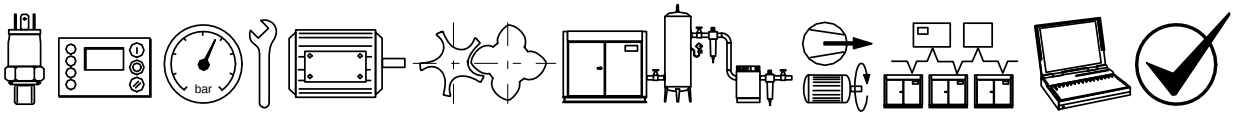


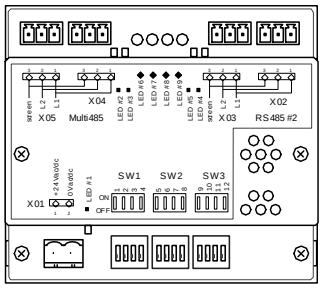
MODBUS
RTU

metacentre™ **System Products**

AirMaster™ **Embedded Controllers**

Scope

This document is applicable to 'direct' AirMaster controller Modbus ports and Metacentre Modbus Gateway module Y07ENER03.00, inclusive variants of the same module, as fitted to Modbus Gateway units Y08CM18.00 and Y08ENER01.00



The Enercon ANI1 or ANI2 Modbus Gateway devices, or CommBox unit, will not comply with all the information, methods and procedures described in this document; use manual MANY0313A.00

MODBUS RTU (Remote Terminal Unit) System Gateway Module

A Metacentre Modbus RTU Gateway module can be used to establish communications with any AirMaster/Metacentre device that is equipped with an RS485 communications port utilising the CMC Multi485 protocol.

MODBUS: Many differing types of data communication media exist of which the most popular, cost efficient and proven for industrialised environments is RS485. The RS485 standard describes the methods and media by which data communication signals are propagated in accordance with the standard. The standard does not, however, describe the timings or format of the data messages that are transferred by the medium. This subject is known as the Protocol. Various equipment manufacturers employ differing types of protocol, the majority of which are designed to satisfy specific application tasks. These are known as proprietary protocols; the differences inherent in each generally preventing one type from being able to communicate successfully with another. There are a number of 'open' protocols on the market, each designed to be generic and non-application specific, most of which can be used under license and/or require the use of additional hardware to implement. Among the 'open' protocols available the most popular is MODBUS RTU; this protocol is free of license and requires no additional hardware. MODBUS RTU is intended to be a true open protocol independent of platform or application and is commonly regarded as the minimum common denominator for systems integration. There are limitations to the MODBUS RTU protocol that can inhibit its use for highly integrated mass data transfer systems; but for single point to point interfacing, at low to moderate speeds, the protocol offers an established and proven industrial default standard that is well understood by system integration and solution providers.

MODBUS Gateway communication is RS485 master-slave. The Gateway acts as a transparent interface to enable a remote 'master' device to be able to communicate with AirMaster/Metacentre Multi485 network enabled 'slave' device(s). The MODBUS RTU data construction and formatting for a 'master' device is the subject of this document. This information is intended for a systems integrator to facilitate set-up of a 'master' device in order to communicate successfully with a controller equipped with a MODBUS port or through a MODBUS Gateway.

The company is unable to offer a 'master' device programming, installation, and commissioning or support service. An appreciation of viability, applicability, costs, support and delivery timescales for this operation, from an in-house or third party specialist source, is the responsibility of the user and should be established prior to implementation of the Modbus link and/or Modbus Gateway facility.

Embedded Controllers

For AirMaster embedded controller products refer to:
MODBUS RTU, Modbus Table(s) and Modbus Table Order Form

Systems Products

For Metacentre compressed air systems products refer to:
MODBUS RTU and Metacentre Compressed Air Systems

Contents:

MODBUS RTU

.

1. MODBUS Table(s)

For each specific type of AirMaster Controller, or differing variants or versions of application software implemented on the same controller type, a 'MODBUS Table' is required. This document discusses generic MODBUS communications and how to implement the software specific 'MODBUS Table' information. For multiple controller systems MODBUS communication formatting may differ from controller to controller and you may require more than one 'MODBUS Table'.

(!) Always check the software variant identification and version number for a controller or unit with the variant and version of the 'MODBUS Table' supplied. In some instances the information contained in a 'MODBUS Table' may not be applicable to a controller or unit installed with the same software variant but a different version number.

2. General

MODBUS RTU (Remote Terminal Unit) is a master-slave type protocol. An AirMaster or Metacentre Unit Controller functions as the slave device. Information requests or commands are communicated from master to slave only. A controller, connected directly, or through a MODBUS Gateway, will always respond to communications from a remote master device in accordance with the MODBUS RTU protocol standard.

3. Communication Link

MODBUS is implemented using a two-wire RS485 industry standard communications link operating in master-slave mode. Some types of AirMaster controller are fitted with a direct MODBUS enabled communications port where a connection can be made directly to the controller. Other controller types are fitted with a CMC Multi485 enabled network port; in this instance a MODBUS Gateway device is required.

(!) polarity of the two RS485 wires (L1+ and L2-) is important; reversal will disrupt communications.

4. RS485 Serial Data Format

RS485 MODBUS ports are 2-wire operating with an asynchronous serial data format:
1 start bit / 8 data bits / 1 stop / no parity (1,8,1,N) - transmitted at 9600 baud.

5. Message Data Format

The bytes of the MODBUS RTU message must be sent in one message package. The RTU protocol allows for a maximum pause of 1.5 byte-times between 2 consecutive bytes of a message.

(!) A pause longer than 1.5 byte-times will render the message invalid and it will be ignored.

Message data format is dependant on function and will consist of a combination of the following elements:

1) Destination address (slave network address)
2) Function Code
3) Data start address (slave register start address)
4) Number of registers, number of bytes of data
5) Message data
6) CRC checksum

5.1 Message Destination Address

The 'destination address' must be correct for the 'slave' controller device for which the message is intended. An address can be from 01Hex to EFHex. If using a MODBUS Gateway the gateway unit is transparent and addresses must be for the destination 'slave' controller or unit.

There is a convention for ‘slave’ controller addresses for CMC Multi485 networked systems. For systems with multiple ‘slave’ controllers each controller must be set with a unique address.

For air compressor systems each air compressor controller must be set with a unique address starting at ‘1’ (01 Hex) increasing sequentially up to ‘12’ (0C Hex). For air compressor systems consisting of air dryer units, each air dryer controller must have a unique address starting at ‘33’ (21 Hex) increasing sequentially.

For systems that contain system control and monitoring units, air compressor management units, or other remote I/O monitoring and control products, unit addresses will be100 (64Hex) and higher.

All controllers are delivered with a network address set for ‘1’ (01 Hex), unless otherwise specified. The network address is menu settable and can be adjusted. The menu address is displayed in decimal values.

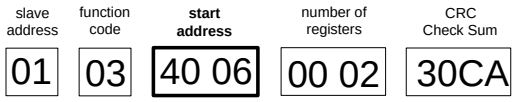
5.2 Message Function Codes

The message function code defines the required data processing operation of the slave controller. Although several types of message function codes are defined by the MODBUS standard, only the message function code types working directly with registers are implemented on controller units:

03H : Read From Register(s) – Get (Get Data) or Adv (Advise Data)
10H : Write To Register(s) – Set (Set Data) or Cmd (Command Instruction)

Any other message function code type will result in an EXCEPTION response.

5.3 Message Data Start Address



The message data start address (16bit word) designates the initial register address location in the controller from which the data is processed. Start address information is contained in the ‘MODBUS Table’.

Note: high-byte transmitted first followed by low-byte.

5.4 Message Data

The message data content depends on the message function code type.

03Hex – Get (read from register)

Slave address + function code ’03 Hex’ + start address of registers in slave memory +
16bit integer value that determines the size (in 16bit ‘word’ registers) of the message data being requested (00 02 = 2 registers of data). This is the number of 16bit registers to read. This information is contained in the ‘MODBUS Table’.

10Hex – Set (write to register)

slave address	function code	start address	number of registers to be set	number of bytes of data	**DATA** 1st register byte 0	byte 1	**DATA** 2nd register byte 2	byte 3	CRC Check Sum
01	10	40 18	00 02	04	00	00	1B	5F	880E

Slave address + function code ’10 Hex’ + start address of register(s) in slave memory to be set + 16bit (integer valve of the number of registers to be set) + 8bit ‘byte’ (integer value for the number of following data bytes) then the ‘data’ itself. This information is contained in the ‘MODBUS Table’.

Note: A function ’10 Hex’ Set message also requires an additional byte defining the number of ‘data’ bytes in the data message. This will always be the number of ‘registers’ multiplied by 2 as each ‘data’ register consists of 2 bytes (if number of ‘data’ registers = 2 then number of ‘data’ bytes = 4).

5.5 Message CRC Checksum

slave address	function code	start address	number of registers	**CRC Check Sum**
01	03	40 06	00 02	30CA

The CRC (Cyclical Redundancy Check) is a check-sum generated by means of ‘A001H polynomial’.

