WinMark Pro Application Note

SYNRAD External Communications Server (SynComm) for Controlling FH Flyer Marking Heads Using the Modbus® Protocol

This Application Note provides information about using the SYNRAD external communications server (SynComm), available on FH Flyer marking heads and Fenix Flyer Laser Markers running firmware version 2.66 or above, on a network using the Modbus® protocol.

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.

Table of Contents

SynComm Modbus I/P Protocol ..... pg 2
Modbus/IP Packet Structure and SynComm Packets ..... pg 3
SynComm User-Defined Structure ..... pg 4
Modbus/IP Flyer Marking Head Functions ..... pg 5
- Marking Commands ..... pg 7
- Filestore Management Commands ..... pg 16
- Date/Time Management Commands ..... pg 24
- Flyer Head Management Commands ..... pg 28
- I/O Management Commands ..... pg 36

SynComm Modbus-Asynchronous Flyer Marking Head Functions ..... pg 40
- Modbus-Asynchronous Marking Events ..... pg 41
- Modbus-Asynchronous I/O Events ..... pg 43

SynComm Smart FH Protocol ..... pg 45
Appendix: List of Flyer Marking Head System Parameter Names ..... pg 46

SynComm Modbus/IP Protocol

Many of the features necessary for marking head control are performed through a user-defined function code that is part of the Modbus/IP protocol. In order to use SynComm within a Modbus/IP network, one of the user-defined function codes (65–72 and 100–110) must be available. The user-defined function code is set by the user in WinMark Pro using the ***Modbus User Function*** property on the "Device" (Flyerxxxxx) tab. The default value is 67 (0x43 in hexadecimal).

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

Guidelines for using the SynComm 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 (Mobus.h and Modbus.c contain endian conversion routines to aid in parsing data).
- Modbus is a request/reply (master/slave) protocol. FH Flyer is set to be a Modbus slave device (server).

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 does ***not*** equal UserFunction (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.

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

7 Bytes	**1 Byte**	**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

Fields	Length	Description	Client	Marking Head
Transaction Identifier	2 Bytes	Identification of a Modbus Request/Response transaction.	Initialized by the client (Request)	Recopied by the server (Flyer) from the received Request
Protocol Identifier	2 Bytes	0 = Modbus protocol	Initialized by the client (Request)	Recopied by the server (Flyer) from the received Request
Length	2 Bytes	Number of following bytes	Initialized by the client (Request)	Initialized by the server (Flyer) (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client (Request)	Recopied by the server (Flyer) from the received Request

For all Modbus commands, the MBAP Header field values are defined as follows:

Transaction Identifier = 0
Protocol Identifier = 0
Length = defined by specific command
Unit Identifier = 0

Function Code Format

The single-byte Function Code directs FH Flyer to perform one of four functions using the following values. The first three functions are Modbus Commands while the fourth command, User-Defined, allows the Client to direct specific Flyer Marking Head Commands using the SynComm User-Defined Structure.

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

SynComm User-Defined Structure

When the single-byte Function Code contained within the Modbus/IP packet structure is set to the SynComm User-Defined code (within the range of 0x41 to 0x48 or 0x64 to 0x6E; 0x43 by default), the packet structure for all subsequent communications will contain an extra header structure (SynHeader) to facilitate the transmission of all Flyer Marking Head Commands.

Instead of referring to a packet comprised of an MBAP Header, a Function Code, the SynHeader, and Data; all future references to the MBAP header will be labeled sMBAP and will include the Function Code, so that sMBAP = 8 bytes. The chart below illustrates the SynComm packet structure.

8 Bytes	**4 Bytes**	**248 Bytes**
sMBAP Header	SynHeader	Data

SynHeader Format

Fields	Length	Description	Client	Marking Head
SynCode	2 Bytes	Identification of the SynComm Function Code.	Initialized by the Client (Request)	Recopied by the server (Flyer) from the received Request
SynError	1 Byte	Error Code returned by Flyer head. No Error = 0	Set to zero by Client (Request)	Initialized by the server (Flyer) (Response)
WaitForEOM	1 Byte	Used by Mark command. Instructs Flyer to wait until EOM before responding to Client Request	Initialized by the client (Request)	Recopied by the server (Flyer) from the received Request

Flyer Marking Head Commands

For all commands where the sMBAP function code is User-Defined, the sMBAP Header fields are defined as follows:

*Transaction Identifier = 0
Protocol Identifier = 0
Length = defined by specific command
Unit Identifier = 0
Function Code = User-Defined (within range of 0x41–0x48, 0x64–0x6E; 0x43 by default)

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

Modbus/IP Flyer Marking Head Functions:

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

Marking Commands (page 7):

- 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 16):

- 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 24):

- 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 28):

- 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 36):

- **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 36, 37, and 38.

