

WinMark Pro Application Note

Controlling Flyer/Fenix Flyer Marking Heads Using Modbus® Protocols

This revised Application Note provides information about communicating to FH Flyer marking heads and Fenix Flyer Laser Markers over an Ethernet network using the Modbus® protocol.

The Modbus protocol is handled in the Flyer head by an external communications server named SynComm. SynComm allows users to access various marking head functions via Flyer's Ethernet port using one of three different protocols: (1) Modbus/IP protocol for interaction with PLCs or other MODBUS network devices; (2) Modbus-Asynchronous protocol, a SYNRAD-modified Modbus protocol for peer-to-peer communications; and (3) SmartFH protocol, provided as legacy support for customers who have upgraded existing FH Smart systems to FH Flyer and wish to continue using custom programs written specifically for FH Smart marking heads.

Important Note: The SmartFH protocol is intended for *legacy support only* (i.e. systems where Flyer is replacing an existing FH Smart marking head. For maximum flexibility, newly integrated systems incorporating FH Flyer marking heads should use Modbus/IP or Modbus-Asynchronous protocols.

SynComm Modbus/IP Protocol

There are two “flavors” of SYNRAD’s implementation of Modbus I/P: (1) as described in Section 1, the first is a basic set of register-based commands using Modbus function codes 3, 4, 6, and 16 that many Programmable Logic Controllers (PLCs) support and (2) as described in Section 2, an extended set of commands using a Modbus user-defined function code in the range of 65–72 or 100–110. The extended (user-defined) command set includes advanced Filestore and head management options, but is less widely supported by the PLC community.

This document describes all the information necessary to control an FH Flyer marking head or Fenix Flyer Laser Marker using either “flavor” of Modbus/IP. For complete information on the full Modbus/IP specification, see <http://www.modbus-ida.org/>. In addition, we have posted sample Visual Basic and Visual C++ code at http://www.winmark.com/products/winmark_activexsamples.html. These samples provide the functions necessary to create commands for communicating with a Flyer head and are designed to be easily incorporated into customer applications.



Table of Contents

Section 1 – Using Basic (Register-Based) Modbus/IP Command Function Codes	pg 3
Register-Based Modbus/IP Message Structure	pg 4
Register-Based Modbus/IP Flyer Marking Head Functions	pg 5
I/O Commands	pg 6
Mark Status Commands	pg 9
Head Temperature Commands	pg 13
Marking Head Status Commands	pg 15
Date/Time Commands	pg 19
Filestore Usage Commands	pg 21
Error Code Commands	pg 23
File Commands	pg 24
File Properties Commands	pg 28
System Parameters Commands	pg 30
Section 2 – Using Extended (User-Defined) Modbus Command Function Codes	pg 32
User-Defined Modbus/IP Packet Structure and SynComm Packets	pg 33
User-Defined SynComm Command/Response Structure	pg 34
User-Defined Modbus/IP Flyer Marking Head Functions	pg 35
Marking Commands	pg 37
Filestore Management Commands	pg 49
Date/Time Management Commands	pg 58
Flyer Head Management Commands	pg 66
I/O Management Commands	pg 76
SynComm Modbus-Asynchronous Flyer Marking Head Functions	pg 84
Modbus-Asynchronous Marking Events	pg 85
Modbus-Asynchronous I/O Events	pg 87
SynComm Smart FH Protocol	pg 90
Appendix A: List of Flyer Marking Head System Parameter Names	pg 91
Appendix B: Flyer EOM Response Data Map	pg 93
Appendix C: MODBUS and SYNRAD Error Codes	pg 94
Appendix D: Flyer’s MODBUS Register Memory Map	pg 96

Section 1 – Using Basic (Register-Based) Modbus/IP Command Function Codes

Guidelines for using the register-based Modbus/IP protocol:

- The Flyer head **MUST** be set to operate in stand-alone mode (***Standalone Marking*** property on “Device” tab set to “Yes”).
- For register-based Modbus operation, the ***Modbus User Function*** property (in WinMark Pro on the “Device” tab) is not used—it can be left at its default value of 67 (0x43 hexadecimal). However you can use a mix of register-based and user-defined based commands to control Flyer and if this is the case, the ***Modbus User Function*** must be set to a decimal value in the range of 65–72 or 100–110. The WinMark Pro default value is 67 (0x43 hexadecimal).
- On the “Device” tab, set the ***External Communications Server*** property to “Modbus”.
- Flyer’s communications server listens on the default Modbus port (Port 502).
- WinMark Pro, WinMark Launcher, and DigiScope applications must be closed after the head is configured for Modbus/Modbus Async operation.
- Filenames must be preceded by the “/” character; for example, Test.mkh becomes /Test.mkh.
- Modbus limits message elements (16-bit words) to a maximum of 120 (or 240 bytes)
- Modbus is a big endian protocol. In our Modbus VB example code posted at: http://www.winmark.com/products/winmark_activexsamples.html, Mobus.h and Modbus.c contain endian conversion routines to aid in parsing data.
- Modbus is a client/server (or command/response) protocol. The Modbus equipped PLC or computer is the client (that commands), the FH Flyer acts as a server (who responds).
- All character strings must be null-terminated (designated as “\00” in Section 1).

Important Note: Register addresses for each function described in this section are zero-based, meaning they start at zero (0). Some PLCs do not support zero-based addressing so if your PLC falls in this category, you must add one (1) to the register address specified by your Message command. For example, to read Flyer’s Input Register (0x0000), specify a starting address of “0” in the Message instruction if your PLC supports zero-based addressing. For PLCs that do not support zero-based addresses, specify a starting address of “1”.



Register-Based Modbus/IP Message Structure

This section describes the format for using a Modbus function code to read or write a Flyer Modbus register using Modbus TCP/IP.

Function Codes

The following Modbus command function codes are available for Flyer/Fenix Flyer control:

Function Code	Decimal (Hex)	Command
3	(0x3)	Read Holding Registers
4	(0x4)	Read Input Registers
6	(0x6)	Write Single Holding Register
16	(0x10)	Write Multiple Holding Registers

Data Types

Typically, the data portion of the message is specified in the PLC's Message (MSG) instruction as a data table address (reading from or writing to). Except for Flyer commands that are formatted as strings, all other data should be read or written as decimal values. The word, or element, sizes listed in [Appendix D](#), Flyer's MODBUS Memory Map, are as follows:

- Word – an unsigned 16-bit integer
- DWord – an unsigned 32-bit integer (a Long)
- SWord – a signed 16-bit integer
- String – ASCII formatted characters

Note: The Modbus standard limits message elements to a maximum of 120 elements (16-bit words) or 240 bytes.

Examples

The examples in this Section were created using an Allen Bradley™ MicroLogix™ 1400 Series B Controller and Rockwell Automation RSLogix Micro Developer software. The formatting of command instructions for your particular PLC may vary.

Modbus/IP Error Checking

When the Modbus command is successful, Flyer returns the function code of the requested operation. If a Modbus error occurs with a register-based Modbus command, the Flyer head returns one byte (equal to the transmitted function code + 0x0080). On receipt of a Modbus error, the user should read register 0x0066 (102 decimal) for a more descriptive Synrad error code, if available, and then perform any tasks necessary to ensure data integrity before sending further commands. See [Appendix C](#) for lists of Modbus and SYNRAD error codes.

Common errors are “broken communications” errors, caused by Flyer not being set to Stand-alone mode (set **Standalone Marking** property to “Yes”) and/or not configuring the SynComm server for Modbus operation (set **External Communications Server** property to “Modbus”).

Register-Based Modbus/IP Flyer Marking Head Functions:

The following functions are currently available via Modbus/IP for Flyer/Fenix Flyer marking head control:

I/O Commands (page 6):

- Read Input Register Read (Get) the status of Flyer's 8-bit input register
- Read/Write Output Register Read (Get) or Write (Set) Flyer's 8-bit output register

Mark Status Commands (page 9):

- Read/Write Marking Status Read (Get) mark status or Write (Start/Abort) mark session
- Read All Mark Status Read (Get) Mark Count, Current Piece, Ticks, Tick Min, Tick Max, Servo Status, and Head Uptime values from the current mark session

Head Temperature Commands (page 13):

- Read Head Temperatures Read (Get) actual Flyer front (power amp) temperature, rear (CPU) temperature and front/rear (CPU) over-temperature status

Marking Head Status Commands (page 15):

- Read Marking Head Status Read (Get) Head Type, current mark status – marking/not marking, and Stand-alone operation status
- Read/Write Network Status Read (Get) Network Share status or Write (Refresh) Network Share connection

Date/Time Commands (page 19):

- Read/Write All Date/Time Status Read (Get) or Write (Set) Flyer's current local date (Year, Month, Day of Week, Day) and time (Hour, Minute, Second, and Milliseconds)

Filestore Usage Commands (page 21):

- Read Filestore Usage Read (Get) number of bytes Used and Available in Flyer Filestore

Error Code Commands (page 23):

- Read Synrad Error Code Read (Get) any Synrad error code generated by Flyer during a Modbus operation

File Commands (page 24):

- Read/Write Filename Read (Get) current filename or Write (Load) a file into RAM for marking
- Write Network File Write (Load) a file located on a network share into Flyer RAM for marking

File Properties Commands (page 28):

- Read/Write Property Value Write (Set) file *Object Name*, Write (Set) file *Property Name*, and Read (Get) or Write (Set) value of selected object property

System Parameters Commands (page 30):

- Read/Write Parameter Value Write (Set) head parameter by name and Read (Get) or Write (Set) value of selected head parameter

I/O Commands

Read Input Register: Read (Get) the status of Flyer's 8-bit input register.

Command (from Client):

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read one (1) word, or element,
- beginning at Flyer Register Address 0 (0x0000).

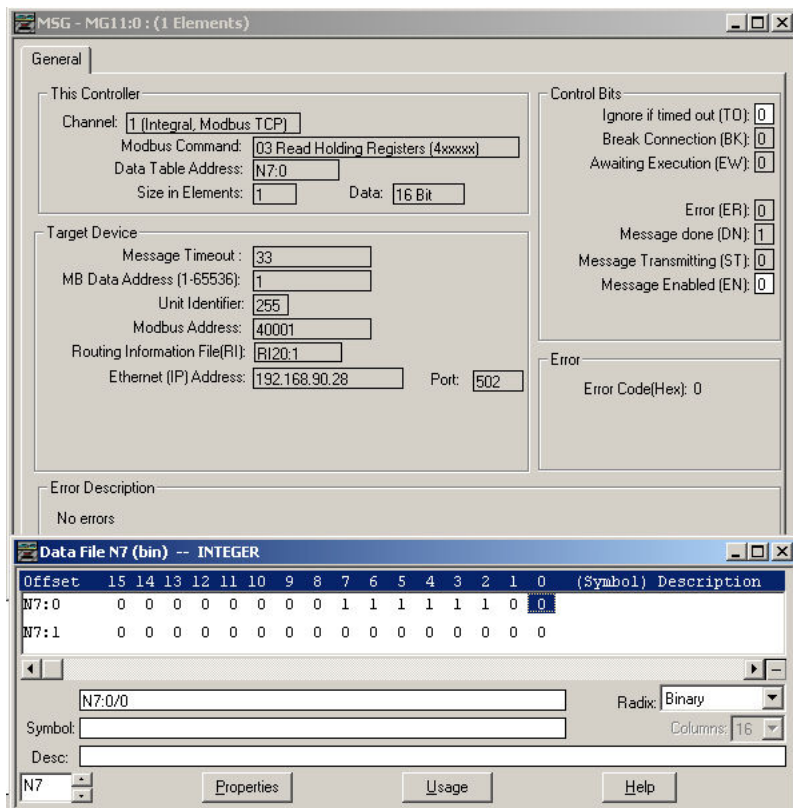
Response (from Flyer):

The contents of Register 0 are sent to the specified data table address.

Example:

Data table N7 contents reflect Flyer's current input bit status in a binary format, where the LSB (N7:0/0) is input IN0 status and the MSB (N7:0/7) is IN7 status. As seen in the screen shot, inputs IN0 and IN1 are OFF, while inputs IN2 through IN7 are ON. Decimal and hexadecimal data table equivalents of the binary input status (1111 1100) are 252 and 0x00FC respectively.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (0) has been incremented by 1 (MB Data Address is set to 1).



The screenshot shows two windows from a software interface. The top window is titled "MSG - MG11:0 : (1 Elements)" and has a "General" tab. It contains configuration fields for "This Controller" (Channel: 1, Modbus Command: 03 Read Holding Registers, Data Table Address: N7:0, Size in Elements: 1, Data: 16 Bit) and "Target Device" (Message Timeout: 33, MB Data Address: 1, Unit Identifier: 255, Modbus Address: 40001, Routing Information File: RI20:1, Ethernet (IP) Address: 192.168.90.28, Port: 502). On the right, "Control Bits" are set to 0 for Ignore if timed out (TO), Break Connection (BK), Awaiting Execution (EW), Error (ER), Message Transmitted (ST), and Message Enabled (EN). The "Error" section shows "Error Code(Hex): 0". The "Error Description" section shows "No errors".

The bottom window is titled "Data File N7 (bin) -- INTEGER". It displays a table of bit values for N7:0 and N7:1. The N7:0 row shows bits 15 through 0 with values 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0. The N7:1 row shows all bits as 0. Below the table, the "Symbol" field is set to "N7:0/0", the "Radix" is "Binary", and the "Columns" is "16".

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	(Symbol)	Description
N7:0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0		
N7:1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Note: By reading two words (or elements) beginning at Register 0, you can read both input and output status with a single MSG instruction.



Read/Write Output Register: Read (Get) or Write (Set) Flyer's 8-bit output register.

Command (from Client):

If Read

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read one (1) word, or element,
- beginning at Flyer Register Address 2 (0x0002).

If Write

- Set Message (MSG) instruction to send Modbus command function code 06, Write Single Holding Register,
- and write one (1) word, or element,
- beginning at Flyer Register Address 2 (0x0002).
- Valid data is a byte value formatted so the LSB corresponds to output OUT0 and the MSB corresponds to OUT7 and where a '1' sets the output ON and '0' sets it OFF.

Response (from Flyer):

If Read

The contents of Flyer Register 2 are sent to the specified data table address.

If Write

The value of the specified data table address is sent to Flyer Register 2.

[Example on next page]

Read/Write Output Register (cont):

Example:

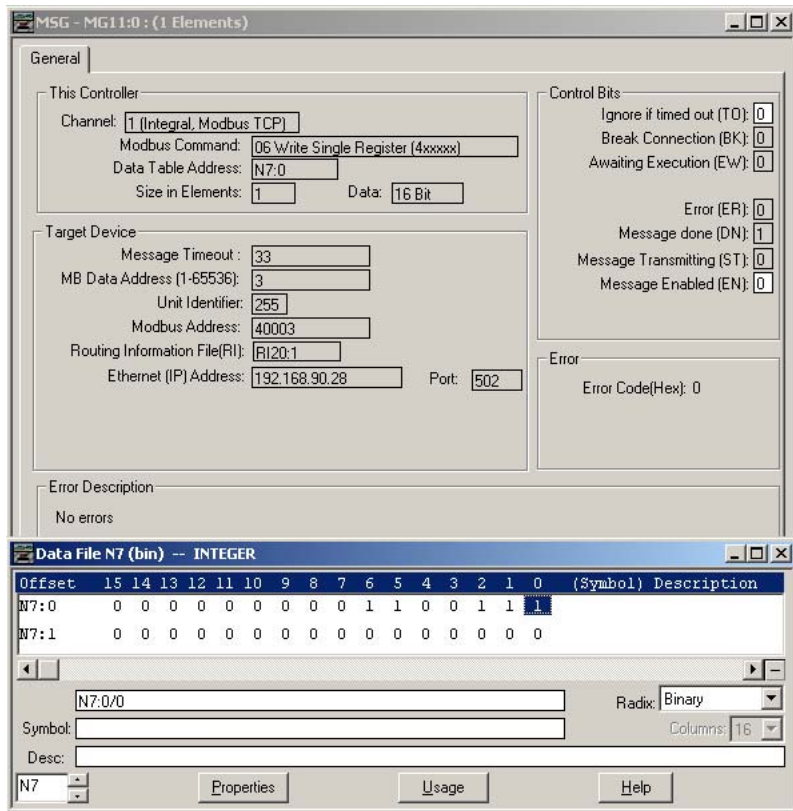
If Read

The specified data table reflects Flyer’s current output bit status, where the LSB is output OUT0 status and the MSB holds OUT7 status.

If Write

Data table N7 contains the desired Flyer output state in binary format, where the LSB (N7:0/0) is output OUT0 status and the MSB (N7:0/7) is OUT7 status. As seen in the screen shot, Flyer outputs OUT0, OUT1, OUT2, OUT5, and OUT6 will be turned ON, while outputs OUT3, OUT4, and OUT7 will be turned OFF. Decimal and hexadecimal data table equivalents of the binary output status (0110 0111) are 103 and 0x0067 respectively.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (2) has been incremented by 1 (MB Data Address is set to 3).



Note: By reading two words (or elements) beginning at Register 0, you can read both input and output status with a single MSG instruction.

Mark Status Commands

Read/Write Marking Status: Read (Get) Flyer's marking status or Write (Set) to start or abort a mark session.

Command (from Client):

If Read

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read one (1) word, or element,
- beginning at Flyer Register Address 4 (0x0004).

If Write

- Set Message (MSG) instruction to send Modbus command function code 06, Write Single Holding Register,
- and write one (1) word, or element,
- beginning at Flyer Register Address 4 (0x0004).
- Valid data is a 16-bit word containing decimal: '1' – start marking; or '2' – abort mark session.

Response (from Flyer):

If Read

The contents of Flyer Register 4 are sent to the specified data table address. Valid data is a 16-bit word containing a decimal value of: '0' – head is idle; '1' – head is marking; or '2' – mark session aborted.

If Write

The value of the specified data table address is sent to Flyer Register 4 and a mark session is started or aborted.

[Example on next page]

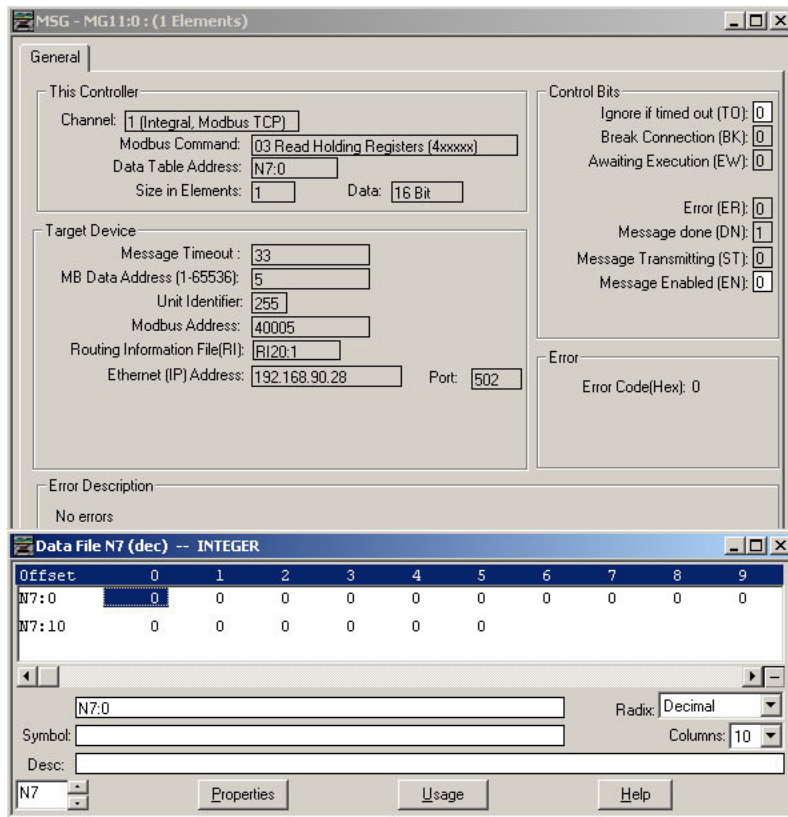
Read/Write Marking Status (cont):

Example:

If Read

The specified data table reflects Flyer's current mark status as shown in the screen shot. The value of N7:0 is decimal 0, which indicates the Flyer head is idle (not marking and last mark was not aborted).

