

FF Link/B

Explicit Parameter Read / Write

Technical Application Note

A-FFL/B

Document No. D122-017

Document Revision 1.2

05/2024

CONTENTS

| | | |
|-----|--|----|
| 1 | Preface | 2 |
| 1.1 | Purpose of this Document | 2 |
| 1.2 | Additional Information | 2 |
| 1.3 | Support..... | 2 |
| 2 | Asynchronous Parameters | 3 |
| 3 | Explicit Block Parameter Messaging | 4 |
| 3.1 | Single Parameter Read / Write | 5 |
| 3.2 | Parameter List Read / Write | 11 |
| 4 | Decoding FMS Errors..... | 18 |



1 PREFACE

1.1 PURPOSE OF THIS DOCUMENT

This document will assist the user to configure Logix explicit messages to read and write H1 device block parameters.

1.2 ADDITIONAL INFORMATION

The following resources contain additional information that can assist the user with the module installation and operation.

| Resource | Link |
|---|---|
| Slate Installation | http://www.aparian.com/software/slate |
| FF Link User Manual FF Link Datasheet Example Code & UDTs | https://www.aparian.com/products/fflinkb |
| Ethernet wiring standard | www.cisco.com/c/en/us/td/docs/video/cds/cde/cde205_220_420/installation/guide/cde205_220_420_hig/Connectors.html |
| CIP Routing | The CIP Networks Library, Volume 1, Appendix C:Data Management |
| Modbus | http://www.modbus.org |
| FOUNDATION™ Fieldbus | FOUNDATION™ Fieldbus System Engineering Guidelines (AG-181) FOUNDATION™ Fieldbus Technical Overview (FD-043) www.fieldcommgroup.org/technologies/foundation-fieldbus |

1.3 SUPPORT

Technical support will be provided via the Web (in the form of user manuals, FAQ, datasheets etc.) to assist with installation, operation, and diagnostics.

For additional support the user can use either of the following:

| Resource | Link |
|---------------------|--|
| Contact us web link | www.aparian.com/contact-us |
| Support email | support@aparian.com |

2 ASYNCHRONOUS PARAMETERS

Asynchronous parameters can be used to read or write data from the Primary Interface to specific block parameters which are not accessible through the normal connector and wire method. Within each Macro Cycle a configurable number of asynchronous parameters will be executed (see the *Asynchronous Parameter Rate* in the general H1 configuration).

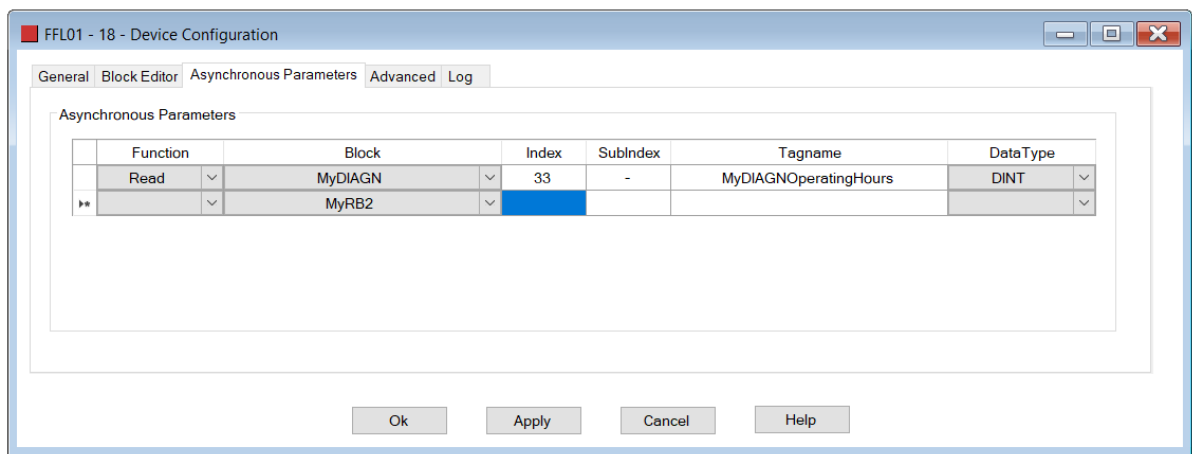


Figure 2.1 – Asynchronous Parameters

Although these block parameters are read, or written to, asynchronously from the connector compel strategy, they are executed continuously.

Reading block parameters continuously is relatively harmless, however, continuously writing certain block parameters may have a negative effect on the performance of the H1 device.

In most cases, it is often more useful to trigger the writing of a specific block parameter only when required. For example, when changing calibration figures, or perhaps changing dynamic factors as the application prepares to manufacture a different product.

