uses a big-endian byte order. For non-standard client devices use this object to reverse the byte order. Refer to *Application Note* or contact support for help. Server operation is always big-endian	BO	Obj\ENum; Adjustable Values: 0=Big-endian, 1=Little-endian Default: big-endian
Formula *y* User defined mathematical formula, used in decoding values from device. The formula number, *y*, is in the range 1…20	F*y*	Fixed Container: *[Standard\AMFormula]*
Debug Enable This will store additional debug information in the record file. Use this option only when instructed by North Support	DE	Obj\NoYes; Adjustable

Formula Setup

Object Type: *[Standard\AMFormula]*

A standard formula setup is a maths module that allows values to be converted into engineering units.

This module allows a simple formula to be applied to a Modbus register value so that the resulting object value contains a meaningful value.

The module can convert the number into an object value using the formula:

```
real-value = (M x raw-value) + A
```

The formula values M and A are engineer-defined. When writing, the module applies the formula below:

```
raw-value = (real-value – A) / M
```

Example

A register value contains a temperature value, where value 0 = -50°C and value 65535 = +50°C.

If A is set to -50 and M=0.0015259, then the formula would be:

temperature = (0.0015259 x register-value) + -50

So, the following temperatures can be calculated from the register values:

Register value	**Temperature**
0	-50°C
32767	0°C
49150	25°C
65535	50°C

The module contains the following objects:

Description	Reference	Type
Addition value	A	Obj\Float; Adjustable
Multiplication value	M	Obj\Float; Adjustable

Modbus System

Object Type: *[Modbus]*

The Modbus system contains objects to access the Modbus client devices available.

Description	Reference	Type
Unit x The unit address, *x,* can be in the range 1…30.	U*x*	Fixed container, one of the following: Default Modbus Device *[Modbus\Default]* If *Device Type* is configured then the container will be of the type [Modbus*Device Type*]. See *Modbus TCP Client Setup*
Serial Address y The serial address, y, can be in the range 1…247.	A*y*	Fixed container, one of the following: Default Modbus Device *[Modbus\Default]* If *Device Type* is configured then the container will be of the type [Modbus*Device Type*]. See *Modbus Serial Client Setup*

Default Modbus Device

Object Type: [Modbus\Default]

A default Modbus device contains a generic list of objects that enable you to access the values in a device. Use this with the device manufacturer's register list. For a full description of Modbus values and how to decode them, refer to *Operation as a Modbus Client* earlier in this document.

Frequently Used Objects

Here we list a summary of the most frequently used decode objects. Refer to the *Notes* section below for additional information, and *Full Object List* below for the complete list of objects available.

Description	Reference	Type
Coil *r* – State The digital output, *r*, is in the range 0…65535 (see note 1)	C*r*.A	Obj\OffOn; Adjustable
Discrete Input *r* – State The digital input, *r*, is in the range 0…65535 (see note 1)	A*r*.A	Obj\OffOn
Holding Register *r* – Unsigned 16-bit Integer The register, *r*, is in the range 0…65535 (see note 1).	D*r*.B	Obj\Num; Range: 0…65535; Adjustable
Holding Register *r* – Unsigned 16-bit (x10)	D*r*.B26	Obj\Float; Range: 0…6553.5; Adjustable
Holding Register *r* – Unsigned 16-bit (x100)	D*r*.B27	Obj\Float; Range: 0…655.35; Adjustable
Holding Register *r* – Unsigned 16-bit (x1000)	D*r*.B28	Obj\Float; Range: 0…65.535; Adjustable
Holding Register *r* – Signed 16-bit Integer	D*r*.C	Obj\Num; Range: -32768…32767; Adjustable
Holding Register *r* – Signed 16-bit (x10)	D*r*.C26	Obj\Float; Range: -3276.8…3276.7; Adjustable
Holding Register *r* – Signed 16-bit (x100)	D*r*.C27	Obj\Float; Range: -327.68…327.67; Adjustable
Holding Register *r* – Signed 16-bit (x1000)	D*r*.C28	Obj\Float; Range: -32.768…32.767; Adjustable
Holding Register *r* – IEEE Float Single precision, stored in two registers (MSW, LSW)	F*r*.J	Obj\Float; Adjustable
Holding Register *r* – Unsigned 32-bit Integer Stored in two registers (MSW,LSW) (see note 2)	F*r*.B	Obj\Num; Range: 0…4294967295; Adjustable
Holding Register *r* – Signed 32-bit Integer Stored in two registers (MSW,LSW) (see note 2)	F*r*.C	Obj\Num; Range: -2147483648…2147483647; Adjustable
Holding Register *r* – Unsigned 64-bit Integer Stored in four registers (MSW…LSW) (see note 2)	H*r*.B	Obj\Text; Range: 0…18446744073709551615; Adjustable
Holding Register *r* – Signed 64-bit Integer Stored in four registers (see note 2)	H*r*.C	Obj\Text; Range: -923372036854775808… 923372036854775807; Adjustable
Holding Register *r* – IEEE Double Float Double precision, stored in four registers (MSW… LSW) (see note 2)	H*r*.J	Obj\Text; Adjustable

Description	Reference	Type
Holding Register *r* – Bit *b* The bit number, *b*, can be in the range 0…15. Refer to *Single Bit of Register* section earlier in this manual	D*r*.K*b*	Obj\OffOn; Adjustable
Input Register *r* – Unsigned 16-bit Integer The register, *r*, is in the range 0…65535 (see note 1)	B*r*.B	Obj\Num; Range: 0…65535
Input Register *r* – Unsigned 16-bit (x10)	B*r*.B26	Obj\Float: Range: 0…6553.5
Input Register *r* – Unsigned 16-bit (x100)	B*r*.B27	Obj\Float; Range: 0…655.35
Input Register *r* – Unsigned 16-bit (x1000)	B*r*.B28	Obj\Float; Range: 0…65.535
Input Register *r* – Signed 16-bit Integer	B*r*.C	Obj\Num; Range: -32768…32767
Input Register *r* – Signed 16-bit (x10)	B*r*.C26	Obj\Float: Range: -3276.8…3276.7
Input Register *r* – Signed 16-bit (x100)	B*r*.C27	Obj\Float: Range: -327.68…327.67
Input Register *r* – Signed 16-bit (x1000)	B*r*.C28	Obj\Num; Range: -32.768…32.767
Input Register *r* – IEEE Float Single precision, stored in two registers (MSW,LSW)	N*r*.J	Obj\Float
Input Register *r* – Unsigned 32-bit Integer Stored in two registers (MSW,LSW) (see note 2)	N*r*.B	Obj\Num; Range: 0…4294967295
Input Register *r* – Signed 32-bit Integer Stored in two registers (MSW,LSW) (see note 2)	Nr.C	Obj\Num; Range: -2147483648…2147483647
Input Register *r* – Unsigned 64-bit Integer Stored in four registers (MSW…LSW) (see note 2)	P*r*.B	Obj\Text; Range: 0…18446744073709551615
Input Register *r* – Signed 64-bit Integer Stored in four registers (MSW…LSW) (see note 2)	P*r*.C	Obj\Text; Range: -923372036854775808… 923372036854775807
Input Register *r* – IEEE Double Float Double precision, stored in four registers (MSW… LSW) (see note 2)	P*r*.J	Obj\Text
Input Register *r* - Bit *b* The bit number, *b*, can be in the range 0…15. Refer to *Single Bit of Register* section earlier in this manual	B*r*.K*b*	Obj\OffOn

Notes

1. The discrete input, coil, or register number, *r*, is in the range 0…65535. Manufacturers sometimes document the register number in the range 1…65536. To rescale these in the 0 to 65535 range from the Modbus protocol, you will need to subtract 1. For example, register 100 becomes 99.
2. Large numbers: 64-bit and 32-bit values can contain up to 20 significant figures. Numbers this size are ok for displaying to a user, but may be too large to perform accurate maths functions. These values can be read in blocks of six significant figures by appending the object reference with a block number. Block 1 reads the six least significant figures, block 2 the next six significant figures, etc.
   For example, if object 'H1.B' reads the 64-bit value '6744073709551615', then object 'H1.B.1' will read the least six significant figures '551615', object 'H1.B.2' the value '073709', and object 'H1.B.3' the value '6744'.