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (4) has been incremented by 1 (MB Data Address is set to 5).



The image shows two screenshots from a software interface. The top screenshot is the 'MSG - MG11:0 : (1 Elements)' configuration window, 'General' tab. It shows settings for 'This Controller' (Channel: 1, Modbus Command: 03 Read Holding Registers (4xxxxx), Data Table Address: N7:0, Size in Elements: 1, Data: 16 Bit) and 'Target Device' (Message Timeout: 33, MB Data Address (1-65536): 5, Unit Identifier: 255, Modbus Address: 40005, Routing Information File (RI): RI20.1, Ethernet (IP) Address: 192.168.90.28, Port: 502). Control Bits are set to 0 for Ignore if timed out (TO), Break Connection (BK), Awaiting Execution (EW), Error (ER), Message Transmitted (ST), and Message Enabled (EN). The Error Code (Hex) is 0. The bottom screenshot is the 'Data File N7 (dec) -- INTEGER' window, showing a table of data for N7:0 and N7:10. The value for N7:0 is 0.

Offset	0	1	2	3	4	5	6	7	8	9
N7:0	0	0	0	0	0	0	0	0	0	0
N7:10	0	0	0	0	0	0				

If Write

The specified data table address, N7:0 in this example, would contain a decimal value of '1' to start marking, or a value of '2' to abort the mark session.



Read All Mark Status: Read (Get) all mark status for the current mark session or read one or more property registers.

Note: Because you can specify only one starting address per MSG instruction, you can only read multiple registers (properties) sequentially.

Command (from Client):

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read two (2) words, or elements, per property.
- Valid Flyer Registers and properties are:

Register Address 6 (0x0006)	Mark Count; Mark Count property value in current file
Register Address 10 (0x000A)	Current Piece; number of piece currently being marked
Register Address 14 (0x000E)	Ticks; total number of ticks in mark session (100 ticks = 1 second)
Register Address 18 (0x0012)	Tick Min; minimum number of ticks for any piece in mark session
Register Address 22 (0x0016)	Tick Max; maximum number of ticks for any piece in mark session
Register Address 26 (0x001A)	Servo Status; values indicating state of Flyer's XY galvanometers
Register Address 32 (0x0020)	Head Uptime; number of seconds since the last boot-up

Response (from Flyer):

The contents of the specified Flyer Register(s) are sent to the specified data table address.

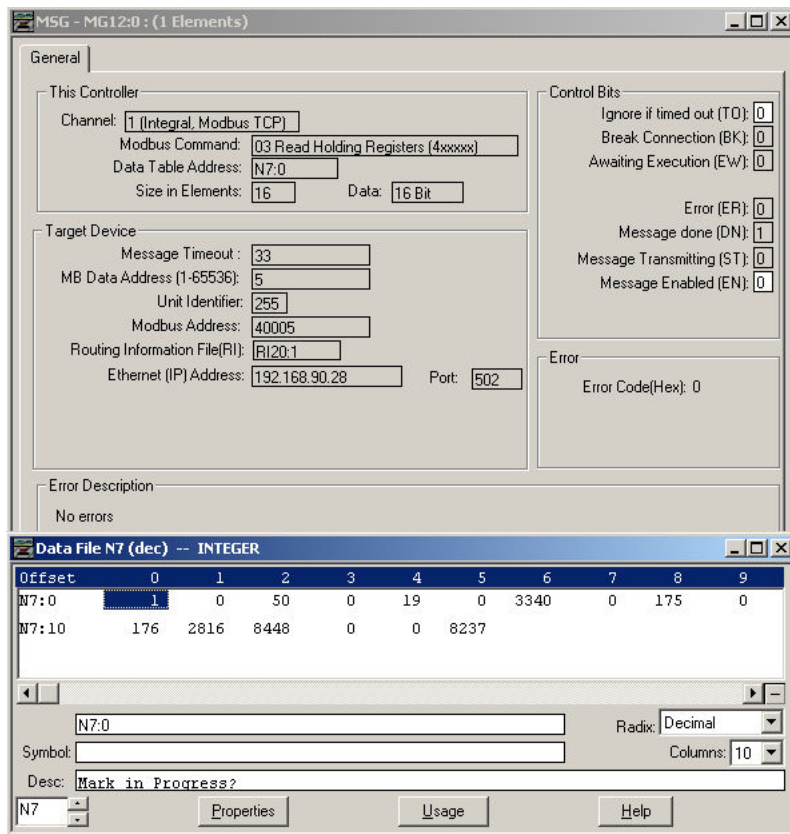
[Example on next page]

Read All Mark Status (cont):

Example:

The screen shot shows the decimal results of reading all mark status, including current marking status (consisting of only 1 word), after reading 16 total words within a single MSG instruction beginning at register address 4. N7:0 shows Flyer status ('1' – head is marking); N7:2 shows the Mark Count property of the current file is set to 50 pieces; N7:4 shows the Current Piece being marked is number 19 out of 50; in N7:6, the total number of Ticks in the mark session so far is 3340 (33.40 seconds); N7:8 shows Tick Min is currently 175 (1.75 seconds) per piece; N7:10 shows Tick Max is 176 (1.76 seconds) per piece; N7:11 and N7:12 show Flyer's servo status; and N7:15 displays the number of seconds that have elapsed since the Flyer head last booted-up (8237 seconds = 2 hours, 17 minutes, 17 seconds).

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (4) has been incremented by 1 (MB Data Address is set to 5).



The screenshot displays two windows from a software interface. The top window, titled "MSG - MG12:0 : (1 Elements)", shows configuration for a Modbus message. The "General" tab is active, showing "This Controller" settings: Channel: 1 (Integral, Modbus TCP), Modbus Command: 03 Read Holding Registers (4xxxx), Data Table Address: N7:0, Size in Elements: 16, and Data: 16 Bit. The "Target Device" section includes Message Timeout: 33, MB Data Address (1-65536): 5, Unit Identifier: 255, Modbus Address: 40005, Routing Information File (RI): RI20.1, and Ethernet (IP) Address: 192.168.90.28 with Port: 502. The "Control Bits" section shows various flags like Ignore if timed out (TO), Break Connection (BK), Awaiting Execution (EW), Error (ER), Message done (DN), Message Transmitting (ST), and Message Enabled (EN), all set to 0 or 1. The "Error" section shows Error Code (Hex): 0. The "Error Description" section shows "No errors".

The bottom window, titled "Data File N7 (dec) -- INTEGER", displays a table of register values:

Offset	0	1	2	3	4	5	6	7	8	9
N7:0	1	0	50	0	19	0	3340	0	175	0
N7:10	176	2816	8448	0	0	8237				

Below the table, there are input fields for "Symbol" (N7:0), "Radix" (Decimal), "Columns" (10), and "Desc" (Mark in Progress?). Buttons for "Properties", "Usage", and "Help" are visible at the bottom.



Head Temperature Commands

Read Head Temperatures: Read (Get) actual Flyer front (power amp) temperature, rear (CPU) temperature and front/rear (CPU) over-temperature status. Temperatures are displayed in tenths of a degree Celsius.

Note: Because you can specify only one starting address per MSG instruction, you can only read multiple registers (properties) sequentially.

Command (from Client):

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read two (2) words, or elements, per property.
- Valid Flyer Registers and properties are:

Register Address 36 (0x0024)	Current front (power amp) temperature, in Celsius
Register Address 38 (0x0026)	Current rear (CPU) temperature, in Celsius
Register Address 40 (0x0028)	Front over-temp status; '0' if OK, '1' if overtemp condition exists
Register Address 42 (0x002A)	Rear over-temp status; '0' if OK, '1' if overtemp condition exists

Response (from Flyer):

The contents of the specified Flyer Register(s) are sent to the specified data table address. Valid data sets are 16-bit words containing a decimal value for front/rear temperatures in tenths of a degree Celsius or a decimal '0' if no over-temperature condition exists or a '1' if an overtemp fault has occurred.

[Example on next page]

Read Head Temperatures (cont):

Example:

The screen shot shows the decimal results of reading head temperature and over-temperature status after reading 4 words within a single MSG instruction beginning at register address 36. N7:0 shows the current front (power amp) temperature as 355 in tenths of a degree Celsius, which equals 35.5 °C (or 95.9 °F); N7:1 shows the current rear (CPU) temperature as 308 in tenths of a degree Celsius, which equals 30.8 °C (or 87.4 °F); N7:2 displays front over-temperature status as 0, meaning a front overtemp fault does **not** exist; and N7:3 displays rear over-temperature status as 0, meaning a rear overtemp fault does **not** exist.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (36) has been incremented by 1 (MB Data Address is set to 37).

The screenshot displays two windows from a PLC configuration software. The top window, titled "MSG - MG11:0 : (1 Elements)", shows the configuration for a Modbus Read Holding Registers instruction. The "General" tab is active, showing a channel of 1 (Integral, Modbus TCP), a Modbus Command of "03 Read Holding Registers (4xxxx)", and a Data Table Address of "N7:0". The "Target Device" section shows a Message Timeout of 33, MB Data Address of 37, Unit Identifier of 255, Modbus Address of 40037, Routing Information File (RI) of RI20:1, and Ethernet (IP) Address of 192.168.90.28 on Port 502. The "Control Bits" section shows various status bits like Ignore if timed out (TO), Break Connection (BK), Awaiting Execution (EW), Error (ER), Message done (DN), Message Transmitting (ST), and Message Enabled (EN), all set to 0 or 1. The "Error" section shows an Error Code (Hex) of 0. The "Error Description" section shows "No errors".

The bottom window, titled "Data File N7 (dec) -- INTEGER", shows a table of data for the N7 register. The table has columns for Offset (0-9) and rows for N7:0 and N7:10. The data for N7:0 is 355, 308, 0, 0, 0, 0, 0, 0, 0, 0. The data for N7:10 is 0, 0, 0, 0, 0, 0.

Offset	0	1	2	3	4	5	6	7	8	9
N7:0	355	308	0	0	0	0	0	0	0	0
N7:10	0	0	0	0	0	0	0	0	0	0

Below the table, there are fields for "Symbol" (N7:0), "Radix" (Decimal), "Columns" (10), and "Desc". At the bottom, there are buttons for "Properties", "Usage", and "Help".



Marking Head Status Commands

Read Marking Head Status: Read (Get) Head Type, current marking status – marking/not marking, and Stand-alone operation status. You can also Read (Get) status of the Network Share – available/not-available (not connected).

Note: Because you can specify only one starting address per MSG instruction, you can only read multiple registers (properties) sequentially.

Command (from Client):

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read one (1) word, or element, per property.
- Valid Flyer Registers and properties are:

Register Address 56 (0x0038)	Head Type; '1' if Flyer/Fenix Flyer
Register Address 58 (0x003A)	Marking status; '0' if idle, '1' if marking
Register Address 60 (0x003C)	Standalone Marking enabled?; '0' if No, '1' if Yes
Register Address 62 (0x003E)	Network Share Available?; '0' if No, '1' if Yes

Response (from Flyer):

The contents of the specified Flyer Register(s) are sent to the specified data table address.

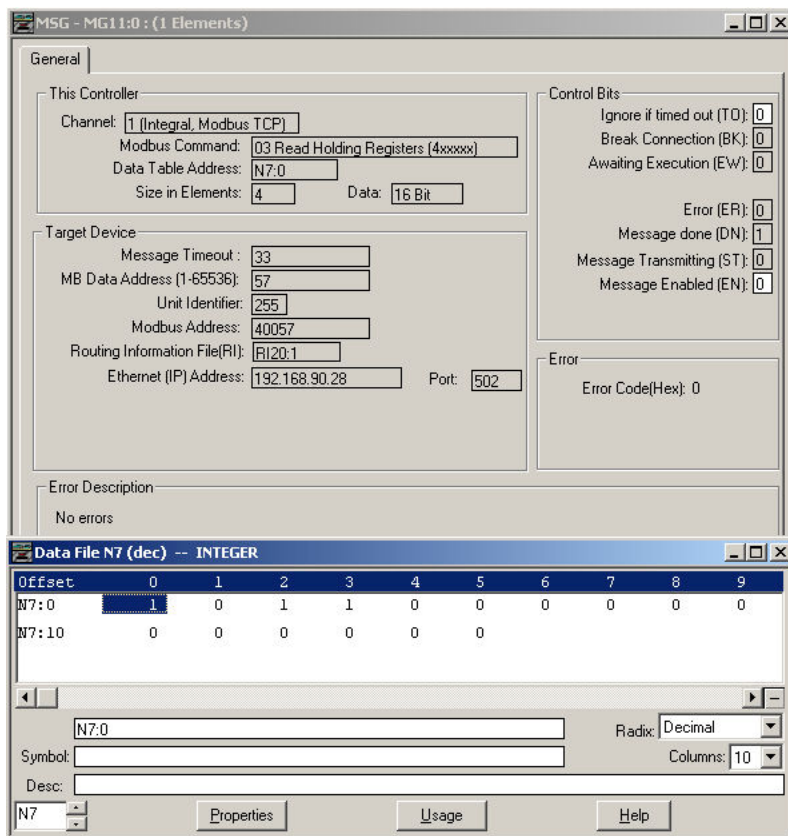
[Example on next page]

Read Marking Head Status (cont):

Example:

The screen shot shows the decimal results of current marking head status after reading 4 words within a single MSG instruction beginning at register address 56. N7:0 shows the Head Type property as 1 (Flyer); N7:1 is 0, so the head is idle (not marking); N7:2 is 1, meaning the Standalone Marking property is enabled (Yes); and N7:3 is 1 to indicate a network share is connected and available.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (56) has been incremented by 1 (MB Data Address is set to 57).



Read/Write Network Status: Read (Get) network share status or Write (Refresh) the network share connection.

Command (from Client):

If Read

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read one (1) word, or element,
- beginning at Flyer Register Address 62 (0x003E).

If Write

- Set Message (MSG) instruction to send Modbus command function code 06, Write Single Holding Register,
- and write one (1) word, or element,
- beginning at Flyer Register Address 62 (0x003E).
- Valid data is a 16-bit word containing any decimal value—the act of writing the register is all that is required to restore (refresh) the connection between Flyer and the network share. We recommend Reading network status after a Write operation to verify the network share folder is available.

Response (from Flyer):

If Read

The contents of Flyer Register 62 are sent to the specified data table address. Valid data is a 16-bit word containing a decimal value of: '0' if the network share is **not** available; or '1' when the share is connected and available.

If Write

The value of the specified data table address is sent to Flyer Register 62. We recommend Reading network status after a Write operation to verify the network share is available.

[Example on next page]

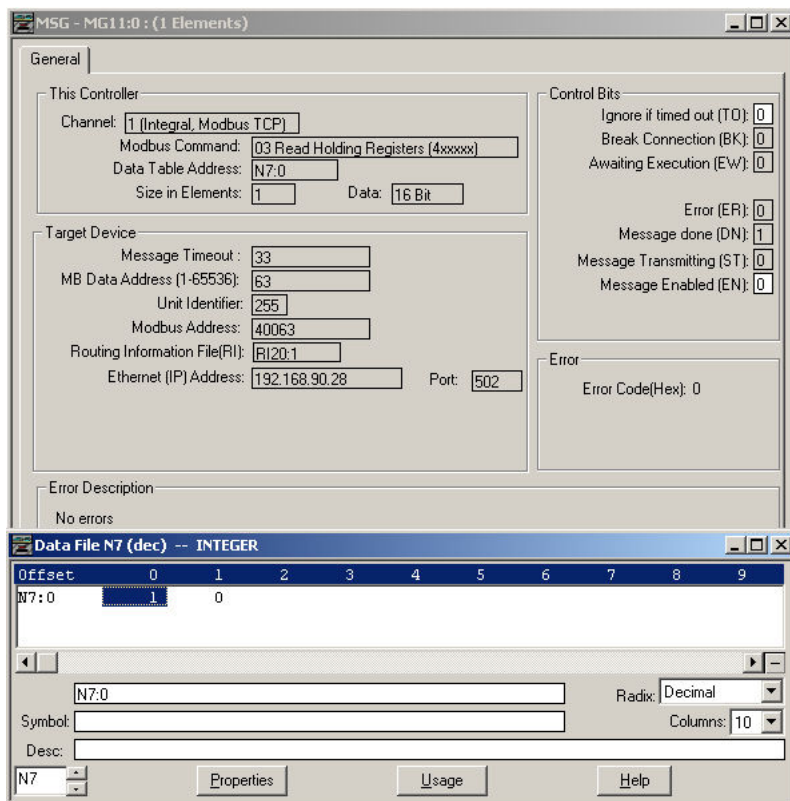
Read/Write Network Status (cont):

Example:

If Read

The specified data table reflects the state of Flyer's network share connection as shown in the screen shot. The value of N7:0 is decimal 1, which indicates the network share folder is connected and available.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (62) has been incremented by 1 (MB Data Address is set to 63).



If Write

Any data table value written to Flyer Register 62 initiates a refresh of the network share connection.

Date/Time Commands

Read/Write All Date/Time Status: Read (Get) Flyer's current local date/time status or Write (Set) new local date/time values.

Note: Because you can specify only one starting address per MSG instruction, you can only read or write multiple registers (properties) sequentially.

Command (from Client):

If Read

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read one (1) word, or element, per property.
- beginning at Flyer Register Address 64 (0x0040).
- Valid Flyer Registers and properties are:

Register Address 64 (0x0040)	Year, 4-digit
Register Address 66 (0x0042)	Month, 1–12 where January = 1, February = 2, etc.
Register Address 68 (0x0044)	Day Of Week, 0–6 where Sunday = 0, Monday = 1, etc.
Register Address 70 (0x0046)	Day, 1–31
Register Address 72 (0x0048)	Hour, 0–23
Register Address 74 (0x004A)	Minute, 0–59
Register Address 76 (0x004C)	Second, 0–59
Register Address 78 (0x004E)	Milliseconds, always zero (0)

If Write

- Set Message (MSG) instruction to send Modbus command function code 16, Write Multiple Holding Registers,
- and write eight (8) words, or elements,
- beginning at Flyer Register Address 64 (0x0040).

Important Note: When writing Date/Time values you must Write to all eight Date/Time registers.

Response (from Flyer):

If Read

The contents of the specified Flyer Register(s) are sent to the specified data table address.

If Write

The decimal values in the specified data table are sent to Flyer Register addresses 64, 66, 68, 70, 72, 74, 76, and 78.