The CRC is two bytes containing a 16-bit binary value (word). The CRC value is calculated by the transmitting device that appends the CRC to the end of the message. The receiving device recalculates the CRC value prior to processing of a received message and compares the result to the actual CRC value appended to the message. If the two values do not match the message is regarded as invalid. The CRC is initiated by first preloading a 16bit register to all 1's (FFFF Hex). Then a process begins of applying each consecutive 8bit byte of the message to the register contents using an exclusive ‘OR’ calculation. The result is shifted one bit in the direction of the least significant bit (LSB), with the most significant bit (MSB) set at ‘0’. The LSB is then examined; if ‘1’ the register content is applied to the polynomial value ‘A001’ Hex (1010 0000 0000 0001) using an exclusive ‘OR’ calculation - if ‘0’ no exclusive OR takes place. This process is repeated until eight ‘bit’ shifts have been performed. After the eighth bit shift, the next 8bit message byte is applied to the register contents using an exclusive ‘OR’ calculation. The bit shift and re-calculation process is then repeated again. When all message bytes have been processed the final content of the 16bit register is the message CRC value.

Only the 8bits of ‘data’ in each message character is used for generating the CRC; start, stop and parity bits are ignored.

Note: When the 16bit CRC value is appended to a message, the low order byte must be transmitted first followed by the high order byte. An incorrect or byte reversed check sum will render the message invalid and it will be ignored.

6. Slave Response Timeout

A slave controller may not answer immediately. Ensure the ‘slave timeout’ setting of the ‘master’ device is set to a value no less than 500ms. If the ‘slave’ device fails to receive a valid message due to a communication disruption, parity error, CRC error or other reasons, no response is given and the master must process a timeout condition in this instance. If the ‘slave’ receives a valid message that cannot be processed an exception response will be returned.

7. Message Answer From Slave to Master
The format of the 'slave' controller answer is similar to the original master request format; the message data content depends on the message function code type.

The 'address' and 'code' of the slave answer is identical to the original request message; the address is the 'slave' device address and the 'code' is a repeat of received function code type from the master. The remainder of the message is dependant on the requested function code type. The CRC checksum is re-calculated for the answer message characters using the specified CRC process.

03Hex – Get: read from register (or 'Adv' Advise)

slave address	function code	start address	number of bytes of data	DATA 1st register byte 0	DATA 1st register byte 1	DATA 2nd register byte 2	DATA 2nd register byte 3	CRC Check Sum
01	03	40	18	04	00	00	1B	5F BE61

1) slave address 1byte
2) function code 1byte
3) bytes of data 1byte (number of bytes in 'data' answer)
4) data (high byte of each register transmitted first)
5) CRC checksum 2bytes (low byte first followed by high byte)

10Hex – Set: write to register (or 'Cmd' Command)

slave address	function code	start address	number of registers set	CRC Check Sum
01	10	40 18	00 02	D40F

1) slave address 1 byte
2) function code 1 byte
3) number of registers set 2 bytes
4) CRC checksum 2 bytes (low byte first followed by high byte)

8. Exception Response
If the 'slave' device receives a request that cannot be processed an 'exception response' is given. An exception response message consists of the following elements:
1) Slave Network Address (1 byte): Slave address identification
2) Function Code (1 byte): In a normal response, the slave repeats the function code of the original master request. All function codes have an MSB (most significant bit) of 0 (values are all below 80 hexadecimal). In an exception response, the slave sets the MSB of the function 'code' to 1. This makes the 'code' value 80 Hex greater than the received 'code' value from the master.
3) Data (1 byte): The 'data' response will contain a '1 byte' value exception code.
4) CRC Checksum (2 byte).

slave address	function code	error code	CRC Check Sum
01	90	04	4DC3

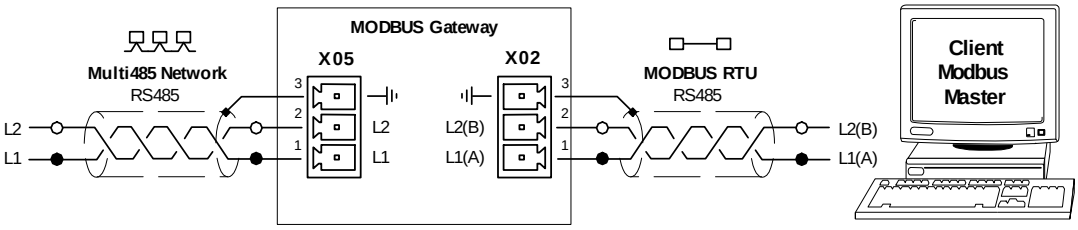
Exception Codes:
01H Illegal Function Code
The requested 'code' function is not supported.
02H Illegal Data Address
The requested 'data start address' is not supported.
03H Illegal Data Value
The requested 'data' value is not supported.
04H Function Error
The slave cannot execute the request or the request type is inhibited.

9. Troubleshooting

***Problem**: No 'slave' response or corrupt MODBUS message*
Solution: - Check that the 'slave' controller is set for the anticipated slave address
- Check that all 'slave' controllers are set with a unique system address
- Check that the controller is set for MODBUS RTU mode (if applicable)
- Check that the 'master' is operating in MODBUS RTU mode
- Check that the 'master' baud rate, parity bit and number of stop bits are correct
- Check that the 'master 'response timeout is set for a minimum of 500ms
- Check that the 'master' is implementing the specified CRC check sum process
- Check RS485 wiring polarity and security of connections

***Problem:** Last character of MODBUS message is corrupted*
Solution: - Add a delay of 2ms after last character received before releasing RTS signal

***Problem:** The MODBUS master message is reflected in the slave answer*
Solution: - Inhibit RX/TX echo on 'master' device communications port



MODBUS Tables:

A ‘MODBUS table’ describes the “items” used to access information in the memory registers of different types of controller, or similar controllers using different application software variants or versions. The MODBUS Table will contain the valid message items (“Name”) together with the Function Code (Function), Register Start Address (“Register Address”), Register Size (“Register Length”) and a definition for coding and decoding the item data (“Coding”). A ‘MODBUS Table' order form, detailing the required order information, can be found on the last page of this document.

1. Table Item Format

Each ‘item’ of a ‘MODBUS Table’ will define the massage format to read or set the information contained in the slave controller register(s):-

Name	Descriptive ‘name’ or ‘item tag’ for the data item. The ‘Name’ is not used in code or message formatting and serves only as a reference for the defined item.
Function	The Hex code required that instructs the slave (AirMaster controller) to perform a GET, ADV (Advise), CMD (command) or SET function.
Register Address	The slave controller register start address for the defined processing function.
Register Length	The number of registers to be processed.
Coding	How to construct or interpret the data elements of a message.
Menu	Controller menu item reference.

Note: see “MODBUS RTU” for a detailed description of ‘Function’, ‘Register Address’ and ‘Register Length’ formats.

2. Name and Function

The ‘name’ for each table item will always start with 3 characters that describe the function type:

Adv	Advise Function (03Hex) – same format as a Get function, see ‘Advise Function’.
Get	Read from register (03Hex)
Set	Write to register (10Hex)
Cmd	Command (10Hex) – same format as a Set function; will instruct the slave to perform a defined action or process

3. Coding

Item coding definitions specify the ‘number of data bytes’ and the ‘data conversion type’. In some instances a data message may contain multiple sets of data items; an ‘Advise’ message for example. In this instance the ‘start location of data’ within the message is also specified to enable extraction of the required data item from the entire message data.

Number of data bytes:
This specifies the length of the item data in bytes (6 = 6 bytes (3 registers) of data)

Start location of data bytes:

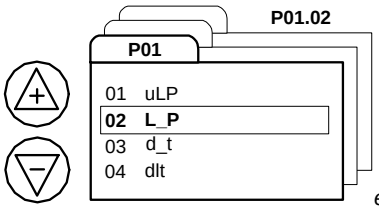
slave address	function code	number of data bytes to follow	1st register byte 0	1st register byte 1	**2nd register byte 2**	**2nd register byte 3**	3rd register byte 4	3rd register byte 5	CRC Check Sum
01	03	06	09	00	**00**	**65**	00	A8	304D

If a data message consists of more than one set of data items (multiple item data message) the ‘start location’ specifies where the first byte of the data associated with in item begins. If, for example, a 6 byte (3 register) answer is returned that consists of three different ‘2 byte’ item data values, a ‘start location of data bytes’ = ‘2’ indicates that the item data starts with the 3rd byte (byte 2) of the data message. The 1st byte of a data message is regarded as byte 0(zero). In this instance the ‘number of data bytes’ will be ‘2’ indicating that the data associated with the item is 2 bytes of data in length. A ‘start location’ of byte ‘2’ and register length of ‘1’ (register = 2 bytes) means the data is contained in the 3rd and 4th bytes of the data message. If no ‘start location’ is specified then data associated with the item will start with the first byte (byte 0) of the message data.

Data Conversion Type:
This specifies how to interpret the data; refer to the 'Data Conversion Type' list in the Modbus Table. For example: If the 'Data Conversion Type' = CODED, STATUS then the decimal integer value of the data has a defined meaning; refer to the 'STATUS' Coded data list in the 'MODBUS Table' for definitions. If the 'Data Conversion Type' = PSI then the decimal integer value of the data is 'pressure' in 'psi' units.

4. Menu Reference

The menu structure of a controller has menu pages that contain a number of menu page 'items'.



example menu and items

For example: menu pages P00, P01, P02 > P'n'.

Menu page P00 is the normal running list of display items that can be accessed and viewed on the controller display without access code. These items are 'read only' and consist of status, hours run and other general control or monitoring value(s).

Menu pages other than P00 are setup and configuration items that require an 'access code' when accessing the items on the controller display.

Each menu page has a list of items that are referenced '1, 2, 3 > 'n'. For example, a menu reference of P01.02 "L_P" refers to menu item '2' of menu page P01. Each menu page item reference also has a two or three alphanumeric character item identification that is displayed by the controller. With a menu reference the controller manual can be examined to determine the exact function, definition, scope and limits for the specified item value.

