Communications

Last updated: 08/09/2021

Introduction

The drive itself provides either one or two Modbus RTU interfaces:

1. A keypad port is always present and is used to connect a keypad directly to the front of the drive. An adapter is available to convert this to an EIA485 physical interface with an RJ45 connector. This can be used for a remote keypad connection, or it can be used to connect to a Modbus RTU master.
2. A user 485 port is present on some product derivatives which provides an EIA485 physical interface which can be connected to a Modbus RTU master. If this port is provided it is via an RJ45 connector on the front of the drive or a three-way terminal block in place of the position feedback interface connection.

Both interfaces use the same protocol as described in this document. The hardware description for each interface is beyond the scope of this document. The transport layer is provided by Modbus RTU and the drive responds to the following function codes.

Modbus function code	Function	Comments
3	Read multiple	Used to read drive parameters
6	Write a single	Used to a write drive parameter
16	Write multiple drive parameters	Used to write drive parameters
23	Read and write multiple drive parameters	Used to read and write drive parameters
64	Legacy CMP protocol	Used to read or write drive parameters.
65	Keypad protocol	This is not for general use and is not described in this document.
66	ECMP	Used to read and write drive parameters. Includes additional functionality including accessing the drive file system.

The set-up for the keypad port is as follows. This is fixed apart from the Modbus node address which can be set with a user parameter.

Function	Setting
Modbus node address (1-16)	*Keypad Port Serial Address* (11.090)
Data bits, stop bits and parity	8 data bits, 1 stop bit, no parity
Selection of modified parameter access up to parameter 255	Parameter numbers allowed up to 255
Baud rate	115200
Minimum Comms Transmit delay	0
Silent period	3.5 bytes

The set-up for the user 485 port is more flexible and defined by user parameters as follows.

Function	Drive parameter
Modbus node address (1-247)	*Serial Address* (11.023)
Data bits, stop bits and parity	*Serial Mode* (11.024)
Selection of modified parameter access up to parameter 255	*Serial Mode* (11.024)
Baud rate	*Serial Baud Rate* (11.025)
Minimum Comms Transmit delay	*Minimum Comms Transmit Delay* (11.026)
Silent Period	*Silent Period* (11.027)

The set-up data is only taken from user parameters for either port when *Reset Serial Communications* (11.020) is set to one, so that it is possible to update the set-up parameters via serial communications without disturbing the set-up while communicating. *Reset Serial Communications* (11.020) automatically resets itself to zero after the set-up changes have been made.

Throughout this document where communications messages are shown each element in the message is one byte. Each port has a 512 byte fixed comms buffer. However, 12 bytes is used for internal signalling. Therefore the maximum message size is 500 bytes.

Where examples are given throughout this document the following colour scheme is used to highlight different areas of the messages.

Modbus Frame
Common section for the protocol defined by the function code
Request/response specific data area
Highlighted data referred to in the text

Although this document describes communication based on Modbus RTU via the drive ports, it is possible to send the ECMP PDU via other protocols though an option module. See the section at the end of this document for details.

This document relates to V01.22.00.00 firmware and higher.

Modbus RTU

General

All messages, except for the slave response when an exception is produced, use the following format. Each section is one byte except the PDU (Protocol Data Unit) which varies in length depending on the function code.

Request
Slave Modbus node address
Function code
PDU Start

PDU End
CRC LSB
CRC MSB
Silent period

Response
Slave Modbus node address
Function code
PDU Start

PDU End
CRC LSB
CRC MSB

The CRC is calculated as CRC16 specified for Modbus RTU ($X^{16} + X^{12} + X^5 + 1$).

If the Modbus node address in the master request is 0 then this is a broadcast message and the slave will perform the required action, but it will not produce a response.

The following responses are possible related to this section:

Response	**Reason**
No response	Modbus node address is 0, or out of range. Modbus node address does not match the slave address. CRC is incorrect.
Exception code 1	The function code is not supported
Response controlled by PDU data	None of the above is true

An exception response is always in the following form:

Response
Modbus node address
0x80 | Function code
Exception value
CRC LSB
CRC MSB

Drive parameters are selected using a “Start Register Address”. The following sections show how this address is derived and then how it is used in different function codes to read or write drive parameters.

Register Addresses

When reading or writing drive user parameters Modbus uses a register addressing scheme to define the first parameter with the “Start Register Address”. If multiple parameters are accessed these must be consecutive after the first parameter. Each register is a 16 bit value, and so it is large enough to accommodate 1, 8 or 16 bit parameters. It is possible to combine 2 registers to access 32 bit parameters or to read values in 32 bit floating point format. Three possible types can be specified as follows.

Type	**Number of registers required**	**Data format**
0	1	16 bit integer
1	2	32 bit integer. The first register contains the most significant 16 bits.
2	2	32 bit floating point. The first register contains the most significant 16 bits.
3	Not applicable	Not allowed

It is possible to use any of the 3 types with any drive parameter, but the result may be truncated, incorrectly sign extended or not written when the 16 or 32 bit integer types are used. See the sections below on reading or writing parameters.

The format of the register address is as given below when the standard menu/parameter format is used. This format is used by the user 485 port when selected by Serial Mode (11.024) with its default value. This mode is not used by the keypad port. The possible range of menu numbers is 0 to 162 and the possible range of parameter numbers is from 0 to 99. It is not possible to access parameter 00.000 directly, but it can be indirectly accessed via parameter 01.000.

15 14	13 0
Type	(Menu x 100) + Parameter number - 1

The table below gives some example register values for accessing parameter 01.006.

Type	Register address
0	0x0000 + [(1 x 100) + 6] - 1 = 0x0069
1	0x4000 + [(1 x 100) + 6] - 1 = 0x4069
2	0x8000 + [(1 x 100) + 6] - 1 = 0x8069

The format of the register address is as given below when the modified menu/parameter format is used. This format can be selected with Serial Mode (11.024). This mode is always used by the keypad port. This modified mode reduces the menu range (0 to 63) and increases the parameter range to 0 to 255. It is not possible to access parameter 00.000 directly, but it can be indirectly accessed via parameter 01.000.

15 14	13 0
Type	(Menu x 256) + Parameter number - 1

The table below gives some example register values for accessing parameter 01.006.

Type	Register address
0	0x0000 + [(1 x 255) + 6] - 1 = 0x0105
1	0x4000 + [(1 x 100) + 6] - 1 = 0x4105
2	0x8000 + [(1 x 100) + 6] - 1 = 0x8105

Reading Drive Parameters

One or more consecutive drive parameters can be read in one Modbus exchange beginning at the parameter specified as the Start Register Address. The value may be truncated or incorrectly signed extended when 16 or 32 bit integer types are specified. The following example parameters show how the data is read using these type formats.

Parameter	Value	Attributes
Parameter1 (20.021)	0xF2345678	32 bit signed or unsigned
Parameter2 (18.011)	0x1234	16 bit signed or unsigned
Parameter3 (18.012)	0xF234	16 bit signed or unsigned

Parameter	Type format	Register data	Comments
Parameter1 (20.021)	1 (32 bit)	0xF234 0x5678	32 bit value transferred without modification
Parameter1 (20.021)	0 (16 bit)	0x5678	32 bit value truncated
Parameter2 (18.011)	1 (32 bit)	0x0000 0x1234	16 bit value is sign extended
Parameter2 (18.011)	0 (16 bit)	0x1234	16 bit value is transferred without modification
Parameter3 (18.012)	1 (32 bit)	0xFFFF 0xF234	16 bit value is sign extended (incorrect for unsigned parameters)
Parameter3 (18.012)	0 (16 bit)	0xF234	16 bit value is transferred without modification

Writing Drive Parameters

One or more consecutive drive parameters can be written in one Modbus exchange beginning at the parameter specified as the start register address. The value may be incorrectly signed extended or not written when a 16 integer type is specified. The following example parameters show how the data is written using these type formats

Parameter	Attributes
Parameter1 (20.021)	32 bit signed or unsigned
Parameter2 (18.011)	16 bit signed or unsigned

Parameter	Type format	Register data	Value written	Comments
Parameter1 (20.021)	1 (32 bit)	0xF234 0x5678	0xF2345678	32 bit value transferred without modification
Parameter1 (20.021)	0 (16 bit)	0x1234	0x00001234	Positive 16 bit value zero extended
Parameter1 (20.021)	0 (16 bit)	0xF234	0xFFFFF234	Negative 16 bit value negative sign extended (incorrect for unsigned parameters)
Parameter2 (18.011)	1 (32 bit)	0x0000 0x1234	0x1234	16 bit value transferred without modification
Parameter2 (18.011)	1 (32 bit)	0x0000 0xF234	0xF234	16 bit value transferred without modification
Parameter2 (18.011)	1 (32 bit)	0x1234 0x5678	None	Value of the parameter is exceeded so no value written
Parameter2 (18.011)	0 (16 bit)	0x1234	0x1234	16 bit value transferred without modification
Parameter2 (18.011)	0 (16 bit)	0xF234	0xF234	16 bit value transferred without modification (The value is a negative number, and so if the parameter was unsigned it would exceed the range and no value is written.)

Function code 3: Read Multiple

Request to read one or more 16 bit registers.

Request PDU
Start Register Address MSB
Start Register Address LSB
Number (n) of 16 bit registers to read MSB
Number (n) of 16 bit registers to read LSB

Response PDU
Length of register data in bytes (2n)
Register data 0 MSB
Register data 0 LSB

Register data N MSB
Register data N LSB

The response PDU is produced except under the following conditions.