[Example on next page]

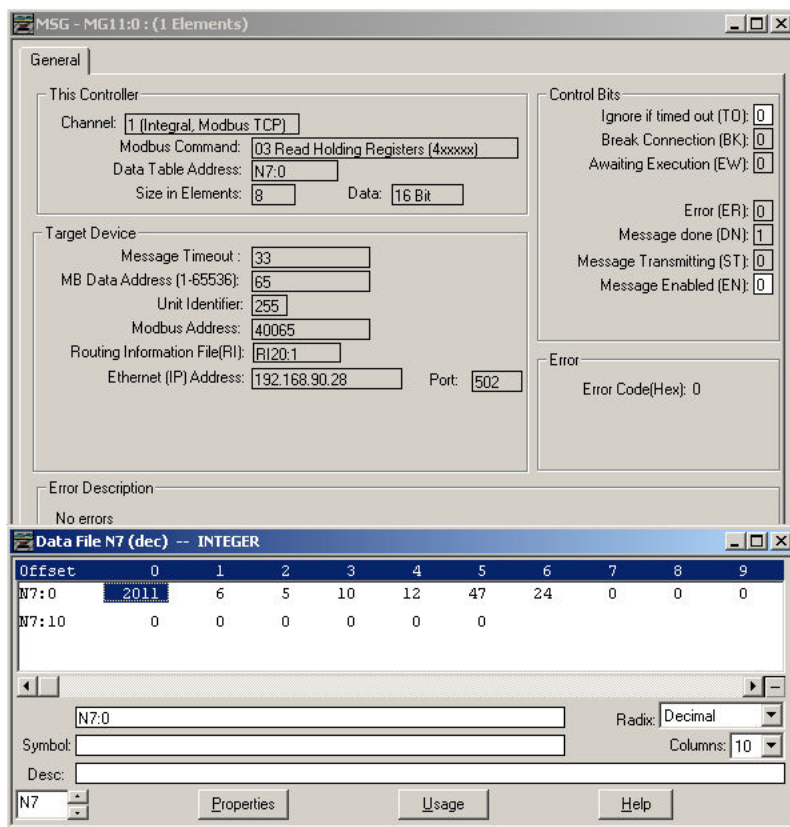
Read/Write All Date/Time Status (cont):

Example:

If Read

The specified data table reflects Flyer’s current local date/time status as seen in the screen shot. N7:0 is the 4-digit year, 2011; N7:1 is the month, 6 equals June; N7:2 is day of the week, 5 equals Friday; N7:3 is the day, the 10th; N7:4 is the hour, 12 PM; N7:5 are the minutes, 47; N7:6 are the seconds, 24; and N7:7 are milliseconds, always zero (0).

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (64) has been incremented by 1 (MB Data Address is set to 65).



If Write

The decimal values contained in the specified data table are sent to Flyer Register addresses 64, 66, 68, 70, 72, 74, 76, and 78.

Important Note: When writing Date/Time values you must Write to all eight Date/Time registers.



Filestore Usage Commands

Read Filestore Usage: Read (Get) current Used and Available Filestore usage in bytes.

Note: Because you can specify only one starting address per MSG instruction, you can only read multiple registers (properties) sequentially.

Command (from Client):

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read two (2) words, or elements, per property.
- Valid Registers and properties are:

Register Address 84 (0x0054) Number of bytes currently Used in Filestore
Register Address 88 (0x0058) Number of bytes currently Available in Filestore

Response (from Flyer):

The contents of the specified Flyer Register(s) are sent to the specified data table address.

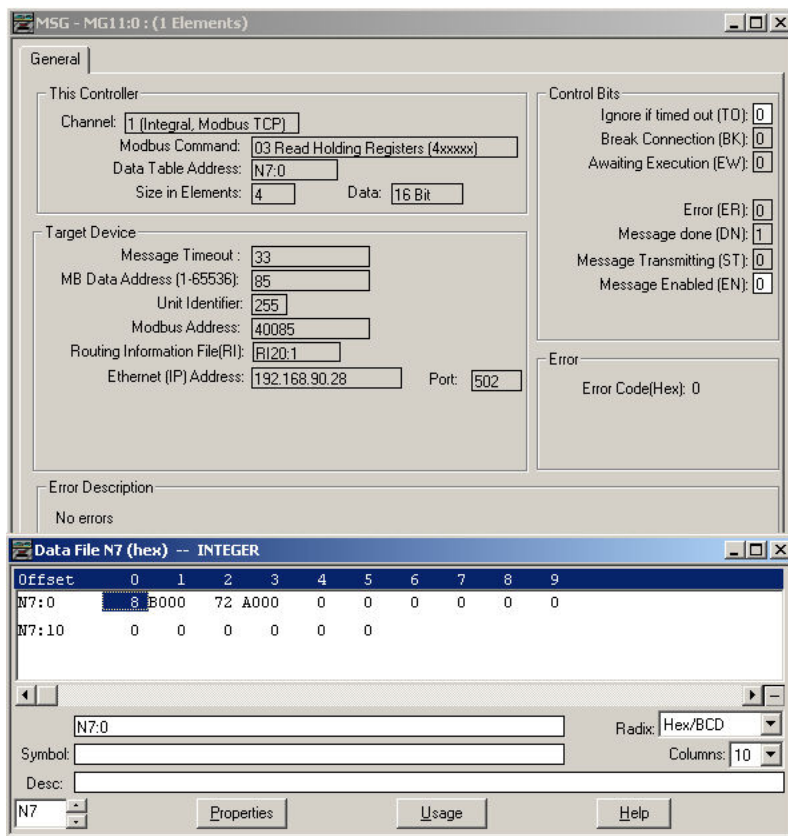
[Example on next page]

Read Filestore Usage (cont):

Example:

The screen shot shows current Flyer Filestore usage after reading 4 words within a single MSG instruction beginning at register address 84. This hexadecimal view of data table N7 shows addresses N7:0/N7:1 with Used filespace in hex bytes, where 0008 B000 equals 569,344 bytes while N7:2/N7:3 contains Available filespace in hex bytes, where 72 A000 equals 7,512,064 bytes.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (84) has been incremented by 1 (MB Data Address is set to 85).





Error Code Commands

Read Synrad Error Code: Read (Get) any Synrad error code generated by Flyer during a Modbus operation.

Command (from Client):

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read one (1) word, or element,
- beginning at Flyer Register Address 102 (0x0066).

Response (from Flyer):

The contents of Flyer Register 102 are sent to the specified data table address.

Example:

On receipt of a Modbus error (see list in [Appendix C](#)), query Register 102 for a more specific Synrad Error Code (also shown in [Appendix C](#)).

File Commands

Read/Write Filename: Read (Get) the current file loaded into Flyer RAM or Write (Load) a file from the Filestore into Flyer RAM for marking.

Note: Use the **Write Network File** command to load a file located on a network share into Flyer RAM for marking.

Command (from Client):

If Read

- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read a maximum 120 words, or elements (limited by Modbus specification),
- beginning at Flyer Register Address 256 (0x0100).

If Write

- Set Message (MSG) instruction to send Modbus command function code 16, Write Multiple Holding Register,
- and write a maximum of 120 words, or elements (limited by Modbus standard),
- beginning at Flyer Register Address 256 (0x0100).
- The path or filename must be preceded by a '/' character and the ASCII string must be null-terminated (0x0000 or decimal 0; **not** 0x0030 or decimal 48 – the keyboard '0' character).

Response (from Flyer):

If Read

The contents of Flyer Register 256 are sent to the specified data table address.

If Write

The ASCII string in the data table is sent to the specified Flyer Register(s) causing Flyer to load the specified file (from the Filestore) into Flyer RAM.

[Example on next page]

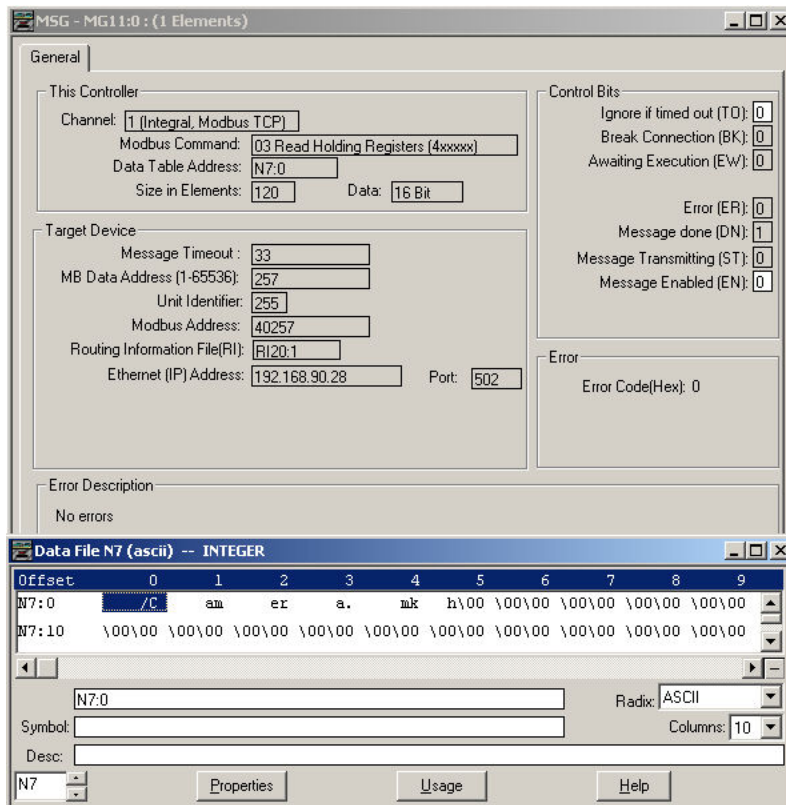
Read/Write Filename (cont):

Example:

If Read

This screen shot shows an ASCII view of data table N7 after a **Read/Write Filename** command where words N7:0–N7:5 show the null-terminated filename of the current file loaded in Flyer RAM (/C am er a. mh h\00). The returned filename, Camera.mkh is always preceded by a ' ' character. The Read command returns the full path, filename, and extension of the file; for example: "/Camera.mkh\00" or "/mydir/myotherfile.mkh\00" or, when the file is located on a network share, "/network/sharedir/sharefile.mkh\00".

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (256) has been incremented by 1 (MB Data Address is set to 257).



If Write

The ASCII string in the data table is sent to the specified Flyer Register(s) causing Flyer to load the specified file (from the Filestore) into Flyer RAM.

Write Network File: Write (Load) a file from a network file share into Flyer RAM for marking.

Note: Use the **Read/Write Filename** command to load a file saved in the Flyer Filestore into RAM for marking.

Command (from Client):

- Check beforehand that the network share is properly configured and available. Read Flyer Register Address 62 (0x003E) and verify the return value is a decimal '1' to indicate the share folder is connected and available. If necessary, open WinMark Pro and configure network share settings.
- Set Message (MSG) instruction to send Modbus command function code 16, Write Multiple Holding Register,
- and write a maximum of 120 words, or elements (limited by Modbus standard),
- beginning at Flyer Register Address 1024 (0x0400).
- The path or filename must be preceded by a '/' character and the ASCII string must be null-terminated (0x0000 or decimal 0; **not** 0x0030 or decimal 48 – the keyboard '0' character).

Response (from Flyer):

The ASCII string in the data table is sent to the specified Flyer Register(s) causing Flyer to load the specified file (from the network share folder) into Flyer RAM for marking. If necessary, use the **Read/Write Filename** command to verify the network file is loaded into Flyer RAM.

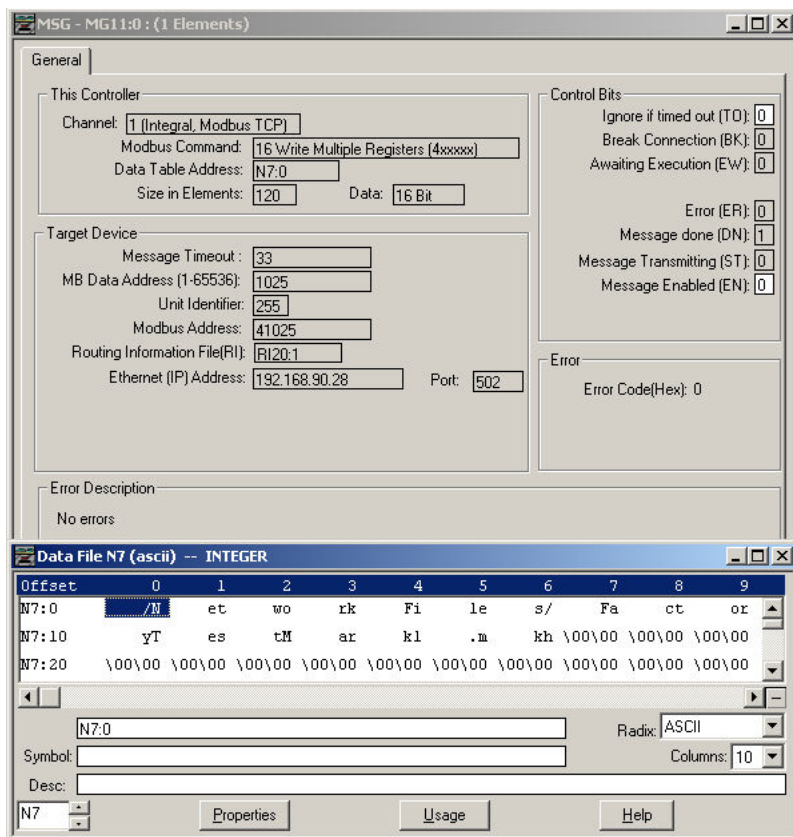
[Example on next page]

Write Network File (cont):

Example:

This screen shot shows an ASCII view of data table N7 where words N7:0–N7:16 show the null-terminated path/filename of a file located in a subdirectory on a network share that will be loaded into Flyer RAM as the current file for marking. The specified full path and filename, NetworkFiles/FactoryTestMark1.mkh (/N et wo rk Fi le s/ Fa ct or yT es tM ar k1 .m kh \00\00), is always preceded by a '/' character and the ASCII string must be null-terminated.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register address (1024) has been incremented by 1 (MB Data Address is set to 1025).



File Properties Commands

Read/Write Property Value: Write (Set) an *Object Name*, Write (Set) an *Object Property*, and then Read (Get) or Write (Set) a property value.

Note: To obtain a list of valid *Object Name* and *Property Names* for a given mark file, open the mark file in WinMark Pro and from the *File* menu, click *Print All Properties*.

Command (from Client):

Read Property Value

- Set Message (MSG) instruction to send Modbus function code 16, Write Multiple Holding Registers, and write a maximum of 19 words, or elements (the ASCII string must be null-terminated), beginning at Flyer Register Address 504 (0x01F8).
- Set Message (MSG) instruction to send Modbus function code 16, Write Multiple Holding Registers, and write a maximum of 23 words, or elements (the ASCII string must be null-terminated), beginning at Flyer Register Address 544 (0x0220).
- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers, and read a maximum of 60 words, or elements, beginning at Flyer Register Address 592 (0x0250).
- Valid Registers and properties are:

Register Address 504 (0x01F8) Object Name
Register Address 544 (0x0220) Object Property
Register Address 592 (0x0250) Property Value

Write Property Value

- Set Message (MSG) instruction to send Modbus function code 16, Write Multiple Holding Registers, and write a maximum of 19 words, or elements (the ASCII string must be null-terminated), beginning at Flyer Register Address 504 (0x01F8).
- Set Message (MSG) instruction to send Modbus function code 16, Write Multiple Holding Registers, and write a maximum of 23 words, or elements (the ASCII string must be null-terminated), beginning at Flyer Register Address 544 (0x0220).
- Set Message (MSG) instruction to send Modbus command function code 16, Write Multiple Holding Registers, and write a maximum of 60 words, or elements (the ASCII string must be null-terminated), beginning at Flyer Register Address 592 (0x0250).
- Valid Registers and properties are:

Register Address 504 (0x01F8) Object Name
Register Address 544 (0x0220) Object Property
Register Address 592 (0x0250) Property Value



Read/Write Property Value (cont):

Important Note: After the initial Write of *Object Name* and *Property Name* registers and a Write or Read of the Property Value register, you can then Write or Read the Property Value by performing a Write or Read of only the Property Value, Flyer Register 592 (0x0250). However, to prevent errors, you should always Write (specify) an *Object Name* and *Property Name* before Writing/Reading a Property Value.

Response (from Flyer): If Read Property Value

The ASCII strings in the specified data table addresses for *Object Name* and *Property Name* are sent to the specified Flyer Registers and the contents (Property Value) of the specified Flyer Register(s) are sent to the specified data table address.

If Write Property Value

The ASCII strings in the specified data table addresses for *Object Name* and *Property Name* are sent to the specified Flyer Registers and the ASCII string in the specified data table address for Property Value is sent to the specified Flyer Register(s) causing Flyer to update the Property Value of the specified object in the mark file stored in Flyer RAM for marking.

Example:

This screen shot shows an ASCII view of data table N7 where words N7:0–N7:3 show the null-terminated *Object Name* (Drawing) and words N7:40–N7:44 show the null-terminated *Property Name* (Mark Count). In this case, the value read for the Drawing's Mark Count property (word N7:90) is 50.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register addresses have been incremented by 1.

The screenshot displays three Modbus configuration windows and an ASCII data table view. The top-left window (MSG - MG11:0) is configured for '16 Write Multiple Registers (4xxxxx)' with Data Table Address N7:0 and Size in Elements 19. The top-right window (MSG - MG12:0) is also configured for '16 Write Multiple Registers (4xxxxx)' with Data Table Address N7:40 and Size in Elements 23. The bottom-left window (MSG - MG14:0) is configured for '13 Read Holding Registers (4xxxxx)' with Data Table Address N7:90 and Size in Elements 87. The bottom-right window shows the 'Data File N7 (ascii) - INTEGER' view, which is a table of 10 columns (Offset 0-9) and 11 rows (N7:0-N7:100). The data in the table is as follows:

Offset	0	1	2	3	4	5	6	7	8	9
N7:0	Dr	aw	in	g\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:10	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:20	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:30	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:40	Ma	rk	Co	un	t\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:50	\00\00	\00\00	\00\00	\00\00	[N7:34]	\00\00	\00\00	\00\00	\00\00	\00\00
N7:60	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:70	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:80	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:90	50	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:100	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00



System Parameters Commands

Read/Write Parameter Value: Write (Set) a head parameter and then Read (Get) or Write (Set) a head parameter value.

Note: See [Appendix A](#) for a list of valid Flyer/Fenix Flyer head parameter names.

Command (from Client):

Read Parameter Value

- Set Message (MSG) instruction to send Modbus command function code 16, Write Multiple Holding Registers,
- and write a maximum of 43 words, or elements (the ASCII string must be null-terminated),
- beginning at Flyer Register Address 768 (0x0300).
- Set Message (MSG) instruction to send Modbus command function code 03, Read Holding Registers,
- and read a maximum of 60 words, or elements,
- beginning at Flyer Register Address 856 (0x0358).
- Valid Registers and properties are:

Register Address 768 (0x0300) Head Parameter
Register Address 856 (0x0358) Parameter Value

Write Parameter Value

- Set Message (MSG) instruction to send Modbus command function code 16, Write Multiple Holding Registers,
- and write a maximum of 43 words, or elements (the ASCII string must be null-terminated),
- beginning at Flyer Register Address 768 (0x0300).
- Set Message (MSG) instruction to send Modbus command function code 16, Write Multiple Holding Registers,
- and write a maximum of 60 words, or elements (the ASCII string must be null-terminated),
- beginning at Flyer Register Address 857 (0x0358).
- Valid Registers and properties are:

Register Address 768 (0x0300) Head Parameter
Register Address 856 (0x0358) Parameter Value

Important Note: After the initial Write of the Head Parameter register and a Write or Read of the Parameter Value register, you can then Write or Read the Parameter Value by performing a Write or Read of only the Parameter Value register, Address 856 (0x0358). However, to prevent errors, you should always Write (specify) a Head Parameter before Writing/Reading a Parameter Value.