Items that do not have a 'Menu' reference are general controller status or menu page P00 items.

5. 'Adv' Advise Function

The 'Advise' function is a special type of 'Get' function that can only be implemented when using a MODBUS Gateway. If utilising a direct to slave controller MODBUS connection 'Advise' functions are not available. Each 'Advise' function item has an equivalent 'Get' function; both functions will provide a similar result.

Controllers on a Multi485 network will routinely broadcast key value and status specific data to all other controllers on the network. This information is used, for example, by a system management unit for systems monitoring and control functions. A MODBUS Gateway will automatically capture, store and continuously update these information items for each controller on the network.

This facility provides a method of retrieving 'Adv' data items directly from the MODBUS Gateway resulting in a faster response time for information requests from a master. The method also has the advantage of reducing the amount of data traffic on the Multi485 network enabling system management controllers to perform there functions without potential communication delays. For this reason MODBUS 'Adv' functions are preferable to 'Get' functions when implemented on a Multi485 network that consists of a system management controller with multiple machine controllers.

6. 'Adv' Advise Function – Single Item Format Option

Controllers or units on a Multi485 network routinely broadcast general status and key performance information. The Modbus Gateway will capture and store each 'Broadcast' detected. The Gateway 'Broadcast' registers will always contain the latest 'broadcast' information for each controller or unit on the Multi485 network. When a Modbus 'Adv' request is made the Modbus Gateway will respond immediately with information from it's own 'Broadcast' registers for the unit address specified. This function reduces network activity and enables a faster Modbus response to commonly requested data.

A standard 'Advise' function defined in the 'MODBUS Table' will show the entire 'broadcast' being returned as a response. The table will define for each 'name' item where in the returned data message the actual requested data can be found. The 'master' must then extract the required data from the returned data message. This method is very efficient as the master can extract all 'broadcast' data from the single returned data message without the need to perform multiple requests for each individual data item contained in a single slave controller 'broadcast' message.

Some 'master' devices may not be equipped with the necessary data message memory to handle a large message of many bytes or have the ability to extract multiple data items from a single data message item. In this instance an alternative 'Advise' function request method can be implemented. If the 'Advise' items of a 'MODBUS Table' are examined it will be seen that the 'Register Address' for each individual 'Advise' item contained in a single slave controller 'broadcast' message will have the same start address (Register Address). If the entire 'broadcast' data message is 7 registers (14 bytes) in length and only the 2nd register (2 bytes) of item data is required, it is possible to specify a 'Register Address' that is 2 bytes higher (skip the first 2 bytes of the broadcast data message) with a 'Register Length' that is consistent with the required item data length. This will instruct the MODBUS Gateway to extract the 2 bytes of required item data from the entire broadcast data message and only return the required 2 bytes of data as a response. Using this method an 'Advise' function can be handled by a 'master' in exactly the same way as a 'Get' function.

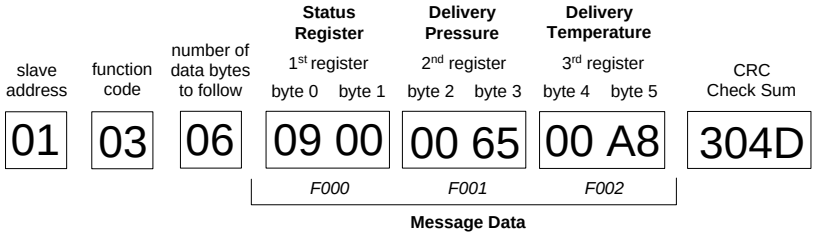
For Example (AdvDeliveryPressure):

The 'broadcast' of an example slave controller may be 6 bytes of data (3 registers) in length starting at register address location 'F000' Hex. The 1st byte (byte 0) is 8bits coded status, the 2nd byte (byte 1) is 8bits status flags which together form a single 16bit status register (1st register). The 3rd and 4th bytes (byte 2 and byte 3) are a single16bit register (2nd register) containing a 'delivery pressure' value. The 5th and 6th bytes (byte 4 and byte 5) are a single 16bit register (3rd register) containing a 'delivery temperature' value. From an example 'MODBUS Table' it may be seen that the 'Register Address' for all four of these separate 'Adv' items is 'F000 Hex' (the start address of the entire 'Broadcast' message that contains the data specified).

An entire 'Broadcast' message may, for example, contain 3 registers (6 bytes) of data. For a particular item the 'Modbus Table' may show the 'start address' for the entire broadcast to be 'F000' with a length of 3 registers (6 bytes). The Modbus Table will indicate that the required data is 2 bytes long (number of data bytes) starting at the 2nd byte of data in the entire broadcast (start location of data bytes).

Name	: AdvDeliveryPressure
Modbus Function	: 03
Modbus Register Address	: F000 *(start address of entire Broadcast message)*
Modbus Register Length	: 0003 *(length of entire Broadcast message)*
Coding	: Number of data bytes = 2 *(length of AdvDeliveryPressure data)* Start location of data bytes = 2 *(the 2 data bytes of the AdvDeliveryPressure data item start at byte 2 in the Broadcast message = bytes 2 and 3 of the message)* Data Conversion Type = PSI
Master Request Message	"01 03 F000 0003 36CB" (36CB = CRC check sum)
Slave Answer Message	"01 03 06 09 00 <u>00 65</u> 00 A8 304D" (304D = CRC)
Coding = PSI	'00 65'Hex = 101 decimal = 101 psi

ⓘ *Register addresses shown are examples only*

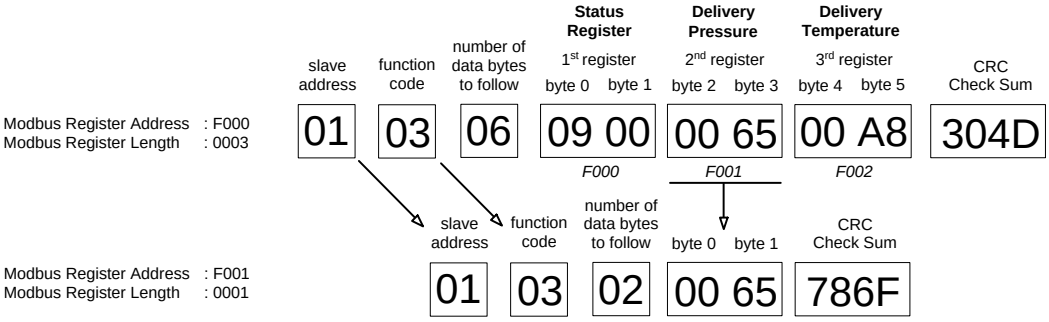


If only the 'delivery pressure' (AdvDeliveryPressure) data value is required a new 'Advise' request message format can be constructed from the Modbus Table definition:

Name : AdvDeliveryPressure
Modbus Function : 03
Modbus Register Address : F001 *(start at the second register, byte 2, of the Broadcast)*
Modbus Register Length : 0001 *(only return one register, 2 bytes, of data)*

Using the new 'Advise' message format the Modbus Gateway will return only the 2nd Broadcast message register (2 bytes) containing the 'delivery pressure' data value.

Master Request Message "01 03 F001 0001 E6CA" (E6CA = CRC check sum)
Slave Answer Message "01 03 02 00 65 786F" (786F = CRC check sum)
Coding = PSI '00 65'Hex = 101 decimal = 101 psi

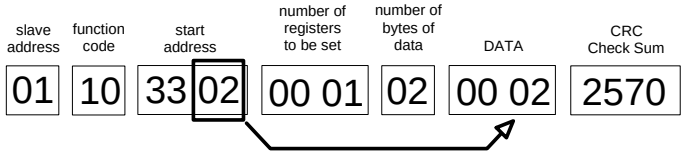


It is only possible to manipulate a Modbus message format using 'registers' (1 register = 2 bytes = 1 word = 16bits); it is not possible to manipulate addresses or register lengths to a single byte of data. At least one register (2 bytes) of data must be specified even if only one byte of information is required. The 'master' must extract the required byte of data from the returned message.

The data type (the definition of the returned data) may be different when using an 'Advise' function than it is when using a 'Get' function for the same information. The 'delivery pressure' returned by an 'Advise' function will be 2 bytes in length and will represent pressure as an integer value in 'psi' units (PSI). The 'delivery pressure' returned by a 'Get' function may, for example, be 4 bytes (2 registers) in length and represent a 32bit signed integer value in miliBar units (mBAR). Always check the item 'Coding' definition to establish the data definition type.

Register addresses shown are examples only

7. 'Cmd' Command Function
A 'Command' function will instruct the 'slave' controller or unit to execute a pre-defined action or process. With a command type message the content of the 'message data' from the 'master' must always be the same value as the 'lower byte' of the command register address. For example: if the command item 'Register Address' = 3302 then the 'data' value must be '00 02' Hex.



It is the act of setting the specified register in the 'slave' controller with the defined 'data' value that initiates the action or process. An incorrect 'data' value will result in an exception response. If the 'command' is accepted the 'slave' will answer with a normal 'Set' register response. If the slave is unable to execute the command it will give a code '04' exception response.

Example:
Using a command function item to set the specified item register to the correct value, the 'slave' controller is instructed to perform the defined action or process. In the case of a 'CmdStart' item, for example, the 'slave' controller is instructed to start the machine. The implementation of a 'Cmd' function message by the 'master' is identical to a 'Set' function message; both operations use function code '10 Hex' to write data to a slave controller register.