You can also use a formula with a block number. The object has the format ‘H1.B28.1’. This will apply the formula first then access the requested block of six significant figures.
If the formula uses a divisor, then the value will be formatted to three decimal places. For example, object ‘H1.B28.1’ will read the value ‘551.615’.

Full Object List

The Modbus driver contains the following objects:

Description	Reference	Type
Function *f*, Entry *r* – Decode *d* The function, *f*, is in the range A…U (see note 3) The entry, *r*, is in the range 0…65535 (see note 1) The decode, *d*, is in the range A…J, L, or M, or O…P (see also note 4). Refer to *Value Decoding* earlier in this manual. For 2 and 4 register values, see note 5.	*Fr.d*	The value type is dependent on the function, f, and decode, d. Adjustable for Coils and Holding Registers
Function *f*, Entry *r* – Decode *d*, Formula *z* As above, but with formula, *z*, applied (see note 2)	*fr.dz*	The value type is dependent on the function, f, decode, d, and formula, z. Adjustable for Coils and Holding Registers
Function *f*, Entry *r* – Bit *b* Refer to *Value Decoding: Single Bit of Register* earlier in this manual. The function, *f*, is in the range D…L, B, N…T (see note 3) The entry, *r*, is in the range 0…65535 (see note 1) The bit number, *b*, can be in the range 0…15.	*Fr*.K*b*	Obj\OffOn; Adjustable for Holding Registers
Function *f*, Entry *r* – Bit Mask *m* Refer to *Value Decoding: Bit Mask* earlier in this manual. The function, *f*, is in the range D…F, B, N (see note 3) The entry, *r*, is in the range 0…65535 (see note 1) The bit mask, *m*, is a number in the range 0…65535.	*Fr*.L*m*	Obj\Num
Function *f*, Entry *r* – Bit Mask *m*, Formula *z* As above, but with formula, *z*, applied (see note 2)	*fr*.L*m.z*	Obj\Float
Function *f*, Entry *r* – Bit Mask *m*, Shift *s* Refer to *Value Decoding: Bit Mask with Bit Shift* earlier in this manual. The function, *f*, is in the range D…F, B, N (see note 3) The entry, *r*, is in the range 0…65535 (see note 1) The bit mask, *m*, is a number in the range 0…65535. The bit shift, *s*, is a number in the range 1…32.	*Fr*.N*m.s*	Obj\Num

Notes

1. The discrete input, coil, or register number, *r*, is in the range 0…65535. Manufacturers sometimes document the register number in the range 1…65536. To rescale these in the 0 to 65535 range from the Modbus protocol, you will need to subtract 1. E.g. Register 100 becomes 99.

2. An optional formula number, *z*, may be applied where indicated above. The formula number is in the range 1…40, where 1…20 refer to user-defined formula, and 21…40 are fixed as follows:

Formula	Multiply	Add
21	10	0
22	100	0
23	1000	0
24	10000	0
25	100000	0
26	0.1	0
27	0.01	0
28	0.001	0
29	0.0001	0
30	0.00001	0

Formula	Multiply	Add
31	2	0
32	5	0
33	0.2	0
34	0.5	0
35	0.05	0
36	0.005	0
37	0.000001	0
38	1	0
39	1	0
40	Four quadrant power factor (Float decode only)	

3. The function, *f*, is the Modbus function or command and can be in the range A…U. Refer to the *Function Codes* section earlier in this document.

Driver Function	Table	Action	Function Code (read)	Function Code (write)
C	Coils	Read/Write digital output	01	05
A	Discrete Inputs	Read digital input	02	
D	Holding Registers	Read/Write 1 output register	03	06
E	Holding Registers	Read/Write 1 output multi-registers	03	16
F	Holding Registers	Read/Write 2 output multi-registers	03	16
G	Holding Registers	Read/Write 3 output multi-registers	03	16
H	Holding Registers	Read/Write 4 output multi-registers	03	16
I	Holding Registers	Read/Write 6 output multi-registers	03	16
J	Holding Registers	Read/Write 8 output multi-registers	03	16
K	Holding Registers	Read/Write 10 output multi-registers	03	16
L	Holding Registers	Read/Write 16 output multi-registers	03	16
B	Input Registers	Read 1 input register	04	
N	Input Registers	Read 2 input multi-registers	04	
O	Input Registers	Read 3 input multi-registers	04	
P	Input Registers	Read 4 input multi-registers	04	
Q	Input Registers	Read 6 input multi-registers	04	
R	Input Registers	Read 8 input multi-registers	04	
S	Input Registers	Read 10 input multi-registers	04	
T	Input Registers	Read 16 input multi-registers	04	
U	Coils	Read/Write 1 digital output	01	15

4. The decode, *d*, is in the range A…Q. Refer to the *Value Decoding* section earlier in this document.

Decode	Use	Object Type
A	Digital State	Obj\OffOn
B	Unsigned Integer	Obj\Num
C	Signed Integer	Obj\Num
D	BCD in lower nibbles only	Obj\Num
E	BCD in register	Obj\Num
F	ASCII in LSB	Obj\Text
G	ASCII string	Obj\Text
H	Unsigned Integer in LSB	Obj\Num
I	Unsigned Integer in MSB	Obj\Num
J	IEEE Float (MSW, LSW order)	Obj\Float
M	IEEE Float (LSW, MSW order)	Obj\Float
K	Single bit of register	Obj\OffOn
L	Bit Mask	Obj\Num
N	Bit Mask with Bit Shift	Obj\Num
O	Unsigned Integer (LSW, MSW order)	Obj\Num
P	Signed Integer (LSW, MSW order)	Obj\Num
Q	Special Decode – Ask North	<various>

5. Large numbers: 64-bit and 32-bit values can contain up to 20 significant figures. Numbers this size are ok for displaying to a user but may be too large to perform accurate maths functions. These values can be read in blocks of six significant figures by appending the object reference with a block number. Block 1 reads the six least significant figures, block 2 the next six significant figures, etc.
   For example, if object ‘H1.B’ reads the 64-bit value ‘6744073709551615’, then object ‘H1.B.1’ will read the least six significant figures ‘551615’, object ‘H1.B.2’ the value ‘073709’, and object ‘H1.B.3’ the value ‘6744’.
   You can also use a formula with a block number. The object has the format ‘H1.B28.1’. This will apply the formula first then access the requested block of six significant figures.
   If the formula uses a divisor, then the value will be formatted to three decimal places. For example, object ‘H1.B28.1’ will read the value ‘551.615’.

Appendix A: Modbus Register List

When the Modbus TCP Server is enabled, or Modbus Serial Mode is set to 'Server', the driver presents values from the North device's Essential Data and Extra Data as Modbus values.