Read/Write Parameter Value (cont):

Response (from Flyer):

If Read Parameter Value

The ASCII string in the specified data table address for Head Parameter is sent to the specified Flyer Registers and the contents (Parameter Value) of the specified Flyer Register(s) are sent to the specified data table address.

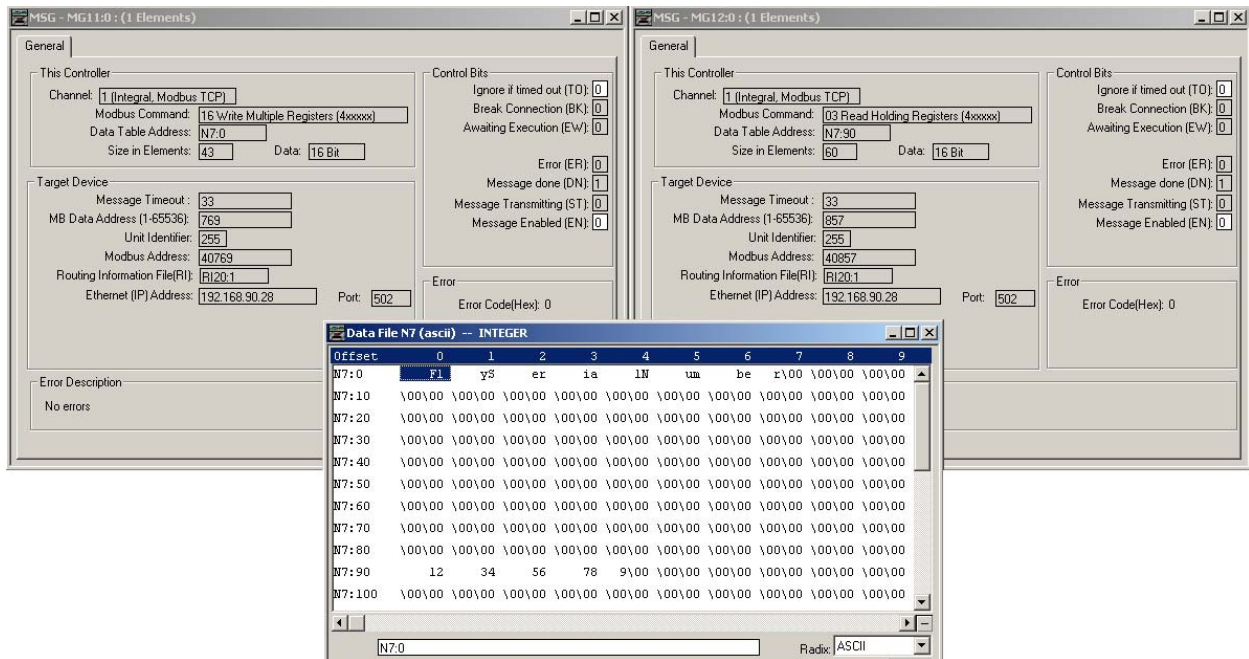
If Write Parameter Value

The ASCII string in the specified data table address for Head Parameter is sent to the specified Flyer Registers and the ASCII string in the specified data table address for Property Value is sent to the specified Flyer Register(s) causing Flyer to update the Head Parameter value in Flyer's configuration file.

Example:

This screen shot shows an ASCII view of data table N7 after a **Read/Write Parameter Value** command where words N7:0–N7:7 show the null-terminated Head Parameter (FlySerialNumber). In this case, the value read for the Parameter Value (words N7:90–N7:94) is 123456789.

Note: The PLC used for the screen shot does not support zero-based addressing so the starting register addresses have been incremented by 1.



The screenshot displays two configuration windows for Modbus messages (MSG - MG110 and MSG - MG120) and a data table window (Data File N7 (ascii) -- INTEGER).

MSG - MG110 (1 Elements) Configuration:

- Channel: 1 (Integral, Modbus TCP)
- Modbus Command: 16 Write Multiple Registers (4xxxx)
- Data Table Address: N7:0
- Size in Elements: 43
- Data: 16 Bx
- Target Device:
 - Message Timeout: 33
 - MB Data Address (1-65536): 769
 - Unit Identifier: 255
 - Modbus Address: 40769
 - Routing Information File(RI): RI201
 - Ethernet (IP) Address: 192.168.90.28
 - Port: 502
- Control Bits: Ignore if timed out (TO): 0, Break Connection (BK): 0, Awaiting Execution (Ew): 0
- Error (ER): 0
- Message done (DN): 1
- Message Transmitting (ST): 0
- Message Enabled (EN): 0
- Error Code(Hex): 0

MSG - MG120 (1 Elements) Configuration:

- Channel: 1 (Integral, Modbus TCP)
- Modbus Command: 03 Read Holding Registers (4xxxx)
- Data Table Address: N7:90
- Size in Elements: 60
- Data: 16 Bx
- Target Device:
 - Message Timeout: 33
 - MB Data Address (1-65536): 857
 - Unit Identifier: 255
 - Modbus Address: 40857
 - Routing Information File(RI): RI201
 - Ethernet (IP) Address: 192.168.90.28
 - Port: 502
- Control Bits: Ignore if timed out (TO): 0, Break Connection (BK): 0, Awaiting Execution (Ew): 0
- Error (ER): 0
- Message done (DN): 1
- Message Transmitting (ST): 0
- Message Enabled (EN): 0
- Error Code(Hex): 0

Data File N7 (ascii) -- INTEGER:

Offset	0	1	2	3	4	5	6	7	8	9
N7:10	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:20	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:30	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:40	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:50	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:60	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:70	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:80	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:90	12	34	56	78	9\00	\00\00	\00\00	\00\00	\00\00	\00\00
N7:100	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00	\00\00

Section 2 – Using Extended (User-Defined) Modbus Command Function Codes

Guidelines for using the User-Defined Modbus/IP protocol:

- The Flyer head MUST be set to operate in stand-alone mode (**Standalone Marking** property on “Device” tab set to “Yes”).
- On the “Device” tab, set the **Modbus User Function** property to a decimal value in the range of 65–72 or 100–110. The default value is 67 (0x43 hex).
- On the “Device” tab, set the **External Communications Server** property to “Modbus”.
- SynComm listens on the default Modbus port (502).
- Modbus is a big endian protocol. In our Modbus VB example code posted at: http://www.winmark.com/products/winmark_activexsamples.html, Mobus.h and Modbus.c contain endian conversion routines to aid in parsing data.
- Modbus is a client/server (or command/response) protocol. The Modbus equipped PLC or computer is the client (that commands), the FH Flyer acts as a server (who responds).

Note: For customers who wish to write applications where the Flyer marking head is the only device on the network, use the SynComm Modbus-Asynchronous protocol. The Modbus-Asynchronous protocol provides additional features that are not part of the standard Modbus/IP protocol such as I/O events, log messages, and intermediate end of mark messages.

- All character strings must be null terminated (designated as “\0” in this document). String length routines do not include the null character as part of string length so you must account for this when parsing data.

Modbus/IP error checking:

- Customers must check for errors per the Modbus I/P specification. The Flyer head returns nine bytes (sMBAP Header = 8 Bytes, Error Code = 1 Byte) when a Modbus error (i.e., Illegal Function Call) occurs. If the returned sMBAP Header.FunctionCode is equal to the transmitted function code + 0x80, then a Modbus error has occurred (the Flyer default is 67) then a Modbus error has occurred. On receipt of a Modbus error, the customer program should flush its send buffers (or perform any other tasks to ensure data integrity of the Ethernet connection) and then resend the last command.



User-Defined Modbus/IP Packet Structure and SynComm Packets

This section describes the encapsulation of a Modbus request or response when it is carried on a Modbus TCP/IP network.

Standard Modbus Packet Structure

The packet structure, as shown below, consists of a 7-byte Modbus Application Protocol (MBAP) Header; a 1-byte Function Code; and up to 252 bytes of Data.

7 Bytes	1 Byte	0 to 252 Bytes
MBAP Header	Function Code	Data

The 7-byte MBAP header, as described below, consists of four fields: a 2-byte Transaction Identifier, followed by a 2-byte Protocol Identifier, a 2-byte Length field, and a 1-byte Unit Identifier field.

MBAP Header Format

Field	Length (Bytes)	Typical Value	Description
Transaction Identifier (TI)	2	00 00*	Identification of a Modbus transaction.
Protocol Identifier (PI)	2	00 00	0 = Modbus protocol.
Length (LN)	2	XX XX	Number of bytes following, including Unit Identifier, Function Code and data.
Unit Identifier (UI)	1	00	Identification of a remote slave connected on a serial line or on other buses.

* The only exception is when using the **Get File List** command.

Function Code

The single-byte Function Code directs Flyer to perform one of four functions using the following values. The first three functions are standard Modbus commands, while the fourth function utilizes the User-Defined extensions within the Modbus specification to direct the Flyer head to perform specific SynComm commands.

Read Holding Registers	0x03
Read Input Registers	0x04
Write Single Register	0x06
SynComm command	0x43 (default) or choose within range 0x41–0x48, 0x64–0x6E

Data

The data portion of the packet is defined by each individual Modbus command. For the custom SynComm packets, the data portion of the packet also includes the SynComm header (SynHeader).



User-Defined SynComm Command/Response Structure

When the Function Code contained within the Modbus/IP packet structure is set to the SynComm User-Defined code, the Flyer head interprets the packet as a SynComm command.

Important Note: The User-Defined SynComm examples in Section 2 will assume: **(1)** the function code being used is the Flyer default value of 0x43 (decimal 67) and **(2)** all two digit numeric values are shown in hexadecimal format, so the '0x' hexadecimal prefix can be omitted for clarity. A displayed value of '43' equals 0x0043.

The SynComm packet data includes an extra header (SynHeader):

SynHeader Format

Fields	Length	Description
SynCode (SC)	2 Bytes	Code used to identify the SynComm command/response.
SynError (SE)	1 Byte	Error code returned by Flyer head.
Wait for End of mark (WE)	1 Byte	Used by Mark command. Instructs Flyer to wait until the end of the mark before responding to the command packet.

The SynComm command packet sent by the client structure includes the Modbus header, the function code, the SynHeader and any data that is required for the command:

No. of Bytes:	7							1	4			0 to 248	
Description:	MBAP Header							FC	SynHeader			Data	
Fields:	TI	PI	LN	UI					SC	SE	WE		
Typ. Value (in hex):	00	00	00	00	XX	XX	00	43*	XX	XX	00	XX	XX XX XX ...

Flyer/Fenix Flyer responds to the SynComm command packet by copying back the Modbus header (with an updated Length value), the function code, the SynHeader (with the appropriate SynError value) and any data that is expected in the response:

No. of Bytes:	7							1	4				0 to 248
Description:	MBAP Header							FC	SynHeader				Data
Fields:	TI	PI	LN	UI					SC	SE	WE		
Typ. Value (in hex):	00	00	00	00	XX	XX	00	43*	XX	XX	XX	XX	XX XX XX ...

*User-Defined (within range of 0x41–0x48, 0x64–0x6E; 0x43 by default): determined by the 'Modbus User Function' property value set in the Flyer head.

User-Defined Modbus/IP Flyer Marking Head Functions:

The following functions are currently available via Modbus/IP for FH Flyer marking head control:

Marking Commands (page 37):

- Load File Load a file from Flyer Filestore into RAM for marking
- Load Network File Load a file from network file share into RAM for marking
- Get Current File Returns the name of the file currently loaded into RAM
- Get Property Value Returns the current value for a specific Object or Mark property
- Set Property Value Sets a new value for a specific Object or Mark property
- Mark File Begin marking the file currently loaded into RAM
- Mark Status Returns the current mark status
- Abort Mark Aborts marking

Filestore Management Commands (page 49):

- Make Directory Create a new directory in the Flyer Filestore
- Delete File/Directory Delete a file or directory from the Flyer Filestore
- Get File List Retrieve a list of files and directories in the Flyer Filestore
- Get Filestore Usage Returns Used and Available space in the Flyer Filestore
- Rename File Rename a file in the Flyer Filestore
- Copy File Copy a file residing in the Flyer Filestore to another Filestore location
- Erase Filestore Erases all files from the Flyer Filestore
- Update Network Mount Refreshes the network file share connection

Date/Time Management Commands (page 58):

- Get UTC Time Retrieves marking head UTC time
- Get Local Time Retrieves marking head local time
- Set UTC Time Sets marking head UTC time
- Set Local Time Sets marking head local time
- Get DST String Retrieves marking head DST information
- Set DST String Sets marking head DST information

Flyer Head Management Commands (page 66):

- Get Head Status Returns marking head type, marking status, stand-alone status, and network file share status
- Get Head Temperature Returns Flyer front/rear temperatures and over temperature condition
- Get Head Uptime Returns time in seconds since Flyer last powered on
- Get System Parameter Returns the value of the specified system parameter
- Set System Parameter Sets the value of the specified system parameter
- Reboot Marking Head Reboots the Flyer marking head
- *Upgrade Firmware Upgrade Flyer Firmware

* Uses two commands: Begin Firmware Upgrade and Download Firmware Packet



I/O Management Commands (page 76):

- **Read Flyer I/O Returns Flyer input and output bit status
- **Read Flyer Inputs Reads (gets) Flyer input bit status
- **Write Flyer Outputs Writes (sets) Flyer output bit status
- Set Wait Digital Waits and returns when bits and bit mask match input state or timeout occurs.

** using standard Modbus commands. See pages 76, 78, and 80.



Marking Commands

Load File: Loads a file from the Flyer Filestore into Flyer RAM for marking.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	01	00	00	Filename

The FileName string must:

- contain the hexadecimal byte representation of the filename
- be prefixed by a forward slash (“/”)
- contain the full path and extension, such as “/MyFolder/File1.mkh”
- be terminated with a NULL character (00)

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	01	XX	00	(None)

Where SE is the SynError byte:

- SE = 00 Success
- SE = 21 Error loading file
- SE = 30 Unable to complete request – marking in progress

**Since no data is sent, return packet length is always six bytes*

Example:

To load 'File1.mkh', send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 01 00 00

Data = ASCII hexadecimal byte representation of filename:

/ F i l e 1 . m k h [NULL]
2F 46 69 6C 65 31 2E 6D 6B 68 00

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 01 00 00 2F 46 69 6C 65 31 2E 6D 6B 68 00
Count = 17 bytes, so the length bytes = 00 11

So, to load 'File1.mkh', the packet would be:

00 00 00 00 00 11 00 43 00 01 00 00 2F 46 69 6C 65 31 2E 6D 6B 68 00
MBAP header FC SynHeader Data



Load Network File: Loads a file from a network file share into Flyer RAM for marking.

Command (from Client):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	0C	00	00	FileName

The FileName string must:

- contain the hexadecimal byte representation of the filename
- be prefixed by a forward slash (“/”)
- contain the full path and extension, such as “/MyShare/MyFile.mkh”
- be terminated with a NULL character (00)

Response (from Flyer):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	0C	XX	00	(None)

Where SE is the SynError byte:

- SE = 00 Success
- SE = 21 Error loading file
- SE = 30 Unable to complete request – marking in progress

**Since no data is sent, return packet length is always six bytes*

Example:

To load ‘/MyShare/MyFile.mkh’, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 0C 00 00

Data = ASCII hexadecimal byte representation of filename:

```

/ M y S h a r e / M y F i l e . m k h NULL
2F 4D 79 53 68 61 72 65 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00

```

The length bytes (‘LN LN’) in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 0C 00 00 2F 4D 79 53 68 61 72 65 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00

Count = 26 bytes, so the length bytes = 00 1A

So, to load ‘/MyShare/MyFile.mkh’ from the network, the packet would be:

00 00 00 00 00 1A 00 43 00 0C 00 00 2F 4D 79 53 68 61 72 65 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00

MBAP header FC SynHeader Data



Get Current File: Returns the name of the file currently loaded into Flyer RAM.

Command (from Client):

MBAP Header						FC	SynHeader				Data	
TI	PI		LN*	UI			SC	SE	WE			
00	00	00	00	00	06	00	43	00	05	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header						FC	SynHeader				Data	
TI	PI		LN	UI			SC	SE	WE			
00	00	00	00	XX	XX	00	43	00	05	XX	00	Filename (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the file name

SE = 22 No file loaded – NO DATA IS SENT

SE = 30 Unable to complete request – marking in progress – NO DATA IS SENT

The FileName string:

- contains the hexadecimal byte representation of the filename
- is prefixed by a forward slash (“/”)
- contains the full path and extension, such as “/filestore/myfile.mkh\0” or “/network/myfile.mkh\0”
- is terminated with a NULL character (00)

Example:

To retrieve the file currently loaded in the Flyer head for marking, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 05 00 00

Data: None

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 05 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 05 00 00

MBAP header FC SynHeader

If Flyer currently has “/filestore/myfile.mkh” loaded into RAM and is not marking, it responds with:

00 00 00 00 00 1C 00 43 00 05 00 00 2F 66 69 6C 65 73 74 6F 72 65 2F 6D 79 66 69 6C 65 2E 6D 6B 68 00

MBAP header FC SynHeader

Data

Where:

Data = ASCII hexadecimal byte representation of filename:

2F 66 69 6C 65 73 74 6F 72 65 2F 6D 79 66 69 6C 65 2E 6D 6B 68 00
/ f i l e s t o r e / m y f i l e . m k h NULL



Get Property Value: Returns the current Property Value for the mark file currently loaded in Flyer RAM, given the *Object Name* and *Property Name* of the file object. Valid object and property names for a given mark file can be obtained from WinMark Pro (from the *File* menu, click *Print All Properties*).

Command (from Client):

MBAP Header				FC	SynHeader			Data				
TI	PI	LN	UI		SC	SE	WE					
00	00	00	00	XX	XX	00	43	00	07	00	00	ObjectName, PropertyName

The ObjectName and PropertyName strings must:

- contain the hexadecimal byte representation of the object and property names
- each be terminated with a NULL character (00)

Response (from Flyer):

MBAP Header				FC	SynHeader			Data				
TI	PI	LN	UI		SC	SE	WE					
00	00	00	00	XX	XX	00	43	00	07	XX	00	PropertyValue (or None)

Where SE is the SynError byte:

- SE = 00 Success
- SE = 21 Error loading file – NO DATA IS SENT
- SE = 22 No file loaded – NO DATA IS SENT
- SE = 23 Invalid *Object Name* and/or *Property Name* – NO DATA IS SENT
- SE = 30 Unable to complete request – marking in progress – NO DATA IS SENT
- SE = 31 Flyer not in Stand-alone mode – NO DATA IS SENT

Example:

A file currently loaded in Flyer RAM contains an object named Text1, with a TextCaption value of 'MyValue'. To retrieve the Text1 caption, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 07 00 00

Data = ASCII hexadecimal byte representation of object and property names, with each value terminated with NULL characters:

```
T e x t 1 NULL T e x t C a p t i o n NULL
54 65 78 74 31 00 54 65 78 74 43 61 70 74 69 6F 6E 00
```

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 07 00 00 2F 54 65 78 74 31 00 54 65 78 74 43 61 70 74 69 6F 6E 00
Count = 25 bytes, so the length bytes = 00 19

So, to get the Text1 TextCaption value, the packet would be:

00 00 00 00 00 19 00 43 00 07 00 00 2F 54 65 78 74 31 00 54 65 78 74 43 61 70 74 69 6F 6E 00
MBAP header FC SynHeader Data

If Flyer can return the 'MyValue' caption without error, the return packet is:

00 00 00 00 00 1C 00 43 00 07 00 00 4D 79 56 61 6C 75 65 00
MBAP header FC SynHeader Data

Where:

Data = ASCII hexadecimal byte representation of the text caption:

```
4D 79 56 61 6C 75 65 00
M y V a l u e NULL
```




Set Property Value: Sets a new Property Value for a mark file loaded in Flyer RAM given the *Object Name*, *Property Name*, and *Property Value*. Valid object and property names for a given mark file can be obtained from WinMark Pro (from the *File* menu, click *Print All Properties*).