Name	: CmdStart
Modbus Function	: 10
Modbus Register Address	: 3300
Modbus Register Length	: 0001
Coding	: Number of data bytes = 1

CmdStart (to slave at address '01' Hex)

Master Command Message	"01 10 3300 0001 02 0000 A553" (A553 = CRC check sum)
Slave Answer Message	"01 10 3300 0001 0E8D" if start command executed or "01 90 04 4D C3" exception response if not executed, '90' = repeat of '10' function code with MSB set to '1' and '04' = exception error code.

(!) *Register addresses shown are examples only*

8. 'Get' Function
Using the MODBUS Table a read data (Get) function message can be constructed:

Name	: GetDeliveryPressure
Modbus Function	: 03
Modbus Register Address	: 4006
Modbus Register Length	: 0002
Coding	: Number of data bytes = 4 Data Conversion Type = mBAR

GetDeliveryPressure (request to slave at address '01' Hex)

Master Request Message	"01 03 4006 0002 31CA" (31CA = CRC check sum)
Slave Answer Message	"01 03 04 00 00 1B 58 F139" (F139 = CRC check sum)
Coding = mBAR	1B 58Hex = 7000 decimal = 7000 miliBar (7.0 bar)

(!) *Register addresses shown are examples only*

9. 'Set' Function
Using the MODBUS Table a write data (Set) function message can be constructed:

Name	: SetLoadPressure
Modbus Function	: 10
Modbus Register Address	: 4018
Modbus Register Length	: 0002
Coding	: Number of data bytes = 4 Data Conversion Type = mBAR

SetLoadPressure (to slave at address '01' Hex)

Master Write Message	"01 10 4018 0002 04 0000 1B58 C9CC" (C9CC = CRC)
Slave Answer Message	"01 10 4018 0002 D40F" (D40F = CRC check sum)
Coding = mBAR	0000 1B58Hex = 7000 decimal = 7000 miliBar (7.0 bar)

ⓘ *Register addresses shown are examples only*

10. Data Coding Definitions:

Definitions for 'data units' and 'data conversion types' are listed for each 'item' in the "MODBUS Table" document.

All 'data' values are 'whole' numbers (integers); decimal places are not permitted in MODBUS data messages.

All 'data' values are unsigned (always positive) unless otherwise stated. Values specified as 'SIGNED' in the MODBUS Table can be negative in accordance to the standard data convention for 'signed' number values.

10.1 Data Types

Each standard definition will start with a “key” word that defines the data type:-

The following are selected examples; data types not included below are detailed in individual ‘Modbus Tables’

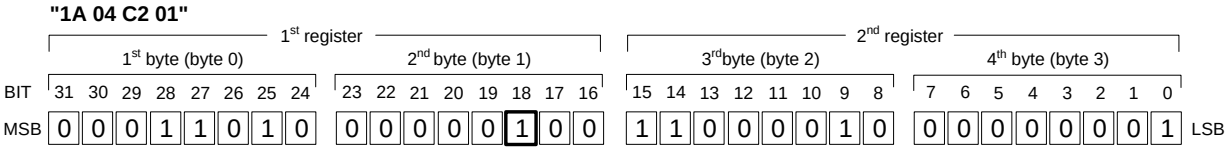
Type	Description
Coded	a decimal value that has a defined definition; see the ‘Coded’ lists in the ‘MODBUS Table’ for value definitions
Value	a ‘whole’ number or value in the specified units
Pressure	a ‘whole’ number defining a pressure in the specified units
Temperature	a ‘whole’ number defining a temperature in the specified units
Time	a ‘whole’ number defining a time period in the specified units
Electrical	a ‘whole’ number defining a volt, amp, power, or speed value in the specified units
Clock	Clock values are relevant to real time clock functions; for example pressure schedules. These ‘whole number’ unsigned values are ‘packaged’ multiple values and must be interpreted as follows.

Clock Data Type	Coding
HH_MM	1) Divide the value by 60 = Hours (0 to 23) 2) The remainder (modulus) = Minutes (0 to 59) Example for a value of ‘1050’ Hours = 1050 / 60 = 17.5 = 17 Hours Minutes = remainder = 30 = 30 Minutes Time = 17:30 (5:30pm)
D_HH_MM	1) Divide the value by 10000 = Day (1 = Monday, 7 = Sunday) 2) Divide the remainder (modulus) by 60 = Hours (0 to 23) 3) The remainder (modulus) = Minutes Example for a value of ‘31050’ Day = 31050 / 10000 = 3.105 = 3 = Wednesday Hours = = remainder / 60 = 17.5 = 17 Hours Minutes = remainder = 30 = 30 Minutes Day/Time = Wednesday 17:30 (5:30pm)
YYYY_DD_MO	1) Divide the value by 10000 = Year 2) Divide the remainder (modulus) by 100 = Day (1 to 31) 3) The remainder (modulus) = Month (1 to 12) Example for a value of ‘20051605’ Year = 20051605 / 10000 = 2005.1605 = Year 2005 Day = remainder / 100 = 16.05 = Day 16 Month = remainder = 5 = Month 5 Date = 16th May 2005

10.2 Data Units

The ‘MODBUS Table’ will define the ‘data units’ for each item.
Data unit definitions are specified in the ‘MODBUS Table’ as a separate list; for example:

BINARY:
A ‘Binary’ unit value is a collection of individual status ‘bits’ where each ‘bit’ represents a particular state or condition. The ‘MODBUS Table’ will reference each ‘bit’ with a number where ‘bit 0(zero)’ is the ‘least significant bit’ (LSB).



A reference to ‘Bit 18’ equates to bit ‘2’ of byte ‘1’ in the answer data message.
If the ‘bit’ is ‘1’ then the condition is ‘TRUE’.

Decimal Places:
Numbers with decimal places (eg 20.55) are not permissible in MODBUS data transfer – all numbers must be integer ‘whole’ numbers. To provide ‘decimal place’ accuracy some data values are multiplied by 10, 100 or 1000 and transmitted as a ‘whole’ number (integer). In this instance the ‘Data Units’ will specify that the number represents a value to one or more decimal places.

For example: PERCENT_DP2 = Percent to 2 decimal places
“2055” divided by 100 = 20.55%

If the ‘Data Units’ specifies “to 1 decimal place”, divide the number by 10 to convert to the correct engineering units. If the ‘Data Units’ specifies “to 2 decimal places”, divide the number by 100; if 3 decimal places divide by 1000.

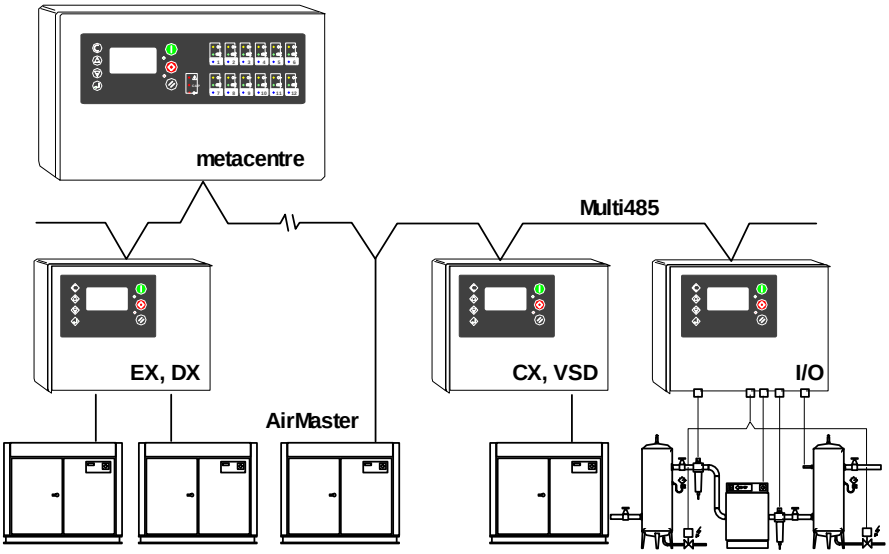
metacentre™ Compressed Air Systems

Metacentre systems comprise of modular networked units. This document describes the 'Broadcast' (Adv message request type) data items that are the same for all Metacentre systems regardless of arrangement, composition or individual unit model or variant.

Metacentre system 'Broadcast' data items comprise of all status and key performance value items typically required for system monitoring and performance reporting.

For 'Get' and 'Set' items a 'Modbus Table' for each individual system unit, model variant and software version, will be required.

1. General Metacentre System Components



Note: *example only; systems will differ from installation to installation*

Metacentre Unit: Compressed Air System Management Unit
Monitors and controls the air compressors, all general system and air compressor related information is marshalled and available from this unit. A system will only contain one Metacentre system management controller.

EX, DX and CX: Extension to the Metacentre Unit
For connection to remote compressor(s) or other specialised compressor integration. All common air compressor status information is sent to, and available from, the Metacentre system management unit. A system may contain multiple EX, DX or CX units.

VSD, VSD-V and VSD-R: Extension to the Metacentre Unit
For connection to variable speed compressor(s) or other specialised compressor integration. All common air compressor status information is sent to, and available from, the Metacentre system management unit. A system may contain multiple VSD, VSD-V or VSD-R units.

I/O: Monitoring/Control of Auxiliary Equipment and/or Sensors
For monitoring and/or control of auxiliary compressed air equipment (for example: dryer, auto drain, filtration differential, isolation valves, cooling water towers/pumps, ventilation) or sensors (for example: pressure, pressure differential, dewpoint, air flow, temperature). A system may contain multiple IO units.