Marking Commands

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

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	FileName (null-terminated string)

sMBAP Header→Length = Offset + FileName Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x1
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

The FileName string must be prefix by a backslash (“/”) and must contain the full path and extension. For example: “/myfile.mkh\0”; “/mydir/myfile.mkh\0”.

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x21 Error loading file

==========

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

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	FileName (null-terminated string)

sMBAP Header→Length = Offset + FileName Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0xC
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

The FileName string must be prefix by a backslash (“/”) must contain the full path and extension. For example: “/myfile.mkh\0”; “/mydir/myfile.mkh\0”

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x21 Error loading file

==========

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

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x5
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response, no Error (from Flyer):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	FileName (null-terminated string)

sMBAP Header→Length = Offset + FileName Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success
FileName→ = returns the full path, filename, and extension of the file.

For example: “/filestore/myfile.mkh\0”; “/network/myfile.mkh\0”; /filestore/mydir/myfile.mkh\0”.

Response, if Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→ SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→ SynErr = 0x22 No file loaded

==========

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

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	ObjectName and PropertyName (null-terminated strings)

sMBAP Header→Length = Offset + ObjectName Length + PropertyName Length + 2 (Null characters); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x7
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

For example, specifying the ObjectName = “Text1\0” and the PropertyName = “TextCaption\0”, returns PropertyValue = “MyText\0”.

Response, no Error (from Flyer):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	Property Value (null-terminated string)

sMBAP Header→Length = Offset + PropertyValue Length + 1(Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x21 Error loading file

Response, if Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x22 No file loaded
SynHeader→SynErr = 0x31 Flyer head not in stand-alone mode
SynHeader→SynErr = 0x23 Invalid ObjectName and/or PropertyName

==========

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*).

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	ObjectName ,PropertyName, and PropertyValue (null-terminated strings)

sMBAP Header→Length = Offset + ObjectName Length + PropertyName Length + PropertyValue Length + 3 (Null characters); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x6
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

For example: specify ObjectName = “Text1\0”, PropertyName = “TextCaption\0”, and PropertyValue = “My New Text\0” to change the TextCaption value of the Text1 object to “My New Text”.

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 No Error
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x22 No file loaded
SynHeader→SynErr = 0x25 Invalid ObjectName, PropertyName, or PropertyValue

==========

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

0x1: Wait For End-of-Mark: The Flyer head does not reply to the **Mark File** request 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. The reply includes an End of Mark structure (description given below).

0x0: Don't Wait For End-of-Mark: The Flyer head replies immediately to the **Mark File** request and returns the Mark Count for the file. Polling via the **Mark Status** command is required to determine when the mark is completed

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x20
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x1 Wait for End Of Mark
OR
SynHeader→WaitForEOM = 0x0 Don't Wait For End of Mark

Response, no Error, Wait For End of Mark (from Flyer):

8 Bytes	4 Bytes	26 Bytes
sMBAP Header	SynHeader	sEOM

sMBAP Header→Length = Offset + size of sEOM; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success
SynHeader→WaitForEOM = 0x1

sEOM Structure:

Name	Size	Description
MarkStatus	2 Bytes	Decimal value indicating the Mark Status: Idle = 0 Marking = 1 Aborted = 2
Reserved	2 Bytes	
EOMResponse	4 Bytes	Hexadecimal value containing status information regarding the state of the Flyer head:
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)

Mark File (cont.):

Response, no Error, Don't Wait For End of Mark (from Flyer):

8 Bytes	4 Bytes	4 Bytes
sMBAP Header	SynHeader	MarkCount DWORD

sMBAP Header→Length = Offset + 4 bytes; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success
SynHeader→WaitForEOM = 0x0
MarkCount→ = returns the total number of pieces to be marked

Response, if Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→ SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→ SynErr = 0x22 No file loaded
SynHeader→ SynErr = 0x31 Flyer head not in stand-alone mode