3 EXPLICIT BLOCK PARAMETER MESSAGING

To simplify the explicit reading and writing of block parameters in Logix two L5X examples are provided:

- **Single Parameter Read/Write** (*FMSEExample.L5X*)
 - Example of a single parameter read /write.
- **Parameter List Read/Write** (*FMSListProcessing.L5X*)
 - Provides a method to execute a pre-configured list of parameter read/write instructions.

These both include an example routine, a number of tags, AOIs (Add-on-Instructions) and UDTs (User-Defined Data types).

The examples can be downloaded from the Aparian website here:

<https://www.aparian.com/products/fflinkb#downloads>

Both examples require the user to first note the specific device, block and parameter attributes for each block parameter they wish to read or write, including:

- Station Address
- Block Index
- Index
- Sub-Index
- Data Type

The Block Index can be viewed in Slate by right-clicking on the specific block (when online with the device) and selecting the **Header Details** option. The Object Index shown is the required Block Index.

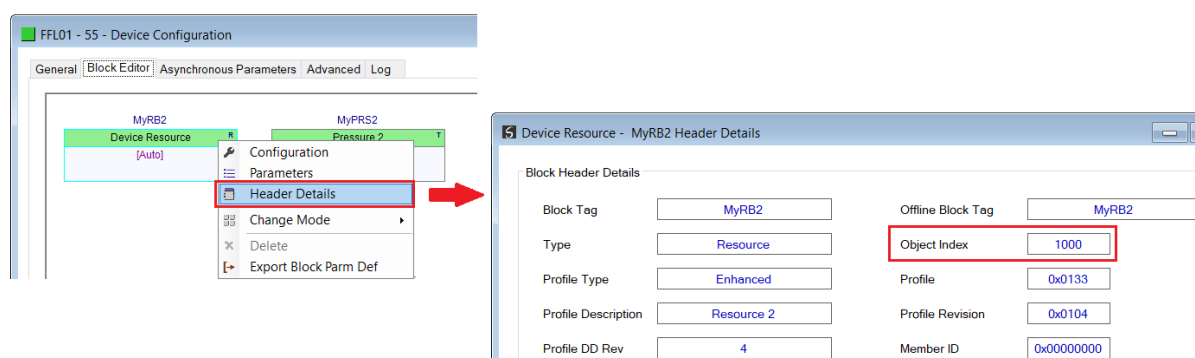
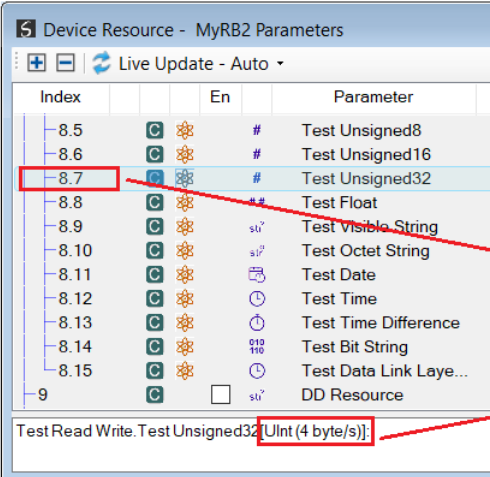


Figure 3.1 – Block Object Index

The parameter Index, SubIndex and Data Type can be retrieved from the Block Parameter window.



| Index | En | Parameter |
|-------|----|-------------------------|
| 8.5 | # | Test Unsigned8 |
| 8.6 | # | Test Unsigned16 |
| 8.7 | # | Test Unsigned32 |
| 8.8 | # | Test Float |
| 8.9 | st | Test Visible String |
| 8.10 | st | Test Octet String |
| 8.11 | st | Test Date |
| 8.12 | st | Test Time |
| 8.13 | st | Test Time Difference |
| 8.14 | st | Test Bit String |
| 8.15 | st | Test Data Link Layer... |
| 9 | st | DD Resource |

Test Read Write. Test Unsigned32 [Uint (4 byte/s)]

Figure 3.2 – Parameter Index, Sub-Index, and Data Type

3.1 SINGLE PARAMETER READ / WRITE

To import this example into your existing FF Link Studio 5000 project, right-click on a Program and select the **Import Routine** option.