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN	UI		SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	06	00	00	ObjectName, PropertyName, PropertyValue

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	06	XX	00	(None)

*Since no data is sent, command packet length is always six bytes

Where SE is the SynError byte:

- SE = 00 Success
- SE = 22 No file loaded
- SE = 25 Invalid *Object Name*, *Property Name*, or *Property Value*
- SE = 30 Unable to complete request – marking in progress

Example:

A file currently loaded in Flyer RAM contains an object named Text1, with a TextCaption value of 'MyText'. To change the Text1 caption to 'NewText', send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 06 00 00

Data = ASCII hexadecimal byte representation of object and property names, with each value terminated with NULL characters:

T e x t 1 NULL T e x t C a p t i o n NULL N e w T e x t NULL
54 65 78 74 31 00 54 65 78 74 43 61 70 74 69 6F 6E 00 4E 65 77 54 65 78 74 00

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 06 00 00 54 65 78 74 31 00 54 65 78 74 43 61 70 74 69 6F 6E 00 4E 65 77 54 65 78 74 00

Count = 32 bytes, so the length bytes = 00 20

So, to change the Text1 TextCaption property to 'NewText', the packet would be:

00 00 00 00 00 20 00 43 00 06 00 00 54 65 78 74 31 00 54 65 78 74 43 61 70 74 69 6F 6E 00 4E 65 77 54 65 78 74 00

MBAP header FC SynHeader

Data



Mark File: Begin marking the file currently loaded in Flyer RAM. There are two options for this command:

- Don't Wait For End-of-Mark: The Flyer head immediately sends the **Mark File** response packet and returns the Mark Count for the file. Polling via the **Mark Status** command is required to determine when the mark is completed.
- Wait For End-of-Mark: The Flyer head does not send the **Mark File** response packet until marking is complete. Ethernet timeout values should be adjusted according when using this option since mark times may vary depending on the nature of the marking.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	20	00	XX	(None)

**Since no data is sent, command packet length is always six bytes*

Where is the Wait for End of mark byte:

WE = 00 Respond immediately - don't wait for end of mark

WE = 01 Wait for end of mark

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN	UI		SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	20	XX	XX	Status (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the mark status (see below)

SE = 22 No file loaded – NO DATA IS SENT

SE = 30 Unable to complete request – marking in progress – NO DATA IS SENT

SE = 31 Flyer not in Stand-alone mode – NO DATA IS SENT

The content and formatting of the status data returned on a successful **Mark File** response depends on the value of the WE byte:

WE = 00 Status is a four byte Long Integer value indicating the number of pieces to mark

WE = 01 Status is a 28 byte value indicating:

Field	Length	Description
MarkStatus	2 Bytes	Decimal value indicating the Mark Status: Idle = 0 Marking = 1 Aborted = 2
Reserved	2 Bytes	
EOMResponse	4 Bytes	Binary value containing status information regarding the state of the Flyer head (see Appendix B)
CurrentPiece	4 Bytes	Decimal value indicating the current piece marked
Ticks	4 Bytes	Number of total ticks for entire mark (100 ticks = 1 second)
MarkCount	4 Bytes	Total number of pieces to be marked
TickMin	4 Bytes	Minimum tick count per piece (100 ticks = 1 second)
TickMax	4 Bytes	Maximum tick count per piece (100 ticks = 1 second)

[Examples on next pages]



Mark File (cont.):

Example 1 - Wait for end of mark:

To command Flyer to mark the file currently loaded in Flyer RAM and respond after the mark session is complete, send:

MBAP Header = 00 00 00 00 LN LN 00
Function Code = 43
SynHeader = 00 20 00 01
Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 20 00 01
Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:
00 00 00 00 00 06 00 43 00 20 00 01
MBAP header FC SynHeader

If the Flyer encounters an error executing the command, it responds with:
00 00 00 00 00 06 00 43 00 20 SE 01
MBAP header FC SynHeader

Where SE is the SynError byte:
SE = 22 No file loaded
SE = 30 Unable to complete request – marking in progress
SE = 31 Flyer not in Stand-alone mode

If the Flyer marks the file without error, it responds with:
00 00 00 00 00 22 00 43 00 20 00 01 00 00 03 04 00 00 00 00 00 00 02 0C 00 00 01 10 00 00 02 0C 00 00 00 FF 00 00 01 1C
MBAP header FC SynHeader Data

Where:
Data = mark statistics:
00 00 03 04 00 00 00 00 00 00 02 0C 00 00 01 10 00 00 02 0C 00 00 00 FF 00 00 01 1C
MkSt Rsvd EOM Resp CurrPc Ticks MarkCount TickMin TickMax

MarkStatus = 00 00 = head is idle
EOMResponse = 00 00 00 00 = no faults detected
CurrentPiece = 00 00 02 0C = piece # 524 marked
Ticks = 00 00 0F 10 = cycle time of last piece marked is 272 ticks = 2.72 seconds
MarkCount = 00 00 02 0C = 524 pieces are to be marked in this batch
TickMin = 00 00 00 FF = minimum cycle time for this batch so far is 255 ticks = 2.55 seconds
TickMax = 00 00 01 1C = maximum cycle time for this batch so far is 284 ticks = 2.84 seconds



Mark File (cont.):

Example 2 – Don't Wait for end of mark:

To command Flyer to mark the file currently loaded in Flyer RAM and respond immediately, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 20 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 20 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 20 00 00

MBAP header FC SynHeader

If the Flyer encounters an error executing the command, it responds with:

00 00 00 00 00 06 00 43 00 20 SE 00

MBAP header FC SynHeader

Where SE is the SynError byte:

SE = 22 No file loaded

SE = 30 Unable to complete request – marking in progress

SE = 31 Flyer not in Stand-alone mode

If the Flyer begins the mark session without error, it responds with:

00 00 00 00 00 06 00 43 00 20 00 00 00 00 10 00

MBAP header FC SynHeader Data

Where:

Data = MarkCount = 00 00 10 00 = 4096 pieces are to be marked in this batch



Mark Status: Returns the current marking status and statistics.

Command (from Client):

MBAP Header				FC	SynHeader			Data	
TI	PI	LN*	UI		SC	SE	WE		
00	00	00	00	06	00	25	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header				FC	SynHeader			Data				
TI	PI	LN	UI		SC	SE	WE					
00	00	00	00	XX	XX	00	43	00	25	XX	00	Status (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the mark status (see below)

SE = 31 Flyer not in Stand-alone mode – NO DATA IS SENT

Where Status contains current mark statistics:

Field	Length	Description
MarkStatus	2 Bytes	Decimal value indicating the Mark Status: Idle = 0 Marking = 1 Aborted = 2
Reserved	2 Bytes	
EOMResponse	4 Bytes	Binary value containing status information regarding the state of the Flyer head (see Appendix B)
CurrentPiece	4 Bytes	Decimal value indicating the current piece marked
Ticks	4 Bytes	Number of total ticks for entire mark (100 ticks = 1 second)
MarkCount	4 Bytes	Total number of pieces to be marked
TickMin	4 Bytes	Minimum tick count per piece (100 ticks = 1 second)
TickMax	4 Bytes	Maximum tick count per piece (100 ticks = 1 second)

[Example on next page]



Mark Status (cont.):

Example:

To command Flyer to report current mark status, send:
MBAP Header = 00 00 00 LN LN 06 00
Function Code = 43
SynHeader = 00 25 00 00
Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:
00 00 00 00 LN LN 00 43 00 25 00 00
Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:
00 00 00 00 00 06 00 43 00 25 00 00
MBAP header FC SynHeader

If the Flyer encounters an error executing the command, it responds with:
00 00 00 00 00 06 00 43 00 25 SE 01
MBAP header FC SynHeader

Where SE is the SynError byte:
SE = 31 Flyer not in Stand-alone mode

If the Flyer marks the file without error, it responds with:
00 00 00 00 00 22 00 43 00 25 00 00 00 00 03 04 00 00 00 00 00 00 02 0C 00 00 01 10 00 00 02 0C 00 00 00 FF 00 00 01 1C
MBAP header FC SynHeader Data

Where:

Data = mark statistics:
00 00 03 04 00 00 00 00 00 00 02 0C 00 00 01 10 00 00 02 0C 00 00 00 FF 00 00 01 1C
MkSt Rsvd EOM Resp CurrPc Ticks MarkCount TickMin TickMax

MarkStatus = 00 00 = head is idle
EOMResponse = 00 00 00 00 = no faults detected
CurrentPiece = 00 00 02 0C = piece # 524 marked
Ticks = 00 00 0F 10 = cycle time of last piece marked is 272 ticks = 2.72 seconds
MarkCount = 00 00 02 0C = 524 pieces are to be marked in this batch
TickMin = 00 00 00 FF = minimum cycle time for this batch so far is 255 ticks = 2.55 seconds
TickMax = 00 00 01 1C = maximum cycle time for this batch so far is 284 ticks = 2.84 seconds



Abort Mark: Abort the current mark session.

Command (from Client):

MBAP Header						FC	SynHeader				Data	
TI	PI		LN*	UI			SC	SE	WE			
00	00	00	00	00	06	00	43	00	21	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header						FC	SynHeader				Data	
TI	PI		LN	UI			SC	SE	WE			
00	00	00	00	XX	XX	00	43	00	21	XX	00	Status (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the mark status (see below)

SE = 31 Flyer not in Stand-alone mode – NO DATA IS SENT

Where Status contains current mark statistics:

Field	Length	Description
MarkStatus	2 Bytes	Decimal value indicating the Mark Status: Idle = 0 Marking = 1 Aborted = 2
Reserved	2 Bytes	
EOMResponse	4 Bytes	Binary value containing status information regarding the state of the Flyer head (see Appendix B)
CurrentPiece	4 Bytes	Decimal value indicating the current piece marked
Ticks	4 Bytes	Number of total ticks for entire mark (100 ticks = 1 second)
MarkCount	4 Bytes	Total number of pieces to be marked
TickMin	4 Bytes	Minimum tick count per piece (100 ticks = 1 second)
TickMax	4 Bytes	Maximum tick count per piece (100 ticks = 1 second)

[Example on next page]



Abort Mark (cont.):

Example:

To command Flyer to abort the current mark session, send:

MBAP Header = 00 00 00 LN LN 06 00

Function Code = 43

SynHeader = 00 21 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 21 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 21 00 00

MBAP header FC SynHeader

If the Flyer encounters an error executing the command, it responds with:

00 00 00 00 00 06 00 43 00 21 SE 01

MBAP header FC SynHeader

Where SE is the SynError byte:

SE = 31 Flyer not in Stand-alone mode

If the Flyer aborts the mark session without error, it responds with:

00 00 00 00 00 22 00 43 00 21 00 00 00 00 03 04 00 00 00 00 00 00 02 0C 00 00 01 10 00 00 02 0C 00 00 00 FF 00 00 01 1C

MBAP header FC SynHeader

Data

Where:

Data = mark statistics:

00 00 03 04 00 00 00 00 00 00 02 0C 00 00 01 10 00 00 02 0C 00 00 00 FF 00 00 01 1C

MkSt Rsvd EOM Resp CurrPc Ticks MarkCount TickMin TickMax

MarkStatus = 00 00 = head is idle

EOMResponse = 00 00 00 00 = no faults detected

CurrentPiece = 00 00 02 0C = piece # 524 marked

Ticks = 00 00 0F 10 = cycle time of last piece marked is 272 ticks = 2.72 seconds

MarkCount = 00 00 02 0C = 524 pieces are to be marked in this batch

TickMin = 00 00 00 FF = minimum cycle time for this batch so far is 255 ticks = 2.55 seconds

TickMax = 00 00 01 1C = maximum cycle time for this batch so far is 284 ticks = 2.84 seconds



Filestore Management Commands

Make Directory: Create a new directory in the Flyer Filestore.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	0A	00	00	DirectoryName

The DirectoryName string must:

- contain the hexadecimal byte representation of the filename
- be prefixed by a forward slash (“/”)
- reference the Flyer Filestore root directory, such as “/filestore/MyDir”
- be terminated with a NULL character (00)

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	0A	XX	00	(None)

Where SE is the SynError byte:

- SE = 00 Success
- SE = 2A Invalid directory name or directory already exists
- SE = 30 Unable to complete request – marking in progress

**Since no data is sent, return packet length is always six bytes*

Example:

To create a directory called ‘filestore/MyDir’, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 0A 00 00

Data = ASCII hexadecimal byte representation of filename:

/ f i l e s t o r e / M y D i r NULL
2F 66 69 6C 65 73 74 6F 72 65 2F 4D 79 44 69 72 00

The length bytes (‘LN LN’) in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 0A 00 00 2F 66 69 6C 65 73 74 6F 72 65 2F 4D 79 44 69 72 00
Count = 23 bytes, so the length bytes = 00 17

So, to create the directory ‘filestore/MyDir’, the packet would be:

00 00 00 00 00 17 00 43 00 0A 00 00 2F 66 69 6C 65 73 74 6F 72 65 2F 4D 79 44 69 72 00

MBAP header

FC SynHeader

Data



Delete File/Directory: Delete a file or directory from the Flyer Filestore.

Important Note: Because only empty directories can be deleted, you must first delete all files within the directory.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	02	00	00	File/Directory Name

The File/Directory Name string:

- must contain the hexadecimal byte representation of the file or directory name
- must be prefixed by a forward slash (“/”)
- is referenced from the Flyer Filestore root directory, but does not include ‘filestore’ in the path, such as “/MyFile.mkh” or “/MyDir”
- must be terminated with a NULL character (00)

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	02	XX	00	(None)

Where SE is the SynError byte:

- SE = 00 Success
- SE = 28 Invalid file or directory name or directory is not empty
- SE = 30 Unable to complete request – marking in progress

**Since no data is sent, return packet length is always six bytes*

Example:

To delete a directory in the Filestore called ‘/MyDir’, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 02 00 00

Data = ASCII hexadecimal byte representation of filename:

/ M Y D i r NULL
2F 4D 79 44 69 72 00

The length bytes (‘LN LN’) in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 02 00 00 2F 4D 79 44 69 72 00

Count = 13 bytes, so the length bytes = 00 0D

So, to delete the directory ‘/MyDir’, the packet would be:

00 00 00 00 00 0D 00 43 00 02 00 00 2F 4D 79 44 69 72 00

MBAP header FC SynHeader Data



Get File List: Requests a list of files and directories currently contained in the Flyer Filestore.

Important Note: Because Modbus is a Command /Response protocol, there is a specific Command/Response sequence necessary to ensure the receipt of all packets. Data packets are tracked using the Transaction Identifier field in the Modbus MBAP Header, along with Packet Header information returned by the Flyer head that provides the current packet and total number of packets to be transferred. When the current packet equals total packets, the transaction is complete.

Command (from Client):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN*	UI		SC		SE	WE			
XX	XX	00	00	00	06	00	43	00	03	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Where TI value indicates the packet number commanded by the Client, starting with 00 00 and incrementing up to total number of packets to be returned by the Flyer head.

Response (from Flyer):

MBAP Header							FC	SynHeader			Packet Header		Data			
TI	PI		LN	UI		SC		SE	WE	CP	TP					
XX	XX	00	00	XX	XX	00	43	00	03	XX	00	XX	XX	XX	XX	FileList (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the filelist

SE = 30 Unable to complete request – marking in progress – NO DATA IS SENT

The Flyer return data includes a Packet Header:

Fields	Length	Description
CurrentPacket (CP)	2 Bytes	Current packet sent by Flyer head
TotalPackets (TP)	2 Bytes	Total packets to be sent by Flyer head

The FileList data:

- contains the hexadecimal byte representation of the file and directory names in the Flyer Filestore
- is sent as a collection of null-terminated strings
- directory folders are prefixed with a double backslash “//” and do not include the /filestore reference, such as “//MyDir”
- filenames are referenced from the Flyer Filestore root directory, but do not include ‘filestore’ in the path, such as “/MyFile.mkh” or “/MyDir/MyFile.mkh”

For instance, if the Flyer Filestore contains two mark files - one contained in a directory (“MyDir/MyFile.mkh”), the other contained in the root Filestore folder (“MyOtherFile.mkh”), Flyer would return just a single packet with the FileList string being:

“//MyDir[NULL]/MyDir/MyFile.mkh[NULL]/MyOtherFile.mkh[NULL]”

[Example on next page]



Get File List (cont):

Example:

To command Flyer to report the contents of the Filestore, send:

MBAP Header = 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 03 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 03 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the command packet would be:

00 00 00 00 00 06 00 43 00 03 00 00

MBAP header FC SynHeader

If the Flyer Filestore contains a single directory ("MyDir") holding a single mark file, "MyFile.mkh", and another file, "MyOtherFile.mkh", located in the root directory of the Filestore, and can return the FileList without error, the return packet is:

00 00 00 00 00 35* 00 43 00 03 00 00 00 00 00 01 &
MBAP header FC SynHeader PacketHdr

2F 2F 4D 79 44 69 72 00 2F 4D 79 44 69 72 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00 &
Data: String1 Data: String2

2F 4D 79 4F 74 68 65 72 46 69 6C 65 2E 6D 6B 68 00
Data: String3

Where:

*There are 00 35 bytes (53 bytes decimal) of data following the LN bytes in the MBAP header

The data field contains the NULL terminated directory and filename strings from the Flyer Filestore:

String1:

2F 2F 4D 79 44 69 72 00
/ / M Y D i r NULL

String2:

2F 4D 79 44 69 72 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00
/ M Y D i r / M Y F i l e . m k h NULL

String3:

2F 4D 79 4F 74 68 65 72 46 69 6C 65 2E 6D 6B 68 00
/ M Y O t h e r F i l e . m k h NULL



Get Filestore Usage: Returns the amount of used and available space (in bytes) in the Flyer Filestore.

Command (from Client):

MBAP Header						FC	SynHeader				Data	
TI	PI		LN*	UI			SC	SE	WE			
00	00	00	00	00	06	00	43	00	04	00	00	(None)

*Since no data is sent, command packet length is always six bytes

Response (from Flyer):

MBAP Header						FC	SynHeader				Data	
TI	PI		LN	UI			SC	SE	WE			
00	00	00	00	XX	XX	00	43	00	04	XX	00	FileSpace (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the file name

SE = 24 Failed to retrieve Filestore information – NO DATA IS SENT

The FileSpace value contains two four-byte values indicating memory usage and available memory.