==========

Mark Status: Returns the current marking status.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x25
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes	26 Bytes
sMBAP Header	SynHeader	sEOM

sMBAP Header→Length = Offset + size of sEOM; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success

sEOM Structure:

Name	Size	Description
MarkStatus	2 Bytes	Decimal value indicating the Mark Status: Idle = 0 Marking = 1 Aborted = 2
Reserved	2 Bytes	
EOMResponse	4 Bytes	Hexadecimal value containing status information regarding the state of the Flyer head:
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)

==========

Abort Mark: Abort the current mark session.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x21
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes	26 Bytes
sMBAP Header	SynHeader	sEOM

sMBAP Header→Length = Offset + size of sEOM; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success

sEOM Structure:

Name	Size	Description
MarkStatus	2 Bytes	Decimal value indicating the Mark Status: Idle = 0 Marking = 1 Aborted = 2
Reserved	2 Bytes	
EOMResponse	4 Bytes	Hexadecimal value containing status information regarding the state of the Flyer head:
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)

==========

Filestore Management Commands

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

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	DirectoryName (null-terminated string)

sMBAP Header→Length = Offset + DirectoryName Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0xA
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Directory names must be prefixed by a backslash and are referenced from the root directory ("/filestore") so that entering "/mydir" is the same as "/filestore/mydir".

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x2A Invalid directory name or directory exists

==========

Delete File/Directory: Delete a file or directory from the Flyer Filestore. Because only empty directories can be deleted, you must first delete all files within the directory.

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	FileName (null-terminated string)

sMBAP Header→Length = Offset + FileName Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x3
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Filenames must be prefixed by a backslash and are referenced from the root directory ("/filestore); so that entering "/myfile.mkh" is the same as "/filestore/myfile.mkh".

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x28 Invalid file or directory or the directory is not empty

==========

Get File List: Returns the current list of files and directories in the Flyer Filestore. This is the only Request that may require multiple packets to complete the transaction.

Important Note: Because Modbus is a Request/Response protocol, there is a specific Request/Response sequence necessary to ensure the receipt of all packets. Transaction tracking is accomplished using the Transaction Identifier field in the Modbus MBAP header and a SynComm packet structure that provides the current packet and total number of packets to be transferred. When the current packet equals total packets, the transaction is complete.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Transaction Identifier = N; where N is the packet number requested by the Client
sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x3
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes	4 Bytes	up to 244 Bytes
sMBAP Header	SynHeader	sPacket	Filelist data (collection of null-terminated strings)

sMBAP Header→Transaction Identifier = N; where N is the packet number returned by the Flyer head
sMBAP Header→Length = Offset + Filelist data length; where Offset = 10 bytes (sPacket + SynHeader + Unit Identifier + Function Code)
SynHeader→ErrCode = 0 Success

Important Note: The first **Get File List** request sets sMBAP Header→Transaction Identifier = 0. If the sPacket→TotalPackets reply is > 1, then each subsequent **Get File List** request increments the sMBAP Header→Transaction Identifier by one, until sPacket→CurrentPacket = sPacket→TotalPackets. For a C++ code example, see FileListCmd() in Modbus.c in the sample application files posted with this document.

sPacket Structure:

Name	Size	Description
CurrentPacket	2 Bytes	Current packet sent by Flyer head
TotalPackets	2 Bytes	Total packets to be sent by Flyer head

The FileList data is sent as a collection of null-terminated strings. These strings will be either a directory folder or a filename with the full path length referenced from the root directory (/filestore); so that reading "/filestore/myfile.mkh" is the same as = "/myfile.mkh". Directory folders are prefixed by a double backslash "//" also referenced from the root directory ("/filestore").

Example: FileList Data = "//mydir\0/mydir/myfile.mkh\0/myotherfile.mkh\0"; indicates the Flyer Filestore contains a folder ("/filestore/mydir") holding a single mark file, "/filestore/mydir/myfile.mkh", and another file, "/filestore/myotherfile.mkh", located outside the "mydir" folder in the root directory of the Filestore.