For further information on Modbus function codes supported, refer to the section *Operation as a Modbus Server*.

Essential Data

Essential Data contains 640 values on Commander, and 1280 values on ObSys. These values are arranged in pages of 10, 16, 32, or 48 objects.

The values may be accessed using any of the supported Modbus function codes. A single register address range is used for all functions – 0…639 on Commander, and 0…1279 on ObSys.

This address maps to the Essential Data page/object reference as shown below. Download this list as a spreadsheet from *http://resource.northbt.com/driver/Modbus_RegisterList.xlsx*

Both the Modbus TCP Server Setup and Serial Server Setup objects contain a **Register List** (RL) object detailing the registers available from the North device.

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
0	P1.O1	P1.O1	P1.O1	P1.O1
1	P1.O2	P1.O2	P1.O2	P1.O2
2	P1.O3	P1.O3	P1.O3	P1.O3
3	P1.O4	P1.O4	P1.O4	P1.O4
4	P1.O5	P1.O5	P1.O5	P1.O5
5	P1.O6	P1.O6	P1.O6	P1.O6
6	P1.O7	P1.O7	P1.O7	P1.O7
7	P1.O8	P1.O8	P1.O8	P1.O8
8	P1.O9	P1.O9	P1.O9	P1.O9
9	P1.O10	P1.O10	P1.O10	P1.O10
10	P2.O1	P1.O11	P1.O11	P1.O11
11	P2.O2	P1.O12	P1.O12	P1.O12
12	P2.O3	P1.O13	P1.O13	P1.O13
13	P2.O4	P1.O14	P1.O14	P1.O14
14	P2.O5	P1.O15	P1.O15	P1.O15
15	P2.O6	P1.O16	P1.O16	P1.O16
16	P2.O7	P2.O1	P1.O17	P1.O17
17	P2.O8	P2.O2	P1.O18	P1.O18
18	P2.O9	P2.O3	P1.O19	P1.O19
19	P2.O10	P2.O4	P1.O20	P1.O20
20	P3.O1	P2.O5	P1.O21	P1.O21
21	P3.O2	P2.O6	P1.O22	P1.O22
22	P3.O3	P2.O7	P1.O23	P1.O23
23	P3.O4	P2.O8	P1.O24	P1.O24
24	P3.O5	P2.O9	P1.O25	P1.O25
25	P3.O6	P2.O10	P1.O26	P1.O26
26	P3.O7	P2.O11	P1.O27	P1.O27
27	P3.O8	P2.O12	P1.O28	P1.O28

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
28	P3.O9	P2.O13	P1.O29	P1.O29
29	P3.O10	P2.O14	P1.O30	P1.O30
30	P4.O1	P2.O15	P1.O31	P1.O31
31	P4.O2	P2.O16	P1.O32	P1.O32
32	P4.O3	P3.O1	P2.O1	P1.O33
33	P4.O4	P3.O2	P2.O2	P1.O34
34	P4.O5	P3.O3	P2.O3	P1.O35
35	P4.O6	P3.O4	P2.O4	P1.O36
36	P4.O7	P3.O5	P2.O5	P1.O37
37	P4.O8	P3.O6	P2.O6	P1.O38
38	P4.O9	P3.O7	P2.O7	P1.O39
39	P4.O10	P3.O8	P2.O8	P1.O40
40	P5.O1	P3.O9	P2.O9	P1.O41
41	P5.O2	P3.O10	P2.O10	P1.O42
42	P5.O3	P3.O11	P2.O11	P1.O43
43	P5.O4	P3.O12	P2.O12	P1.O44
44	P5.O5	P3.O13	P2.O13	P1.O45
45	P5.O6	P3.O14	P2.O14	P1.O46
46	P5.O7	P3.O15	P2.O15	P1.O47
47	P5.O8	P3.O16	P2.O16	P1.O48
48	P5.O9	P4.O1	P2.O17	P1.O49
49	P5.O10	P4.O2	P2.O18	P1.O50
50	P6.O1	P4.O3	P2.O19	P1.O51
51	P6.O2	P4.O4	P2.O20	P1.O52
52	P6.O3	P4.O5	P2.O21	P1.O53
53	P6.O4	P4.O6	P2.O22	P1.O54
54	P6.O5	P4.O7	P2.O23	P1.O55
55	P6.O6	P4.O8	P2.O24	P1.O56

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
56	P6.O7	P4.O9	P2.O25	P1.O57
57	P6.O8	P4.O10	P2.O26	P1.O58
58	P6.O9	P4.O11	P2.O27	P1.O59
59	P6.O10	P4.O12	P2.O28	P1.O60
60	P7.O1	P4.O13	P2.O29	P1.O61
61	P7.O2	P4.O14	P2.O30	P1.O62
62	P7.O3	P4.O15	P2.O31	P1.O63
63	P7.O4	P4.O16	P2.O32	P1.O64
64	P7.O5	P5.O1	P3.O1	P2.O1
65	P7.O6	P5.O2	P3.O2	P2.O2
66	P7.O7	P5.O3	P3.O3	P2.O3
67	P7.O8	P5.O4	P3.O4	P2.O4
68	P7.O9	P5.O5	P3.O5	P2.O5
69	P7.O10	P5.O6	P3.O6	P2.O6
70	P8.O1	P5.O7	P3.O7	P2.O7
71	P8.O2	P5.O8	P3.O8	P2.O8
72	P8.O3	P5.O9	P3.O9	P2.O9
73	P8.O4	P5.O10	P3.O10	P2.O10
74	P8.O5	P5.O11	P3.O11	P2.O11
75	P8.O6	P5.O12	P3.O12	P2.O12
76	P8.O7	P5.O13	P3.O13	P2.O13
77	P8.O8	P5.O14	P3.O14	P2.O14
78	P8.O9	P5.O15	P3.O15	P2.O15
79	P8.O10	P5.O16	P3.O16	P2.O16
80	P9.O1	P6.O1	P3.O17	P2.O17
81	P9.O2	P6.O2	P3.O18	P2.O18
82	P9.O3	P6.O3	P3.O19	P2.O19
83	P9.O4	P6.O4	P3.O20	P2.O20
84	P9.O5	P6.O5	P3.O21	P2.O21
85	P9.O6	P6.O6	P3.O22	P2.O22
86	P9.O7	P6.O7	P3.O23	P2.O23
87	P9.O8	P6.O8	P3.O24	P2.O24
88	P9.O9	P6.O9	P3.O25	P2.O25
89	P9.O10	P6.O10	P3.O26	P2.O26
90	P10.O1	P6.O11	P3.O27	P2.O27
91	P10.O2	P6.O12	P3.O28	P2.O28
92	P10.O3	P6.O13	P3.O29	P2.O29
93	P10.O4	P6.O14	P3.O30	P2.O30
94	P10.O5	P6.O15	P3.O31	P2.O31
95	P10.O6	P6.O16	P3.O32	P2.O32
96	P10.O7	P7.O1	P4.O1	P2.O33
97	P10.O8	P7.O2	P4.O2	P2.O34
98	P10.O9	P7.O3	P4.O3	P2.O35
99	P10.O10	P7.O4	P4.O4	P2.O36
100	P11.O1	P7.O5	P4.O5	P2.O37
101	P11.O2	P7.O6	P4.O6	P2.O38