Alternative response	Reason
Exception response with exception code 2	Number of registers to read is greater than 16 Number of registers n is not even for 32 bit or float type reads The response data will not fit in the response buffer Any parameter to be read does exist or is write-only The type is not 16 bit, 32 bit or float

The following example shows an exchange to obtain the value of parameter 01.006 (which has a value of 50.0) as a 32 bit integer. The drive is set up for standard menu/parameter format.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x03	Function code 3	0x03	Function code 3
0x40	Start Register Address = 0x4069 (Type = 32 bit integer)	0x04	Length of data in bytes = 4
0x69		0x00	4 bytes of data = 500 (parameter value = 50.0)
0x00	Number 0f register = 2	0x00	
0x02		0x01	
0x01	CRC MSB	0xF4	
0xD7	CRC LSB	0xFA	CRC MSB
		0x24	CRC LSB

Function code 6: Write Single

Request to write one 16 bit register.

Request PDU
Start Register Address MSB
Start Register Address LSB
Register data MSB
Register data LSB

Response PDU
Start Register Address MSB
Start Register Address LSB
Register data MSB
Register data LSB

The response PDU is produced except under the conditions given in the table below.

Alternative response	Reason
Exception response with exception code 2	The type is not 16 bit
Exception response with exception code 4	The parameter to write does not exist, is read-only or the data to be written is outside the range of the parameter

The following example shows an exchange to write 50.0 to parameter 01.006. The drive is set up for standard menu/parameter format.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x06	Function code 6	0x06	Function code 6
0x00	Start Register Address = 0x0069 (Type = 16 bit integer)	0x00	Start Register Address = 0x0069 (Type = 16 bit integer)
0x69		0x69	

0x01	Write value = 500	0x01	2 bytes of data = 500
0xF4		0xF4	
0x59	CRC MSB	0x59	CRC MSB
0xC1	CRC LSB	0xC1	CRC LSB

Function code 16: Write Multiple

Request to write one or more 16 bit registers.

Request PDU
Start Register Address MSB
Start Register Address LSB
Number (N) of 16 bit registers to write MSB
Number (N) of 16 bit registers to write LSB
Length of register data in bytes (2N)
Register data 0 MSB
Register data 0 LSB

Register data N MSB
Register data N LSB

Response PDU
Start Register Address MSB
Start Register Address LSB
Number (N) of 16 bit registers successfully written MSB
Number (N) of 16 bit registers successfully written LSB

The response PDU is produced except under the conditions given in the table below. If the response PDU is produced, then parameter writing stops when the first error occurs (e.g. if the parameter does not exist). The actual number of registers written is given in the response.

Alternative response	**Reason**
Exception response with exception code 2	Number of registers to write is greater than 16 Length of register data in bytes (2n) is not equal to the number of registers to write (n) multiplied by 2 The type is not 16 bit, 32 bit or float

The following example shows an exchange to write 50.0 to parameter 01.006 and 10.0 to parameter 01.007 as 32 bit integers. The drive is set up for standard menu/parameter format.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x10	Function code 16	0x10	Function code 6
0x40	Start Register Address = 0x4069 (Type = 32 bit integer)	0x40	Start Register Address = 0x4069 (Type = 32 bit integer)
0x69		0x69	
0x00	Number 0f registers = 4	0x00	2 bytes of data written
0x04		0x04	
0x08	Length of register data = 8	0x04	CRC MSB
0x00	Write value = 500	0x16	CRC LSB
0x00			
0x01			
0xF4			
0x00	Write value = 100		
0x00			
0x00			
0x64			
0x65	CRC MSB		
0xC9	CRC LSB		

Function code 23: Read/write Multiple

Request to write one or more 16 bit registers and then read one or more 16 bit registers.

Request PDU
Start Register Address MSB
Start Register Address LSB
Number (N) of 16 bit registers to read MSB
Number (N) of 16 bit registers to read LSB
Start Register Address to write MSB
Start Register Address to write LSB
Number (N) of 16 bit registers to write MSB
Number (N) of 16 bit registers to write LSB
Length of register data in bytes (2N)
Register data 0 MSB
Register data 0 LSB

Register data N MSB
Register data N LSB

Response PDU
Length of register data in bytes (2N)
Register data 0 MSB
Register data 0 LSB

Register data N MSB
Register data N LSB

It should be noted that the request PDU is the request PDU for function code 3 (Read Multiple) followed by the request PDU for function code 16 (Write Multiple). The write is performed first and continues until any of the errors given for function code 16 (Write Multiple) occur. Some parameters may have been written when an error is detected, but no indication is given about how many parameters have been written successfully. The read is always performed even if an error is detected during writing. Any of the errors given for function code 3 (Read Multiple) can occur and the exception response is the same as for function code 3 (Read Multiple).

The following example shows an exchange to obtain the value of parameter 01.006 which has a value of 50.0 as a 32 bit integer, and then write 50.0 to parameter 01.021 and 10.0 to parameter 01.022 as 32 bit integers. The drive is set up for standard menu/parameter format.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x17	Function code 23	0x17	Function code 23
0x40	Start Register Address to Read = 0x4069 (01.006, 32 bit integer)	0x04	Length of data in bytes = 4
0x69		0x00	4 bytes of data = 500 (parameter value = 50.0)
0x00	Number of registers = 2	0x00	
0x02		0x01	
0x40	Start Register Address to Write = 0x4078 (01.021, 32 bit integer)	0xF4	
0x78		0xF9	CRC MSB
0x00	Number of registers = 4	0x30	CRC LSB
0x04			
0x08	Length of register data = 8		
0x00	Write value = 500		
0x00			
0x00			
0xF4			
0x00	Write value = 100		
0x00			
0x00			
0x64			
0xB9	CRC MSB		
0x9A	CRC LSB		

Function code 64: CMP message

When function code 64 (0x40) is used the PDU contains a legacy CMP message. The destination can be the drive, or an option module fitted to the drive. The destination is specified within the PDU. The contents of the PDU are described in the section on CMP.

Function code 65: Keypad message

When function code 65 (x041) is used the PDU contains a keypad message. This protocol is designed for a keypad to communicate with the drive and is not for general use. It is not described in this document.

Function code 66: ECMP messages

When a function code 66 (0x42) is used the PDU contains an ECMP message. The destination can be the drive, or an option module fitted to the drive. The destination is specified within the PDU. This protocol provides a number of features including reading/writing parameters, reading/writing files, etc. The contents of the PDU are described in the section on ECMP.

CMP

General

The PDU for a CMP exchange with the drive is shown below.

Request PDU
Destination = 0
Destination sub-node (Any value as not used)
Op Code
Status = ST_REQ = 0
Process ID (Any value can be used)
Start of Op Code specific data request

End of Op Code specific data request

Response PDU
Destination = 0
Destination sub-node (Same value as in request)
Op Code
Status
Process ID (Same value as in request)
Start of Op Code specific data response

End of Op Code specific data response

The following Op Code values are supported

Op Code Identifier	**Op Code**	**Description**
OC_DEVICE_ID	0x50	Device ID
OC_RD_OBJ	0x10	Read object
OC_WR_OBJ	0x11	Write object
OC_RD_OBJLIST	0x14	Read object list
OC_WR_OBJLIST	0x15	Write object list

Status should always be ST_REQ (0) in the request. Status in the response indicates whether the exchange was successful or the reason it failed. The possible Status values are given in the table below. If an error occurs the Object Type in the response is ORed with 0x80. When a list of objects is read or written in one exchange the Status cannot indicate which read/writes were successful or failed, but the Object Type in the response for each individual object will show this.

Status Identifier	**Status**	**Description**
ST_REQ	0	Request
ST_ACK	1	Response okay
ST_UNKNOWN	-2	Parameter does not exist
ST_RDONLY	-3	Parameter is read-only
ST_WRONLY	-4	Parameter is write-only
ST_OVRANGE	-5	Write value is outside parameter range
ST_BADTYPE	-7	Indicates required number of decimal places error.
ST_FAIL	-10	General failure. See Op Code sections for the reason for failure.

The Op Code specific data is described below for each of the supported Op Codes. It should be noted that values consisting of more than one byte are in Little Endian format, i.e. LS byte first and MS byte last. For example, the version number in the Device ID response is 00.02.00.00 and is given in the response as 0x00, 0x00, 0x02, 0x00.

If Destination is non-zero it will route the request to an option module fitted in the Option Slot given by the Destination. The description of CMP supported by option modules is beyond the scope of this document. It should be noted that there is one buffer within the drive for routing CMP messages to option modules which is shared by both the keypad and user port. If an attempt is made to use one port after the other port has sent a CMP message to an option module and is waiting for a response, then the new request will be ignored and there will be no response to this request.

Device ID (0x50)

Request to identify the drive. There is no Op Code specific data request.

Op Code specific data response
CMP version LSB

CMP version MSB
Target ID LSB

Target ID MSB
Device ID string first character

Device ID string last character

The following example shows an exchange to obtain the device ID.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x40	Function code 64	0x40	Function code 64
0x00	Destination = 0	0x00	Destination = 0
0x00	Destination sub-node = 0	0x00	Destination sub-node = 0
0x50	Op Code = OC_DEVICE_ID	0x50	Op Code = OC_DEVICE_ID
0x00	Status = ST_REQ = 0	0x01	Status = ST_ACK = 1
0x01	Process ID = 1	0x01	Process ID = 1
0xC4	CRC MSB	0x00	CMP version = 00.02.00.00
0xD1	CRC LSB	0x00	
		0x02	
		0x00	
		0xFF	Target ID = 0x000007FF
		0x07	
		0x00	
		0x00	
		0x65	‘e’ (Device ID string)
		0x43	‘C’
		0x4D	‘M’
		0x50	‘P’
		0x00	NULL
		0x0B	CRC MSB
		0xC9	CRC LSB

Read object (0x10)

Request to read one parameter from the drive.

Op Code specific data request
Object Type = 1
Parameter Number
Menu Number
Bit Number

Op Code specific data response
Object Type = 1
Parameter Number
Menu Number
Bit Number
Register Data Decimal Places
Register Data LSB

Register Data MSB

If the Object Type has any value other than 1 or the bit number is outside the range from 0 to 32 then the Status in the response is ST_FAIL (-10). If any failure occurs Object Type is the response is ORed with 0x80 and the Register Data Type and Register Data are not included in the response.