==========

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

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x4
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response, no Error (from Flyer):

8 Bytes	4 Bytes	8 Bytes
sMBAP Header	SynHeader	sFileSpace

sMBAP Header→Transaction Identifier = N; where N is the packet number returned by the Flyer head.
sMBAP Header→Length = Offset + Filelist data length; where Offset = 10 bytes (sPacket + SynHeader + Unit Identifier + Function Code)
SynHeader→ErrCode = 0 Success

sFileSpace Structure:

Name	Size	Description
Used	4 Bytes	Bytes used in the Flyer Filestore
Available	4 Bytes	Bytes available in the Filestore

Response, if Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→ErrCode = 0x24 Failed to retrieve filestore information

==========

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

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	SourceName (null-terminated string) DestName (null-terminated string)

sMBAP Header→Length = Offset + SourceName Length + DestName Length + 2 (Null characters); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x9
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

SourceName and DestName must be prefixed by a backslash and are referenced from the root directory ("/filestore") so that entering "/mydir" is the same as "/filestore/mydir".

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x29 Invalid SourceName and/or DestName

==========

Copy File: Copy a file residing in the Flyer Filestore to another Filestore location.

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	SourceName (null-terminated string) DestName (null-terminated string)

sMBAP Header→Length = Offset + SourceName Length + DestName Length + 2 (Null characters); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x8
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

SourceName and DestName must be prefixed by a backslash and are referenced from the root directory ("/filestore") so that entering "/mydir" is the same as "/filestore/mydir".

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x29 Invalid SourceName and/or DestName

==========

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

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0xB
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x2B Unable to erase Flyer Filestore

==========

Update Network Mount: Refreshes the network file share connection.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0xD
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→SynErr = 0x30 Unable to complete request – marking in progress
SynHeader→SynErr = 0x2C Unable to connect to network file share

==========

Date/Time Management Commands

Get UTC Time: Retrieves the Flyer head's UTC time.
Get Local Time: Retrieves the Flyer head's local time.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x40 UTC Time
SynHeader→SynCode = 0x41 Local Time
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response, no Error (from Flyer):

8 Bytes	4 Bytes	16 Bytes
sMBAP Header	SynHeader	sSystemTime

sMBAP Header→Length = Offset; where Offset = 22 bytes (sSystemTime + SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success

sSystemTime Structure size = 16 Bytes:

Name	Size	Description
Year	2 Bytes	Four digit year
Month	2 Bytes	January = 1
DayOfWeek	2 Bytes	Sunday=0
Day	2 Bytes	1-31
Hour	2 Bytes	0-59
Minute	2 Bytes	0-59
Second	2 Bytes	0-59
Milliseconds	2 Bytes	Currently always zero

Response, if Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x10 Error retrieving UTC time
SynHeader→SynErr = 0x11 Error retrieving local time

==========

Set UTC Time: Sets the Flyer head's UTC time.
Set Local Time: Sets the Flyer head's local time.

Request (from Client):

8 Bytes	4 Bytes	16 Bytes
sMBAP Header	SynHeader	sSystemTime

sMBAP Header→Length = Offset; where Offset = 22 bytes (sSystemTime + SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x42 UTC Time
SynHeader→SynCode = 0x43 Local Time
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

sSystemTime Structure size = 16 Bytes:

Name	Size	Description
Year	2 Bytes	Four digit year
Month	2 Bytes	January = 1
DayOfWeek	2 Bytes	Sunday=0
Day	2 Bytes	1-31
Hour	2 Bytes	0-59
Minute	2 Bytes	0-59
Second	2 Bytes	0-59
Milliseconds	2 Bytes	Currently always zero

Response, no Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success
SynHeader→SynErr = 0x12 Error setting UTC time
SynHeader→SynErr = 0x13 Error setting local time

==========

Get DST String: Retrieves the Flyer head's Daylight Savings Time (DST) information.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x44
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response, no Error (from Flyer):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	DST String (null-terminated string)

sMBAP Header→Length = Offset + DSTString Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success

The DST information string is received from Flyer as a formatted string containing the following information:

"PST+8PDT+7,M3.2.0/03:00,M11.1.0/02:00", where:

'PST+8' indicates the standard time bias (in this case Pacific Standard Time) with a +8 hour offset from UTC time and segment

'PDT+7,' indicates the daylight time bias (in this case Pacific Daylight Time) with a +7 hour offset from UTC time and segment

'M3.2.0/03:00' indicates when the changeover from standard time to daylight savings time occurs —in this case M3.2.0; where '3" indicates the third month (March), '2' indicates the second week beginning on '0' (day 0); which is Sunday, and segment

'/03:00,' indicates that the change occurs at 3:00 AM on the specified date and segment

'M11.1.0/02:00' indicates when the changeover from daylight savings time to standard time occurs—in this case M11.1.0 indicates the first Sunday (day 0) of the 11th month (November). Segment '/02:00' indicates that the change occurs at 2:00 AM on the specified date.