2. Metacentre System Network Addresses

Network Address:- Decimal *(Hexadecimal)*

Metacentre Unit 101(65)

AirMaster Compressor Controller, EX, DX, CX, VSD, VSD-V or VSD-R

Compressor 1 = 1(01)	Compressor 2 = 2(02)	Compressor 3 = 3(03)	Compressor 4 = 4(04)
Compressor 5 = 5(05)	Compressor 6 = 6(06)	Compressor 7 = 7(07)	Compressor 8 = 8(08)
Compressor 9 = 9(09)	Compressor 10 = 10(0A)	Compressor 11 = 11(0B)	Compressor 12 = 12(0C)

I/O	I/O Box 1 = 105(69)	I/O Box 2 = 112(70)	I/O Box 3 = 106(6A)
I/O Box 4 = 107(6B)	I/O Box 5 = 108(6C)	I/O Box 6 = 109(6D)	I/O Box 7 = 110(6E)
I/O Box 8 = 111(6F)	I/O Box 9 = 113(71)	I/O Box 10 = 114(72)	> I/O Box 30 = 134(86)

Note: I/O Unit 2 address (70Hex) is intentionally out of sequence; this is not a print error.

3. Communication Link

To Interface with a Metacentre product that is equipped with a Multi485 enabled network port, or to interface with multiple Metacentre products operating on a single Multi485 system network, a MODBUS Gateway unit is required. The MODBUS Gateway forms the interface between the Multi485 protocol and MODBUS RTU master/slave communications link.

MODBUS Gateway connectivity is implemented using a two-wire RS485 industry standard communications link operating in point-to-point, master-slave, mode. In use the MODBUS Gateway is transparent and each Metacentre system unit is accessible using individual system device addresses.

(!) Polarity of the two Modbus RS485 wires (L1+ or ‘A’ and L2- or ‘B’) is important; reversal will inhibit communications and result in error.

4. RS485 Modbus Serial Data Format

The MODBUS Gateway port operates with an asynchronous serial data format:

1 start bit / 8 data bits / 1 stop / no parity - 9600baud.

1-8-1-N-9600

5. Data Coding

The following are selected examples; data types not included below are detailed in the 'Modbus Tables'

Value : The number is the value in the specified engineering units
The engineering units will differ dependant on unit set-up or item definition.

PSI : Pressure in 'psi'

BAR : Pressure in 'Bar'

FAH : Temperature in ^{o}F

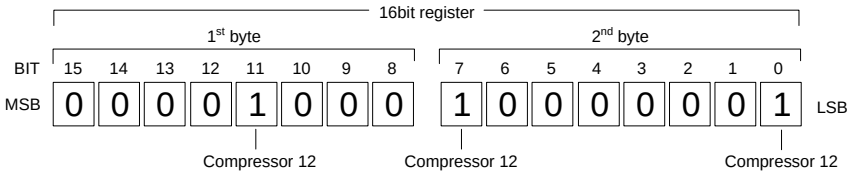
CEL : Temperature in ^{o}C

HRS : Hours

% : Percentage 0 to 100

BOOLEAN : The number will be 0 or greater than 0, (Boolean: 0 = False, 1 = True)

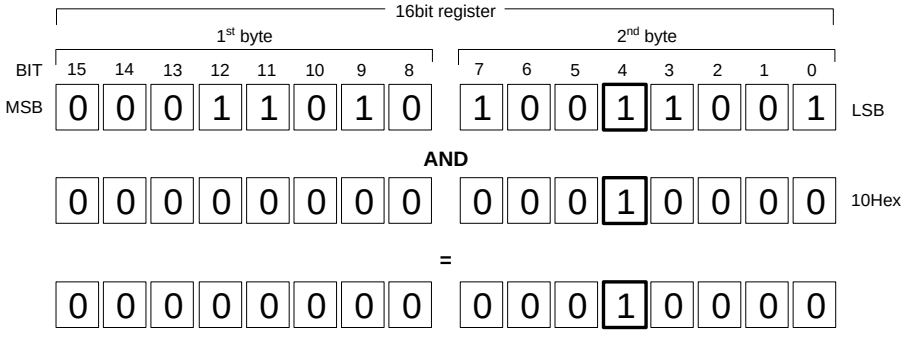
BINARY : The number represents a 16bit (two byte) binary value of 16bit flags.
The value must be interpreted in terms of each 'bit' as a set of sixteen Boolean (0 or 1) flags. These values are compressor related or I/O Box Input related. For compressor related items the least significant bit (Bit 0) represents compressor 1. For unit inputs the least significant bit (Bit 0) generally represents input 1.



The example illustrates the bit pattern for a value of '08 81 Hex'. This value is interpreted as a 'true' condition with respect to the item definition for compressors 1, 8 and 12. If the 'item' definition is 'Compressors Running' then compressors 1, 8 and 12 are in a 'running' condition.

BIT *'n'* : A Boolean (true/false) can be established from examining the specified 'bit' of the 16bit register. If the item specifies 'Bit 4' then the 4th bit should be examined:

Note: The LSB (least significant bit) of a register or byte is regarded as Bit 0(zero)



= 16 decimal (condition is TRUE)

The 4th bit of a register can be extracted by 'masking' the register content with "10 Hex"; if the resulting value is greater than 0(zero) then the condition is 'True', if the result is 0(zero) then the condition is 'False'.

6. Metacentre System 'Adv' Items

6.1 Metacentre Unit - System Management Controller

Item: **AdvAvailableStatus**
Description: Compressor Not Available
(Unavailable to respond to a load command without manual intervention)

Modbus Function : 03
Modbus Register Address : F020
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 0 = compressor 1, Bit 1 = compressor 2 …)
Function: 0 = Available (OK)
1 = Not Available (stopped or shutdown trip)

Master Command Message "65 03 F020 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 0000 (+CRC check sum)"

Item: **AdvRunningStatus**
Description: Compressor Motor Running

Modbus Function : 03
Modbus Register Address : F021
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 0 = compressor 1, Bit 1 = compressor 2 …)
0 = stopped/standby, 1 = Motor running

Master Command Message "65 03 F021 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 0441 (+CRC check sum)"

Item: **AdvLoadedStatus**
Description: Compressor Loaded (producing Air)

Modbus Function : 03
Modbus Register Address : F022
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 0 = compressor 1, Bit 1 = compressor 2 …)
0 = Off Load, 1 = On Load (producing air)

Master Command Message "65 03 F022 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 0041 (+CRC check sum)"

Item: **AdvAlarmStatus**
Description: Compressor Alarm (Warning)
Note: for shutdown trip use AdvAvailableStatus

Modbus Function : 03
Modbus Register Address : F023
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 0 = compressor 1, Bit 1 = compressor 2 …)
0 = OK, 1 = Alarm (Warning)

Master Command Message "65 03 F023 0001(+CRC check sum)"
Slave Answer Message "65 03 02 0000 (+CRC check sum)"

Item: **AdvCompressorMaintenance**
Description: Compressor has been removed from Metacentre management control.
Compressor stopped for service/maintenance or inhibited from system use.

Modbus Function : 03
Modbus Register Address : F087
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 0 = compressor 1, Bit 1 = compressor 2 …)
0 = OK, 1 = Service/Maintenance

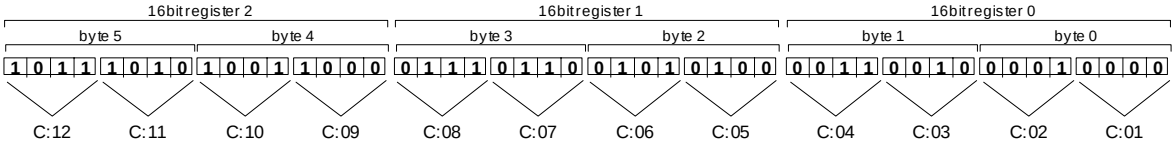
Master Command Message "65 03 F087 0001(+CRC check sum)"
Slave Answer Message "65 03 02 0021 (+CRC check sum)"

Example answer shows compressors 1 and 6 in Service/Maintenance ('0021' Hex = bits 0 and 5 = '1')

Item: **AdvCompressorSequence**
Description: The sequence assignment for each compressor in the system

Modbus Function : 03
Modbus Register Address : F063
Modbus Register Length : 0003
Coding : Number of data bytes = 6

Master Command Message "65 03 F063 0003 (+CRC check sum)"
Slave Answer Message "65 03 06 BA98 7654 3210 (+CRC check sum)"



C:01 = Compressor 1, C:02 = Compressor 2 . . .

Registers 0, byte 0, bits 0 to 3 = sequence assignment for Compressor 1
Registers 0, byte 0, bits 4 to 7 = sequence assignment for Compressor 2
Registers 0, byte 1, bits 0 to 3 = sequence assignment for Compressor 3
Registers 0, byte 1, bits 4 to 7 = sequence assignment for Compressor 4
Registers 1, byte 2, bits 0 to 3 = sequence assignment for Compressor 5
Registers 1, byte 2, bits 4 to 7 = sequence assignment for Compressor 6
Registers 1, byte 3, bits 0 to 3 = sequence assignment for Compressor 7
Registers 1, byte 3, bits 4 to 7 = sequence assignment for Compressor 8
Registers 2, byte 4, bits 0 to 3 = sequence assignment for Compressor 9
Registers 2, byte 4, bits 4 to 7 = sequence assignment for Compressor 10
Registers 2, byte 5, bits 0 to 3 = sequence assignment for Compressor 11
Registers 2, byte 5, bits 4 to 7 = sequence assignment for Compressor 12

Interpret the specified 4'bits' as a number from 0 to 15 where:
0 = 'A' 1 = 'B' 2 = 'C' 3 = 'D' 4 = 'E' 5 = 'F'
6 = 'G' 7 = 'H' 8 = 'I' 9 = 'J' 10 = 'K' 11 = 'L'
12 to 15 not used.