If Bit Number is 0 the whole value is returned. If the Bit Number is between 1 and 32 then the state of bit (Bit Number – 1) is returned.

The parameter value is returned as an integer and Register Data Decimal Places Type is the number of decimal places plus 2. For example, if the parameter value is 50.0 then the Register Data is 500 and the Register Data Decimal Places is 3. If the parameter has more than 6 decimal places Register Data Decimal Places is set to -1.

The following example shows an exchange to obtain the value of parameter 01.006 which has a value of 50.0.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x40	Function code 64	0x40	Function code 64
0x00	Destination = 0	0x00	Destination = 0
0x00	Destination sub-node = 0	0x00	Destination sub-node = 0
0x10	Op Code = OC_RD_OBJ	0x10	Op Code = OC_RD_OBJ
0x00	Status = ST_REQ = 0	0x01	Status = ST_ACK = 1
0x01	Process ID = 1	0x01	Process ID = 1
0x01	Object Type = 1	0x01	Object Type = 1
0x06	Parameter Number = 6	0x06	Parameter Number = 6
0x01	Menu Number = 1	0x01	Menu Number = 1
0x00	Bit Number = 0	0x00	Bit Number = 0
0xCC	CRC MSB	0x03	Register Data Decimal Places = 1
0xA0	CRC LSB	0xF4	Register Data = 500
		0x01	
		0x00	
		0x00	
		0x11	CRC MSB

		0x1C	CRC LSB

Write object (0x11)

Request to write one parameter to the drive. There no Op Code specific data response.

Op Code specific data request
Object Type = 1
Parameter Number
Menu Number
Bit Number
Register Data Decimal Places
Register Data LSB

Register Data MSB

If the Object Type has any value other than 1 or the bit number is outside the range from 0 to 32 or Register Data Decimal Places is greater than 8 (6 decimal places) then the Status in the response is ST_FAIL (-10).

If Bit Number is 0 the whole value is written. If the Bit Number is between 1 and 32 then the state of bit (Bit Number – 1) is written. It should be noted that to write one bit of a parameter a read-modify-write operation is required. If another process modifies the parameter between the read and write, then the final value may be incorrect.

The parameter value is written with the Register Data value including the number of decimal places defined by Register Data Decimal Places. The number of decimal places is Register Data Decimal Places minus 2. For example, if the Register Data is 500 and Register Data Decimal Places is 3 then 50.0 is written to the parameter. If the Register Data Decimal Places is -1, then the raw integer value is written. For example, if the Register Data is 1234, Register Data Decimal Places is -1 and the destination parameter has 3 decimal places, then the value written is 1.234.

The following example shows an exchange to write parameter 01.006 with 50.0.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x40	Function code 64	0x40	Function code 64
0x00	Destination = 0	0x00	Destination = 0
0x00	Destination sub-node = 0	0x00	Destination sub-node = 0
0x11	Op Code = OC_WR_OBJ	0x11	Op Code = OC_WR_OBJ
0x00	Status = ST_REQ = 0	0x01	Status = ST_ACK = 1
0x01	Process ID = 1	0x01	Process ID = 1
0x01	Object Type = 1	0x95	CRC MSB
0x06	Parameter Number = 6	0x55	CRC LSB
0x01	Menu Number = 1		
0x00	Bit Number = 0		
0x03	Register Data Decimal Places = 1		
0xF4	Register Data = 500		
0x01			
0x00			
0x00			
0xE8	CRC MSB		
0x23	CRC LSB		

Read object list (0x14)

This Op Code operates in a similar way to Read Object (see Read Object for details) but can read between 1 and 10 objects. If an attempt is made to read more than 10 objects, the Status is set to ST_FAIL (-10) and the original request is returned without the additional response data. Otherwise the full response is returned. If a failure occurs for an individual read the Register Data and Register Data Decimal Places are not added to the response and the corresponding Object Type is ORed with 0x80. Status does not indicate the reason for the error and will always be ST_ACK (1).

Op Code specific data request
Number of Objects
Following bytes until end of Op Code specific request (i.e. Number of Objects x 4)
Object Type 1 = 1
Parameter Number 1
Menu Number 1
Bit Number 1

Object Type N = 1
Parameter Number N
Menu Number N
Bit Number N

Op Code specific data response
Number of Objects
Following bytes until end of Op Code specific response

Object Type = 1
Parameter Number 1
Menu Number 1
Bit Number 1

Object Type N = 1
Parameter Number N
Menu Number N
Bit Number N
Register Data 1 LSB

Register Data 1 MSB
Register Data 1 Decimal Places

Register Data N LSB

Register Data N MSB
Register Data N Decimal Places

The following example shows an exchange to obtain the value of parameter 01.006 which has a value of 50.0 and parameter 01.016 which has a value of 10.0.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x40	Function code 64	0x40	Function code 64
0x00	Destination = 0	0x00	Destination = 0
0x00	Destination sub-node = 0	0x00	Destination sub-node = 0
0x14	Op Code = OC_RD_OBJLIST	0x14	Op Code = OC_RD_OBJ
0x00	Status = ST_REQ = 0	0x01	Status = ST_ACK = 1
0x01	Process ID = 1	0x01	Process ID = 1
0x02	Number of Objects = 2	0x02	Number of Objects = 2
0x08	Number of Objects x 4 = 8	0x12	Number of Objects x 9 = 18
0x01	Object Type = 1	0x01	Object Type = 1
0x06	Parameter Number = 6	0x06	Parameter Number = 6
0x01	Menu Number = 1	0x01	Menu Number = 1
0x00	Bit Number = 0	0x00	Bit Number = 0
0x01	Object Type = 1	0x01	Object Type = 1
0x10	Parameter Number = 16	0x10	Parameter Number = 16
0x01	Menu Number = 1	0x01	Menu Number = 1
0x00	Bit Number = 0	0x00	Bit Number = 0
0x45	CRC MSB	0xF4	Register Data 1 = 500
0x46	CRC LSB	0x01	
		0x00	
		0x00	
		0x03	Register Data 1 Decimal Places = 1
		0x64	Register Data 2 = 100
		0x00	
		0x00	
		0x00	
		0x03	Register Data 2 Decimal Places = 1
		0xFF	CRC MSB
		0x0D	CRC LSB

Write object list (0x15)

This Op Code operates in a similar way to Write Object (see Write Object for details) but can write between 1 and 10 objects. If an attempt is made to write more than 10 objects, the Status is set to ST_FAIL (-10) in the response. If a failure occurs for an individual write the corresponding Object Type in the response is ORed with 0x80. Status does not indicate the reason for the error and will always be ST_ACK (1).

Op Code specific data request
Number of Objects
Following byte until the end of this request = Number of Objects x 9
Object Type 1 = 1
Parameter Number 1
Menu Number 1
Bit Number 1

Object Type N = 1

Parameter Number N
Menu Number N
Bit Number N
Register Data 1LSB

Register Data 1 MSB
Register Data 1 Decimal Places

Register Data N LSB

Register Data N MSB
Register Data N Decimal Places

Op Code specific data response
Number of Objects
Following bytes until the end of this request = Number of Objects x 9
Object Type = 1
Parameter Number 1
Menu Number 1
Bit Number 1

Object Type N = 1
Parameter Number N
Menu Number N
Bit Number N

The following example shows an exchange to write 50.0 to parameter 01.006 and 10.0 to parameter 01.016.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x40	Function code 64	0x40	Function code 64
0x00	Destination = 0	0x00	Destination = 0
0x00	Destination sub-node = 0	0x00	Destination sub-node = 0
0x15	Op Code = OC_WR_OBJLIST	0x15	Op Code = OC_WR_OBJLIST
0x00	Status = ST_REQ = 0	0x01	Status = ST_ACK = 1
0x01	Process ID = 1	0x01	Process ID = 1
0x02	Number of Objects = 2	0x02	Number of Objects = 2
0x12	Number of Objects x 9 = 18	0x08	Number of Objects x 4 = 8
0x01	Object Type = 1	0x01	Object Type = 1
0x06	Parameter Number = 6	0x06	Parameter Number = 6
0x01	Menu Number = 1	0x01	Menu Number = 1
0x00	Bit Number = 0	0x00	Bit Number = 0
0x01	Object Type = 1	0x01	Object Type = 1
0x10	Parameter Number = 16	0x10	Parameter Number = 16
0x01	Menu Number = 1	0x01	Menu Number = 1
0x00	Bit Number = 0	0x00	Bit Number = 0
0xF4	Register Data 1 = 500	0xBA	CRC MSB
0x01		0x04	CRC LSB
0x00			
0x00			
0x03	Register Data 1 Decimal Places = 1		
0x64	Register Data 2 = 100		
0x00			
0x00			
0x00			
0x03	Register Data Decimal Places 2 = 1		
0xE2	CRC MSB		
0x5D	CRC LSB		

ECMP

General

This document describes the sub-set of ECMP commands and features supported by the drive. The PDU for an ECMP exchange with the drive is shown below. It should be noted that values consisting of more than one byte are in Big Endian format, i.e. MS byte first and LS byte last. For example, the maximum response size in the request 0x0200 bytes and is given in the request as 0x02, 0x00.

Request PDU
Destination Addressing Scheme
Destination Address (Only with Default Route Addressing Scheme)
Source Addressing Scheme
Source Address (Only with Default Route Addressing Scheme)
Transaction ID
Response Size
Response Size
Request Code
Option Code = 0
Start of Request Code specific data

End of Request Code specific data

Response PDU
Destination Addressing Scheme
Destination Address (Only with Default Route Addressing Scheme)
Source Addressing Scheme
Source Address (Only with Default Route Addressing Scheme)
Transaction ID
Status
Chunk ID = 0
Request Code
Option Code = 0
Start of Response Code specific data

End of Response Code specific data

The following ECMP commands are supported.