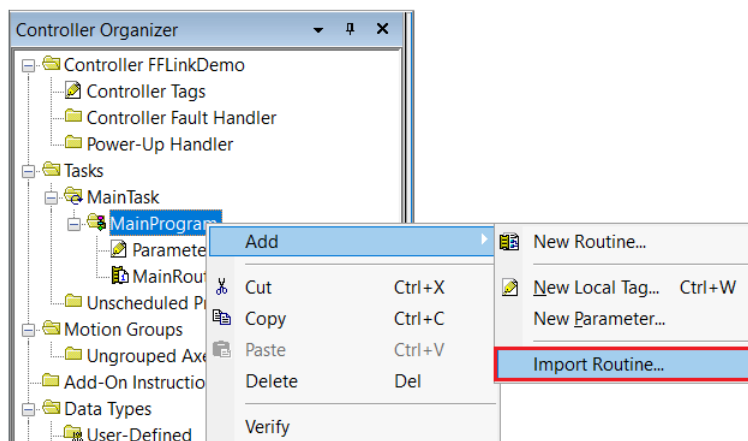


Figure 3.3 – Logix Routine Import – Single Parameter

Select the **FMSEExample.L5X** file.

NOTE: This file can be downloaded from the Aparian website:

<https://www.aparian.com/products/fflinkb#downloads>

In addition to the **FMSEExample** routine, the import will create a number of AOIs and UDTs as follows:

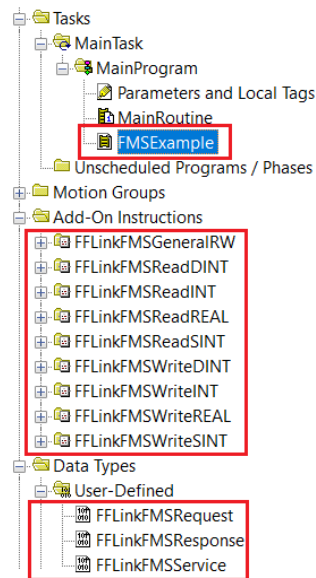


Figure 3.4 – Logix Imported Routine, AOI, and UDTs

The example shows the implementation of a Write REAL parameter, as follows:

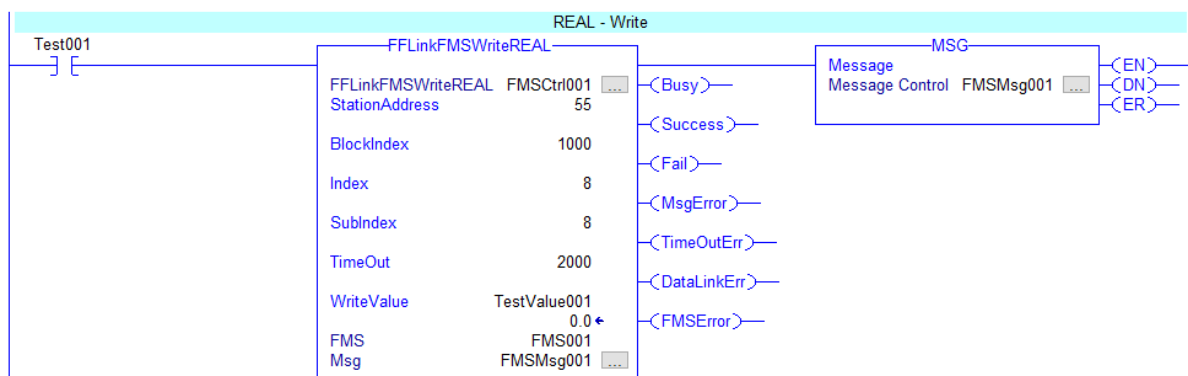


Figure 3.5 – FMS Write Real rung

Depending on the required function (Read/Write) and Data Type, this AOI should be substituted, as follows:

| <i>Data Type</i> | <i>Read AOI</i> | <i>Write AOI</i> |
|------------------|-------------------|--------------------|
| SINT | FFLinkFMSReadSINT | FFLinkFMSWriteSINT |
| INT | FFLinkFMSReadINT | FFLinkFMSWriteINT |
| DINT | FFLinkFMSReadDINT | FFLinkFMSWriteDINT |
| REAL | FFLinkFMSReadREAL | FFLinkFMSWriteREAL |

Table 3.1. – AOI Data Types

In the AOI, create a unique backing tag and enter the appropriate AOI parameters.

| <i>Parameter</i> | <i>Description</i> |
|------------------|---|
| StationAddress | The Station Address of the H1 device hosting the block. |
| BlockIndex | The online Block Index (address) of the block holding the required parameter. This index can be viewed in Slate, in the block Header Details. |
| Index | The parameter index of the required parameter. |
| SubIndex | The parameter subindex of the of the required parameter. NOTE: Select 0, for a parameter with no sub-parameters. |
| Timeout | The maximum amount of time in milliseconds the FF Link module will wait for an H1 device to respond to this FMS request. This should be typically twice the MacroCycle time. |
| ReadValue | For read messages, this will be the destination tag of the read value. |
| WriteValue | For write messages, this will be the source tag of the write value. |
| FMS | A tag of UDT type FFLinkFMSService which contains the raw Field Message Specification Request and Response structures. NOTE: This tag must be unique for each explicit message. |
| Msg | The Logix Message instruction tag. NOTE: This tag must be unique for each explicit message. |

Table 3.2. – AOI parameter structure

To configure the Logix Message instruction, select the “...” button adjacent to the tag.