Example:

To query Flyer for the Filestore memory usage, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 04 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 04 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 04 00 00

MBAP header FC SynHeader

If the Flyer encounters an error executing the command, it responds with:

00 00 00 00 00 06 00 43 00 04 SE 01

MBAP header FC SynHeader

Where SE is the SynError byte:

SE = 24 Failed to retrieve Filestore information

If the Flyer returns the Filestore memory information without error, it responds with:

00 00 00 00 00 0E 00 43 00 04 00 00 00 0A AE 60 00 76 39 A0

MBAP header FC SynHeader Data

Where:

Data = memory usage and availability:

00 0A AE 60 00 76 39 A0

MemUsed MemAvail

MemUsed = 00 0A AE 60 = 700,000 bytes used; MemAvail = 00 76 39 A0 = 7,748,000 bytes available



Rename File: Rename a file residing in the Flyer Filestore.

Command (from Client):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	09	00	00	SourceName, DestName

The SourceName and DestName strings:

- must contain the hexadecimal byte representation of the file source and destination paths
- must be prefixed by a forward slash (“/”)
- are referenced from the Flyer Filestore root directory, but do not include ‘filestore’ in the path, such as ‘/MyFile.mkh’ and ‘/MyDir/MyFile.mkh’
- must be terminated with NULL characters (00)

Response (from Flyer):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	09	XX	00	(None)

**Since no data is sent, command packet length is always six bytes*

Where SE is the SynError byte:

- SE = 00 Success
- SE = 29 Invalid SourceName and/or DestName
- SE = 30 Unable to complete request, marking in progress

Example:

To rename ‘/MyFile.mkh’ to ‘/MyNewFile.mkh’, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 09 00 00

Data = ASCII hexadecimal byte representation of source and destination filenames:

/ M y F i l e . m k h NULL / M y N e w F i l e . m k h NULL
2F 4D 79 46 69 6C 65 2E 6D 6B 68 00 2F 4D 79 4E 65 77 46 69 6C 65 2E 6D 6B 68 00

The length bytes (‘LN LN’) in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 09 00 00 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00 2F 4D 79 4E 65 77 46 69 6C 65 2E 6D 6B 68 00

Count = 33 bytes, so the length bytes = 21

So, to rename ‘/MyFile.mkh’ to ‘/MyNewFile.mkh’, the packet would be:

00 00 00 00 00 21 00 43 00 09 00 00 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00 2F 4D 79 4E 65 77 46 69 6C 65 2E 6D 6B 68 00

MBAP header FC SynHeader

Data



Copy File: Copy a file residing in the Flyer Filestore to another Filestore location with the same or new filename or copy to current location as a new filename.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	08	00	00	SourceName, DestName

The SourceName and DestName strings:

- must contain the hexadecimal byte representation of the file source and destination paths
- must be prefixed by a forward slash (“/”)
- are referenced from the Filestore root directory, but do not include ‘/filestore’ in the path, such as ‘/MyFile.mkh’ and ‘/MyDir/MyFile.mkh’
- must be terminated with NULL characters (00)

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	08	XX	00	(None)

**Since no data is sent, command packet length is always six bytes*

Where SE is the SynError byte:

- SE = 00 Success
- SE = 29 Invalid SourceName and/or DestName
- SE = 30 Unable to complete request, marking in progress

Example:

To copy ‘/MyFile.mkh’ to ‘/MyNewFile.mkh’, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 08 00 00

Data = ASCII hexadecimal byte representation of source and destination filenames:

/ M y F i l e . m k h NULL / M y N e w F i l e . m k h NULL
2F 4D 79 46 69 6C 65 2E 6D 6B 68 00 2F 4D 79 4E 65 77 46 69 6C 65 2E 6D 6B 68 00

The length bytes (‘LN LN’) in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 08 00 00 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00 2F 4D 79 4E 65 77 46 69 6C 65 2E 6D 6B 68 00

Count = 33 bytes, so the length bytes = 21

So, to copy ‘/MyFile.mkh’ to ‘/MyNewFile.mkh’, the packet would be:

00 00 00 00 00 21 00 43 00 08 00 00 2F 4D 79 46 69 6C 65 2E 6D 6B 68 00 2F 4D 79 4E 65 77 46 69 6C 65 2E 6D 6B 68 00

MBAP header FC SynHeader

Data



Erase Filestore: Erases all files from the Flyer Filestore and *reboots the marking head.*

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	0B	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	0B	XX	00	(None)

**Since no data is sent, command packet length is always six bytes*

Where SE is the SynError byte:

- SE = 00 Success
- SE = 2B Unable to erase Flyer Filestore
- SE = 30 Unable to complete request – marking in progress

Example:

To erase the Filestore and reboot the Flyer head, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 0B 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 0B 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 0B 00 00

MBAP header FC SynHeader



Update Network Mount: Refreshes the network file share connection.

Command (from Client):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	0D	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	0D	XX	00	(None)

**Since no data is sent, command packet length is always six bytes*

Where SE is the SynError byte:

- SE = 00 Success
- SE = 2C Unable to connect to network file share
- SE = 30 Unable to complete request – marking in progress

Example:

To refresh the Flyer's network file share connection, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 0D 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 0D 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 0D 00 00

MBAP header FC SynHeader



Date/Time Management Commands

Get Flyer Time and Date: Retrieves the Flyer head's time and date information, either as Local time or Coordinated Universal Time (UTC).

Command (from Client):

MBAP Header				FC	SynHeader			Data
TI	PI	LN*	UI		SC	SE	WE	
00	00	00	00	43	00	XX	00	(None)

**Since no data is sent, command packet length is always six bytes*

Where SC is the SynCode byte, determining whether Local or UTC time should be returned:

SC = **40** Return UTC time

SC = **41** Return Local time

Response (from Flyer):

MBAP Header				FC	SynHeader			Data
TI	PI	LN	UI		SC	SE	WE	
00	00	00	00	43	00	XX	XX	DateTime (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the DateTime values (see below)

SE = 10 Error retrieving UTC time – NO DATA IS SENT

SE = 11 Error retrieving Local time – NO DATA IS SENT

Where SC is the SynCode byte, determining whether Local or UTC time should be returned:

SC = 40 Return UTC time

SC = 41 Return Local time

Where DateTime contains:

Field	Length	Description
Year	2 Bytes	Four-digit year
Month	2 Bytes	1–12, where January = 1, February = 2, etc.
DayOfWeek	2 Bytes	0–6, where Sunday = 0, Monday = 1, etc.
Day	2 Bytes	1–31
Hour	2 Bytes	0–23
Minute	2 Bytes	0–59
Seconds	2 Bytes	0–59
Milliseconds	2 Bytes	Always zero (0)

[Example on next page]



Get Flyer Time and Date (cont):

Example:

To get Flyer's Local time and date information, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 41 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 41 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 41 00 00

MBAP header FC SynHeader

If the Flyer returns the time and date information without error, it responds with:

00 00 00 00 00 16 00 43 00 41 00 00 07 DB 00 04 00 03 00 1A 00 09 00 2C 00 36 00 00

MBAP header FC SynHeader Data

Where:

Data = date/time information:

07 DB 00 04 00 03 00 1A 00 09 00 2C 00 36 00 00
Year Month DofW Day Hour Min Sec MSec

Year = 07 DB = 2011

Month = 00 04 = April

DofW = 00 03 = Wednesday

Day = 00 1A = 26th

Hour = 00 09 = 9AM

Min = 00 2C = 44

Sec = 00 36 = 54

MSec = 00 00 = millisecond count (always zero)

Resulting in a local date/time of Wednesday, April 26, 2011 at 9:44:54 AM.



Set Flyer Time and Date: Sets the Flyer head's time and date information, either as Local time or Coordinated Universal Time (UTC).

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	XX	00	00	DateTime

**Since no data is sent, command packet length is always six bytes*

Where SC is the SynCode byte, determining whether the time values are Local or UTC time:

SC = 42 Set UTC time

SC = 43 Set Local time

Where DateTime contains:

Field	Length	Description
Year	2 Bytes	Four digit year
Month	2 Bytes	1-12 where January = 1, February = 2, etc.
DayOfWeek	2 Bytes	0-6 where Sunday = 0, Monday = 1, etc.
Day	2 Bytes	1-31
Hour	2 Bytes	0-23
Minute	2 Bytes	0-59
Seconds	2 Bytes	0-59
Milliseconds	2 Bytes	Always zero (0)

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	XX	XX	00	(None)

Where SE is the SynError byte:

SE = 00 Success

SE = 12 Error setting UTC time

SE = 13 Error setting Local time

Where SC is the SynCode byte, determining whether the time values are Local or UTC time:

SC = 42 Set UTC time

SC = 43 Set Local time

[Example on next page]



Set Flyer Time and Date (cont):

Example:

To set Flyer's Local time and date settings to Wednesday, April 26, 2011 at 9:44:54 AM, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 43 00 00

Data =

Year = 2011 = 07 DB

Month = April = 00 04

DofW = Wednesday = 00 03

Day = 26th = 00 1A

Hour = 9AM = 00 09

Min = 44 = 00 2C

Sec = 54 = 00 36

MSec = millisecond count (always zero) = 00 00

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 43 00 00 07 DB 00 04 00 03 00 1A 00 09 00 2C 00 36 00 00

Count = 22 bytes, so the LN bytes = 00 16

So, the packet would be:

00 00 00 00 00 16 00 43 00 43 00 00 07 DB 00 04 00 03 00 1A 00 09 00 2C 00 36 00 00

MBAP header

FC SynHeader

Data



Get DST String: Retrieves the Flyer head's Daylight Savings Time (DST) and Standard Time (ST) definition data.

Command (from Client):

MBAP Header						FC	SynHeader			Data
TI	PI		LN*	UI	SC		SE	WE		
00	00	00	00	00	06	00	44	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header						FC	SynHeader			Data		
TI	PI		LN	UI	SC		SE	WE				
00	00	00	00	XX	XX	00	43	00	44	XX	00	DSTString (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the DSTString (see below)

SE = 14 Unable to retrieve DST information – NO DATA IS SENT

The DSTString value received from Flyer:

- contains the hexadecimal byte representation of the DSTString information
- is terminated with a NULL character (00)
- contains three comma delimited values:
 - indicating local DST and standard time zone and their bias values from UTC time
 - indicating the date and time when local time changes from ST to DST
 - indicating the date and time when local time changes from DST to ST

For example, the DSTString value might look like: "PST+8PDT+7,M3.2.0/03:00,M11.1.0/02:00"

Where the three values indicate:

PST+8PDT+7 the local standard and daylight time zone and bias - in this case:

PST Pacific Standard Time

+8 the offset required to convert local standard time to UTC time (8 hours)

PDT Pacific Daylight Time

+7 the offset required to convert local daylight time to UTC time (7 hours)

M3.2.0/03:00 the changeover from standard time to daylight savings time occurs - in this case:

M3 month 3 (March)

.2 week 2

.0 day 0 (Sunday)

/03:00 3:00 AM

M11.1.0/02:00 when the changeover from daylight savings time to standard time occurs - in this case:

M11 month 11 (November)

.1 week 1

.0 day 0 (Sunday)

/02:00 2:00 AM

[Example on next page]



Get DST String (cont):

Example:

To get Flyer's Daylight Savings Time information, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 44 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 44 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 44 00 00

MBAP header FC SynHeader

If the Flyer contains a DSTString value of "PST+8PDT+7,M3.2.0/03:00,M11.1.0/02:00", and returns without error, it responds with:

00 00 00 00 00 2C 00 43 00 44 00 00 50 53 54 2B 38 50 44 54 2B 37 2C &

MBAP header FC SynHeader Data

4D 33 2E 32 2E 30 2F 30 33 3A 30 30 2C &

Data

4D 31 31 2E 31 2E 30 2F 30 32 3A 30 30 00

Data

Where:

Data = date/time information string:

50 53 54 2B 38 50 44 54 2B 37 2C

P S T + 8 P D T + 7 ,

4D 33 2E 32 2E 30 2F 30 33 3A 30 30 2C

M 3 . 2 . 0 / 0 3 : 0 0 ,

4D 31 31 2E 31 2E 30 2F 30 32 3A 30 30 00

M 1 1 . 1 . 0 / 0 2 : 0 0 NULL

Note: See the value definitions above.



Set DST String: Sets the Flyer head's Daylight Savings Time (DST) and Standard Time (ST) definition data.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	45	00	00	DSTString (None)

The DSTString value sent to Flyer must:

- contain the hexadecimal byte representation of the DSTString information
- be terminated with a NULL character (00)
- contain three comma delimited values:
 - indicating local DST and standard time zone and their bias values from UTC time
 - indicating the date and time when local time changes from ST to DST
 - indicating the date and time when local time changes from DST to ST

For example, the DSTString value** might look like: "PST+8PDT+7,M3.2.0/03:00,M11.1.0/02:00"

Where the three values indicate:

PST+8PDT+7 the local standard and daylight time zone and bias - in this case:

- PST Pacific Standard Time
- +8 the offset required to convert local standard time to UTC time (8 hours)
- PDT Pacific Daylight Time
- +7 the offset required to convert local daylight time to UTC time (7 hours)

M3.2.0/03:00 the changeover from standard time to daylight savings time occurs - in this case:

- M3 month 3 (March)
- .2 week 2
- .0 day 0 (Sunday)
- /03:00 3:00 AM

M11.1.0/02:00 when the changeover from daylight savings time to standard time occurs - in this case:

- M11 month 11 (November)
- .1 week 1
- .0 day 0 (Sunday)
- /02:00 2:00 AM

** the correct DST string formatting may be derived by using WinMark's Set Time and Date interface (in WinMark, select Tools/Set Time and Date) to set the Flyer time and date settings, then reading the DST string out of the head using the Get DST String command.

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	45	00	00	(None)

*Since no data is sent, command packet length is always six bytes

Where SE is the SynError byte:

- SE = 00 Success
- SE = 15 Unable to set DST information



Set DST String (cont):

Example:

To get Flyer's Daylight Savings Time information, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 45 00 00

Data = ASCII hexadecimal byte representation of the DST String:

50 53 54 2B 38 50 44 54 2B 37 2C
P S T + 8 P D T + 7 ,

4D 33 2E 32 2E 30 2F 30 33 3A 30 30 2C
M 3 . 2 . 0 / 0 3 : 0 0 ,

4D 31 31 2E 31 2E 30 2F 30 32 3A 30 30 00
M 1 1 . 1 . 0 / 0 2 : 0 0 NULL

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 45 00 00 50 53 54 2B 38 50 44 54 2B 37 2C 4D 33 2E 32 2E 30 2F 30 & 33 3A 30 30 2C 4D 31 31 2E 31 2E 30 2F 30 32 3A 30 30 00

Count = 44 bytes, so the LN bytes = 00 2C

So, the packet would be:

00 00 00 00 00 2C 00 43 00 45 00 00 50 53 54 2B 38 50 44 54 2B 37 2C &
MBAP header FC SynHeader Data

4D 33 2E 32 2E 30 2F 30 33 3A 30 30 2C &
Data

4D 31 31 2E 31 2E 30 2F 30 32 3A 30 30 00
Data

Flyer Head Management Commands

Get Marking Head Status: Returns marking head type, marking status, Stand-alone status, and network file share status in a Head Status structure (sHeadStats)

Command (from Client):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	52	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN	UI		SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	52	XX	00	HeadStatus (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the head status (see below)

Where HeadStatus contains the current Flyer head status:

Field	Length	Description
MarkingHeadType (HT)	1 Byte	1 = Flyer
Marking (MK)	1 Byte	Current marking status where 0 = idle; 1 = marking
Standalone (SA)	1 Byte	Stand-alone mode enabled where 0 = No; 1 = Yes
NetworkShareConnected (SC)	1 Byte	Network share available where 0 = No; 1 = Yes

[Example on next page]



Get Marking Head Status (cont):

Example:

To command Flyer to report its status, send:

MBAP Header = 00 00 00 LN LN 06 00

Function Code = 43

SynHeader = 00 52 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 52 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 52 00 00

MBAP header FC SynHeader

If the Flyer is able to return the head status without error, it responds with:

00 00 00 00 00 0A 00 43 00 52 00 00 01 00 01 01

MBAP header FC SynHeader Data

Where:

Data = head status:

01 00 01 01

HT MK SA SC

HT = 01 = head type is Flyer

MK = 00 = head is not marking

SA = 01 = head is in Stand-alone mode

SC = 01 = network share is connected and available



Get Head Temperature: Returns the measured temperature and over-temperature status from Flyer's front and rear temperature sensors.

Command (from Client):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	50	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN	UI		SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	50	XX	00	TempInfo (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the head status (see below)

Where TempInfo contains Flyer's head temperature information:

Field	Length	Description
FrontTemp (FT)	4 Bytes	Floating point front sensor temperature in degrees Celsius
RearTemp (RT)	4 Bytes	Floating point rear sensor temperature in degrees Celsius
FrontOvertemp (FOT)	1 Byte	Overtemp fault at front temp sensor? (FALSE = 0; TRUE = 1)
RearOvertemp (ROT)	1 Byte	Overtemp fault at rear temp sensor? (FALSE = 0; TRUE = 1)

[Example on next page]



Get Head Temperature (cont):

Example:

To command Flyer to report its temperature status, send:

MBAP Header = 00 00 00 LN LN 06 00

Function Code = 43

SynHeader = 00 50 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 50 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 50 00 00

MBAP header FC SynHeader

If the Flyer is able to return the head temperature status without error, it responds with:

00 00 00 00 00 10 00 43 00 50 00 00 42 11 85 1F 41 F7 85 1F 00 00

MBAP header FC SynHeader Data

Where:

Data = head temperature status:

42 11 85 1F 41 F7 85 1F 00 00
FT RT FOT ROT

FT = 4211851F = floating point representation of 36.38 = front temperature of the Flyer in °C

RT = 41F7851F = floating point representation of 30.94 = rear temperature of the Flyer in °C

FOT = 00 = overtemp condition has not been detected at the front temperature sensor

ROT = 00 = overtemp condition has not been detected at the rear temperature sensor



Get Head Uptime: Retrieves the number of seconds Flyer has been running since the last boot-up sequence. This value resets every time the head is powered down or rebooted, so it indicates only the duration of the current running state, whether the head is marking or idle.

Command (from Client):

MBAP Header						FC	SynHeader			Data		
TI	PI		LN*	UI			SC	SE	WE			
00	00	00	00	00	06	00	43	00	51	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header						FC	SynHeader			Data		
TI	PI		LN	UI			SC	SE	WE			
00	00	00	00	XX	XX	00	43	00	51	XX	00	Uptime (or None)

Where SE is the SynError byte:

SE = 00 Success – data portion contains the head uptime value

The Uptime data is a long integer value indicating uptime in seconds. For instance, if the head has been up and running for 69,874 seconds since the last boot up sequence, Uptime = 00 01 10 F2.

Example:

To command the Flyer to report the head uptime data, send:

MBAP Header = 00 00 00 LN LN 06 00

Function Code = 43

SynHeader = 00 51 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 51 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 51 00 00

MBAP header FC SynHeader

If the Flyer is able to return the head uptime value without error, it responds with:

00 00 00 00 00 0A 00 43 00 51 00 00 00 01 10 F2

MBAP header FC SynHeader Data

Where:

Data = 00 01 10 F2 = 69874 seconds the head has been up and running since the last boot up sequence.