Request	Command	Request	Command
General		0x22	FileWrie
0x00	Identify	0x23	FileClose
0x01	Info	0x24	FileInfo
0x02	Interrogate	0x25	FileDelete
0x03	Reset	0x26	FileState
Parameter		0x27	FilePos
0x10	Read	User Program	
0x11	ReadWithType	0x60	ProgramControl
0x12	Write	0x61	ProgramStatus
0x13	ObjectInfo	Misc	
0x14	GetNextObject	0x74	ModbusPDU
File		0x7E	Diagnostics
0x20	FileOpen		
0x21	FileRead		

Destination Addressing Scheme / Destination Address

If Destination Addressing Scheme is Intercept (1) then the request is directed at the drive and the Destination Address is omitted from the request. The example below uses the Intercept addressing scheme (highlighted in orange) for both the source and destination.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x03	Reset Request Code	0x83	0x80 | Reset Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x01	Reset Type = 1	0x01	Reset Type = 1
0x91	CRC MSB	0x00	Reset Status = 0
0xBC	CRC LSB	0x55	CRC MSB
		0x8E	CRC LSB

If the Destination Addressing Scheme is Default Route (2) then the destination is defined by the Destination Address as given below. If any other Destination Addressing Scheme is used there will be no response. Bits 0-3 of the Destination Address are the physical destination.

Bits 3-0	Physical Destination
0	Drive
1-4	Option Slots 1-4
5-14	Not valid
15	Broadcast message to all physical destinations

Bits 7-4 of the Destination Address are the logical destination which is a sub-node within the physical destination. The logical destinations outside the drive are not defined in this document, but the logical destination within the drive are as follows.

Bits 7-4	Logical Destination
0	Drive
1	Keypad connected to the user 485 port
2	Keypad connected to the keypad port
15	Broadcast message to all logical addresses in this table

The table below gives some examples of Destination Addresses.

Destination Address	Destination
0x00	Drive
0x01	Logical address 0 in option slot 1
0x0F	Logical address 0 in the drive and all option slots
0x20	Keypad on the User 485 port

The example below uses the Default Route addressing scheme for the destination where the physical destination is the drive and the logical destination is the drive, i.e. the drive itself handles and responds to the request. The source uses the Intercept addressing scheme. The addressing scheme area is highlighted in orange. Note that the Source Addressing Scheme and the Source Address in the request will become the Destination Addressing Scheme and Destination Address in the response and vice versa.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x02	Destination Addressing Scheme = 2	0x01	Destination Addressing Scheme = 1
0x00	Destination Address = 0 (Drive)	0x02	Source Addressing Scheme = 2
0x01	Source Addressing Scheme = 1	0x00	Source Address = 0 (Drive)
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x03	Reset Request Code	0x83	0x80 | Reset Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x01	Reset Type = 1	0x01	Reset Type = 1
0xE1	CRC MSB	0x00	Reset Status = 0
0x77	CRC LSB	0xB7	CRC MSB
		0xC9	CRC LSB

When a broadcast message is sent each of the destination nodes is expected to perform the required action but not produce a response. If a broadcast message is sent to the drive it will only perform an action if the Request Code is Reset or ProgramControl. All other request codes would be expected to give a response as a result of their action; therefore, no action is taken in response to a broadcast message.

Throughout the rest of this document the Intercept (1) addressing scheme is used for both the destination and source.

Source Addressing Scheme / Source Address
See Destination Addressing Scheme / Destination Address.

Transaction ID
The Transaction ID in the request is used as the Transaction ID in the response. Values other than 0 are recommended.

Response Size
Bits 11-0 specify the maximum length of the Response PDU in bytes. If the response is going to be longer than this then no response is produced. Bits 15-12 indicate that “chunking” is required. This feature is not supported and if these bits are non-zero a chunking error is produced.

Request Code
The Request Code specifies the action required, and if it is supported, the Response Code is the Request Code ORed with 0x80. If it is not supported, then an error response is given as shown in the example below where the Request Code is 4.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x04	Unsupported request code	0xFD	0x80 | Error Response Code (0x7D)
0x00	Option Code = 0	0x00	Option code = 0
0xAD	CRC MSB	0x00	Request Code error
0x20	CRC LSB	0x04	Request Code
		0x4D	
		0xF5	

Status

The response Status is always OK (0) unless a chunking error is produced as shown in the example below where an Identify request has been made, but the chunks allowed in the Response Size (MS 4 bits) has been set to 15. The Status is Chunking Error (-3).

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0xF2	Chunks allowed = 15 Maximum response size = 0x200	0xFD	Status = -3
0x00		0x00	Chunk ID = 0
0x00	Identify Request Code	0x29	CRC MSB
0x00	Option Code = 0	0x8E	CRC LSB
0x00	Number of Attributes = 0		
0xE0	CRC MSB		
0x69	CRC LSB		

Option Code

This must always be zero otherwise an error response is given as shown in the example below where the Option Code is 5.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x00	Identify Request Code	0xFD	Response Code = 0x80 | Request Code
0x05	Option Code = 5	0x00	Option Code = 0
0x00	Number of Attributes = 0	0x01	Option Code error
0xA3	CRC MSB	0x05	Option Code
0x2C	CRC LSB	0x8D	
		0xA5	

The following sections cover all the Request Codes available to users and give the format used in the Request code specific data in the request and the response.

Identify (0x00)

Identifies the drive product type, derivative, factory fit option module (option slot 4 if present). One additional attribute can be selected to give further information.

Request Code specific data request
Number of Attributes (0 or 1, limited to 1 if greater)
Attribute Type

Response Code specific data response
Category of product = 0 (Indicates this is a drive and not an option module)
Identifier size = 4 (Number of bytes present before the required attribute)
Product Type (11.063)
Drive Derivative (11.028)
Option Slot 4 *Module ID* (24.001) MSB
Option Slot 4 *Module ID* (24.001) LSB
Attribute Type
Attribute data start

Attribute data end

The following attribute types can be specified.

Attribute Type	Meaning	Description
0	Manufacturer name	“Control Techniques”
1	Product family	“GT8”
2	Product model	*Product Identifier Characters* (11.064) with each byte as a character followed by *Drive Rating And Configuration* (11.065) with each decimal digit as a character followed by *Additional Identifier Characters 1* (11.091) with each byte as a character followed by *Additional Identifier Characters 2* (11.092) with each byte as a character
3	Serial number	[*Serial Number MS* (11.053) x 10^9] + *Serial Number LS* (11.052) with each decimal digit as a character
4	Order number	Responds with no attributes
5	Date code	*Drive Date Code* (11.054) with each decimal digit as a character
6	Device name	*Drive Name Characters 1-4* (11.079) to *Drive Name Characters 13-16* (11.082) with each byte as a character

7	Version summary	"FW=#AABBCCDD;IM=#EE;DI=VFF.GG" AABBCCDD is the drive firmware version EE is the supported value for the user program/derivative images FF.GG=Derivative image version number if present

The following example shows an exchange using the Identify Request Code including Attribute 1 (Product family) with an M700 drive which has an - EtherNet factory fit option in Option Slot 4.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x00	Identify Request Code	0x80	0x80 | Identify Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x01	Number of Attributes = 1	0x00	Category = 0 (Drive)
0x01	Attribute = 1 (Product family)	0x04	Identifier Size = 4 bytes
0xBC	CRC MSB	0x00	*Product Type* (11.063)
0x28	CRC LSB	0x01	*Drive Derivative* (11.028)
		0x01	Option Slot 4 Module ID = 430
		0xAE	
		0x01	Number of Attributes = 1
		0x01	Attribute type = 1
		0x00	Length of attribute string MSB
		0x03	Length of attribute string LSB
		0x47	'G'
		0x54	'T'
		0x38	'8'
		0x8F	CRC MSB
		0x6C	CRC LSB

Info (0x01)

Gives information about communications with the drive and any other available ECMP servers.

Request Code specific data request

Response Code specific data response
Buffer Length MSB
Buffer Length LSB
Max Response Time MSB
Max Response Time LSB
Max Handle Periods MSB
Max Handle Periods LSB
Number of default routes
Start of default route addresses

End of default route addresses

Buffer Length

The maximum allowed length of the request and response excluding the Modbus data, Destination Addressing Scheme/Address and Source Addressing Scheme/Address. In the example below the actual buffer length is 500 bytes, but the response gives a buffer length of 492 bytes to allow 8 bytes for the Modbus Node Address, Modbus Function Code, maximum destination and source addressing information and the Modbus CRC.

Max Response Time and Max Handle Periods

Not supported by the drive.

Number of default routes

The number of ECMP servers that can be accessed using the default route addressing scheme. The default route addresses data then gives the destination address for each server that is available.

The following example shows an exchange using the Info Request Code with a drive which has a remote keypad connected to the user 485 port (physical address 0, logical address 2), an option module in Option Slot 2 (physical address 2, logical address 0) and an option module in Option Slot 3 (physical address 3, logical address 0).

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x01	Info Request Code	0x81	0x80 | Info Request Code

0x00	Option Code = 0	0x00	Option code = 0
0xAE	CRC MSB	0x01	Buffer length = 492 bytes
0x70	CRC LSB	0xEC	
		0x00	Max Response Time = 0
		0x00	
		0x00	Max Handle Periods = 0
		0x00	
		0x03	Number of default routes
		0x20	Keypad Destination Address
		0x02	Option Slot 2 Destination Address
		0x03	Option Slot 3 Destination Address
		0x8E	CRC MSB
		0xE7	CRC LSB

Interrogate (0x02)

Indicates if the specified ECMP request code is supported or not. This request can only check for one request code, and so the request code specific data is fixed apart from the required request code to check.

Request Code specific data request
Item Type = 0 (Request Code)
Number of request codes in following list = 1
Request Code

Response Code specific data response
Item Type = 0
Number of request codes in following list = 1
Request Code
Item Support. 0=not supported, 1=supported.