Response, if Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x14 Unable to retrieve DST information

==========

Set DST String: Sets the Flyer head's Daylight Savings Time (DST) information.

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	DST String (null-terminated string)

sMBAP Header→Length = Offset + DSTString Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x45
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

The DST information string is sent to Flyer as a formatted string containing the following information:
"PST+8PDT+7,M3.2.0/03:00,M11.1.0/02:00", where:
'PST+8'indicates the standard time bias (in this case Pacific Standard Time) with a +8 hour offset from UTC time and segment
'PDT+7,' indicates the daylight time bias (in this case Pacific Daylight Time) with a +7 hour offset from UTC time and segment
'M3.2.0/03:00' indicates when the changeover from standard time to daylight savings time occurs —in this case M3.2.0; where '3" indicates the third month (March), '2' indicates the second week beginning on '0' (day 0); which is Sunday, and segment
'/03:00,' indicates that the change occurs at 3:00 AM on the specified date and segment
'M11.1.0/02:00' indicates when the changeover from daylight savings time to standard time occurs—in this case M11.1.0 indicates the first Sunday (day 0) of the 11th month (November). Segment '/02:00' indicates that the change occurs at 2:00 AM on the specified date.

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success
SynHeader→SynErr = 0x15 Unable to set DST information

==========

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)

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP→Length = Offset
where Offset = 6 Bytes (SynHeader + Unit Identifier + Function Code)

SynHeader→SynCode = 0x52
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes	4 Bytes
sMBAP Header	SynHeader	sHeadStats

sMBAP→Length = Offset
where Offset = 10 Bytes (sHeadStats + SynHeader + Unit Identifier + Function Code)

SynHeader→ SynErr = 0 Success

sHeadStats Structure size = 4 Bytes:

Name	Size	Description
Marking Head Type	1 Byte	1 = Flyer
IsMarking	1 Byte	FALSE = 0; TRUE = 1
IsStandalone	1 Byte	FALSE = 0; TRUE = 1
IsNetworkShareConnected	1 Byte	FALSE = 0; TRUE = 1

==========

Get Head Temperature: Returns the measured air temperature from Front and Rear locations as well as over-temperature status from these locations.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x50
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes	10 Bytes
sMBAP Header	SynHeader	sTemperatureInfo

sMBAP Header→Length = Offset; where Offset = 16 bytes (sTemperatureInfo + SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success

sTemperatureInfo Structure size = 10 Bytes:

Name	Size	Description
FrontTemp	4 Bytes float	Floating point Front sensor temperature in Celsius
RearTemp	4 Bytes float	Floating point Rear sensor temperature in Celsius
FrontOvertemp	1 Byte	FALSE = 0; TRUE = 1
RearOvertemp	1 Byte	FALSE = 0; TRUE = 1

==========

Get Head Uptime: Returns the number of seconds since the Flyer head was last booted up.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x51
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes	4 Bytes
sMBAP Header	SynHeader	Uptime DWORD

sMBAP Header→Length = Offset; where Offset = 16 bytes (sTemperatureInfo + SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0 Success

Uptime DWORD = number of seconds since last boot up.

==========

Get System Parameter: Returns a string value of the specified system parameter. See the Appendix for a list of valid system parameters.

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	SysParamName (null-terminated string)

sMBAP Header→Length = Offset + SysParamName Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x32
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Example: SysParamName = “FlyIpAddress\0”

Response, no Error (from Flyer):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	SysParamValue (null-terminated string)

sMBAP→Length = Offset + SysParamName Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0

Example: SysParamValue = “192.168.90.32\0”

Response, if Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x26 Invalid system parameter

==========

Set System Parameter: Sets a string value for the specified system parameter. See the Appendix for a list of valid system parameters.