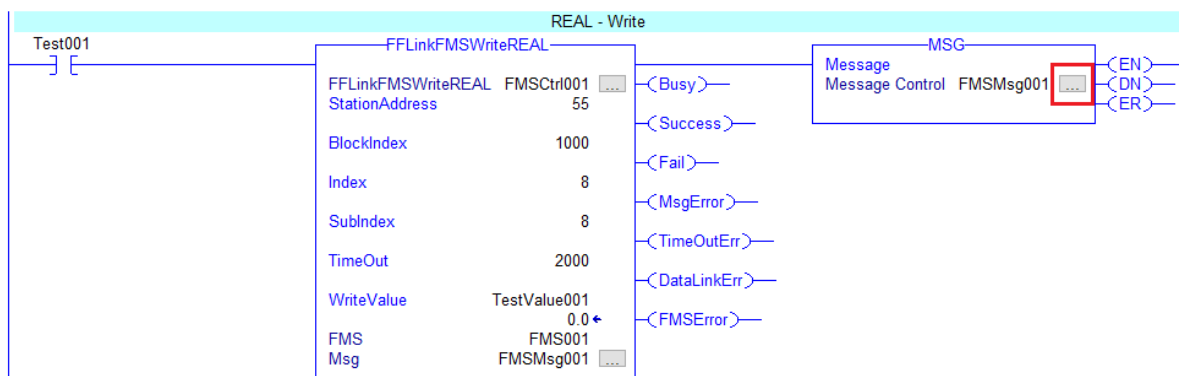


Figure 3.6 – FMS Write Real rung – Message Configuration

The message instruction must be configured as follows:

Figure 3.7 – Asynchronous Parameter Message Instruction Settings

| <i>Attribute</i> | <i>Value</i> |
|---------------------|--|
| Message Type | CIP Generic |
| Service Type | Custom |
| Service Code | 66 (hex) |
| Class | 434 (hex) |
| Instance | 1 |
| Attribute | 0 (hex) |
| Source | [FMS].Request (Where [FMS] represents the FMS tag configured in the AOI.) Example: FMS001.Request |
| Source Length | 32 |
| Destination Element | [FMS].Response (Where [FMS] represents the FMS tag configured in the AOI.) Example: FMS001.Request |

Table 3.3. – MSG instruction parameters

Ensure the Path in the Communication tab is set to the specific FF Link module.

Message Configuration - FMSMsg001

Configuration Communication Tag

☒ Path: FFL01 Browse...

☐ Broadcast

Communication Method

☒ CIP ☐ DH+ Channel: 'A' Destination Link: 0

☐ CIP With Source ID Source Link: 0 Destination Node: 0 (Octal)

☒ Connected ☐ Cache Connections ☐ Large Connection

☐ Enable ☐ Enable Waiting ☐ Start ☐ Done Done Length: 0

☐ Error Code: Extended Error Code: ☐ Timed Out

Error Path:
Error Text:

OK Cancel Apply Help

Figure 3.8 – Asynchronous Parameter Message Instruction Path

To execute the explicit message set the input condition (example Test001) to true.

The AOI will configure the message request and the Message sent to the FF Link module. Once the message completes, the AOI will process the result.

The following output parameters provide a status of the explicit message:

| Output Parameter | Description |
|------------------|--|
| Busy | The message instruction is waiting for a reply from the FF Link. |
| Success | The explicit message was successful. |
| Fail | The explicit message failed. |
| MsgError | The Logix message instruction failed. Check the communication path, and message instruction parameters (Class, Instance, Attribute etc.) |
| TimeOutErr | The FMS Message timed-out. Check the TimeOut parameter in the AOI, and confirm that the H1 device is online. |
| DataLinkErr | The message failed due to an error at the DataLink level. Check device station address. |
| FMSError | The message failed due to an error at the FMS level. Check Index, Sub Index and Data Type. Check the device's permissions. (Grant / Deny parameters in Resource Block) |
| FMSErrorCode | The raw FMS Error Code associated with the FMS Error. See the section on FMS Error decoding at the end of this document. |

Table 3.4. – Explicit Message Parameters

To re-trigger the explicit message, the input condition must first be set to false, and then to true.