The following example shows an exchange using the Interrogate Request Code to determine if the drive supports the Write Request Code, which it does.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x02	Interrogate Request Code	0x82	0x80 | Interrogate Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Item Type = 0	0x00	Item Type = 0
0x01	Count = 1	0x01	Count = 1
0x12	Write Request Code	0x12	Write Request Code
0x81	CRC MSB	0x01	Write Request Code supported
0xBC	CRC LSB	0x23	CRC MSB
		0xB9	CRC LSB

Reset (0x03)

Requests a drive reset. If the Reset Type is 1 then the reset is equivalent to a user reset (i.e. pressing the reset button on the keypad or writing 1 to *Drive Reset* (10.033)). If the Reset Type is 255 then the drive processor is reset into its bootloader. Reset Status is always set to 0 to indicate success. If any other reset type is used no action is taken and the Reset Status is set to -1.

Request Code specific data request
Reset Type

Response Code specific data response
Reset Type
Reset Status

The following example shows an exchange using the Reset Request Code to initiate a reset equivalent to a user reset.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x03	Reset Request Code	0x83	0x80 | Reset Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x01	Reset Type = 1	0x01	Reset Type = 1
0x91	CRC MSB	0x00	Reset Status = 0
0xBC	CRC LSB	0x55	CRC MSB

		0x8E	CRC LSB

Read (0x10)

Request the value, of up to 10 parameters.

Request Code specific data request
Parameter Addressing Scheme = 0
Number of Parameters = 1 to 10
Parameter 1 Menu Number MSB = 0
Parameter 1 Menu Number LSB
Parameter 1 Parameter Number MSB = 0
Parameter 1 Parameter Number LSB

Parameter N Menu Number MSB = 0
Parameter N Menu Number LSB
Parameter N Parameter Number MSB = 0
Parameter N Parameter Number LSB

Response Code specific data response
Number of Parameters = 0 to 10
Parameter 1 Status
Parameter 1 Data Type
Parameter 1 Data MSB (1 to 4 bytes)

Parameter 1 Data LSB

Parameter N Status
Parameter N Data MSB (1 to 4 bytes)

Parameter N Data LSB

If there is an error that prevents any reading from taking place (e.g. the request Parameter Addressing Scheme is non-zero) then the error response is given as shown below, otherwise the response will include the data for as many parameters as will fit into the maximum possible response size. If an attempt is made to read more than 10 parameters, then only 10 will be read.

Response Code specific data response
Number of Parameters = 0
Parameter Status

The response for each parameter includes Parameter Status, Parameter Data Type and Parameter Data.

Parameter Status

Parameter Status	**Meaning**
0	Okay
-1	Parameter Addressing Scheme error (not zero)
-4	Parameter does not exist
-8	Not readable
-11	Request invalid
-12	Response too big

Parameter Data Type

Parameter Data Type	**Meaning**	**Description**
0	Boolean	Unsigned 8 bit with a value of 0 or 1
1	INT8	Signed 8 bit
2	UINT8	Unsigned 8 bit
3	INT16	Signed 16 bit
4	UINT16	Unsigned 16 bit
5	INT32	Signed 32 bit

The following example shows an exchange using the Read Request Code to obtain the value of parameter 01.006 which is a signed 32 bit parameter with a value of 50.0 and parameter 01.014 which is a signed 8 bit parameter with a value of 0.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0

0x00		0x00	Chunk ID = 0
0x10	Read Request Code	0x90	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Parameter Addressing Scheme = 0	0x02	Number of Parameters = 2
0x02	Number of Parameters = 2	0x00	Parameter 1 Status = Okay
0x00	Parameter 1 Menu Number = 1	0x05	Parameter 1 Data Type = 5 (INT32)
0x01		0x00	Parameter 1 Data = 500
0x00	Parameter 1 Parameter Number = 6	0x00	
0x06		0x01	
0x00	Parameter 2 Menu Number = 1	0xF4	
0x01		0x00	Parameter 2 Status = Okay
0x00	Parameter 2 Menu Number = 14	0x01	Parameter 2 Data Type = 1 (INT8)
0x0E		0x00	Parameter 2 Data = 0
0x6F	CRC MSB	0x50	CRC MSB
0x2D	CRC LSB	0xA0	CRC LSB

ReadWithType (0x11)

Requests the value of up to 10 parameters with number of decimal places and unit type information.

Request Code specific data request
Parameter Addressing Scheme = 0
Number of Parameters = 1 to 10
Parameter 1 Menu Number MSB = 0
Parameter 1 Menu Number LSB
Parameter 1 Parameter Number MSB = 0
Parameter 1 Parameter Number LSB

Parameter N Menu Number MSB = 0
Parameter N Menu Number LSB
Parameter N Parameter Number MSB = 0
Parameter N Parameter Number LSB

Response Code specific data response
Number of Parameters = 0 to 10
Parameter 1 Status
Parameter 1 Data Type
Parameter 1 Data MSB (1 to 4 bytes)

Parameter 1 Data LSB
Parameter 1 Decimal Places
Parameter 1 Unit Type

Parameter N Status
Parameter N Data MSB (1 to 4 bytes)

Parameter N Data LSB
Parameter N Decimal Places
Parameter N Unit Type

The request and response are the same as Read (0x10) except that the response for each parameter includes the Decimal Places and Unit Type value.

Decimal Places

Number of decimal places used to represent the parameter value.

Unit Type

The Units are used by a keypad connected to the drive. The actual values used are beyond the scope of this document.

The following example shows an exchange using the Read Request Code to obtain the value of parameter 01.006 which is a signed 32 bit parameter with a value of 50.0 and parameter 01.014 which is a signed 8 bit parameter with a value of 0.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x11	ReadWithType Request Code	0x91	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Parameter Addressing Scheme = 0	0x02	Number of Parameters = 2

0x02	Number of Parameters = 2	0x00	Parameter 1 Status = Okay
0x00	Parameter 1 Menu Number = 1	0x05	Parameter 1 Data Type = 5 (INT32)
0x01		0x00	Parameter 1 Data = 500
0x00	Parameter 1 Parameter Number = 6	0x00	
0x06		0x01	
0x00	Parameter 2 Menu Number = 1	0xF4	
0x01		0x01	Number of Decimal Places = 1
0x00	Parameter 2 Menu Number = 14	0x0F	Units = 15
0x0E		0x00	Parameter 2 Status = Okay
0x92	CRC MSB	0x01	Parameter 2 Data Type = 1 (INT8)
0xEE	CRC LSB	0x00	Parameter 2 Data = 0
		0x00	Number of Decimal Places = 0
		0x00	Units = 0
		0xC0	CRC MSB
		0xA0	CRC LSB

Write (0x12)

Requests that the specified data is written to up to 10 parameters.

Request Code specific data request
Parameter Addressing Scheme = 0
Number of Parameters = 1 to 10
Parameter 1 Menu Number MSB = 0
Parameter 1 Menu Number LSB
Parameter 1 Parameter Number MSB = 0
Parameter 1 Parameter Number LSB
Parameter 1 Data Type
Parameter 1 Decimal Places
Parameter 1 Data MSB (1 to 4 bytes)

Parameter 1 Data LSB

Parameter N Menu Number MSB = 0
Parameter N Menu Number LSB
Parameter N Parameter Number MSB = 0
Parameter N Parameter Number LSB
Parameter N Data Type
Parameter N Decimal Places
Parameter N Data MSB (1 to 4 bytes)

Parameter N Data LSB

Response Code specific data response
Number of Parameters = 0 to 10
Parameter 1 Status

Parameter N Status

If there is an error that prevents any writing from taking place (e.g. the request Parameter Addressing Scheme is non-zero) then the error response is given as shown below, otherwise an attempt will be made to write to the specified parameters up to a maximum of 10. The response will include the status for each attempted parameter write.

Response Code specific data response
Number of Parameters = 0
Parameter Status

Parameter Data Type

The parameter data specifies the format of the Parameter Data in the request as given in the table below. The request must contain the correct number of bytes for the selected data type otherwise an incorrect value may be written to a parameter or all the parameters may not be written. The number of bytes corresponding to the specified data type are taken from the request and are cast to the specified data type. For example (INT8)0xFF = -1 and (INT8)0x7F = 127. The value is then written to the parameter as a signed value.

Parameter Data Type	**Meaning**	**Description**
0	Boolean	Unsigned 8 bit with a value of 0 or 1
1	INT8	Signed 8 bit
2	UINT8	Unsigned 8 bit
3	INT16	Signed 16 bit
4	UINT16	Unsigned 16 bit
5	INT32	Signed 32 bit

Parameter Decimal Places

The Parameter Data is written as a decimal number with the specified number of decimal places. For example, if the Parameter Data is 500 and Parameter Decimal Places is 1 then the value is written as 50.0. The Parameter Decimal Places must be between 0 and 9, and the number of decimal places of the parameter – Parameter Decimal Places must be between -9 and 9, otherwise an error is given, and the parameter is not written. If the Parameter Decimal Places is -1, then the raw integer value is written. For example, if the Parameter Data is 1234, Parameter Decimal Places is -1 and the destination parameter has 3 decimal places, then the value written is 1.234.

Parameter Status

Parameter Status	Meaning
0	Okay
-1	Parameter Addressing Scheme error (not zero)
-4	Parameter does not exist
-9	Not writeable
-10	Over-range
-11	Request invalid
-12	Response too big
-13	Decimal places

The following example shows an exchange using the Write Request Code to write 50.0 to parameter 01.006 as a signed 32 bit value, and 1 to parameter 01.014 as a signed 16 bit value.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x12	Write Request Code	0x92	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Parameter Addressing Scheme = 0	0x02	Number of Parameters = 2
0x02	Number of Parameters = 2	0x00	Parameter 1 Status = Okay
0x00	Parameter 1 Menu Number = 1	0x00	Parameter 2 Status = Okay
0x01		0x42	CRC MSB
0x00	Parameter 1 Parameter Number = 6	0x3C	CRC LSB
0x06			
0x05	Parameter 1 Data Type = 5 (INT32)		
0x01	Parameter 1 Decimal Places = 1		
0x00	Parameter 1 Data = 500		
0x00			
0x01			
0xF4			
0x00	Parameter 2 Menu Number = 1		
0x01			
0x00	Parameter 2 Menu Number = 14		
0x0E			
0x03	Parameter 2 Data Type = 3 (INT16)		
0x00	Parameter 2 Decimal Places = 0		
0x00	Parameter 2 Data = 1		
0x01			
0xBF	CRC MSB		
0x6F	CRC LSB		

ObjectInfo (0x13)

Request information about a parameter or menu, or the parameter database version number.