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	SysParamName (null-terminated string) SysParamValue (null-terminated string)

sMBAP Header→Length = Offset + SysParamName Length + SysParamValue Length + 2 (Null chars); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x31
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Example: SysParamName = “FlyIpAddress\0” and SysParamValue = “192.168.90.32\0”

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→ SynErr = 0x27 Invalid system parameter name or value

==========

Reboot Marking Head: Reboots the Flyer head.

Request (from Client):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x53
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success

==========

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

After the last packet is received, the Flyer head will automatically install the new firmware and reboot on the new firmware code, 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 completed the reboot. This reboot process takes approximately 2 minutes to complete.

Important Note: Because Modbus/IP is a Request/Response protocol, there is a specific Request/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 WinMark Pro, if possible.

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.

Begin Firmware Upgrade and **Download Firmware Packet** commands are described on the following page.

Begin Firmware Upgrade: A preamble sent to the marking head to prepare it for the firmware upgrade file download.

Request (from Client):

8 Bytes	4 Bytes	4 Bytes
sMBAP Header	SynHeader	Filesize DWORD

sMBAP Header→Length = Offset + 4 bytes (size of DWORD); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x11
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→SynErr = 0x32 Unable to begin download.

==========

Download Firmware Packet: Send an upgrade file packet up to 248 bytes in length to the Flyer Head. A packet length of zero either cancels the upgrade or signals the end of the download.

Request (from Client):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	Upgrade File Packet

sMBAP Header→TransactionID = Packet Number (zero based)
sMBAP Header→Length = Offset + Packet Length; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x12
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

Response (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→TransactionID = Received Packet Number (zero based)
sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success
SynHeader→SynErr = 0x32 Invalid file type (Not a .fhz download file)
SynHeader→SynErr = 0x33 Packet download failure

==========

I/O Management Commands

To read Flyer input/output bit status (**Read Flyer I/O**), read FH Flyer inputs (**Read Flyer Inputs**), or write to Flyer outputs (**Write Flyer Outputs**), SynComm uses three basic Modbus commands: ***Read Holding Registers***, ***Read Input Registers***, and ***Write Single Register***. 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 a Request from the Client, FH Flyer will return a Response or an Error.

Read Flyer I/O (*Read Holding Registers*): This command returns the current input and output states (registers) of the Flyer marking head or it returns an error. There are two valid addresses and two registers:

Input Register Address = 0x0000
Output Register Address = 0x0001

Request (from Client):

Description	Size	Value
Function Code	1 Byte	0x03
Starting Address	2 Bytes	0x0000 to 0x0001
Quantity of Registers	2 Bytes	1 to 2

Response, no Error (from Flyer):

Description	Size	Value
Function Code	1 Byte	0x03
Byte Count	1 Byte	2 x N*
Register value	N* x 2 Bytes	

*N = Quantity of Registers

Response, if Error (from Flyer):

Description	Size	Value
Error Code	1 Byte	0x83
Exception Code	1 Byte	01 or 02 or 03 or 04 or 06

==========

Read Flyer Inputs (*Read Input Registers*): Returns the current input state (registers) of the Flyer head. There is one valid address and one register:

Input Register Address = 0x0000

Request (from Client):

Description	Size	Value
Function Code	1 Byte	0x04
Starting Address	2 Bytes	0x0000
Quantity of Registers	2 Bytes	1

Response, no Error (from Flyer):

Description	Size	Value
Function Code	1 Byte	0x03
Byte Count	1 Byte	2 x N*
Input Registers	N* x 2 Bytes	

*N = Quantity of Registers

Response, if Error (from Flyer):

Description	Size	Value
Error Code	1 Byte	0x84
Exception Code	1 Byte	01 or 02 or 03 or 04 or 06

==========

Write Flyer Outputs (*Write Single Register*): Writes to the output register of the Flyer head. There is one valid address and one register:

Input Register Address = 0x0001

Request (from Client):

Description	Size	Value
Function Code	1 Byte	0x06
Starting Address	2 Bytes	0x0001
Register Value	2 Bytes	0x0000 to 0x00FF

Response, no Error (from Flyer):

Description	Size	Value
Function Code	1 Byte	0x06
Starting Address	2 Bytes	0x0001
Register Value	2 Bytes	0x0000 to 0x00FF

Response, if Error (from Flyer):

Description	Size	Value
Error Code	1 Byte	0x86
Exception Code	1 Byte	01 or 02 or 03 or 04 or 06

==========

Set Wait Digital: Waits until Flyer's current input state matches the input bits and input bit mask specified by user. This Event can wait indefinitely or timeout through a user-assigned timeout value. The input bits, mask, and timeout values are sent via an sDigitalEvent structure.