3.2 PARAMETER LIST READ / WRITE

The Parameter List example is better suited for when multiple parameter read/write functions are required. Each parameter read/write function is configured by populating a UDT array. These can then be triggered **one at a time**, from either Logix application code or directly from an HMI (Human Machine Interface).

To import this example into your existing FF Link Studio 5000 project, right-click on a Program and select the **Import Routine** option.

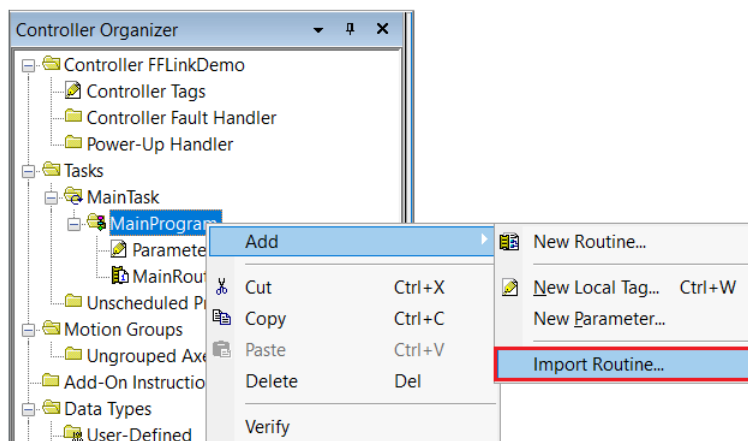


Figure 3.9 – Logix Import – Parameter List

Select the **FMSListProcessing.L5X** file.

Note: This file can be downloaded from the Aparian website:

<https://www.aparian.com/products/fflinkb#downloads>

In addition to the **FMSListProcessing** routine, the import will create a number of AOIs and UDTs as follows:

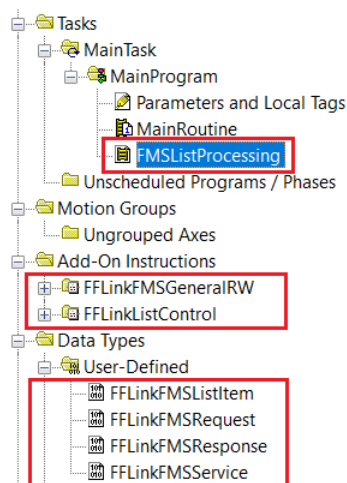


Figure 3.10 – Imported Router, AOI, and UDTs

The **FMSListProcessing** routine contains a single rung that processes the entire list of parameters.

This rung, and all the referenced tags, should be multiplied for each FF Link module that requires explicit parameter read/write functions.

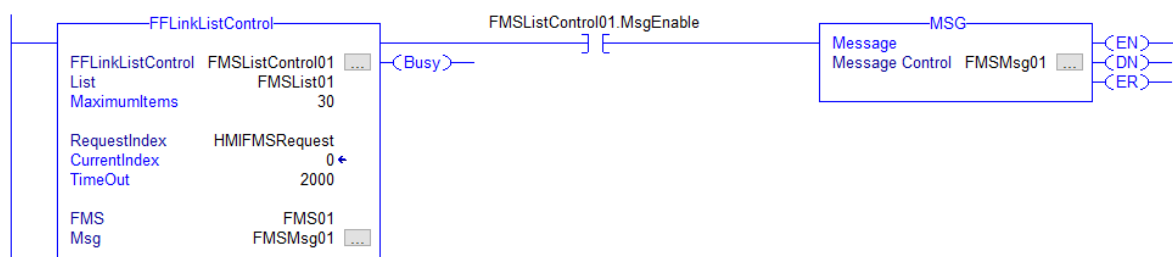


Figure 3.11 – List Processing rung

In addition to the AOI backing tag (e.g. FMSListControl01), the **FFLinkListControl** AOI has the following parameters:

| Parameter | Description |
|--------------|---|
| List | This tag must be an array of the UDT type FFLinkFMSListItem . The size of the array must be at least one greater than the required number of parameters. e.g. FMSList01 Note: The list array index 0 must not be used. |
| MaximumItems | This value should be equal or less to the dimension of the above array. |
| RequestIndex | This tag should be of an INT type and is used to trigger the parameter read/write. Its value corresponds to the position in the above array. For example, to trigger the parameter action defined in position 3, e.g. FMSList01[3], the application code / HMI should set the RequestIndex tag (e.g. HMIFMSRequest) to a value of 3. |

| | |
|--------------|---|
| | Once the action has been processed by the AOI, this tag will be reset to a value of 0. Note: The list array index 0 must not be used. |
| CurrentIndex | This is an AOI output that displays the parameter index that is currently being processed. |
| Timeout | The maximum amount of time in milliseconds the FF Link module will wait for an H1 device to respond to this FMS request. This should be typically twice the MacroCycle time. |
| FMS | A tag of UDT type FFLinkFMSService which contains the raw Field Message Specification Request and Response structures. |
| Msg | The Logix Message instruction tag. |

Table 3.5. – AOI parameters

To configure the Logix Message instruction, select the “...” button adjacent to the tag.