Request Code specific data request
Parameter Addressing Scheme = 0
Count = 1
Info Type
Menu Number MSB
Menu Number LSB
Parameter Number MSB
Parameter Number LSB

Response Code specific data response
Count = 1
Parameter Access Status
Info Type
Info Data MSB

Info Data LSB

If there is an error that prevents any information being returned (e.g. the request Parameter Addressing Scheme is non-zero) then the error response is given as shown below, otherwise an attempt is made to return the required information. A menu and parameter number must be supplied in the request, but these are not always required. If these are not required, then any value may be provided.

Response Code specific data response
Count = 0
Parameter Access Status = -1 (Parameter Addressing Scheme Error) or -11 (Request Invalid)

Info Type
The following information can be requested.

Info Type	**Meaning**	**Description**	**Bytes**
0	No information		0
1	Minimum in menu	Lowest number parameter in specified menu.	2
2	Maximum in Menu	Highest number parameter in specified menu.	2
3	Parameter format	Parameter format information (see table below).	4
4	Minimum value	Parameter Data Type = 5 Minimum parameter value	1 byte = Parameter Data Type (INT32) 4 bytes = Minimum (32 bit signed)
5	Maximum value	Parameter Data Type = 5 Maximum parameter value	1 byte = Parameter Data Type (INT32) 4 bytes = Maximum (32 bit signed)
6	Units	Parameter units used by a keypad connected to the drive. The actual values used are beyond the scope of this document.	1
7	Data type	Parameter Data Type.	1
8	First menu	Lowest numbered menu.	2
9	Last menu	Highest number menu.	2
10	Menu customisation	If the menu has been customised (i.e. option module set-up menu) then the data is version number and CRC. If it has not been customised the version and CRC are both zero.	2 bytes = Version 4 bytes = CRC
11	Database version	Version number and CRC of the core parameter database and the core customisation table.	2 bytes = Core database version 4 bytes = Core database CRC 2 bytes = Core customisation version 4 bytes = Core customisation CRC
12	RAM location	Parameter RAM location in the array containing all parameters of the same size, i.e. if a 32 bit parameter the location is the index into the 32 bit parameter array.	4

The table below gives the meaning of the bits in the parameter format information.

Bit	**Code**	**Function**
31-30		Not assigned
29-22	UNITS	Keypad units
21	FL	Floating point value (always 0)
20	DF3	Keypad display format parameter
19	DF2	
18	DF1	
17	DF0	
16	PR	Pseudo read-only
15	FI	Filtered when displayed by keypad
14	DE	Destination set-up parameter
13	TE	String parameter
12	VM	Variable maximum and minimum
11	DP3	Number of decimal places
10	DP2	
9	DP1	
8	DP0	
7	ND	Parameter has no default
6	RA	Voltage or current rating dependant
5	NC	Not copied to/from SMART or SD card
4	NV	Not visible
3	PT	Protected from destinations
2	NR	Read not allowed
1	W	Write allowed
0	BU	Bit default/Unipolar

The table below gives the possible Parameter Data Types in the response.

Parameter Data Type	**Meaning**	**Description**
0	Boolean	Unsigned 8 bit with a value of 0 or 1
1	INT8	Signed 8 bit
2	UINT8	Unsigned 8 bit
3	INT16	Signed 16 bit
4	UINT16	Unsigned 16 bit
5	INT32	Signed 32 bit

The following example shows an exchange using the ObjectInfo Request Code to request the maximum value of parameter 01.006.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x13	ObjectInfo Request Code	0x83	0x80 | Request code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Parameter Addressing Scheme = 0	0x01	Count = 1
0x01	Count = 1	0x00	Parameter Access Status = Okay
0x05	Info Type = Maximum Value	0x05	Info Type = Maximum Value
0x00	Menu Number MSB	0x05	Data Type = 5 (INT32)
0x01	Menu Number LSB	0x00	Maximum Value = 5990 (599.0)
0x00	Parameter Number MSB	0x00	
0x06	Parameter Number LSB	0x17	
0x8C	CRC MSB	0x66	
0x49	CRC LSB	0x26	CRC MSB
		0x00	CRC LSB

GetNextObject (0x14)

Returns the menu and parameter number of the parameter above the specified parameter. If there are no parameters in the menu above the specified parameter, then the count in the response is 0 and the menu and parameter number are not included in the response.

Request Code specific data request
Parameter Addressing Scheme = 0
Menu Number MSB
Menu Number LSB
Parameter Number MSB
Parameter Number LSB
Count = 1 (must be 1)

Response Code specific data response
Parameter Addressing Scheme = 0
Count
Menu Number MSB
Menu Number LSB
Parameter Number MSB
Parameter Number LSB

If there is an error that prevents any information being returned (e.g. the request Parameter Addressing Scheme is non-zero) then the error response is given as shown below, otherwise an attempt is made to return the required information.

Response Code specific data response
Count = 0
Parameter Access Status = -1 (Parameter Addressing Scheme Error) or -11 (Request Invalid)

The following example shows an exchange using the GetNextObject Request Code to get the next parameter after parameter 01.006 which is parameter 01.007.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x14	GetNextObject Request Code	0x94	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Parameter Addressing Scheme = 0	0x00	Parameter Addressing Scheme = 0
0x00	Menu Number MSB	0x01	Count = 1
0x01	Menu Number LSB	0x00	Menu Number MSB
0x00	Parameter Number MSB	0x01	Menu Number LSB
0x06	Parameter Number LSB	0x00	Parameter Number MSB
0x01	Count = 1	0x07	Parameter Number LSB
0x5A	CRC MSB	0x1D	CRC MSB
0xEA	CRC LSB	0x1E	CRC LSB

FileOpen (0x20)

Request to open the specified file.

Request Code specific data request
Access Mode (0 to 3)
Additional Data Scheme = 0
Filename Size MSB (Filename size must be between 1 and 255)
Filename Size LSB
Filename char0

Filename char N
File handle MSB (File handle is ignored)
File handle LSB

Response Code specific data response
File Status
File handle MSB
File handle LSB

An attempt is made to open the specified file. If successful, the status is Okay (1) and a non-zero file handle is returned. Otherwise the Status indicates the reason for the error and the returned file handle is 0.

Access Mode

The possible access modes are given in the table below.

Access Mode	Meaning	Description
0	Info	Open a file for anything other than reading or writing
1	Read	Open a file for reading
2	Create	Create a new file for writing. If the file already exists, it is over-written.
3	Append	Open a file for writing. New data will be added to the end. If the file does not exist, it will be created.

Unless otherwise specified blocking access is used with all file requests, so that the action is completed and then the drive sends the response. Some actions can take a prolonged period to complete (i.e. writing a file to a media card fitted in the drive). Non-blocking mode can be selected by setting Bit 7 of access mode. When non-blocking mode is used the drive sends the response as soon as the request has been processed and indicates whether the required action has been started successfully or not. The action is then completed in a background thread. It is possible to poll the drive using the FileState request to determine when the action has been completed. The following non-blocking actions are supported when accessing a SMART card fitted in the drive:

1. FileOpen for writing a file. Before another request is sent via the comms interface that requested this as a non-blocking action, FileState must be used to make sure the action is complete. Once non-blocking has been selected using FileOpen, non-blocking is used for FileWrite or FileDelete until the file is closed.
2. FileWrite will use non-blocking if requested when the file was opened successfully. The data to write is not cached, but the comms buffer containing the FileWrite request is used. Therefore, it is important that none of the data is overwritten by subsequent requests. The FileState request sent via Modbus (i.e. including the Modbus CRC) is short enough to avoid overwriting data, and so this can be used to determine when the write is complete.
3. FileDelete will use non-blocking if requested when the file was opened successfully for reading. If a blocking operation is used the file is automatically closed when the delete is complete. This is not the case for non-blocking operation, and so the file must be closed with the FileClose function once FileState function indicates that the delete has been completed.

If an SD card is fitted in the drive, blocking or non-blocking actions can be selected, but all actions apart from deleting “MCDF/000” will be blocking and once the response has been given the action will be complete. Deleting “MCDF/000” will reset the MCDF folder on the card and may take some time. This action must be requested as non-blocking or else an invalid request response it given.

Filename Size and Filename

Filename Size gives the length of Filename in bytes, e.g. if Filename = “/par/diff” then Filename Size = 9. Filename is the file name including its path in ASCII characters. The following files are accessible.

Filename	Function	Access mode	File handle
/par/all	Raw parameter file for all menus	Info/Read	1
/par/NN	Raw parameter file for menu NN	Info/Read	
/par/diff	Required parameter differences from defaults. Defaults are loaded before the file data is written to parameters.	Info/Read/Write	
/par/macro	Differences with no default loading. Defaults are not loaded before the file data is written to parameters.	Info/Read/Write	
/par/boot	Same as /par/diff except that Parameter Cloning (11.042 is set to 4 in the file header. This file is normally intended for internal drive use.	Info/Read	
/scope/00	Scope file	Info/Read	2
/sys/prog/user	User program file	Info/Red/Write	1
/fs/(file path)	SD card file. (file path) is the file path and file name location from the root of the SD card file system.	Info/Read/Write/Append	3

Each file handle has a 5s timeout. If a successful file operation is not performed for 5s the file is automatically closed and the file handle is released.