Get System Parameter: Returns the string value of the specified system parameter. See [Appendix A](#) for a list of valid system parameter names.

Command (from Client):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	32	00	00	SysParamName

The SysParamName string must:

- contain the hexadecimal byte representation of the system parameter
- be terminated with a NULL character (00)

Response (from Flyer):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	32	XX	00	SysParamValue (or None)

Where SE is the SynError byte:

SE = 00 Success

SE = 26 Invalid system parameter – NO DATA IS SENT

Example:

To retrieve Flyer’s Ethernet IP address, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 32 00 00

Data = ASCII hexadecimal byte representation of system parameter name (FlyIpAddress), terminated with a NULL character:

```

F l y I p A d d r e s s NULL
46 6C 79 49 70 41 64 64 72 65 73 73 00

```

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 32 00 00 46 6C 79 49 70 41 64 64 72 65 73 73 00
 Count = 19 bytes, so the length bytes = 00 13

So, to get Flyer’s Ethernet IP Address (FlyIpAddress) value, the packet would be:

00 00 00 00 00 13 00 43 00 32 00 00 46 6C 79 49 70 41 64 64 72 65 73 73 00
 MBAP header FC SynHeader Data

If Flyer’s Ethernet IP address is “192.168.90.32”, and it can return the value without error, the return packet is:

00 00 00 00 00 14 00 43 00 32 00 00 31 39 32 2E 31 36 38 2E 39 30 2E 33 32 00
 MBAP header FC SynHeader Data

Where:

Data = ASCII hexadecimal byte representation of the FlyIpAddress value:

```

31 39 32 2E 31 36 38 2E 39 30 2E 33 32 00
1 9 2 . 1 6 8 . 9 0 . 3 2 NULL

```



Set System Parameter: Sets a string value for the specified system parameter. See [Appendix A](#) for a list of valid system parameter names.

Command (from Client):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN		UI	SC		SE	WE			
00	00	00	00	XX	XX	00	43	00	31	00	00	SysParamName, SysParamValue

The SysParamName and SysParamValue strings must:

- contain the hexadecimal byte representation of the system parameter name and value
- each be terminated with a NULL character (00)

Response (from Flyer):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN*		UI	SC		SE	WE			
00	00	00	00	00	06	00	43	00	31	XX	00	(None)

*Since no data is sent, command packet length is always six bytes

Where SE is the SynError byte:

SE = 00 Success

SE = 27 Invalid system parameter name or value

Example:

To set Flyer's Ethernet IP Address (FlyIpAddress), send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 31 00 00

Data = ASCII hexadecimal byte representation of system parameter name and value, each terminated with a NULL character:

F l y I p A d d r e s s NULL 1 9 2 . 1 6 8 . 9 0 . 3 2 NULL
46 6C 79 49 70 41 64 64 72 65 73 73 00 31 39 32 2E 31 36 38 2E 39 30 2E 33 32 00

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 31 00 00 46 6C 79 49 70 41 64 64 72 65 73 73 00 31 39 32 2E 31 36 38 2E 39 30 2E 33 32 00

Count = 33 bytes, so the length bytes = 00 21

So, to set Flyer's Ethernet IP Address (FlyIpAddress) value, the packet would be:

00 00 00 00 00 21 00 43 00 31 00 00 46 6C 79 49 70 41 64 64 72 65 73 73 00 31 39 32 2E 31 36 38 2E 39 30 2E 33 32 00

MBAP header FC SynHeader

Data

If the Flyer can process the request without error, the return packet is:

00 00 00 00 00 06 00 43 00 31 00 00

MBAP header FC SynHeader



Reboot Marking Head: Initiate a reboot of the Flyer marking head.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	53	00	00	(None)

**Since no data is sent, command packet length is always six bytes*

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	53	XX	00	(None)

Where SE is the SynError byte:

SE = 00 Success

Example:

To command a Flyer head reboot, send:

MBAP Header = 00 00 00 LN LN 06 00

Function Code = 43

SynHeader = 00 53 00 00

Data = none

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 53 00 00

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 43 00 53 00 00

MBAP header FC SynHeader

If the Flyer receives the reboot command without error, it responds with:

00 00 00 00 00 06 00 43 00 53 00 00

MBAP header FC SynHeader



**Begin Firmware Upgrade:
Download Firmware Packet:**

The firmware upgrade procedure loads new firmware into the Flyer head. Upgrading Flyer firmware on the client side consists of opening the new .fhz firmware file and sending it to the Flyer head in 248-byte packets. There are two commands that control the firmware upgrade process: **Begin Firmware Upgrade** – a preamble sent to the head to prepare it for the incoming upgrade file and **Download Firmware Packet** – a 248-byte packet of the upgrade file sent to the marking head.

The typical download sequence is as follows:

```
Open .fhz upgrade file
Send Begin Firmware Upgrade command
While Not EOF
{
    Read 248 bytes from .fhz upgrade file
    Send Download Firmware Packet
}
Send Download Firmware Packet with packet length of zero. (to signal end of file download)
Download complete
```

Note: After the last firmware packet is received, Flyer head automatically installs the new firmware and reboots using the new firmware code, so no further communication will occur. It is the responsibility of the client software to re-establish a connection to the marking head after the head has installed updated firmware and completed the reboot. This reboot process takes approximately 2 minutes to complete.

Important Note: Because Modbus/IP is a Command/Response protocol, there is a specific Command/Response sequence necessary to ensure the receipt of all packets. Transaction tracking is accomplished using the Transaction Identifier field in the Modbus MBAP. A consequence of the Modbus/IP protocol is that download speeds are very slow (approximately 3 minutes) with a total upgrade time (including reboot) of approximate five minutes. For this reason, SYNRAD recommends that you upgrade Flyer firmware using the WinMark Pro Laser Marking Software application, if possible.



Begin Firmware Upgrade: A preamble sent to Flyer to prepare it for the firmware upgrade process.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN	UI		SC		SE	WE			
00	00	00	00	00	0A	00	43	00	11	00	00	Filesize

Where Filesize is a 4 byte value indicating the total bytes in fhz file.

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*	UI		SC		SE	WE			
00	00	00	00	00	06	00	43	00	11	XX	00	None

**Since no data is sent, command packet length is always six bytes*

Where SE is the SynError byte:

- SE = 00 Success
- SE = 32 Unable to begin download

Download Firmware Packet: Send up to 248 bytes of data from the .fhz upgrade file to the Flyer head. A packet length of zero either cancels the upgrade or signals the end of the download.

Command (from Client):

MBAP Header							FC	SynHeader				Data
TI	PI		LN	UI		SC		SE	WE			
XX	XX	00	00	XX	XX	00	43	00	12	00	00	UpgradeData

Where:

- the Transaction ID (TI) is a zero-based count of data packets sent
- UpgradeData is a section of data from the fhz upgrade file.

Response (from Flyer):

MBAP Header							FC	SynHeader				Data
TI	PI		LN*	UI		SC		SE	WE			
XX	XX	00	00	00	06	00	43	00	12	XX	00	None

**Since no data is sent, command packet length is always six bytes*

Where TI is the Transaction ID of the data packet received and SE is the SynError byte:

- SE = 00 Success
- SE = 32 Invalid file type (not a valid .fhz download file)
- SE = 33 Packet download failure

Example:

For sample Visual Basic and Visual C++ code demonstrating the firmware upgrade procedure, see http://www.winmark.com/products/winmark_activexsamples.html. The sample files provide the functions necessary to create commands for communicating with an FH Flyer head and are designed to be easily incorporated into customer applications.

I/O Management Commands

These commands use the basic (register-based) Modbus command function codes to read Flyer input/output bit status (**Read Flyer I/O**), read Flyer inputs (**Read Flyer Inputs**), or write to (set) Flyer outputs (**Write Flyer Outputs**). The protocol for these three commands is copied directly from the “*MODBUS Application Protocol Specification V 1.1b*”. The range of allowable values and addresses shown below are Flyer marking head specific. Upon receiving the command, the Flyer head responds with the I/O data or an error code.

Read Flyer I/O [Read Holding Registers]: Returns the current state of the Flyer’s Input and Output registers.

Command (from Client):

MBAP Header							FC	Data
TI	PI	LN*	UI					
00	00	00	00	00	06	00	03	StartAddress, QtyRegisters

**Since fixed length data values are used, command packet length is always six bytes*

Where:

- StartAddress is a 2-byte value indicating where to start reading, Register 0 (Flyer Input Register) or Register 1 (Flyer Output Register)

Note: If your PLC or Modbus controller does *not* support zero-based addressing, increment the starting register address by one (1).

- QtyRegisters is a 2-byte value indicating the number of registers to be reported, either 1 or 2. Use a value of ‘2’ to read both input and output status with a single command.

Response (from Flyer):

MBAP Header							FC	Data
TI	PI	LN	UI					
00	00	00	00	XX	XX	00	XX	IOData or ExceptionCode

Where FC is the Function Code byte:

FC = 03 Success and IOData contains the requested Register data:

Field	Length	Description
ByteCount (BC)	1 Byte	Number of bytes of register data = 2 bytes for every register requested by QtyRegisters
RegisterData (RD)	(2 x QtyRegisters) Bytes	Binary register data

or FC = 83, Error getting data, and Flyer returns a 1 byte ExceptionCode (EC):

- EC = 01 Illegal Modbus Function
- EC = 02 Illegal Data Address
- EC = 03 Illegal Data Value
- EC = 04 Slave Device (Flyer) Failure
- EC = 06 Slave Device (Flyer) Busy

[Example on next page]



Read Flyer I/O [Read Holding Registers] (cont):

Example:

To command Flyer to report the contents of both the Input and Output registers, send:

MBAP Header = 00 00 00 00 00 06 00

Function Code = 03

Data =

StartAddress = 00 00 (start at Input register)

QtyRegisters = 00 02 (report the status of the Input and Output registers)

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 03 00 00 00 02

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 03 00 00 00 02

MBAP header FC Data

If the Flyer is able to return the I/O status without error, it responds with something like:

00 00 00 00 00 07 00 03 04 00 51 00 62

MBAP header FC Data

Where:

ByteCount = 04

RegisterData = 00 51 00 62:

Inputs = 00 51 of which only the second byte is used is equivalent to binary 0101 0001, where the LSB corresponds to input IN0 status and the MSB corresponds to IN7 status. In this example, inputs IN0, IN4, and IN6 are ON and all other inputs are OFF.

Outputs = 00 62 of which only the second byte is used is equivalent to binary 0110 0010, where the LSB corresponds to output OUT0 status and the MSB corresponds to OUT7 status. In this example, outputs OUT1, OUT5, and OUT6 are ON and all other outputs are OFF.

If the Flyer encounters an error, it responds with something like:

00 00 00 00 00 03 00 83 06

MBAP header FC EC

Where EC is the ExceptionCode, indicating a Slave Device (Flyer) Busy.



Read Flyer Inputs [Read Input Registers]: Returns the current state of Flyer’s input bus, located at Flyer Register address 00 00.

Command (from Client):

MBAP Header							FC	Data
TI	PI		LN*	UI				
00	00	00	00	00	06	00	04	StartAddress, QtyRegisters

**Since fixed length data values are used, command packet length is always six bytes*

Where:

- StartAddress is a 2-byte value = 00 00 (Flyer Input Register)

Note: If your PLC or Modbus controller does *not* support zero-based addressing, increment the starting register address by one (1).

- QtyRegisters is a 2-byte value = 00 01 (read one register)

Response (from Flyer):

MBAP Header							FC	Data
TI	PI		LN	UI				
00	00	00	00	XX	XX	00	XX	IOData or ExceptionCode

Where FC is the Function Code byte:

FC = 04 Success and IOData contains the requested register data:

Field	Length	Description
ByteCount (BC)	1 Byte	Number of bytes of register data = 2 bytes
RegisterData (RD)	2 Bytes	Binary register data

or FC = 84, Error getting data, and Flyer returns a 1 byte ExceptionCode (EC):

- EC = 01 Illegal Modbus Function
- EC = 02 Illegal Data Address
- EC = 03 Illegal Data Value
- EC = 04 Slave Device (Flyer) Failure
- EC = 06 Slave Device (Flyer) Busy

[Example on next page]



Read Flyer Inputs [Read Input Registers] (cont):

Example:

To command Flyer to report the contents of the Input register, send:

MBAP Header = 00 00 00 00 00 06 00

Function Code = 04

Data =

StartAddress = 00 00 (start at Input register)

QtyRegisters = 00 01 (report just the status of the Input register)

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 04 00 00 00 01

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 04 00 00 00 01

MBAP header FC Data

If the Flyer is able to return the head status without error, it responds with something like:

00 00 00 00 00 05 00 04 02 00 51

MBAP header FC Data

Where:

ByteCount = 02

RegisterData = 00 51:

Inputs = 00 51 of which only the second byte is used is equivalent to binary 0101 0001, where the LSB corresponds to input IN0 status and the MSB corresponds to IN7 status. In this example, inputs IN0, IN4, and IN6 are ON and all other inputs are OFF.

If Flyer encounters an error, it responds with something like:

00 00 00 00 00 03 00 84 06

MBAP header FC EC

Where EC is the ExceptionCode, indicating a Slave Device (Flyer) Busy.



Write Flyer Outputs [Write Single Register]: Writes to the Flyer Output bus, located at Flyer Register address 00 01.

Command (from Client):

MBAP Header							FC	Data
TI	PI		LN*	UI				
00	00	00	00	00	06	00	06	StartAddress, Value

**Since fixed length data values are used, command packet length is always six bytes*

Where:

- StartAddress is a 2-byte value = 00 01 (Flyer Output Register)

Note: If your PLC or Modbus controller does **not** support zero-based addressing, increment the starting register address by one (1).

- Value is a 2-byte value = 00 00 to 00 FF (only the least significant byte is used)

Response (from Flyer):

MBAP Header							FC	Data
TI	PI		LN	UI				
00	00	00	00	XX	XX	00	XX	StartAddress, Value or ExceptionCode

Where FC is the Function Code byte:

FC = 06 Success and both StartAddress and Value match those from the command packet

or FC = 86, Error getting data, and Flyer returns 1 byte ExceptionCode (EC):

EC = 01 Illegal Modbus Function

EC = 02 Illegal Data Address

EC = 03 Illegal Data Value

EC = 04 Slave Device (Flyer) Failure

EC = 06 Slave Device (Flyer) Busy

[Example on next page]



Write Flyer Outputs [Write Single Register] (cont):

Example:

To command Flyer to set the Output register so that OUT0, OUT1 and OUT2 are ON and all other outputs are OFF, send:

MBAP Header = 00 00 00 00 00 06 00

Function Code = 06

Data =

StartAddress = 00 01 (start at Output register)

Value = 00 07 (binary 0000 0111)

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes AFTER the length bytes:

00 00 00 00 LN LN 00 06 00 01 00 07

Count = 6 bytes, so the LN bytes = 00 06

So, the packet would be:

00 00 00 00 00 06 00 06 00 01 00 07

MBAP header FC Data

If the Flyer is able to set the head's Output register without error, it responds with:

00 00 00 00 00 06 00 06 00 01 00 07

MBAP header FC Data

If the Flyer is encounters an error, it responds with something like:

00 00 00 00 00 03 00 86 06

MBAP header FC EC

Where EC is the ExceptionCode, indicating a Slave Device (Flyer) Busy.

Set Wait Digital: Commands Flyer to send a Wait Digital Event packet when its input state matches the input pattern specified by the user. Flyer can be commanded to wait indefinitely for the input pattern or send a timeout packet after a specified time period (timeout). The specified inputs, the input state required to fire the Wait Digital Event packet, and the amount of time Flyer should wait for the required input state (timeout value in milliseconds) are passed as EventDefinition data.

Important Note: Since Modbus is a Command/Response protocol, make sure the event timeout does not exceed the Ethernet communications timeout.

Note: This event cannot be set while marking is in progress; the return error (if marking) is a Modbus error (Exception Code = 06).

Command (from Client):

MBAP Header				FC	SynHeader			Data				
TI	PI	LN*	UI		SC	SE	WE					
00	00	00	00	00	0C	00	43	00	23	00	00	EventDefinition

**Since fixed length data values are used, command packet length is always twelve bytes*

Where EventDefinition contains:

Field	Length	Description
InputValue*	1 Bytes	Input state required to fire the event
Mask*	1 Bytes	Bit mask indicating which input bits to check
Timeout	4 Bytes	Signed Integer timeout value, in milliseconds (-1 = wait forever)

**Each input is represented as one bit in the byte value – MSB is IN7, LSB is IN0*

Indicating the desired input pattern to match, which is the logical AND of the InputValue and Mask. For example to check only IN3, IN4 and IN5, the Mask would be 0011 1000 (IN7 is the MSB, IN0 is the LSB) and to fire the event when IN3 and IN5 are active and IN4 is inactive, the InputValue would be 0010 1000. The resultant input pattern to match is xx10 1xxx, where the 'x' input states are ignored.

Response, if no Modbus error occurs and the Flyer inputs satisfy EventDefinition:

MBAP Header				FC	SynHeader			Data				
TI	PI	LN	UI		SC	SE	WE					
00	00	00	00	00	06	00	43	00	23	XX	00	(None)

Where SE is the SynError byte:

SE = 00 Success

SE = 50 Input pattern not met within the Timeout period

Response, if an error occurs:

MBAP Header				FC	Data						
TI	PI	LN*	UI								
00	00	00	00	00	03	00	C3	ExceptionCode = 06 (Flyer is busy)			

**Since fixed length data values are used, command packet length is always three bytes*

[Example on next page]



Set Wait Digital (cont):

Example:

To command Flyer to monitor the inputs for 1 second and report a Wait Digital Event when inputs 3 and 5 are active and input 4 is inactive, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 23 00 00

Data = 28 38 00 00 03 E8:

 InputValue = 0010 1000 = 28

 Mask = 0011 1000 = 38

 Timeout = 1 second = 1000 ms = 00 00 03 E8

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 23 00 00 28 38 00 00 03 E8

Count = 12 bytes, so the LN bytes = 00 0C

So, the packet would be:

00 00 00 00 00 0C 00 43 00 23 00 00 28 38 00 00 03 E8

 MBAP header FC SynHeader Data

If the Flyer is busy marking, it responds with:

00 00 00 00 00 03 00 C6 06

 MBAP header FC EC

If the Flyer head is able to process the command but does not find the defined input pattern within the timeout period, it responds with:

00 00 00 00 00 06 00 43 00 23 50 00

 MBAP header FC SynHeader

If the Flyer head finds the defined input pattern within the timeout period, it responds with:

00 00 00 00 00 06 00 43 00 23 00 00

 MBAP header FC SynHeader

SynComm Modbus-Asynchronous Flyer Marking Head Functions

The SynComm Modbus-Asynchronous protocol is a special case of the User-Defined Modbus protocol designed specifically for FH Flyer marking head use. This exclusive protocol permits unsolicited responses from Flyer that are necessary for capturing Log messages during marking, intermediate End of Mark messages (for files with *Mark Count* values greater than zero) and for processing **Input Change Events**. In order to use the SynComm Modbus-Asynchronous protocol, the following conditions must be met:

1. The Flyer head and Client software must communicate over Modbus port number 502.
2. The Client software must contain a Receive thread to process unsolicited transmissions from the Flyer head.