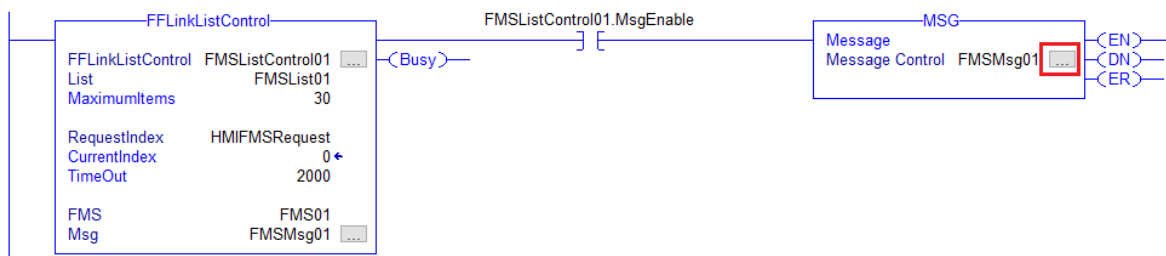


Figure 3.12 – List Processing rung – Message Configuration

The message instruction must be configured as follows:

Figure 3.13 – Asynchronous Parameter Message Instruction Settings

| <i>Attribute</i> | <i>Value</i> |
|---------------------|---|
| Message Type | CIP Generic |
| Service Type | Custom |
| Service Code | 66 (hex) |
| Class | 434 (hex) |
| Instance | 1 |
| Attribute | 0 (hex) |
| Source | [FMS].Request (Where [FMS] represents the FMS tag configured in the AOI.) Example: FMS01.Request |
| Source Length | 32 |
| Destination Element | [FMS].Response (Where [FMS] represents the FMS tag configured in the AOI.) Example: FMS01.Request |

Table 3.6. – Message Instruction Parameters

Ensure the Path in the Communication tab is set to the specific FF Link module.

Message Configuration - FMSMsg01

Configuration Communication Tag

Path:

FFL01

☐ Broadcast

Communication Method

☒ CIP ☐ DH+ Channel: Destination Link:

☐ CIP With Source ID Source Link: Destination Node: (Octal)

☐ Connected ☐ Cache Connections ☐ Large Connection

☐ Enable ☐ Enable Waiting ☐ Start ☐ Done Done Length:

☐ Error Code: Extended Error Code: ☐ Timed Out

Error Path:

Error Text:

Figure 3.14 – Asynchronous Parameter Message Instruction Path

The configuration of each parameter read/write function resides in the list array.

NOTE: Index 0 of the array must not be used.

| Name | Value | Style | Data Type |
|-------------------------------|--------------|---------|-----------------------|
| FMSList01 | { ... } | | FFLinkFMSListItem[40] |
| + FMSList01[0] | { ... } | | FFLinkFMSListItem |
| - FMSList01[1] | { ... } | | FFLinkFMSListItem |
| + FMSList01[1].StationAddress | 55 | Decimal | INT |
| + FMSList01[1].Command | 1 | Decimal | INT |
| + FMSList01[1].BlockIndex | 1000 | Decimal | INT |
| + FMSList01[1].Index | 8 | Decimal | INT |
| + FMSList01[1].SubIndex | 8 | Decimal | INT |
| + FMSList01[1].DataType | 3 | Decimal | INT |
| - FMSList01[1].ValueREAL | 956.11 | Float | REAL |
| + FMSList01[1].ValueDINT | 0 | Decimal | DINT |
| + FMSList01[1].ValueINT | 0 | Decimal | INT |
| + FMSList01[1].ValueSINT | 0 | Decimal | SINT |
| + FMSList01[1].Status | 0 | Decimal | INT |
| - FMSList01[1].Busy | 0 | Decimal | BOOL |
| - FMSList01[1].Success | 1 | Decimal | BOOL |
| - FMSList01[1].Fail | 0 | Decimal | BOOL |
| - FMSList01[1].MsgError | 0 | Decimal | BOOL |
| - FMSList01[1].DataLinkError | 0 | Decimal | BOOL |
| - FMSList01[1].FMSError | 0 | Decimal | BOOL |
| - FMSList01[1].DataTypeError | 0 | Decimal | BOOL |
| - FMSList01[1].TimeOutError | 0 | Decimal | BOOL |
| + FMSList01[1].FMSErrorCode | 16#0000_0000 | Hex | DINT |