(!) In 'Energy' mode (where applicable dependant on model) the sequence assignment may continuously change on a non-routine basis.

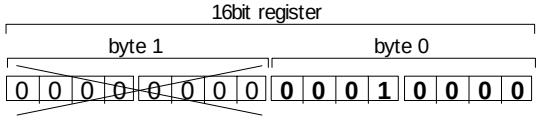
Item: **AdvSystemStatus**
Description: Compressor Alarm (Warning)

Modbus Function : 03
Modbus Register Address : F024
Modbus Register Length : 0001
Coding : Number of data bytes = 1

This register must be logically manipulated and interpreted in the follow manner:

a) Mask the higher order byte (byte 1) of the register, only the lower order byte (byte 0) is applicable:

Note: byte 1 may contain a value



b) Bits 0 to 3 (first four 'bits' of the first byte) must be interpreted as a number from 0 (0 Hex) to 15 (F Hex):
0 = OK
1 = pressure sensor fault
2 = flow sensor fault (if applicable)
3 = 2nd pressure sensor fault (if applicable)
4 = internal 24V fault
5 = external 24V fault
6 = real time clock fault
7 = XPM-LED fault (if applicable)
8 = not used
9 = not used
10 = not used
11 = not used
12 = 'iX' (i-PCB expansion Box for compressors 5 to 8) communications fault
13 = 'iX' (i-PCB expansion Box for compressors 5 to 8) short circuit fault
14 = 'iX' (i-PCB expansion Box for compressors 9 to 12) communications fault
15 = 'iX' (i-PCB expansion Box for compressors 9 to 12) short circuit fault

c) Bits 4 to 7 (last four 'bits' of the first byte) must be interpreted as individual status:
Bit 4 = 0: metacentre off (stopped), 1:metacentre on, operational (standby or running)
Bit 5 = 0: no shutdown, 1: shutdown fault
Bit 6 = 0: no alarm (warning), 1: alarm (warning)
Bit 7 = 0: no insufficient capacity alarm, 1: insufficient capacity alarm

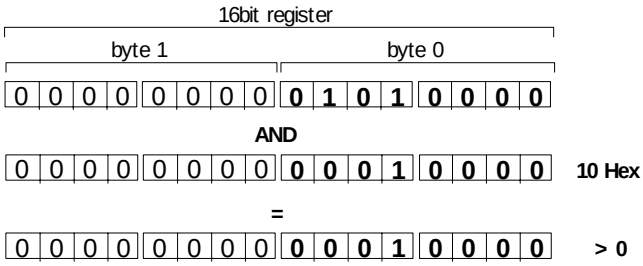
Master Command Message "65 03 F024 0001(+CRC check sum)"
Slave Answer Message "65 03 02 0010 (+CRC check sum)"

For basic Metacentre status information the 'AdvSystemStatus' register can be 'masked':
Metacentre ON If 'register' AND '10 Hex' > 0 Then (metacentre on, operational)
Metacentre Shutdown Trip If 'register' AND '20 Hex' > 0 Then (metacentre shutdown trip)
Metacentre Alarm If 'register' AND '40 Hex' > 0 Then (metacentre alarm, warning)

For example:

Register = 50 Hex
Mask = 10 Hex
Result = 10 Hex

Result is greater than 0(zero)
= Metacentre is on



Note: in the above example the Metacentre has an alarm (warning) condition – bit 6 of byte 0 = '1'

Item: **AdvCompressorPercentage**
Description: Percentage of modulation speed/output (only applicable while compressor running on load)

Modbus Function : 03

Modbus Register Address : F000 (compressor 1) – byte 1
Modbus Register Address : F000 (compressor 2) – byte 0

Modbus Register Address : F001 (compressor 3) – byte 1
Modbus Register Address : F001 (compressor 4) – byte 0

Modbus Register Address : F002 (compressor 5) – byte 1
Modbus Register Address : F002 (compressor 6) – byte 0

Modbus Register Address : F003 (compressor 7) – byte 1
Modbus Register Address : F003 (compressor 8) – byte 0

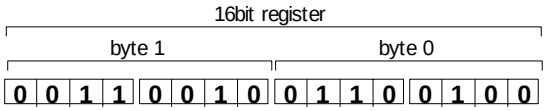
Modbus Register Address : F004 (compressor 9) – byte 1
Modbus Register Address : F004 (compressor 10) – byte 0

Modbus Register Address : F005 (compressor 11) – byte 1
Modbus Register Address : F005 (compressor 12) – byte 0

Modbus Register Length : 0001
Coding : Number of data bytes = 2
% (0 to 100)

Master Command Message "65 03 F002 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 3264 (+CRC check sum)"

The answer register will contain percentage data for 2 compressors; the percentage value for the first compressor can be established from the high byte (byte 1) of data (bits 8 to 15) and the value for the second compressor in the low byte (byte 0) of data (bits 0 to 7).



For example; Modbus Register Address F002 will return the percentage data for compressors 5 and 6. If the answer data is "3264 Hex" then "32" (byte 1) = 50 decimal = compressor 5 is 50% and "64" = 100 decimal = compressor 6 is 100%.

For variable speed (or other variable output type) compressors the % value represents the percentage of the available speed/output modulation span where 0% equates to minimum speed/output. The % value is not representative of %Load.
For example: if modulation span = 50% of rated maximum output of a compressor then:
0 = 50% rated speed/output (when the compressor is loaded – test for loaded state)
50 = 75% of rated speed/output
100 = 100% of rated speed/output

For 3-Step (CX Box) compressors, 0 = offload, 50 = 50% and 100 = 100% load.

For 5-Step (CX Box) compressors, 0 = offload, 25 = 25%, 50 = 50%, 75 = 75% and 100 = 100% load.

For compressors that are load/unload regulated the value will change from 0 = offload to 100 = onload; no intermediate values will occur.

Compressor Run Hours:
The metacentre management unit monitors the running status of each compressor and records the 'running hours' for each independently of the compressor controller. The metacentre will routinely 'broadcast' this data; each 'broadcast' will contain the 'running hours' for one particular compressor. The 'running hours' for each compressor in the system is broadcast sequentially, one after the other, in a continuous repeating cycle; the first 'broadcast' will contain the 'running hours' for compressor 1, the next 'broadcast' will contain the 'running hours' for compressor 2 etc. The compressor for which the 'running hours' is applicable is identified in the 5th byte (byte 4) of a 3 register (6 byte) 'Adv' request answer.

To ensure the 'running hours' value and 'compressor identification' correspond to the same compressor both items must be extracted from the same 3 register (6 byte) 'Adv' request answer.

Item: **AdvRunningHours**
Description: Running hours value for the identified compressor

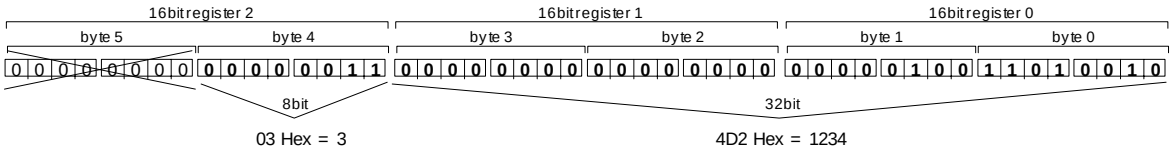
Modbus Function : 03
Modbus Register Address : F080
Modbus Register Length : 0003
Coding : Number of data bytes = 6

Master Command Message "65 03 F080 0003 (+CRC check sum)"
Slave Answer Message "65 03 06 0003 0000 04D2 (+CRC check sum)"

Interpretation:

Hours Run = Evaluate bytes 0 to 3 (registers 0 and 1) as a single '32bit' unsigned integer number; this represents the hours run value.

Compressor Identification = Evaluate byte 4 (low order byte of register 2) as an '8bit' unsigned integer number between 0 and 15; this represents the compressor identification number:
0 = compressor 1
1 = compressor 2
2 = compressor 3 . . . up to 11 = compressor 12



Compressor 4 = 1234 Hours

Registers 0 and 1 (bytes 0 to 3) = 32bit integer value of 'running hours'; unit = hours
Register 2 (byte 4) = 8bit compressor identification number
Register 2 (byte 5) may contain a value; mask register 2 with '0F Hex' or otherwise ignore

To obtain the 'Running Hours' for each compressor in the system, continuously request 'AdvRunningHours' on a routine basis. The system management unit will issue a new 'Broadcast' approximately every 2 seconds, each broadcast will contain 'running hours' information for a different compressor. To capture and maintain a record of the 'running hours' for each compressor in the system issue an 'AdvRunningHours' request, for example, every 15 seconds. After a short period of time the running hours for each compressor in the system will have been received.