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
102	P11.O3	P7.O7	P4.O7	P2.O39
103	P11.O4	P7.O8	P4.O8	P2.O40
104	P11.O5	P7.O9	P4.O9	P2.O41
105	P11.O6	P7.O10	P4.O10	P2.O42
106	P11.O7	P7.O11	P4.O11	P2.O43
107	P11.O8	P7.O12	P4.O12	P2.O44
108	P11.O9	P7.O13	P4.O13	P2.O45
109	P11.O10	P7.O14	P4.O14	P2.O46
110	P12.O1	P7.O15	P4.O15	P2.O47
111	P12.O2	P7.O16	P4.O16	P2.O48
112	P12.O3	P8.O1	P4.O17	P2.O49
113	P12.O4	P8.O2	P4.O18	P2.O50
114	P12.O5	P8.O3	P4.O19	P2.O51
115	P12.O6	P8.O4	P4.O20	P2.O52
116	P12.O7	P8.O5	P4.O21	P2.O53
117	P12.O8	P8.O6	P4.O22	P2.O54
118	P12.O9	P8.O7	P4.O23	P2.O55
119	P12.O10	P8.O8	P4.O24	P2.O56
120	P13.O1	P8.O9	P4.O25	P2.O57
121	P13.O2	P8.O10	P4.O26	P2.O58
122	P13.O3	P8.O11	P4.O27	P2.O59
123	P13.O4	P8.O12	P4.O28	P2.O60
124	P13.O5	P8.O13	P4.O29	P2.O61
125	P13.O6	P8.O14	P4.O30	P2.O62
126	P13.O7	P8.O15	P4.O31	P2.O63
127	P13.O8	P8.O16	P4.O32	P2.O64
128	P13.O9	P9.O1	P5.O1	P3.O1
129	P13.O10	P9.O2	P5.O2	P3.O2
130	P14.O1	P9.O3	P5.O3	P3.O3
131	P14.O2	P9.O4	P5.O4	P3.O4
132	P14.O3	P9.O5	P5.O5	P3.O5
133	P14.O4	P9.O6	P5.O6	P3.O6
134	P14.O5	P9.O7	P5.O7	P3.O7
135	P14.O6	P9.O8	P5.O8	P3.O8
136	P14.O7	P9.O9	P5.O9	P3.O9
137	P14.O8	P9.O10	P5.O10	P3.O10
138	P14.O9	P9.O11	P5.O11	P3.O11
139	P14.O10	P9.O12	P5.O12	P3.O12
140	P15.O1	P9.O13	P5.O13	P3.O13
141	P15.O2	P9.O14	P5.O14	P3.O14
142	P15.O3	P9.O15	P5.O15	P3.O15
143	P15.O4	P9.O16	P5.O16	P3.O16
144	P15.O5	P10.O1	P5.O17	P3.O17
145	P15.O6	P10.O2	P5.O18	P3.O18
146	P15.O7	P10.O3	P5.O19	P3.O19
147	P15.O8	P10.O4	P5.O20	P3.O20

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
148	P15.O9	P10.O5	P5.O21	P3.O21
149	P15.O10	P10.O6	P5.O22	P3.O22
150	P16.O1	P10.O7	P5.O23	P3.O23
151	P16.O2	P10.O8	P5.O24	P3.O24
152	P16.O3	P10.O9	P5.O25	P3.O25
153	P16.O4	P10.O10	P5.O26	P3.O26
154	P16.O5	P10.O11	P5.O27	P3.O27
155	P16.O6	P10.O12	P5.O28	P3.O28
156	P16.O7	P10.O13	P5.O29	P3.O29
157	P16.O8	P10.O14	P5.O30	P3.O30
158	P16.O9	P10.O15	P5.O31	P3.O31
159	P16.O10	P10.O16	P5.O32	P3.O32
160	P17.O1	P11.O1	P6.O1	P3.O33
161	P17.O2	P11.O2	P6.O2	P3.O34
162	P17.O3	P11.O3	P6.O3	P3.O35
163	P17.O4	P11.O4	P6.O4	P3.O36
164	P17.O5	P11.O5	P6.O5	P3.O37
165	P17.O6	P11.O6	P6.O6	P3.O38
166	P17.O7	P11.O7	P6.O7	P3.O39
167	P17.O8	P11.O8	P6.O8	P3.O40
168	P17.O9	P11.O9	P6.O9	P3.O41
169	P17.O10	P11.O10	P6.O10	P3.O42
170	P18.O1	P11.O11	P6.O11	P3.O43
171	P18.O2	P11.O12	P6.O12	P3.O44
172	P18.O3	P11.O13	P6.O13	P3.O45
173	P18.O4	P11.O14	P6.O14	P3.O46
174	P18.O5	P11.O15	P6.O15	P3.O47
175	P18.O6	P11.O16	P6.O16	P3.O48
176	P18.O7	P12.O1	P6.O17	P3.O49
177	P18.O8	P12.O2	P6.O18	P3.O50
178	P18.O9	P12.O3	P6.O19	P3.O51
179	P18.O10	P12.O4	P6.O20	P3.O52
180	P19.O1	P12.O5	P6.O21	P3.O53
181	P19.O2	P12.O6	P6.O22	P3.O54
182	P19.O3	P12.O7	P6.O23	P3.O55
183	P19.O4	P12.O8	P6.O24	P3.O56
184	P19.O5	P12.O9	P6.O25	P3.O57
185	P19.O6	P12.O10	P6.O26	P3.O58
186	P19.O7	P12.O11	P6.O27	P3.O59
187	P19.O8	P12.O12	P6.O28	P3.O60
188	P19.O9	P12.O13	P6.O29	P3.O61
189	P19.O10	P12.O14	P6.O30	P3.O62
190	P20.O1	P12.O15	P6.O31	P3.O63
191	P20.O2	P12.O16	P6.O32	P3.O64
192	P20.O3	P13.O1	P7.O1	P4.O1
193	P20.O4	P13.O2	P7.O2	P4.O2

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
194	P20.O5	P13.O3	P7.O3	P4.O3
195	P20.O6	P13.O4	P7.O4	P4.O4
196	P20.O7	P13.O5	P7.O5	P4.O5
197	P20.O8	P13.O6	P7.O6	P4.O6
198	P20.O9	P13.O7	P7.O7	P4.O7
199	P20.O10	P13.O8	P7.O8	P4.O8
200	P21.O1	P13.O9	P7.O9	P4.O9
201	P21.O2	P13.O10	P7.O10	P4.O10
202	P21.O3	P13.O11	P7.O11	P4.O11
203	P21.O4	P13.O12	P7.O12	P4.O12
204	P21.O5	P13.O13	P7.O13	P4.O13
205	P21.O6	P13.O14	P7.O14	P4.O14
206	P21.O7	P13.O15	P7.O15	P4.O15
207	P21.O8	P13.O16	P7.O16	P4.O16
208	P21.O9	P14.O1	P7.O17	P4.O17
209	P21.O10	P14.O2	P7.O18	P4.O18
210	P22.O1	P14.O3	P7.O19	P4.O19
211	P22.O2	P14.O4	P7.O20	P4.O20
212	P22.O3	P14.O5	P7.O21	P4.O21
213	P22.O4	P14.O6	P7.O22	P4.O22
214	P22.O5	P14.O7	P7.O23	P4.O23
215	P22.O6	P14.O8	P7.O24	P4.O24
216	P22.O7	P14.O9	P7.O25	P4.O25
217	P22.O8	P14.O10	P7.O26	P4.O26
218	P22.O9	P14.O11	P7.O27	P4.O27
219	P22.O10	P14.O12	P7.O28	P4.O28
220	P23.O1	P14.O13	P7.O29	P4.O29
221	P23.O2	P14.O14	P7.O30	P4.O30
222	P23.O3	P14.O15	P7.O31	P4.O31
223	P23.O4	P14.O16	P7.O32	P4.O32
224	P23.O5	P15.O1	P8.O1	P4.O33
225	P23.O6	P15.O2	P8.O2	P4.O34
226	P23.O7	P15.O3	P8.O3	P4.O35
227	P23.O8	P15.O4	P8.O4	P4.O36
228	P23.O9	P15.O5	P8.O5	P4.O37
229	P23.O10	P15.O6	P8.O6	P4.O38
230	P24.O1	P15.O7	P8.O7	P4.O39
231	P24.O2	P15.O8	P8.O8	P4.O40
232	P24.O3	P15.O9	P8.O9	P4.O41
233	P24.O4	P15.O10	P8.O10	P4.O42
234	P24.O5	P15.O11	P8.O11	P4.O43
235	P24.O6	P15.O12	P8.O12	P4.O44
236	P24.O7	P15.O13	P8.O13	P4.O45
237	P24.O8	P15.O14	P8.O14	P4.O46
238	P24.O9	P15.O15	P8.O15	P4.O47
239	P24.O10	P15.O16	P8.O16	P4.O48