Figure 3.15 – Logix Tag structure

The UDT members are defined as follows:

| <i>Parameter</i> | <i>Description</i> |
|------------------|---|
| StationAddress | The Station Address of the H1 device hosting the block. |
| Command | The command of the parameter function: 1 = Read parameter (from H1 device) 2 = Write parameter (to H1 device) |
| BlockIndex | The online Block Index (address) of the block holding the required parameter. This index can be viewed in Slate, in the block Header Details. |
| Index | The parameter index of the required parameter. |
| SubIndex | The parameter subindex of the of the required parameter. Note: Select 0, for a parameter with no sub-parameters. |
| DataType | The DataType of the parameter: 0 = SINT 1 = INT 2 = DINT 3 = REAL |
| ValueREAL | Used for Read/Write functions when the DataType is set to REAL. In the case of a write, this value must be set before the message is triggered. In the case of a read, this tag is populated with the parameter value read. |
| ValueDINT | Used for Read/Write functions when the DataType is set to DINT. In the case of a write, this value must be set before the message is triggered. In the case of a read, this tag is populated with the parameter value read. |

| | |
|---------------|---|
| ValueINT | Used for Read/Write functions when the DataType is set to INT. In the case of a write, this value must be set before the message is triggered. In the case of a read, this tag is populated with the parameter value read. |
| ValueSINT | Used for Read/Write functions when the DataType is set to SINT. In the case of a write, this value must be set before the message is triggered. In the case of a read, this tag is populated with the parameter value read. |
| Status | Status (INT) returned by the FF Link. Bit 0 – Success Bit 1 – Timeout Bit 2 – DataLink Error Bit 3 – InvokeID Mismatch Bit 4 – FMS Error |
| Busy | The message instruction is waiting for a reply from the FF Link. |
| Success | The explicit message was successful. |
| Fail | The explicit message failed. |
| MsgError | The Logix message instruction failed. Check the communication path, and message instruction parameters (Class, Instance, Attribute etc.) |
| DataLinkErr | The message failed due to an error at the DataLink level. Check the device station address. |
| FMSError | The message failed due to an error at the FMS level. Check Index, Sub Index and Data Type. Check the device's permissions. (Grant / Deny parameters in Resource Block) |
| DataTypeError | The parameter value did not match that specified in the request. |
| TimeOutErr | The FMS Message timed-out. Check the TimeOut parameter in the AOI, and confirm that the H1 device is online. |
| FMSErrorCode | The raw FMS Error Code associated with the FMS Error. See the section of FMS Error decoding at the end of this document. |

Table 3.7. – UDT structure

The figure below illustrates which elements in each FMSList item are to be configured by the user and which are updated by the AOI.

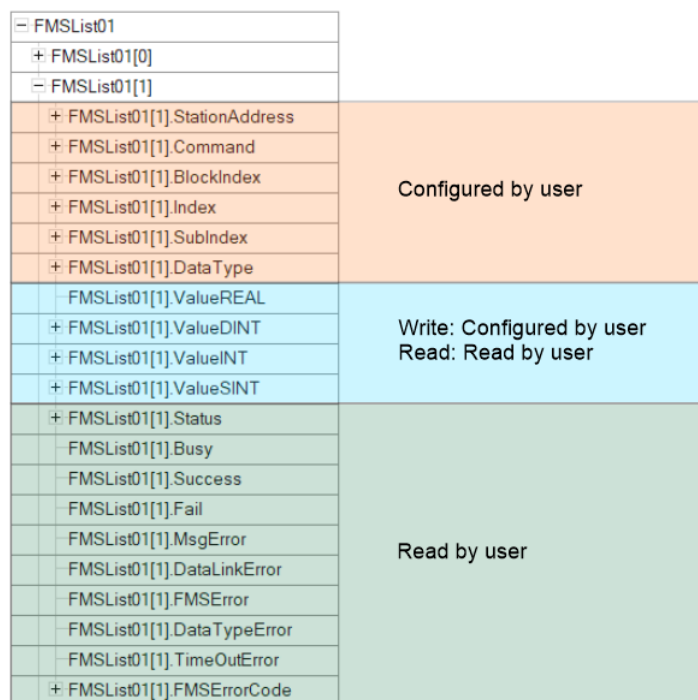


Figure 3.16 – Logix Tag structure regions

To trigger a specific parameter read/write function, write the required array index into the tag specified in AOI parameter RequestIndex. (e.g. HMIFMSRequest).