The following functions are currently available via Modbus-Asynchronous for Flyer marking head control:

[Modbus-Asynchronous Marking Events](#) (page 85):

- Log Message Event Returns log messages generated while marking in progress
- End of Mark Event Returns an sEOM structure for each End of Mark

[Modbus-Asynchronous I/O Events](#) (page 87):

- Set Input Change Create a bit mask to fire an Input Change Event when a masked input changes state
- Input Change Event Returns current input bit state (in sDigitalEvent structure) when a masked input (created by Set Input Change) changes state



Modbus-Asynchronous Marking Events

Log Message Event: Returns any Log messages generated by Flyer during a mark.

Response (from Flyer):

MBAP Header							FC	SynHeader			Data	
TI	PI		LN	UI	SC	SE		WE				
00	00	00	00	XX	XX	00	43	00	10	00	00	LogMessage

Where LogMessage is a null-terminated string.

Example:

If the mark session ends after a user abort, the Flyer would return a log message something like:

00 00 00 00 00 14 00 43 00 10 00 00 2A 2A 2A 41 42 4F 52 54 45 44 2A 2A 2A 00
 MBAP header FC SynHeader Data

Where:

Data = ASCII hexadecimal byte representation of the text caption:

2A 2A 2A 41 42 4F 52 54 45 44 2A 2A 2A 00
 * * * A B O R T E D * * * NULL



End of Mark Event: Returns Flyer mark statistics at the end of a mark piece.

Response (from Flyer):

MBAP Header				FC	SynHeader			Data				
TI	PI	LN	UI		SC	SE	WE					
00	00	00	00	00	22	00	43	00	62	00	00	Status

Where Status is a 28 byte value indicating:

Field	Length	Description
MarkStatus	2 Bytes	Decimal value indicating the Mark Status: Idle = 0 Marking = 1 Aborted = 2
Reserved	2 Bytes	
EOMResponse	4 Bytes	Binary value containing status information regarding the state of the Flyer head (see Appendix B)
CurrentPiece	4 Bytes	Decimal value indicating the current piece marked
Ticks	4 Bytes	Number of total ticks for entire mark (100 ticks = 1 second)
MarkCount	4 Bytes	Total number of pieces to be marked
TickMin	4 Bytes	Minimum tick count per piece (100 ticks = 1 second)
TickMax	4 Bytes	Maximum tick count per piece (100 ticks = 1 second)

Example

After marking a piece in a mark session, the Flyer sends something like:

00 00 00 00 00 22 00 43 00 62 00 01 &
 MBAP header FC SynHeader

00 01 03 04 00 00 00 00 00 00 00 18 00 00 01 10 00 00 02 0C 00 00 00 FF 00 00 01 1C
 Data

Where:

Data = mark statistics:

00 01 03 04 00 00 00 00 00 00 00 18 00 00 01 10 00 00 02 0C 00 00 00 FF 00 00 01 1C
 MkSt Rsvd EOM Resp CurrPc Ticks MarkCount TickMin TickMax

MarkStatus = 00 01 = head is marking

EOMResponse = 00 00 00 00 = no faults detected

CurrentPiece = 00 00 00 18 = piece # 24 marked

Ticks = 00 00 0F 10 = cycle time of last piece marked is 272 ticks = 2.72 seconds

MarkCount = 00 00 02 0C = 524 pieces are to be marked in this batch

TickMin = 00 00 00 FF = minimum cycle time for this batch so far is 255 ticks = 2.55 seconds

TickMax = 00 00 01 1C = maximum cycle time for this batch so far is 284 ticks = 2.84 seconds



Modbus-Asynchronous I/O Events

Set Input Change: Sets the bit mask required to fire an **Input Change Event**. When an input bit changes state and is part of the specified input bit/input bit mask, the Flyer head returns an **Input Change Event** that contains the current value of the input bits. Any inputs whose mask bits are set to 0 are ignored.

Note: This Event cannot be set while marking is in progress; the return error (if marking) is a Modbus error (0x6).

Command (from Client):

MBAP Header				FC	SynHeader			Data
TI	PI	LN*	UI		SC	SE	WE	
00	00	00	00	43	00	60	00	EventDefinition

**Since fixed length data values are used, command packet length is always twelve bytes*

Where EventDefinition contains:

Field	Length	Description
InputValue	1 Bytes	Not used
Mask*	1 Bytes	Bit mask indicating which input bits to check
Timeout	4 Bytes	Not used

**Each input is represented as one bit in the byte value – MSB is IN7, LSB is IN0*

Response, if no error occurs:

MBAP Header				FC	SynHeader			Data
TI	PI	LN	UI		SC	SE	WE	
00	00	00	00	43	00	60	00	(None)

Response, if an error occurs:

MBAP Header				FC	Data
TI	PI	LN*	UI		
00	00	00	00	C3	ExceptionCode = 06 (Flyer is busy)

**Since fixed length data values are used, response packet length is always three bytes*

[Example on next page]



Set Input Change (cont):

Example:

To command Flyer to report an Input Change Event when inputs IN3, IN4, or IN5 change state, send:

MBAP Header = 00 00 00 00 LN LN 00

Function Code = 43

SynHeader = 00 60 00 00

Data = 00 38 00 00 03 E8:

InputValue = 00 (not applicable, always zero)

Mask = 0011 1000 = 38 (fire event only if IN3, IN4, or IN5 change)

Timeout = 00 00 00 00 (not applicable, always zero)

The length bytes ('LN LN') in the MBAP header are determined by the count of all the packet bytes

AFTER the length bytes:

00 00 00 00 LN LN 00 43 00 60 00 00 00 38 00 00 00 00

Count = 12 bytes, so the LN bytes = 00 0C

So, the packet would be:

00 00 00 00 00 0C 00 43 00 60 00 00 00 38 00 00 00 00

MBAP header FC SynHeader Data

If the Flyer is busy marking, it responds with a Modbus error packet:

00 00 00 00 00 03 00 C3 06

MBAP header FC EC

If the Flyer head is able to process the command, it responds with:

00 00 00 00 00 06 00 43 00 60 00 00

MBAP header FC SynHeader



Input Change Event: After being enabled by the **Set Input Change** command, Flyer reports the current input bit status every time any input(s) of interest change state. Note that Input Status is returned for ALL inputs, not just the inputs of interest.

Response (from Flyer):

MBAP Header				FC	SynHeader			Data			
TI	PI	LN	UI		SC	SE	WE				
00	00	00	00	0C	00	43	00	62	00	00	IOData

Where IOData contains:

Field	Length	Description
InputStatus*	1 Bytes	Input status
Mask	1 Bytes	Not used
Timeout	4 Bytes	Not used

**Each input is represented as one bit in the byte value – MSB is IN7, LSB is IN0*

Example:

If Flyer has been commanded to report an Input Change Event when IN3, IN4 or IN5 changes state, and the input status changes from 0000 0000 to 0001 0000, Flyer responds with:

00 00 00 00 00 00 0C 00 43 00 62 00 00 10 00 00 00 00 00
 MBAP header FC SynHeader Data



SynComm SmartFH Protocol

Note: The SmartFH protocol is intended for *legacy support only* (i.e. systems where Flyer is replacing an existing FH Smart marking head. For maximum flexibility, new integrations of Flyer heads should use Modbus/IP or Modbus-Asynchronous protocols.

The SmartFH protocol is described in the *FH Series Smart Marking Head Operator's Manual* and is not repeated here. SynComm provides support for FH Smart's Special Protocol commands with the following exceptions:

Code Download	-- Not Supported
File Download	-- Not Supported
File Upload	-- Not Supported
File Upload By Name	-- Not Supported
Compress Filestore	-- Not Supported
Get File Space	-- Returns Used and Available space on Filestore
Save	-- Not Supported
Save As	-- Not Supported

Important Note: Port number 39538 is reserved for all SmartFH communications.



Appendix A: List of Flyer Marking Head System Parameter Names

Below is the current list of valid system properties for FH Flyer marking heads:

FlyIpAddress	Flyer address when static IP addressing is used. Use standard IP address format. The default Flyer IP address is: 192.168.100.100.
FlyIpMask	The subnet mask required when static IP addressing is used (typically set to 255.255.255.0).
FlyIpBroadcast	The broadcast address required when static IP addressing is used (typically set to 255.255.255.255).
FlyIpGateway	The gateway address required when static IP addressing is used.
FlyIpDns1	The primary DNS server address required when static IP addressing is used.
FlyIpDns2	The secondary DNS server address required when static IP addressing is used.
FlyUseDHCP	Enter “1” to obtain a dynamic IP address from the DHCP server (dynamic IP addressing) or “0” to use the FlyIpAddress property value (static IP addressing).
FlyIpRangex	Limit Flyer access to computers within a defined address range (or ranges). For example (e.g.), enter 192.168.45.6/192.268.45.6 to limit the allowable IP address to a single computer with the IP address 192.168.45.6. Up to ten different ranges, or addresses, can be entered (named as FlyIpRange0 , FlyIpRange1 , ... FlyIpRange9).
EncoderResolution	Floating point number for calculated encoder resolution in pulses per millimeter (e.g., 12.93).
InvertEncoderDirection	Enter “1” for True, “0” for False.
RisingEdgePartSense	Enter “1” for True, “0” for False.
SensorDistance	Floating point number for the measured sensor distance in inches.
QuadEncoder	Enter “1” for True, “0” for False.
UseEncoderless	Enter “1” for True, “0” for False.
UseFixedPartPitch	Enter “1” for True, “0” for False.
PartPitch	In encoderless tracking applications, a floating-point number representing the mark-to-mark distance in inches.
LineSpeed	In encoderless tracking applications, a floating-point value in inches per second equivalent to actual conveyor or part velocity.
MotionVector	Floating point number between 0–360° for angle of tracking orientation.
ObjectName	Alphanumeric name and/or number of a specific Flyer head. In WinMark Pro, the name of the currently selected head is displayed as the label of the “Device” tab and the <i>Mark</i> button.
FlyLens	an integer value indicating the current focusing lens installed on Flyer. Valid selections are: 125HP mm lens – 7; 80 mm lens – 6; 125 mm lens – 5; 200 mm lens – 2; 370 mm lens – 4, and User-defined lens – 128. Any other values cause Flyer to default to a 200 mm lens (FlyLens = 2).
StartUpDrawing	String value containing the path and filename of the file (saved in the Filestore) to load at startup in stand-alone mode.



MarkOnStart	Enter "1" (True) to start-up Flyer in stand-alone mode; "0" (False) to start-up in WinMark control mode. When MarkOnStart is "1", you must specify a StartUpDrawing .
CustomDateCodex	String value containing a unique custom date code definition for files used in stand-alone marking. The number of custom date code formats (named as CustomDateCode0 , CustomDateCode1 , ... CustomDateCodexx) is limited only by system memory.
ShiftCodexx	String value representing hourly shift codes contained in files for stand-alone marking. For example, ShiftCodes0 = A, ShiftCodes1 = A, ... ShiftCodes7 = A, ShiftCodes8 = B, ... ShiftCodes15 = B, ShiftCodes16 = C, ... ShiftCodes23 = C.
FlyUbootVersion	Read-only string containing the U-boot bootloader version.
FlyKernelVersion	Read-only string containing the Linux kernel version.
FlyVersion	Read-only string containing the firmware version.
FlySerialNumber	Read-only string containing the Flyer serial number.
FlyMacAddress	Read-only string containing the Flyer MAC address.
KeyboardLocked	Enter "1" (True) to lock the keyboard; "0" (False) to unlock the keyboard.
FlyFasiEnable	Read-only string indicating whether the FASI switch is On or Off.
FlyPwmGate	Read-only string indicating whether Flyer's PWM output is switched to provide a PWM signal or a Gate signal.
FlyTickleDisable	Read-only string indicating whether the tickle function is switched to Disable (no tickle) or Normal (with tickle).
FlyShareUser	Not implemented. String value containing the user name for connecting to a Windows share on the network.
FlySharePassword	Not implemented. String value containing the password for connecting to a Windows share on the network.
FlyShareServer	Not implemented. String value containing the IP address or DNS name of the server (if DNS is setup correctly).
FlyShareName	Not implemented. The "path" leading to the share "\\server\path".
FlyShareReadOnly	Not implemented. Enter "1" (True) to connect to the share with read-only access; "0" (False) to connect to the share with all permissions normally given to the username specified by FlyShareUser .
ClearingMarkInterval	In stand-alone marking, an integer value representing the number of mark cycles between a clearing mark operation. A value of 0 is off (no clearing mark performed).
ClearingMarkSessionStart	In stand-alone marking, enter "1" (True) to perform a clearing mark at start of mark session; "0" (False) - do NOT perform a clearing mark at start of mark session.



Appendix B: Flyer EOM Response Data Map

The four bytes of data returned as the EOMResponse portion of the End Of Mark data packet are mapped as follows (with bit 31 being the MSB of the left-most byte, bit 0 being the LSB of the right-most byte):

Bit	Function	Bit Definitions (Indicates When Set)
31	Over Temp 2	An over-temperature fault exists on the head
30	Over Temp 1	An over-temperature warning exists on the head
29	Power Fault	The 5V supply for the A/D converters is out of tolerance
28	PWM Fault	The PWM output does not match the commanded value
27	Power Amp Disable	The Servo power amps are disabled
26	Reserved	
25	Reserved	
24	Reserved	
23	Line Speed Error	(Tracking mode only) The part has moved beyond the marking window before completion of the mark
22	Multi-Part Error	(Tracking mode only) A second part has moved into the marking field before the completion of the previous part mark
21	Y Servo Fault	A fault has been detected in the Y axis servo loop
20	X Servo Fault	A fault has been detected in the X axis servo loop
19	Need Tracking Vectors	(Tracking mode only) The servo DSP has not received sine or cosine motion vectors for part movement
18	Need Lens	The servo DSP has not received the lens coefficients needed for optical correction
17	Need Notch	The servo DSP has not received the tuned coefficients for the IIR notch filters used in the servo loop
16	Need Tuning	The servo DSP has not received tuning gains used in the servo loop
15	Reserved	
14	Reserved	
13	Mark Complete	The servo DSP has completed the current mark
12	Reserved	
11	Reserved	
10	Reserved	
9	Reserved	
8	Reserved	
7	Reserved	
6	Reserved	
5	Reserved	
4	Reserved	
3	Reserved	
2	Reserved	
1	Reserved	
0	Reserved	



Appendix C: MODBUS and SYNRAD Error Codes

If the PLC's Message Instruction returns a Modbus error, then perform a read of Register 0x0066 (102 decimal) for a more descriptive SYNRAD file or marking head error code as shown below:

MODBUS Error Codes

Error Code (Hex, Decimal)	Error Name
0x1, 1	MB Illegal Function
0x2, 2	MB Illegal Data Address
0x3, 3	MB Illegal Data Value
0x4, 4	MB Slave Device Failure
0x5, 5	MB Acknowledge
0x6, 6	MB Slave Device Busy

SYNRAD File Error Codes

Error Code (Hex, Decimal)	Error Name
0x20, 32	No Current File
0x21, 33	File Load
0x22, 34	No File Loaded
0x23, 35	Get Property Fail
0x24, 36	File Space Fail
0x25, 37	Set Property Fail
0x26, 38	Get Parameter Fail
0x27, 39	Set Parameter Fail
0x28, 40	File Delete
0x29, 41	File Move
0x2A, 42	File Directory
0x2B, 43	Filestore Erase
0x2C, 44	File Network Refresh
0x2D, 45	Null Terminated String



SYNRAD Marking Head Error Codes

<u>Error Code (Hex, Decimal)</u>	<u>Error Name</u>
0x30, 48	Head Marking
0x31, 49	Head Not In Stand-Alone Mode
0x32, 50	Firmware Upgrade
0x33, 51	Firmware Download
0x40, 64	Get UTC Time
0x41, 65	Get Local Time
0x42, 66	Set UTC Time
0x43, 67	Set Local Time
0x44, 68	Get DST Information
0x45, 69	Set DST Information
0x50, 80	I/O Timeout
0x79, 121	Unknown Command



Appendix D: Flyer's MODBUS Register Memory Map

The chart below shows Flyer's Modbus memory map addressing for Register-Based Modbus functions 3, 4, 6, and 26:

Address	Description	Size	Access	Action
IO				
0x0000	Input Register	WORD	R	Get Inputs
0x0002	Output Register	WORD	R/W	Get/Set Outputs
EOM/Mark Status				
0x0004	Mark Status	WORD	R/W	Write: Mark (no EOM wait). 0 = Idle (Read) 1 = Mark No Wait 2 = Abort Read: Mark Status
0x0006	Mark Count	DWORD	R	Mark Status
0x000A	Current piece	DWORD	R	Mark Status
0x000E	Ticks	DWORD	R	Mark Status
0x0012	Tick Min	DWORD	R	Mark Status
0x0016	Tick Max	DWORD	R	Mark Status
0x001A	Servo Status	DWORD	R	Mark Status
Head Uptime				
0x0020	Head Uptime	DWORD	R	GetUptime
Head Temperature				
*Data in tenths Celsius				
0x0024	Power Amp Temp	SWORD	R	GetTemperature
0x0026	CPU Temp	SWORD	R	GetTemperature
0x0028	Power Amp Overtemp	WORD	R	GetTemperature
0x002A	CPU Overtemp	WORD	R	GetTemperature
RESERVED				
0X002C Through 0X0036	RESERVED	WORDS		
Marking Head Status				
0X0038	Head Type	WORD	R	GetHeadStatus
0X003A	Is Marking	WORD	R	GetHeadStatus
0X003C	Is Standalone	WORD	R	GetHeadStatus
0X003E	Is Network Share	WORD	R/W	Read: GetHeadStatus Write: RefreshNetworkShare
Date/Time				
* Write must send all registers				
0x0040	Year	WORD	R/W	Get/Set Date Time
0x0042	Month	WORD	R/W	Get/Set Date Time
0x0044	DayOfWeek	WORD	R/W	Get/Set Date Time
0x0046	Day	WORD	R/W	Get/Set Date Time
0x0048	Hour	WORD	R/W	Get/Set Date Time
0x004A	Minute	WORD	R/W	Get/Set Date Time
0x004C	Second	WORD	R/W	Get/Set Date Time
0x004E	Milliseconds	WORD	R/W	Get/Set Date Time
Filestore Usage				
0x0054	Used	DWORD	R	GetFilestoreUsage
0x0058	Available	DWORD	R	GetFilestoreUsage

0x005C	RESERVED	DWORD		
0X0060	RESERVED	DWORD		
0x0064	RESERVED	WORD		
0X0066	Synrad Error Code	WORD	R	GetSynError
0X0068 THROUGH 0X0FE	RESERVED	WORDS		
Filename				
0x0100 – 0x01F6	FileName	STRING	R/W	Read: GetCurrentFile Write: LoadFile
File Properties				
0x01F8 - 0x021E	Object/Entity Name	STRING	R/W	Get/Set File Properties
0x0220 - 0x024E	Property Name	STRING	R/W	Get/Set File Properties
0x0250 – 0x02FE	Property Value	STRING	R/W	Get/Set File Properties ** Read/Write to Property value actually triggers the function.
System Parameters				
0x0300 - 0x0356	Parameter Name	STRING	R/W	Get/Set Head Parameters
0x0358 – 0x03F8	Parameter Value	STRING	R/W	Get/Set Head Parameters ** Read/Write to Parameter value actually triggers the function.
Network File				
0X400 – 0X04F8	Network File	STRING	W	Load Network File