Item: **AdvSystemPressure**
Description: System pressure (management system control pressure)

Modbus Function : 03
Modbus Register Address : F026
Modbus Register Length : 0001
Coding : Number of data bytes = 2
PSI *(14.5psi = 1.0bar)*

Master Command Message "65 03 F026 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 0057 (+CRC check sum)"

For example: if answer value = '0057' Hex (57 Hex = 87psi = 6.0bar)

Item: **AdvTargetPressure**
Description: The target pressure that the Metacentre is attempting to maintain

Modbus Function : 03
Modbus Register Address : F025
Modbus Register Length : 0001
Coding : Number of data bytes = 2
PSI *(14.5psi = 1.0bar)*

Master Command Message "65 03 F025 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 0074 (+CRC check sum)"

For example: if answer value = '0074' Hex (74 Hex = 116psi = 8.0bar)

Item: **AdvSystemStatusFlags**
Description: Management system status and function active indicators

Modbus Function : 03
Modbus Register Address : F084
Modbus Register Length : 0001
Coding : Number of data bytes = 2

Interpret each individual 'bit' of the 16bit (1 register) answer:

Bit **Description** *(true when bit = '1')*

0 Start Function Active**
1 Zone Function Active**
2 to 4 *see Operating Mode*
5 Pressure Balancing Function Active**
6 Metacentre Active (on 'and' active – standby, prefill or normal operation)
7 Pressure Schedule Active
8 Low Pressure Alarm (Warning)
9 High Pressure Alarm (Warning)
10 Insufficient Capacity Alarm (Warning)
11 Restricted Capacity Alarm (Warning)*
12 to 14 *see Active Table*
15 Metacentre in Prefill Mode*

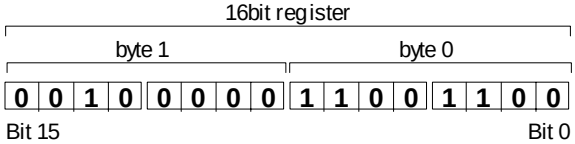
Operating Mode:
Interpret 'bits' 2, 3 and 4 as a '3bit' positive integer number between 0 and 7
1 = FIFO Mode
2 = Energy Control Mode
3 = Time Rotation Mode
4 = Equal Hours Mode

Active Table:
Interpret 'bits' 12, 13 and 14 as a '3bit' positive integer number between 0 and 7
1 = Table 1
2 = Table 2 . . .

* *Applicable to Metacentre models SX and XC only*
** *Applicable to Metacentre model XC only*

Reference the applicable Metacentre manual for detailed descriptions of each function.

Example:

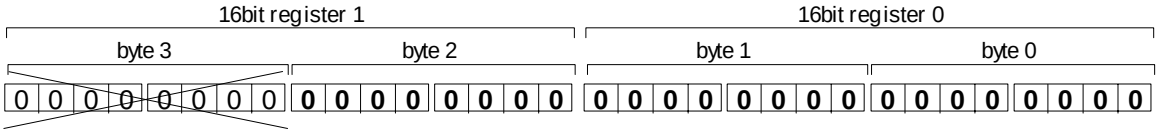


Master Command Message "65 03 F084 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 20CC (+CRC check sum)"

Bits 2, 3 and 4 evaluate to '3' = the Metacentre is operating in 'Time Rotation' Mode
Bits 12, 13 and 14 evaluate to '2' = Metacentre Table 2 is active
Bit 6 = '1' = Metacentre is 'on' and is active
Bit 7 = '1' = the Metacentre 'Pressure Schedule' is active

Item: **AdvRelayStatusFlags**
Description: Additional management system status and function active indicators

Modbus Function : 03
Modbus Register Address : F085
Modbus Register Length : 0002
Coding : Number of data bytes = 3



Note: byte 3 may contain a value

Mask 'byte 3' of register 1 (for example: mask the 2 register '32bit value' with '0F FF' Hex)

Interpret each individual 'bit' of the remaining 24bit (bytes 0, 1 and 2) value:

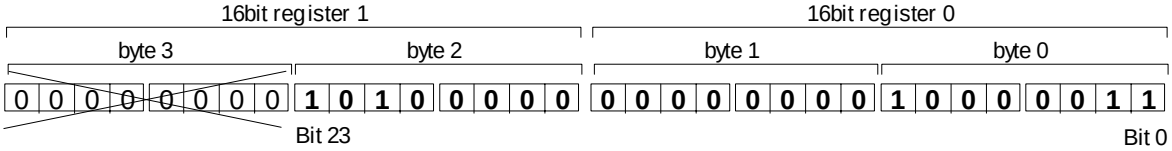
Bit **Description** *(true when bit = '1')*

0 to 15 Virtual Relay 1 to 16 Output States: bit 0 = Virtual Relay 1, bit 1 = Virtual Relay 2 . . . .
16 Start Time Function Fault Condition******
17 Delta Pressure Alarm (Warning) Condition******
18 Alarm Override Condition (a compressor alarm state is being suppressed, inhibited, by a service/maintenance function or menu maintenance function)
19 General I/O Unit Alarm 'A' (alarm) Condition Detected (from any input of any I/O unit in the system)
20 General I/O Unit Alarm 'T' (high level alarm) Condition Detected (from any input of any I/O unit in the system)
21 General I/O Unit Alarm 'S' (signal) Condition Detected (from any input of any I/O unit in the system)
22 General I/O Unit Communications disruption or failure Condition (applicable to any I/O unit in the system)
23 Metacentre Producing Air (maintaining target pressure)

****** *Applicable to Metacentre model XC only*

Reference the applicable Metacentre manual for detailed descriptions of each function.

Example:



Master Command Message "65 03 F085 0002 (+CRC check sum)"
Slave Answer Message "65 03 02 00A0 0083 (+CRC check sum)"

Bits 0, 1 and 7 = '1' = Virtual Relays 1, 2 and 8 are 'on'
Bit 21 = '1' = An I/O Unit has an 'S' signal condition
Bit 23 = '1' = the Metacentre is 'on' and actively maintain target pressure

metacentre™ XC

The following are applicable to system management unit model 'Metacentre XC' only.

Item: **AdvFlow**
Description: Air flow sensor value (if option fitted and function activated)

Modbus Function : 03
Modbus Register Address : F083
Modbus Register Length : 0001
Coding : Number of data bytes = 2
10 x m^3/min *(1 m^3/min = 35.315cfm = 1000 L/min)*

Master Command Message "65 03 F083 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 0064 (+CRC check sum)"

For example: if answer value = '0064' Hex (64 Hex = 100 = 10 m^3/min)

Item: **AdvDeltaPressure**
Description: Differential between primary control pressure and 2nd pressure (if option fitted and function activated)

Modbus Function : 03
Modbus Register Address : F060
Modbus Register Length : 0001
Coding : Number of data bytes = 2
PSI *(14.5psi = 1.0bar)*

Master Command Message "65 03 F025 0001 (+CRC check sum)"
Slave Answer Message "65 03 02 0007 (+CRC check sum)"

For example: if answer value = '0007' Hex (07 Hex = 7psi = 0.5bar)

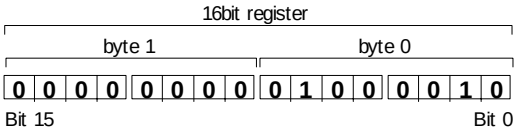
6.2 I/O Unit

Item: **AdvDigInReading**
Description: Status of digital inputs 1 to 8

Modbus Function : 03
Modbus Register Address : F408
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 0 = digital input 1, Bit 1 = digital input 2 …)
Function: 0 = normal state, 1 = not normal state

Note: 'normal state' is dependant on digital input configuration (normally closed or normally open).

Example shows Inputs 2 and 7 in 'not normal' state



The actual digital input descriptions (the switching devices to which the inputs are connected), will be dependent on individual I/O unit set-up. This information should be established from the product supplier who installed and commissioned the I/O unit.

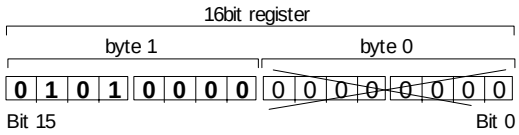
Master Command Message "69 03 F408 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 0042 (+CRC check sum)"

Item: **AdvDigInStateAlm**
Description: Alarm 'A' status for digital inputs 1 to 8

Modbus Function : 03
Modbus Register Address : F422
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 8 = digital input 1, Bit 9 = digital input 2 …)
Function: 0 = OK, 1 = Alarm State 'A'

Disregard (or mask) the lower order byte (byte 0) of the register, only the high order byte (byte 1) is applicable:

Note: byte 0 may contain a value



Example shows Inputs 5 and 7 in 'A' state

The 'alarm' state of a digital input will only apply if the digital input has been configured for 'A' alarm status monitoring; the 'bit' will always be '0' if digital input is not configured for 'A' alarm monitoring function.

Master Command Message "69 03 F422 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 5000 (+CRC check sum)"

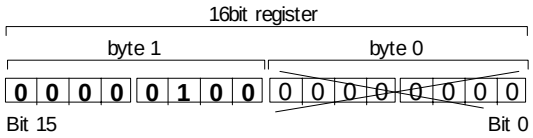
Reference the Metacentre I/O Unit manual for detailed descriptions of each function.

Item: **AdvDigInStateErr**
Description: Alarm 'T' status for digital inputs 1 to 8

Modbus Function : 03
Modbus Register Address : F423
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 8 = digital input 1, Bit 9 = digital input 2 …)
Function: 0 = OK, 1 = High Level Alarm State 'T'

Disregard (or mask) the lower order byte (byte 0) of the register, only the high order byte (byte 1) is applicable:

Note: byte 0 may contain a value



Example shows Input 3 in 'T' state

The 'high level alarm' state of a digital input will only apply if the digital input has been configured for 'T' high level alarm status monitoring; the 'bit' will always be '0' if digital input is not configured for 'T' high level alarm monitoring function.

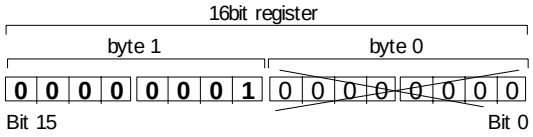
Master Command Message "69 03 F423 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 0400 (+CRC check sum)"

Item: **AdvDigInStateSrel**
Description: Signal 'S' function status for digital inputs 1 to 8

Modbus Function : 03
Modbus Register Address : F424
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 8 = digital input 1, Bit 9 = digital input 2 …)
Function: 0 = OK, 1 = 'S' State

Disregard (or mask) the lower order byte (byte 0) of the register, only the high order byte (byte 1) is applicable:

Note: byte 0 may contain a value



Example shows Input 1 in 'S' state

The 'S' signal state of a digital input will only apply if the digital input has been configured for 'S' signal status monitoring; the 'bit' will always be '0' if digital input is not configured for 'S' monitoring function.