When “/par/diff” or “/par/macro” files are opened for writing the file system operates in a similar way to a user action started via parameter mm.000 which requires that the drive is not enabled and that interaction between user parameters is disabled. The action cannot be started if the drive is enabled (i.e. *Drive Active* (10.002) = 1) or another user action is already active. Any attempt to open the file for writing under these conditions will give the appropriate error status. If the drive is enabled while the file is open for writing a {Data Changing} trip is initiated. This prevents the drive from being enabled during the data transfer. The order of the parameters in the file is not important because all parameter interaction (i.e. parameter routing and variable maximums) is disabled during transfer to a drive.

File Status

The table below gives the Status codes used with all the file operations.

Status	Meaning	Description
0	Processing	A non-blocking action is still active.
1	Okay	Successful
2	Okay more data	Read successful, but more data available
3	Okay end of file	Read successful but end of file reached
-1	File Handle	Invalid file handle
-2	Blocked	A non-blocking action is active and is preventing access to the file
-5	Not found	The specified file does not exist
-6	Read-only	File cannot be opened for writing
-7	Write-only	File cannot be opened for reading
-8	Not created	File cannot be created for writing
-9	No data	There is no data to be read from the file
-10	Wrong mode	The file is not open in the required mode for the requested action
-11	Too big	The data being written is too big for the file or there is no more space
-12	Protected	The requested operation cannot be performed on this file
-13	CRC	The supplied CRC does not match the actual CRC when the file is closed
-14	Length	The supplied length does not match the actual length when the file is closed
-15	Too many open	The file cannot be opened because a file handle is not available
-16	File invalid	The file is invalid or corrupt
-17	Invalid request	The request could not be processed
-18	No append	The file cannot be opened for appending
-19	Invalid state	The drive is in a state which prevents the file from being opened
-20	Incompatible	The file is not compatible with the drive
0x6E to 0x6F	Custom Success	These values can be returned as File Status in the response to a FileWrite(0x22) request code with a parameter difference file (/par/diff). They indicate that the write has been successful up to the point in the file where a marker (option slot delimiter) has been detected which indicates the end of drive parameter data. 0x6E indicates an entry in the file with Menu = Parameter = 0 and Value != 0. 0x6F indicates an entry in the file with Menu = Parameter = 0. Menu 0 parameters cannot be included in this type of file, but these markers can be present when a parameter difference file is stored on an NV media card.
0x70 to 0x7F	Custom Success	The file has been closed successfully (i.e. same as Okay), but Bits 3-0 signify the following warnings which indicate a hardware difference between the source and destination drives: Bit 0: One or more option modules is different. Bit 1: The drive derivatives are different. Bit 2: The drive ratings are different. Bit 3: Not used.

The following example shows an exchange using the FileOpen Request Code to open a “/par/diff” file for reading.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x20	FileOpen Request Code	0xA0	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x01	Access Mode = Read	0x01	File Status = Okay
0x00	Additional Data Scheme = 0	0x00	File Handle = 1
0x00	Filename Length = 9	0x01	
0x09		0x4A	CRC MSB
0x2F	Filename = “/par/diff”	0x38	CRC LSB
0x70			
0x61			
0x72			
0x2F			
0x64			
0x69			
0x66			
0x66			
0xF0	CRC MSB		
0x87	CRC LSB		

FileRead (0x21)

Request to read a specified number of bytes from a file already open for reading. The file must be opened in Read Access Mode.

Request Code specific data request
File Handle MSB
File Handle LSB
Number of Bytes Request MSB
Number of Bytes Request LSB

Response Code specific data response
File Status
Number of Bytes Returned MSB
Number of Bytes Returned LSB
Data 0

Data N

File Status

Unless an error occurs, Status can be used to determine whether there is more data left to read as follows.

Status	Meaning	Description
2	Okay more data	Read successful, but more data available
3	Okay end of file	Read successful but end of file reached
-9	No data	There is no data to be read from the file

The following example shows an exchange using the FileRead Request Code to read 8 bytes from a "/par/diff" file which has been opened in the example for the FileOpen request code. The first 8 bytes are "PAR" followed by the values of parameter 11.077 (4 bytes) and parameter 11.084 (1 byte).

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x21	FileRead Request Code	0xA1	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	File Handle = 1	0x02	File Status = Okay more data
0x01		0x00	Number of Bytes Response = 8
0x00	Number of Bytes Request = 8	0x08	
0x08		0x50	Data = "PAR", 11.077, 11.084
0x76	CRC MSB	0x41	
0x55	CRC LSB	0x52	
		0x00	
		0x00	
		0x00	
		0x00	
		0x00	
		0x03	CRC MSB
		0x93	CRC LSB

FileWrite (0x22)

Request to write a specified number of bytes to a file already open for writing. The file must be opened in Create or Append Access Mode.

Request Code specific data request
File Handle MSB
File Handle LSB
Number of Bytes to Write MSB
Number of Bytes to Write LSB
Data 0

Data N

Response Code specific data response
File Status

The following example shows an exchange using the FileWrite Request Code to write 8 bytes to a "/par/diff" file which has been opened for writing (Create). The first 8 bytes are "PAR" followed by 5 bytes of 0x00.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x22	FilwWrite Request Code	0xA2	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	File Handle = 1	0x01	File Status = Okay
0x01		0xB9	CRC MSB
0x00	Number of Bytes Request = 8	0x9E	CRC LSB
0x08			
0x50	Data = “PAR”, 11.077, 11.084		
0x41			
0x52			
0x00			
0x00			
0x00			
0x00			
0x00			
0xD8	CRC MSB		
0x82	CRC LSB		

FileClose (0x23)

Request to close a file that was given the specified File Handle when opened.

Request Code specific data request
File Handle MSB
File Handle LSB
Number of Data Bytes MSB
Number of Data Bytes
Number of Data Bytes
Number of Data Bytes LSB
CRC MSB
CRC
CRC
CRC LSB

Response Code specific data response
File Status

File Status

The request includes the expected number of data bytes and CRC of the data that has been transferred since the file was opened. These values are compared with the actual values calculated by the drive. If they are the same, then Status = Okay (1) otherwise Status = Length (-14) if the number of data bytes is incorrect or Status = CRC (-13) if the CRC is incorrect. CRC32 is used (reverse polynomial = 0xEDB88320, initialisation value = 0xFFFFFFFF).

If a “/par/diff” or “/par/macro files” is closed successfully with no errors, but there are important hardware difference between the source drive that created the file and the destination drive Status is not set to Okay, but to one of the “Custom success” values which give additional warning information about the difference between the source and destination drives. The “Custom Success” code is 0x70 and is ORed with one or more of the following warning bits.

Bit 0: One or more option modules is different
Bit 1: The drive derivatives are different
Bit 2: The drive ratings are different

The following example shows an exchange using the FileClose Request Code following the examples given for FileOpen and FileRead Request Codes. The correct number of bytes (8) and the correct CRC (0x933EFB9E) are supplied in the request, and so the Status in the response is Okay (1).

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x23	FileClose Request Code	0xA3	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	File Handle = 1	0x01	File Status = Okay
0x01		0xE8	CRC MSB
0x00	Data Bytes = 8	0x5E	CRC LSB
0x00			

0x00			
0x08			
0x93	CRC = 0x933EFB9E		
0x3E			
0xFB			
0x9E			
0x18	CRC MSB		
0x8C	CRC LSB		

FileInfo (0x24)

Requests up to 3 attributes about an open file that was given the specified File Handle when opened. The file can be opened with any Access Mode.

Request Code specific data request
File Handle MSB
File Handle LSB
Number of Attributes (0 to 3)
Attribute Type 1

Attribute Type N

Response Code specific data response
File Status
Number of Attributes
Attribute Type 1
Attribute Data 1 MSB

Attribute Data 1 LSB

Attribute Type N
Attribute Data N MSB

Attribute Data N LSB

Attribute Type

Attribute Type	**Meaning**	**Description**	**Size in bytes**
0	File length	File length in bytes. Returned as 0 if not available.	4
1	File integrity	Indicates the integrity of the storage system.	1
2	CRC	Not supported and indicates an invalid request.	N/A
>2	Not supported	Not supported indicates okay but does not include any attribute data in the response.	N/A

The bits in the file integrity have the following meaning:

Bit	**Meaning**
0	Always 1.
1	If this request is used when a parameter difference file is open for writing and the first 80 bytes of the file have been written (i.e. the file header) this bit indicates if the option module in Slot 1 is different to the one in the file header. This includes a module being fitted or removed since the file was created.
2	As Bit 1, but for Option Slot 2.
3	As Bit 1, but for Option Slot 3.
4	As Bit 1, but for Option Slot 4.

The following example shows an exchange using the FileInfo Request Code to obtain the file integrity for a file that was given File Handle 1 when opened.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x24	FileInfo Request Code	0xA4	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	File Handle = 1	0x01	File Status = Okay
0x01		0x01	Number of Attributes = 1
0x01	Number of Attributes = 1	0x01	File Integrity = true
0x01	Attribute Type 1 = 1 (File Integrity)	0x28	CRC MSB
0xB7	CRC MSB	0x73	CRC LSB

0x96	CRC LSB		

FileDelete (0x25)

Request that a file is deleted. The file will be closed automatically after being deleted if non-blocking operation is not selected when opening the file. The FileClose function must be used to close the file after being deleted if non-blocking operation is used. When a SMART card file is opened for creating the file is deleted, and so it is only possible to open a SMART card file for deleting if the access mode is INFO or READ.

Request Code specific data request
File Handle MSB
File Handle LSB

Response Code specific data response
File Status

The following example shows an exchange using the FileDelete Request Code to request that a file which was given File Handle 1 when opened is deleted. The file in this example cannot be deleted, and the response File Status is -12 (Protected). If the file had been deleted successfully Status would be 1 (Okay).

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x25	FileDelete Request Code	0xA5	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	File Handle = 1	0xF4	File Status = Protected
0x01		0xC8	CRC MSB
0xB6	CRC MSB	0x18	CRC LSB
0xB4	CRC LSB		

FileState (0x26)

Request an indication of whether a non-blocking action is still in progress on the specified file.