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
240	P25.O1	P16.O1	P8.O17	P4.O49
241	P25.O2	P16.O2	P8.O18	P4.O50
242	P25.O3	P16.O3	P8.O19	P4.O51
243	P25.O4	P16.O4	P8.O20	P4.O52
244	P25.O5	P16.O5	P8.O21	P4.O53
245	P25.O6	P16.O6	P8.O22	P4.O54
246	P25.O7	P16.O7	P8.O23	P4.O55
247	P25.O8	P16.O8	P8.O24	P4.O56
248	P25.O9	P16.O9	P8.O25	P4.O57
249	P25.O10	P16.O10	P8.O26	P4.O58
250	P26.O1	P16.O11	P8.O27	P4.O59
251	P26.O2	P16.O12	P8.O28	P4.O60
252	P26.O3	P16.O13	P8.O29	P4.O61
253	P26.O4	P16.O14	P8.O30	P4.O62
254	P26.O5	P16.O15	P8.O31	P4.O63
255	P26.O6	P16.O16	P8.O32	P4.O64
256	P26.O7	P17.O1	P9.O1	P5.O1
257	P26.O8	P17.O2	P9.O2	P5.O2
258	P26.O9	P17.O3	P9.O3	P5.O3
259	P26.O10	P17.O4	P9.O4	P5.O4
260	P27.O1	P17.O5	P9.O5	P5.O5
261	P27.O2	P17.O6	P9.O6	P5.O6
262	P27.O3	P17.O7	P9.O7	P5.O7
263	P27.O4	P17.O8	P9.O8	P5.O8
264	P27.O5	P17.O9	P9.O9	P5.O9
265	P27.O6	P17.O10	P9.O10	P5.O10
266	P27.O7	P17.O11	P9.O11	P5.O11
267	P27.O8	P17.O12	P9.O12	P5.O12
268	P27.O9	P17.O13	P9.O13	P5.O13
269	P27.O10	P17.O14	P9.O14	P5.O14
270	P28.O1	P17.O15	P9.O15	P5.O15
271	P28.O2	P17.O16	P9.O16	P5.O16
272	P28.O3	P18.O1	P9.O17	P5.O17
273	P28.O4	P18.O2	P9.O18	P5.O18
274	P28.O5	P18.O3	P9.O19	P5.O19
275	P28.O6	P18.O4	P9.O20	P5.O20
276	P28.O7	P18.O5	P9.O21	P5.O21
277	P28.O8	P18.O6	P9.O22	P5.O22
278	P28.O9	P18.O7	P9.O23	P5.O23
279	P28.O10	P18.O8	P9.O24	P5.O24
280	P29.O1	P18.O9	P9.O25	P5.O25
281	P29.O2	P18.O10	P9.O26	P5.O26
282	P29.O3	P18.O11	P9.O27	P5.O27
283	P29.O4	P18.O12	P9.O28	P5.O28
284	P29.O5	P18.O13	P9.O29	P5.O29
285	P29.O6	P18.O14	P9.O30	P5.O30

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
286	P29.O7	P18.O15	P9.O31	P5.O31
287	P29.O8	P18.O16	P9.O32	P5.O32
288	P29.O9	P19.O1	P10.O1	P5.O33
289	P29.O10	P19.O2	P10.O2	P5.O34
290	P30.O1	P19.O3	P10.O3	P5.O35
291	P30.O2	P19.O4	P10.O4	P5.O36
292	P30.O3	P19.O5	P10.O5	P5.O37
293	P30.O4	P19.O6	P10.O6	P5.O38
294	P30.O5	P19.O7	P10.O7	P5.O39
295	P30.O6	P19.O8	P10.O8	P5.O40
296	P30.O7	P19.O9	P10.O9	P5.O41
297	P30.O8	P19.O10	P10.O10	P5.O42
298	P30.O9	P19.O11	P10.O11	P5.O43
299	P30.O10	P19.O12	P10.O12	P5.O44
300	P31.O1	P19.O13	P10.O13	P5.O45
301	P31.O2	P19.O14	P10.O14	P5.O46
302	P31.O3	P19.O15	P10.O15	P5.O47
303	P31.O4	P19.O16	P10.O16	P5.O48
304	P31.O5	P20.O1	P10.O17	P5.O49
305	P31.O6	P20.O2	P10.O18	P5.O50
306	P31.O7	P20.O3	P10.O19	P5.O51
307	P31.O8	P20.O4	P10.O20	P5.O52
308	P31.O9	P20.O5	P10.O21	P5.O53
309	P31.O10	P20.O6	P10.O22	P5.O54
310	P32.O1	P20.O7	P10.O23	P5.O55
311	P32.O2	P20.O8	P10.O24	P5.O56
312	P32.O3	P20.O9	P10.O25	P5.O57
313	P32.O4	P20.O10	P10.O26	P5.O58
314	P32.O5	P20.O11	P10.O27	P5.O59
315	P32.O6	P20.O12	P10.O28	P5.O60
316	P32.O7	P20.O13	P10.O29	P5.O61
317	P32.O8	P20.O14	P10.O30	P5.O62
318	P32.O9	P20.O15	P10.O31	P5.O63
319	P32.O10	P20.O16	P10.O32	P5.O64
320	P33.O1	P21.O1	P11.O1	P6.O1
321	P33.O2	P21.O2	P11.O2	P6.O2
322	P33.O3	P21.O3	P11.O3	P6.O3
323	P33.O4	P21.O4	P11.O4	P6.O4
324	P33.O5	P21.O5	P11.O5	P6.O5
325	P33.O6	P21.O6	P11.O6	P6.O6
326	P33.O7	P21.O7	P11.O7	P6.O7
327	P33.O8	P21.O8	P11.O8	P6.O8
328	P33.O9	P21.O9	P11.O9	P6.O9
329	P33.O10	P21.O10	P11.O10	P6.O10
330	P34.O1	P21.O11	P11.O11	P6.O11
331	P34.O2	P21.O12	P11.O12	P6.O12