Important Note: Since Modbus is a request/reply 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 (0x6).

Request (from Client):

8 Bytes	4 Bytes	6 Bytes
sMBAP Header	SynHeader	sDigitalEvent

sMBAP Header→Length = Offset; where Offset = 12 bytes (sDigitalEvent + SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x23
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

sDigitalEvent Structure size = 6 Bytes:

Name	Size	Description
Bits	1 Byte	Bit value to wait for
Mask	1 Byte	The bit mask (valid inputs to check)
Timeout	4 Bytes (int)	Timeout value in milliseconds (-1 is infinite timeout)

Response, no Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success; Input bits match bit/mask values

Response, if Error (from Flyer): * this is a Modbus error

Description	Size	Value
Error Code	1 Byte	User-Defined Code + 0x80
Exception Code	1 Byte	06

==========

SynComm Modbus-Asynchronous Flyer Marking Head Functions

The SynComm Modbus-Asynchronous protocol is a special case of the standard 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 port number 35938 to ensure that communications do not occur over a Modbus port.
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 FH Flyer marking head control:

Modbus-Asynchronous Marking Events (page 41):

- 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 43):

- 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):

8 Bytes	4 Bytes	up to 248 Bytes
sMBAP Header	SynHeader	LogMessage (null-terminated string)

sMBAP Header→Length = Offset + LogMessage Length +1 (Null character); where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x10
SynHeader→SynErr = 0 Success

==========

End of Mark Event: Returns an sEOM structure for each End of Mark message generated by Flyer.
Response (from Flyer):

8 Bytes	4 Bytes	26 Bytes
sMBAP Header	SynHeader	sEOM

sMBAP Header→Length = Offset + size of sEOM; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x62
SynHeader→ErrCode = 0 Success

sEOM Structure:

Name	Size	Description
MarkStatus	2 Bytes	Decimal value indicating the Mark Status: Idle = 0 Marking = 1 Aborted = 2
Reserved	2 Bytes	
EOMResponse	4 Bytes	Hexadecimal value containing status information regarding the state of the Flyer head:
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)

==========

Modbus-Asynchronous I/O Events

Set Input Change: Sets the bit mask required to fire an **Input Change Event**. When an input bit changes value 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.

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

Request (from Client):

8 Bytes	4 Bytes	6 Bytes
sMBAP Header	SynHeader	sDigitalEvent

sMBAP Header→Length = Offset; where Offset = 12 bytes (sDigitalEvent + SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x60
SynHeader→SynErr = 0x0
SynHeader→WaitForEOM = 0x0

sDigitalEvent Structure size = 6 Bytes:

Name	Size	Description
Bits	1 Byte	NA
Mask	1 Byte	The bit mask (specific input bits to check)
Timeout	4 Bytes (int)	NA

Response, no Error (from Flyer):

8 Bytes	4 Bytes
sMBAP Header	SynHeader

sMBAP Header→Length = Offset; where Offset = 6 bytes (SynHeader + Unit Identifier + Function Code)
SynHeader→SynErr = 0x0 Success. Input Change Mask set

Response, if Error (from Flyer): * this is a Modbus error

Description	Size	Value
Error Code	1 Byte	User-Defined Code + 0x80
Exception Code	1 Byte	06

==========

Input Change Event: Returns current input bit status whenever an input that is part of the bit mask (set using **Set Input Change**) changes state. The bits are returned in the sDigitalEvent structure.

Response (from Flyer):

8 Bytes	4 Bytes	6 Bytes
sMBAP Header	SynHeader	sDigitalEvent

sMBAP Header→Length = Offset; where Offset = 12 bytes (sDigitalEvent + SynHeader + Unit Identifier + Function Code)
SynHeader→SynCode = 0x62
SynHeader→SynErr = 0x0 Success. Input state change detected; current bit values returned

sDigitalEvent Structure size = 6 Bytes:

Name	Size	Description
Bits	1 Byte	Returned input bit values
Mask	1 Byte	NA
Timeout	4 Bytes (int)	NA

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: 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 *Mark* 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.

ShiftCodesx 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.