Once the action has been processed by the AOI, this tag will be reset to a value of 0.

NOTE: The list array index 0 cannot be used.

Since these parameters are processed one-at-a-time, it is important to wait until the current action has completed before initiating the next one.

The explicit parameter functions can be triggered by Logix application code or directly from the HMI. An example below shows an FTVView display which allows the user to trigger a number of these parameter functions.

| winMain - /FMS// (Display) | | | | | | | | | |
|----------------------------|---------|-------|-------|----------|-----------|--------|--------|----|---------|
| Description | Station | Block | Index | SubIndex | Data Type | Action | Value | | Status |
| Test Float 8.8 | 55 | 1000 | 8 | 8 | REAL | READ | 55.000 | Go | Success |
| Test Float 8.8 | 55 | 1000 | 8 | 8 | REAL | WRITE | 55.000 | Go | Success |
| Test DINT 8.4 | 55 | 1000 | 8 | 4 | DINT | READ | 56 | Go | Success |
| Test DINT 8.4 | 55 | 1000 | 8 | 4 | DINT | WRITE | 56 | Go | Success |
| Test INT 8.3 | 55 | 1000 | 8 | 3 | INT | READ | 78 | Go | Success |
| Test INT 8.3 | 55 | 1000 | 8 | 3 | INT | WRITE | 78 | Go | Success |
| Test SINT 8.2 | 55 | 1000 | 8 | 2 | SINT | READ | 11 | Go | Success |
| Test SINT 8.2 | 55 | 1000 | 8 | 2 | SINT | WRITE | 11 | Go | Success |

Figure 3.17 – Example HMI Display

4 DECODING FMS ERRORS

The FMS Errors are encoded using a series of *TypeInfo* structures, where the first structure is of type Sequence. The structure format is as follows:

| Bits | | | | | | | |
|------|-----|---|---|--------|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | Tag | | | Length | | | |

The first structure will have the Tag set to zero and the length of 1. So, a value of 0x81. The next structure will have the Tag value set to the **Error Class**, and the length will indicate the length of the **Error Code** to follow (typically 1). The next byte will be the actual **Error Code**.

This can be simplified as follows:

| Byte | Bits | | | | | | | |
|------|-------------|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | Error Class | | | | 0 | 0 | 0 | 1 |
| 2 | Error Code | | | | | | | |
| 3 | n/a | | | | | | | |

The FMS Error Class and Code can then be evaluated using the table below.

For example:
The FMSError bit is set, and the raw FMSErrorCode is 0x66086181.

| Name | Value | Style | Data Type |
|---------------------------|--------------|-------|-----------|
| + FMSCtrl005.FMSErrorCode | 16#6608_6181 | Hex | DINT |

This decodes to an Error Class of 6 (Access), and Error Code of 8 (Type Conflict).

| Byte | Value | Bits | | | | | | | |
|------|-------|------|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0x81 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0x61 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0x08 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Error Class = 6
Error Code = 8

| <i>Class</i> | <i>Class Description</i> | <i>Error Code</i> | <i>Description</i> |
|--------------|--------------------------|-------------------|--------------------------------|
| 1 | VFD State | - | - |
| 2 | Application Reference | 0 | Other |
| | | 1 | Application Unreachable |
| 3 | Definition | 0 | Other |
| | | 1 | Object Undefined |
| | | 2 | Object Attributes Inconsistent |
| | | 4 | Name Already Exists |
| 4 | Resource | 0 | Other |
| | | 1 | Memory Unavailable |
| 5 | Service | 0 | Other |
| | | 1 | Object State Conflict |
| | | 2 | PDU Size |
| | | 3 | Object Constraint Conflict |
| | | 4 | Parameter Inconsistent |
| | | 5 | Illegal Parameter |
| 6 | Access | 0 | Other |
| | | 1 | Object Invalidated |
| | | 2 | Hardware Fault |
| | | 3 | Object Access Denied |
| | | 4 | Invalid Address |
| | | 5 | Object Attribute Inconsistent |
| | | 6 | Object Access Unsupported |
| | | 7 | Object Non-Existent |
| | | 8 | Type Conflict |
| | | 9 | Named Access Unsupported |
| | | 10 | Access to Element Unsupported |
| 7 | Object Dictionary | 0 | Other |
| | | 1 | Name Length Overflow |
| | | 2 | OD Overflow |
| | | 3 | OD Write Protected |
| | | 4 | Extension Length Overflow |
| | | 5 | OD Description Length Overflow |
| | | 6 | Operational Problem |
| 8 | Other | - | - |

Table 4.1. – FMS Error Class and Error Codes