Request Code specific data request
File Handle MSB
File Handle LSB

Response Code specific data response
File Status

For any open file other than a file on a media card File Status is Not in Progress (-4). For an open file on a media card File Status is Processing (0) if a non-blocking action is still in progress, otherwise it is Not in Progress (-4).

The following example shows an exchange using the FileState Request Code.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x26	FileState Request Code	0xA6	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	File Handle = 1	0xFC	File Status = Not in Progress
0x01		0x39	CRC MSB
0xB6	CRC MSB	0xDE	CRC LSB
0xF0	CRC LSB		

FilePos (0x27)

Request to move the current file location pointer. The file can be opened with an Access Mode for reading or writing, but there may be restrictions based on the type of file, for example any file beginning with /par/ must be open for reading only.

Request Code specific data request
File Handle MSB
File Handle LSB
Reference Point
Offset MSB
Offset
Offset
Offset LSB

Response Code specific data response
Offset MSB
Offset
Offset
Offset LSB

Reference Point

Reference Point	Meaning
0	Start of File
1	End of File
2	Current Location

Offset
The offset in the request is a signed value from the Reference Point. The result, which is the offset in the response, is the location pointer in bytes from the start of the file and will be limited to be within the file.

The following example shows an exchange using the FilePos Request Code to offset the position by 2 bytes from the start of the file on an SD card (File Handle = 3). The offset in the response is 2 indicating that the location in the file is now the third byte (byte 0 is the first byte with and offset of 0, etc.)

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x27	FilePos Request Code	0xA7	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	File Handle = 3	0x01	File Status = Okay
0x03		0x00	Offset = 2
0x00	Reference Point = 0	0x00	
0x00		0x00	
0x00		0x02	
0x02	Offset = 2	0x90	CRC MSB
0x77	CRC MSB	0x73	CRC LSB
0x69	CRC LSB		

ProgramControl (0x60)
Stops or starts the on-board user program.

Request Code specific data request
Target = 0
Command
Sub-command = 0

Response Code specific data response
Program Control Status

Command
If Command is 0 or 1, then Onboard User Program Enable (11.047) = Command. Any other value gives an error.

Program Control Status
If Command is out of range, Target > 0 or Sub-command > 0, then Program Control Status is -1. Otherwise it is 0.

The following example shows an exchange using the ProgramControl Request Code to start a user program.

Request		Response	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x60	ProgramControl Request Code	0xE0	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Target = 0	0x00	Program Control Status = 0
0x01	Command = 1	0xD8	CRC MSB
0x00	Sub-command = 0	0x4A	CRC LSB
0xF8	CRC MSB		
0x79	CRC LSB		

ProgramStatus (0x61)

Request the status of the onboard user program. The returned User Program Status is the value of *Onboard User Program Status* (11.048).

Request Code specific data request
Target = 0

Response Code specific data response
Running State
Additional Items = 1
Item Type = 0
User Program Status MSB
User Program Status
User Program Status
User Program Status LSB

Running State

Running State	**Meaning**
0	*Onboard User Program Enable* (11.047) = 0
1	*Onboard User Program Enable* (11.047) = 1
2	Active trip = User Program (249)
3	No user program present

The following example shows an exchange using the ProgramStatus Request Code when there is no user program present.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x61	ProgramStatusl Request Code	0xE1	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Target = 0	0x03	Running State = 3
0xF1	CRC MSB	0x01	Additional Items
0xA2	CRC LSB	0x00	Item Type
		0x00	User Program Status 3
		0x00	
		0x00	
		0x00	
		0xF4	CRC MSB
		0x56	CRC LSB

ModbusPDU (0x74)

Send a Modbus PDU embedded within an ECMP request. The structure of the whole exchange including the outer Modbus frame is shown below. The embedded Modbus PDU can only use Function Code 64 (CMP). Although this can be used via the communications ports on the drive, it is more useful when using communications via an option module. Communication via an option module can only use ECMP, but by embedding a Modbus PDU within the ECMP exchange, it is possible to use CMP communications.

Request	
Outer Modbus Header	Modbus node address and Function Code (66)
ECMP header	Destination, Source, Transaction ID and Maximum response size
ECMP request code	Request Code = 0x74, Object Code = 0
Modbus PDU Size	Size of the embedded PDU in bytes (Function Code and CMP request data)
Modbus Function Code	Function Code = 64
Modbus PDU Request	PDU as specified in the section on CMP
Outer Modbus CRC	CRC applied to the whole exchange

Response	
Outer Modbus Header	Modbus node address and Function Code (66)
ECMP header	Destination, Source, Transaction ID, Status and Chunk ID
ECMP request code	0x80 | Request Code = 0xF4, Object Code = 0
Modbus PDU Size	Size of the embedded PDU in bytes (Function Code and CMP response data)
Modbus Function Code	Function Code = 64
Modbus PDU Response	PDU as specified in the section on CMP
Outer Modbus CRC	CRC applied to the whole exchange

The following example shows an exchange using the ModbusPDU Request Code with an embedded CMP Read object request to obtain the value of parameter 01.006 which has a value of 50.0. The embedded CMP exchange is highlighted in orange.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66

0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x74	ModbusPDU Request Code	0xF4	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Size MSB	0x00	Size MSB
0x0A	Size LSB	0x0F	Size LSB
0x40	Function Code 64	0x40	Function Code 64
0x00	Destination = 0	0x00	Destination = 0
0x00	Destination sub-node = 0	0x00	Destination sub-node = 0
0x10	Op Code = OC_RD_OBJ	0x10	Op Code = OC_RD_OBJ
0x00	Status = ST_REQ = 0	0x01	Status = ST_ACK = 1
0x01	Process ID = 1	0x01	Process ID = 1
0x01	Object Type = 1	0x01	Object Type = 1
0x06	Parameter Number = 6	0x06	Parameter Number = 6
0x01	Menu Number = 1	0x01	Menu Number = 1
0x00	Bit Number = 0	0x00	Bit Number = 0
0x5B	CRC MSB	0x03	Register Data Decimal Places = 1
0x02	CRC LSB	0xF4	Register Data = 500
		0x01	
		0x00	
		0x00	
		0x8E	CRC MSB

Diagnostics (0x7E)

Request diagnostic information. The drive only supports reading one attribute and the attribute must be of Attribute Type Device Status.

Request Code specific data request
Mode = 0 (Read)
Number of Attributes = 1
Attribute Type = 0 (Device Status)

Response Code specific data response
Number of Attributes = 1
Attribute Type = 0
Attribute Size = 1
Device Status

Device Status

Device Status gives the status of the drive as follows:

Bit 0: 1 if *Drive Healthy* (10.001) = 0
Bit 1: *Active Alarm* (10.104) > 0
Bit 2: *Drive Active* (10.002)
Bits 3 to 7: Not used = 0

The following example shows an exchange using the Diagnostic Request Code. Drive Healthy (10.001) = 0, Active Alarm (10.104) = 0 and Drive Active (10.002) = 0.

Request		**Response**	
0x01	Modbus node address = 1	0x01	Modbus node address = 1
0x42	Function code 66	0x42	Function code 66
0x01	Destination Addressing Scheme = 1	0x01	Destination Addressing Scheme = 1
0x01	Source Addressing Scheme = 1	0x01	Source Addressing Scheme = 1
0x01	Transaction ID = 1	0x01	Transaction ID = 1
0x02	Maximum response size = 0x200	0x00	Status = 0
0x00		0x00	Chunk ID = 0
0x7E	Diagnostic Request Code	0xFE	0x80 | Request Code
0x00	Option Code = 0	0x00	Option code = 0
0x00	Mode = 0	0x01	Number of Attributes = 1
0x01	Number of Attributes = 1	0x00	Attribute Type = 0
0x00	Attribute Type = 0	0x01	Attribute Size = 1 byte
0x50	CRC MSB	0x01	Device Status
0x7B	CRC LSB	0x75	CRC MSB
		0x49	CRC LSB

ECMP via Communications Option Modules

Ethernet

It is possible to send an ECMP PDU via Ethernet. This is accomplished using the TCP/IP protocol layer which has built in retransmission features and connection management.

TCP is a stream protocol; it defines a stream of bytes in either direction on a connection identified by a TCP port number. It does not define how that stream is segmented into messages; that is defined by the protocol layer utilising TCP. For the transmission of ECMP messages a stream header provides the mechanism to detect the start of a message within a stream of bytes, how long the message should be and a checksum to enable a software process to verify the 4 bytes are a valid header.

Both the request and response stream headers are in the following format.

Byte	**Request/Respnse**
0	Length of ECMP PDU MSB
1	Length of ECMP PDU LSB
2	0
3	0xFF ^ Byte 0 ^ Byte 1 ^ Byte 2
	ECMP PDU Start

	ECMP PDU End

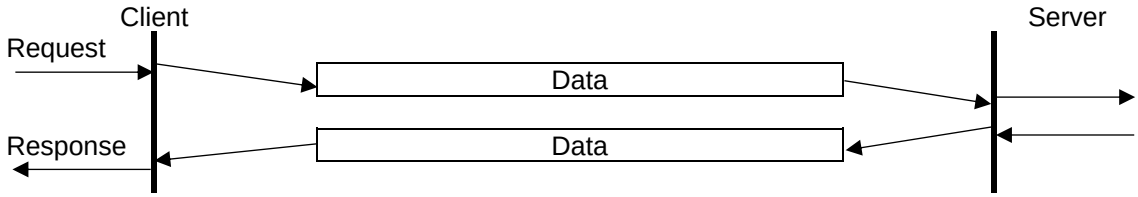
Communication takes the form of a client – server relationship. The drive and its Ethernet interface are always the server.

Client

1. Sends a request to a server over a connection that has been established.
2. Waits for a response; the duration of the wait is configurable by the client.
3. Reads the response, processes it and presents it to the client application.

Server

1. Listens for new connection requests on TCP Port 6160
2. Establishes connections
3. Listens for requests made on a connection
4. Processes requests and responds



The client can use any port number available to it for the purpose of TCP connection so long as the server port 6160 is used.