Master Command Message "69 03 F424 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 0100 (+CRC check sum)"

Reference the Metacentre I/O Unit manual for detailed descriptions of each function.

Item: **AdvAnaInValue**
Description: Value detected on the specified analogue input in the set engineering units

Modbus Function : 03
Modbus Register Address : F400 (analog input 1)
Modbus Register Address : F401 (analog input 2)
Modbus Register Address : F402 (analog input 3)
Modbus Register Address : F403 (analog input 4)
Modbus Register Length : 0001
Coding : Number of data bytes = 2

The actual analogue input description (the device to which the input is connected), and the analogue value engineering units and description (the interpretation of the value), will be dependent on individual I/O Box set-up. This information should be established from the product supplier who installed and commissioned the I/O Unit.

Master Command Message "69 03 F400 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 00CD (+CRC check sum)"

Example shows request for analogue input 1 value.

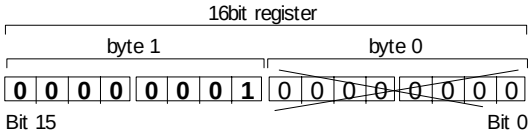
Data Coding: Numbers with decimal places (for example: 20.5) are not permissible in MODBUS data transfer – all numbers must be integer whole numbers. Because of this limitation it is common practise to transmit numbers as engineering units multiplied by a factor of 10, 100 or 1000. For example; to transmit temperature to one decimal place, a detected temperature of 20.5^{o}C will be transmitted as 205 ('CD' Hex). A receiving device can then divide by 10 to obtain the correct value and display to one decimal place (205 = 20.5^{o}C). The transmitted value will be a 'signed' integer to allow negative values.

Item: **AdvAnaInStateAlm**
Description: Alarm 'A' status for analogue inputs 1 to 4

Modbus Function : 03
Modbus Register Address : F420
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 8 = input 1, Bit 9 = input 2 …)
Function: 0 = OK, 1 = Alarm State 'A'

Disregard (or mask) the lower order byte (byte 0) of the register, only the high order byte (byte 1) is applicable:

Note: byte 0 may contain a value



Example shows Input 1 in 'A' state

The 'alarm' state of an analogue input will only apply if the analogue input has been configured for 'A' alarm status monitoring; the 'bit' will always be '0' if analogue input is not configured for 'A' alarm monitoring function.

Master Command Message "69 03 F420 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 0100 (+CRC check sum)"

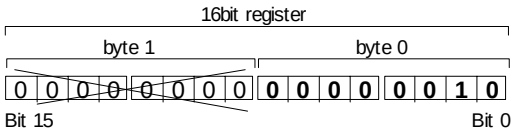
Reference the Metacentre I/O Unit manual for detailed descriptions of each function.

Item: **AdvAnaInStateErr**
Description: High Level Alarm 'T' status for analogue inputs 1 to 4

Modbus Function : 03
Modbus Register Address : F420
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 8 = input 1, Bit 9 = input 2 …)
Function: 0 = OK, 1 = High Level Alarm State 'T'

Disregard (or mask) the high order byte (byte 1) of the register, only the low order byte (byte 0) is applicable:

Note: byte 1 may contain a value



Example shows Input 2 in 'T' state

The 'high level alarm' state of an analogue input will only apply if the analogue input has been configured for 'T' high level alarm status monitoring; the 'bit' will always be '0' if analogue input is not configured for 'T' high level alarm monitoring function.

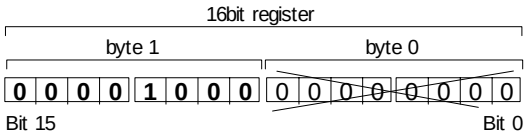
Master Command Message "69 03 F420 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 0002 (+CRC check sum)"

Item: **AdvAnaInStateSrel**
Description: Signal 'S' status for analogue inputs 1 to 4

Modbus Function : 03
Modbus Register Address : F421
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 8 = input 1, Bit 9 = input 2 …)
Function: 0 = OK, 1 = 'S' State

Disregard (or mask) the low order byte (byte 0) of the register, only the high order byte (byte 1) is applicable:

Note: byte 0 may contain a value



Example shows Input 4 in 'S' state

The 'S' signal state of an analogue input will only apply if the analogue input has been configured for 'S' signal status monitoring; the 'bit' will always be '0' if analogue input is not configured for 'S' signal monitoring function.

Master Command Message "69 03 F421 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 0800 (+CRC check sum)"

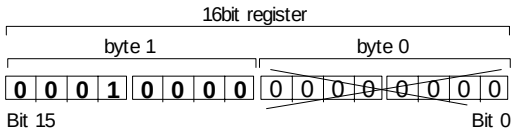
Reference the Metacentre I/O Unit manual for detailed descriptions of each function.

Item: **AdvDigOut**
Description: status for output relays 1 to 6

Modbus Function : 03
Modbus Register Address : F425
Modbus Register Length : 0001
Coding : Number of data bytes = 2
BINARY (Bit 8 = output relay 1, Bit 9 = output relay 2 …)
Function: 0 = off, 1 = on (relay contacts closed)

Disregard (or mask) the lower order byte (byte 0) of the register, only the high order byte (byte 1) is applicable:

Note: byte 0 may contain a value



Example shows Relay Output 5 = ON (bit 12)

Master Command Message "69 03 F425 0001 (+CRC check sum)"
Slave Answer Message "69 03 02 1000 (+CRC check sum)"

Reference the Metacentre I/O Unit manual for detailed descriptions of each function.

6.3 Compressor Pressure/Temperature

The compressor identification network address number must be the Metacentre identification number (compressor 01 to 12 - 01 to 0C Hex).

These values are only available from a unit that is equipped with the applicable input(s) and the input(s) are connected to a sensor and activated. For compressor(s) integrated to a management system unit using a 'hard wire' (i-PCB) interface these values are not available.

Item: **AdvDeliveryPressure**
Description: Compressor delivery pressure

Modbus Function : 03
Modbus Register Address : F001
Modbus Register Length : 0001
Coding : Number of data bytes = 2
PSI *(14.5psi = 1.0bar)*

Master Command Message "01 03 F001 0001 (+CRC check sum)"
Slave Answer Message "01 03 02 0057 (+CRC check sum)"

Example shows request and reply for data from compressor 1 (01 Hex).

For example: if compressor 1 answer value = '0057' Hex (57 Hex = 87psi = 6.0bar)

Item: **AdvTemperature**
Description: Compressor temperature (typically oil temperature but may differ dependant on compressor type and/or model – refer to the compressor manufactures manual)

Modbus Function : 03
Modbus Register Address : F002
Modbus Register Length : 0001
Coding : Number of data bytes = 2
FAH/CEL/CELx10 (dependant on controller/compressor)

Master Command Message "01 03 F002 0001 (+CRC check sum)"
Slave Answer Message "01 03 02 00CD (+CRC check sum)"

Example shows request and reply for data from compressor 1 (01 Hex).

Data Coding: Numbers with decimal places (for example: 20.5) are not permissible in MODBUS data transfer – all numbers must be integer whole numbers. Because of this limitation it is common practise to transmit numbers as engineering units multiplied by a factor of 10, 100 or 1000. For example; to transmit temperature to one decimal place, a detected temperature of 20.5^{o}C will be transmitted as 205 ('CD' Hex). A receiving device can then divide by 10 to obtain the correct value and display to one decimal place (205 = 20.5^{o}C). The transmitted value will be a 'signed' integer to allow negative values.

All compressor status information is available directly from the Metacentre system management unit.

Y05CM27.00 – MODBUS Table, Order Form

For AirMaster Embedded Controller or Metacentre Systems Unit Product.

A ‘MODBUS Table’ describes the “items” used to access information in the memory registers of different types of AirMaster/Metacentre controller, or similar controllers using different application software variants or versions. The ‘MODBUS Table’ will contain the valid message items (“Name”) together with the Function Code (Function), Register Start Address (“Register Address”), Register Size (“Register Length”) and a definition for coding and decoding the item data (“Coding”).

An installation may consist of a number of different controller types, or similar controller types with different application software variants or versions. The appropriate ‘MODBUS Table’ will be required for each unique type and variant.

To order a ‘MODBUS Table’ the following information is required for each unique controller type, or application variant or version. Please complete this form and submit a copy with your Order; alternatively please include the information requested on this form with your order. Any order submitted without this information will be rejected. The ‘MODBUS Table’ will be sent by e-mail; please include your e-mail address with the order.

AirMaster ☐
metacentre ☐

Type	**:**
Part Number	**:**
Serial Number	**:**
Software Version Number	**:**

Your e-mail Address	**:**

Type: Located on the manufacturers’ label attached to the rear or side of the controller (for example: S1, R1, T1)

Part Number: Located on the manufacturers’ label attached to the rear or side of the controller (for example: Y04CM11.00)

Serial Number: Generally a four digit number printed directly after the part number (for example: Y04CCM11.00-0010; where 0010 is the serial number)

Software Version Number: Located on the manufacturers’ label attached to the rear or side of the controller, also displayed by the controller during power-up initialisation (for example: S1CMCSTD.E02; note “E02” version number is important)