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
332	P34.O3	P21.O13	P11.O13	P6.O13
333	P34.O4	P21.O14	P11.O14	P6.O14
334	P34.O5	P21.O15	P11.O15	P6.O15
335	P34.O6	P21.O16	P11.O16	P6.O16
336	P34.O7	P22.O1	P11.O17	P6.O17
337	P34.O8	P22.O2	P11.O18	P6.O18
338	P34.O9	P22.O3	P11.O19	P6.O19
339	P34.O10	P22.O4	P11.O20	P6.O20
340	P35.O1	P22.O5	P11.O21	P6.O21
341	P35.O2	P22.O6	P11.O22	P6.O22
342	P35.O3	P22.O7	P11.O23	P6.O23
343	P35.O4	P22.O8	P11.O24	P6.O24
344	P35.O5	P22.O9	P11.O25	P6.O25
345	P35.O6	P22.O10	P11.O26	P6.O26
346	P35.O7	P22.O11	P11.O27	P6.O27
347	P35.O8	P22.O12	P11.O28	P6.O28
348	P35.O9	P22.O13	P11.O29	P6.O29
349	P35.O10	P22.O14	P11.O30	P6.O30
350	P36.O1	P22.O15	P11.O31	P6.O31
351	P36.O2	P22.O16	P11.O32	P6.O32
352	P36.O3	P23.O1	P12.O1	P6.O33
353	P36.O4	P23.O2	P12.O2	P6.O34
354	P36.O5	P23.O3	P12.O3	P6.O35
355	P36.O6	P23.O4	P12.O4	P6.O36
356	P36.O7	P23.O5	P12.O5	P6.O37
357	P36.O8	P23.O6	P12.O6	P6.O38
358	P36.O9	P23.O7	P12.O7	P6.O39
359	P36.O10	P23.O8	P12.O8	P6.O40
360	P37.O1	P23.O9	P12.O9	P6.O41
361	P37.O2	P23.O10	P12.O10	P6.O42
362	P37.O3	P23.O11	P12.O11	P6.O43
363	P37.O4	P23.O12	P12.O12	P6.O44
364	P37.O5	P23.O13	P12.O13	P6.O45
365	P37.O6	P23.O14	P12.O14	P6.O46
366	P37.O7	P23.O15	P12.O15	P6.O47
367	P37.O8	P23.O16	P12.O16	P6.O48
368	P37.O9	P24.O1	P12.O17	P6.O49
369	P37.O10	P24.O2	P12.O18	P6.O50
370	P38.O1	P24.O3	P12.O19	P6.O51
371	P38.O2	P24.O4	P12.O20	P6.O52
372	P38.O3	P24.O5	P12.O21	P6.O53
373	P38.O4	P24.O6	P12.O22	P6.O54
374	P38.O5	P24.O7	P12.O23	P6.O55
375	P38.O6	P24.O8	P12.O24	P6.O56
376	P38.O7	P24.O9	P12.O25	P6.O57
377	P38.O8	P24.O10	P12.O26	P6.O58

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
378	P38.O9	P24.O11	P12.O27	P6.O59
379	P38.O10	P24.O12	P12.O28	P6.O60
380	P39.O1	P24.O13	P12.O29	P6.O61
381	P39.O2	P24.O14	P12.O30	P6.O62
382	P39.O3	P24.O15	P12.O31	P6.O63
383	P39.O4	P24.O16	P12.O32	P6.O64
384	P39.O5	P25.O1	P13.O1	P7.O1
385	P39.O6	P25.O2	P13.O2	P7.O2
386	P39.O7	P25.O3	P13.O3	P7.O3
387	P39.O8	P25.O4	P13.O4	P7.O4
388	P39.O9	P25.O5	P13.O5	P7.O5
389	P39.O10	P25.O6	P13.O6	P7.O6
390	P40.O1	P25.O7	P13.O7	P7.O7
391	P40.O2	P25.O8	P13.O8	P7.O8
392	P40.O3	P25.O9	P13.O9	P7.O9
393	P40.O4	P25.O10	P13.O10	P7.O10
394	P40.O5	P25.O11	P13.O11	P7.O11
395	P40.O6	P25.O12	P13.O12	P7.O12
396	P40.O7	P25.O13	P13.O13	P7.O13
397	P40.O8	P25.O14	P13.O14	P7.O14
398	P40.O9	P25.O15	P13.O15	P7.O15
399	P40.O10	P25.O16	P13.O16	P7.O16
400	P41.O1	P26.O1	P13.O17	P7.O17
401	P41.O2	P26.O2	P13.O18	P7.O18
402	P41.O3	P26.O3	P13.O19	P7.O19
403	P41.O4	P26.O4	P13.O20	P7.O20
404	P41.O5	P26.O5	P13.O21	P7.O21
405	P41.O6	P26.O6	P13.O22	P7.O22
406	P41.O7	P26.O7	P13.O23	P7.O23
407	P41.O8	P26.O8	P13.O24	P7.O24
408	P41.O9	P26.O9	P13.O25	P7.O25
409	P41.O10	P26.O10	P13.O26	P7.O26
410	P42.O1	P26.O11	P13.O27	P7.O27
411	P42.O2	P26.O12	P13.O28	P7.O28
412	P42.O3	P26.O13	P13.O29	P7.O29
413	P42.O4	P26.O14	P13.O30	P7.O30
414	P42.O5	P26.O15	P13.O31	P7.O31
415	P42.O6	P26.O16	P13.O32	P7.O32
416	P42.O7	P27.O1	P14.O1	P7.O33
417	P42.O8	P27.O2	P14.O2	P7.O34
418	P42.O9	P27.O3	P14.O3	P7.O35
419	P42.O10	P27.O4	P14.O4	P7.O36
420	P43.O1	P27.O5	P14.O5	P7.O37
421	P43.O2	P27.O6	P14.O6	P7.O38
422	P43.O3	P27.O7	P14.O7	P7.O39
423	P43.O4	P27.O8	P14.O8	P7.O40

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
424	P43.O5	P27.O9	P14.O9	P7.O41
425	P43.O6	P27.O10	P14.O10	P7.O42
426	P43.O7	P27.O11	P14.O11	P7.O43
427	P43.O8	P27.O12	P14.O12	P7.O44
428	P43.O9	P27.O13	P14.O13	P7.O45
429	P43.O10	P27.O14	P14.O14	P7.O46
430	P44.O1	P27.O15	P14.O15	P7.O47
431	P44.O2	P27.O16	P14.O16	P7.O48
432	P44.O3	P28.O1	P14.O17	P7.O49
433	P44.O4	P28.O2	P14.O18	P7.O50
434	P44.O5	P28.O3	P14.O19	P7.O51
435	P44.O6	P28.O4	P14.O20	P7.O52
436	P44.O7	P28.O5	P14.O21	P7.O53
437	P44.O8	P28.O6	P14.O22	P7.O54
438	P44.O9	P28.O7	P14.O23	P7.O55
439	P44.O10	P28.O8	P14.O24	P7.O56
440	P45.O1	P28.O9	P14.O25	P7.O57
441	P45.O2	P28.O10	P14.O26	P7.O58
442	P45.O3	P28.O11	P14.O27	P7.O59
443	P45.O4	P28.O12	P14.O28	P7.O60
444	P45.O5	P28.O13	P14.O29	P7.O61
445	P45.O6	P28.O14	P14.O30	P7.O62
446	P45.O7	P28.O15	P14.O31	P7.O63
447	P45.O8	P28.O16	P14.O32	P7.O64
448	P45.O9	P29.O1	P15.O1	P8.O1
449	P45.O10	P29.O2	P15.O2	P8.O2
450	P46.O1	P29.O3	P15.O3	P8.O3
451	P46.O2	P29.O4	P15.O4	P8.O4
452	P46.O3	P29.O5	P15.O5	P8.O5
453	P46.O4	P29.O6	P15.O6	P8.O6
454	P46.O5	P29.O7	P15.O7	P8.O7
455	P46.O6	P29.O8	P15.O8	P8.O8
456	P46.O7	P29.O9	P15.O9	P8.O9
457	P46.O8	P29.O10	P15.O10	P8.O10
458	P46.O9	P29.O11	P15.O11	P8.O11
459	P46.O10	P29.O12	P15.O12	P8.O12
460	P47.O1	P29.O13	P15.O13	P8.O13
461	P47.O2	P29.O14	P15.O14	P8.O14
462	P47.O3	P29.O15	P15.O15	P8.O15
463	P47.O4	P29.O16	P15.O16	P8.O16
464	P47.O5	P30.O1	P15.O17	P8.O17
465	P47.O6	P30.O2	P15.O18	P8.O18
466	P47.O7	P30.O3	P15.O19	P8.O19
467	P47.O8	P30.O4	P15.O20	P8.O20
468	P47.O9	P30.O5	P15.O21	P8.O21
469	P47.O10	P30.O6	P15.O22	P8.O22

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
470	P48.O1	P30.O7	P15.O23	P8.O23
471	P48.O2	P30.O8	P15.O24	P8.O24
472	P48.O3	P30.O9	P15.O25	P8.O25
473	P48.O4	P30.O10	P15.O26	P8.O26
474	P48.O5	P30.O11	P15.O27	P8.O27
475	P48.O6	P30.O12	P15.O28	P8.O28
476	P48.O7	P30.O13	P15.O29	P8.O29
477	P48.O8	P30.O14	P15.O30	P8.O30
478	P48.O9	P30.O15	P15.O31	P8.O31
479	P48.O10	P30.O16	P15.O32	P8.O32
480	P49.O1	P31.O1	P16.O1	P8.O33
481	P49.O2	P31.O2	P16.O2	P8.O34
482	P49.O3	P31.O3	P16.O3	P8.O35
483	P49.O4	P31.O4	P16.O4	P8.O36
484	P49.O5	P31.O5	P16.O5	P8.O37
485	P49.O6	P31.O6	P16.O6	P8.O38
486	P49.O7	P31.O7	P16.O7	P8.O39
487	P49.O8	P31.O8	P16.O8	P8.O40
488	P49.O9	P31.O9	P16.O9	P8.O41
489	P49.O10	P31.O10	P16.O10	P8.O42
490	P50.O1	P31.O11	P16.O11	P8.O43
491	P50.O2	P31.O12	P16.O12	P8.O44
492	P50.O3	P31.O13	P16.O13	P8.O45
493	P50.O4	P31.O14	P16.O14	P8.O46
494	P50.O5	P31.O15	P16.O15	P8.O47
495	P50.O6	P31.O16	P16.O16	P8.O48
496	P50.O7	P32.O1	P16.O17	P8.O49
497	P50.O8	P32.O2	P16.O18	P8.O50
498	P50.O9	P32.O3	P16.O19	P8.O51
499	P50.O10	P32.O4	P16.O20	P8.O52
500	P51.O1	P32.O5	P16.O21	P8.O53
501	P51.O2	P32.O6	P16.O22	P8.O54
502	P51.O3	P32.O7	P16.O23	P8.O55
503	P51.O4	P32.O8	P16.O24	P8.O56
504	P51.O5	P32.O9	P16.O25	P8.O57
505	P51.O6	P32.O10	P16.O26	P8.O58
506	P51.O7	P32.O11	P16.O27	P8.O59
507	P51.O8	P32.O12	P16.O28	P8.O60
508	P51.O9	P32.O13	P16.O29	P8.O61
509	P51.O10	P32.O14	P16.O30	P8.O62
510	P52.O1	P32.O15	P16.O31	P8.O63
511	P52.O2	P32.O16	P16.O32	P8.O64
512	P52.O3	P33.O1	P17.O1	P9.O1
513	P52.O4	P33.O2	P17.O2	P9.O2
514	P52.O5	P33.O3	P17.O3	P9.O3
515	P52.O6	P33.O4	P17.O4	P9.O4

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
516	P52.O7	P33.O5	P17.O5	P9.O5
517	P52.O8	P33.O6	P17.O6	P9.O6
518	P52.O9	P33.O7	P17.O7	P9.O7
519	P52.O10	P33.O8	P17.O8	P9.O8
520	P53.O1	P33.O9	P17.O9	P9.O9
521	P53.O2	P33.O10	P17.O10	P9.O10
522	P53.O3	P33.O11	P17.O11	P9.O11
523	P53.O4	P33.O12	P17.O12	P9.O12
524	P53.O5	P33.O13	P17.O13	P9.O13
525	P53.O6	P33.O14	P17.O14	P9.O14
526	P53.O7	P33.O15	P17.O15	P9.O15
527	P53.O8	P33.O16	P17.O16	P9.O16
528	P53.O9	P34.O1	P17.O17	P9.O17
529	P53.O10	P34.O2	P17.O18	P9.O18
530	P54.O1	P34.O3	P17.O19	P9.O19
531	P54.O2	P34.O4	P17.O20	P9.O20
532	P54.O3	P34.O5	P17.O21	P9.O21
533	P54.O4	P34.O6	P17.O22	P9.O22
534	P54.O5	P34.O7	P17.O23	P9.O23
535	P54.O6	P34.O8	P17.O24	P9.O24
536	P54.O7	P34.O9	P17.O25	P9.O25
537	P54.O8	P34.O10	P17.O26	P9.O26
538	P54.O9	P34.O11	P17.O27	P9.O27
539	P54.O10	P34.O12	P17.O28	P9.O28
540	P55.O1	P34.O13	P17.O29	P9.O29
541	P55.O2	P34.O14	P17.O30	P9.O30
542	P55.O3	P34.O15	P17.O31	P9.O31
543	P55.O4	P34.O16	P17.O32	P9.O32
544	P55.O5	P35.O1	P18.O1	P9.O33
545	P55.O6	P35.O2	P18.O2	P9.O34
546	P55.O7	P35.O3	P18.O3	P9.O35
547	P55.O8	P35.O4	P18.O4	P9.O36
548	P55.O9	P35.O5	P18.O5	P9.O37
549	P55.O10	P35.O6	P18.O6	P9.O38
550	P56.O1	P35.O7	P18.O7	P9.O39
551	P56.O2	P35.O8	P18.O8	P9.O40
552	P56.O3	P35.O9	P18.O9	P9.O41
553	P56.O4	P35.O10	P18.O10	P9.O42
554	P56.O5	P35.O11	P18.O11	P9.O43
555	P56.O6	P35.O12	P18.O12	P9.O44
556	P56.O7	P35.O13	P18.O13	P9.O45
557	P56.O8	P35.O14	P18.O14	P9.O46
558	P56.O9	P35.O15	P18.O15	P9.O47
559	P56.O10	P35.O16	P18.O16	P9.O48
560	P57.O1	P36.O1	P18.O17	P9.O49
561	P57.O2	P36.O2	P18.O18	P9.O50

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
562	P57.O3	P36.O3	P18.O19	P9.O51
563	P57.O4	P36.O4	P18.O20	P9.O52
564	P57.O5	P36.O5	P18.O21	P9.O53
565	P57.O6	P36.O6	P18.O22	P9.O54
566	P57.O7	P36.O7	P18.O23	P9.O55
567	P57.O8	P36.O8	P18.O24	P9.O56
568	P57.O9	P36.O9	P18.O25	P9.O57
569	P57.O10	P36.O10	P18.O26	P9.O58
570	P58.O1	P36.O11	P18.O27	P9.O59
571	P58.O2	P36.O12	P18.O28	P9.O60
572	P58.O3	P36.O13	P18.O29	P9.O61
573	P58.O4	P36.O14	P18.O30	P9.O62
574	P58.O5	P36.O15	P18.O31	P9.O63
575	P58.O6	P36.O16	P18.O32	P9.O64
576	P58.O7	P37.O1	P19.O1	P10.O1
577	P58.O8	P37.O2	P19.O2	P10.O2
578	P58.O9	P37.O3	P19.O3	P10.O3
579	P58.O10	P37.O4	P19.O4	P10.O4
580	P59.O1	P37.O5	P19.O5	P10.O5
581	P59.O2	P37.O6	P19.O6	P10.O6
582	P59.O3	P37.O7	P19.O7	P10.O7
583	P59.O4	P37.O8	P19.O8	P10.O8
584	P59.O5	P37.O9	P19.O9	P10.O9
585	P59.O6	P37.O10	P19.O10	P10.O10
586	P59.O7	P37.O11	P19.O11	P10.O11
587	P59.O8	P37.O12	P19.O12	P10.O12
588	P59.O9	P37.O13	P19.O13	P10.O13
589	P59.O10	P37.O14	P19.O14	P10.O14
590	P60.O1	P37.O15	P19.O15	P10.O15
591	P60.O2	P37.O16	P19.O16	P10.O16
592	P60.O3	P38.O1	P19.O17	P10.O17
593	P60.O4	P38.O2	P19.O18	P10.O18
594	P60.O5	P38.O3	P19.O19	P10.O19
595	P60.O6	P38.O4	P19.O20	P10.O20
596	P60.O7	P38.O5	P19.O21	P10.O21
597	P60.O8	P38.O6	P19.O22	P10.O22
598	P60.O9	P38.O7	P19.O23	P10.O23
599	P60.O10	P38.O8	P19.O24	P10.O24
600	P61.O1	P38.O9	P19.O25	P10.O25
601	P61.O2	P38.O10	P19.O26	P10.O26
602	P61.O3	P38.O11	P19.O27	P10.O27
603	P61.O4	P38.O12	P19.O28	P10.O28
604	P61.O5	P38.O13	P19.O29	P10.O29
605	P61.O6	P38.O14	P19.O30	P10.O30
606	P61.O7	P38.O15	P19.O31	P10.O31
607	P61.O8	P38.O16	P19.O32	P10.O32

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
608	P61.O9	P39.O1	P20.O1	P10.O33
609	P61.O10	P39.O2	P20.O2	P10.O34
610	P62.O1	P39.O3	P20.O3	P10.O35
611	P62.O2	P39.O4	P20.O4	P10.O36
612	P62.O3	P39.O5	P20.O5	P10.O37
613	P62.O4	P39.O6	P20.O6	P10.O38
614	P62.O5	P39.O7	P20.O7	P10.O39
615	P62.O6	P39.O8	P20.O8	P10.O40
616	P62.O7	P39.O9	P20.O9	P10.O41
617	P62.O8	P39.O10	P20.O10	P10.O42
618	P62.O9	P39.O11	P20.O11	P10.O43
619	P62.O10	P39.O12	P20.O12	P10.O44
620	P63.O1	P39.O13	P20.O13	P10.O45
621	P63.O2	P39.O14	P20.O14	P10.O46
622	P63.O3	P39.O15	P20.O15	P10.O47
623	P63.O4	P39.O16	P20.O16	P10.O48

Modbus Register Address	Essential Data Object Reference (Objects configured per page)			
	x10	x16	x32	x64
624	P63.O5	P40.O1	P20.O17	P10.O49
625	P63.O6	P40.O2	P20.O18	P10.O50
626	P63.O7	P40.O3	P20.O19	P10.O51
627	P63.O8	P40.O4	P20.O20	P10.O52
628	P63.O9	P40.O5	P20.O21	P10.O53
629	P63.O10	P40.O6	P20.O22	P10.O54
630	P64.O1	P40.O7	P20.O23	P10.O55
631	P64.O2	P40.O8	P20.O24	P10.O56
632	P64.O3	P40.O9	P20.O25	P10.O57
633	P64.O4	P40.O10	P20.O26	P10.O58
634	P64.O5	P40.O11	P20.O27	P10.O59
635	P64.O6	P40.O12	P20.O28	P10.O60
636	P64.O7	P40.O13	P20.O29	P10.O61
637	P64.O8	P40.O14	P20.O30	P10.O62
638	P64.O9	P40.O15	P20.O31	P10.O63
639	P64.O10	P40.O16	P20.O32	P10.O64

Extra Data

Extra Data contains 1024 values. These values are arranged in 16 pages of 64 objects.

Once the interface is started, the values may be accessed using any of the supported Modbus function codes. These values appear in registers that follow on from the existing Essential Value registers – i.e. Registers 640…1663 on Commander, or 1280…2303 on ObSys.

For a complete list of Modbus registers and the Extra Data page/object reference, download the spreadsheet from *http://resource.northbt.com/driver/Modbus_RegisterList.xlsx*

Driver Versions

Version	Build Date	Details
1.0	10/9/2012	Driver released (renamed from JBus).
1.0	29/8/2013	Change Modbus system scan to detect online devices. Default baud rate to 9600 on initialisation Address Start and Count now initialised
1.0	6/9/2013	Fixed problem with bit write
1.1	7/2/2014	Moved objects AC, AS, TXB and BO to advanced setup object (A) Add Intelligent scan object to disable new scan method (A.IS) Add exception device typelist (A.Ux.A, A.Ux.DT) Default TXB to 10ms on initialisation
1.1	19/5/2014	On initialisation, default baud rate to 19200, and E81 byte format
2.0	8/2/2016	Added Reply Timeout object (TO) to advanced setup. 64-bit value support added Decode types H & I, now read before writing value. Added formulas 37 and 40 Added ability to read in blocks of 6 sig figs, for 32-bit and 64-bit values
2.0	18/6/2018	Fix reading 64-bit float values when 0 (previously returned blank)
2.0	17/8/2018	Added function code 'U' to support Modbus function code 15
3.0	2/12/2019	Combined existing Modbus, ModbusTCP, and ModbusSlave drivers in this new version. Added decode 'N' for bit mask with bit shift.
3.0	03/02/2020	Fix documentation of input/holding multi-register counts (5,6,7,8 should be 6,8,12,16)
3.0	01/06/2020	Added decode 'O' and 'P' for LSW,MSW order. Added undocumented special decode mechanism 'Q' Changed multi-register count from 12 to 10 registers. Resolved issue on rounding floating-point numbers when writing
3.0	16/11/2022	TCP server is no longer enabled by default
3.0	01/03/2023	Fix issue when decoding 4-register values (function 'H')

Next Steps…

If you require help, contact support on 01273 694422 or visit *www.northbt.com/support*

North Building Technologies Ltd
+44 (0) 1273 694422
support@northbt.com
www.northbt.com

Author: JF
Checked by: TM

Document issued 20/11/2023.