A Sierra Monitor Company

Driver Manual

(Supplement to the FieldServer Instruction Manual)

FS-8704-14 EtherNet/IP

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after November 2014

Driver Version:	1.09
Document Revision:	3

TABLE OF CONTENTS

1 ETHERNET/IP DESCRIPTION

The Ethernet IP driver allows the FieldServer to transfer data to and from devices over Ethernet using the EtherNet/IP protocol. The FieldServer can emulate either a Server or Client.

EtherNet/IP uses CIP (Control and Information Protocol), the common network, transport and application layers also shared by ControlNet and DeviceNet. EtherNet/IP then makes use of standard Ethernet and TCP/IP technology to transport CIP communications packets. The result is a common, open application layer on top of open and highly popular Ethernet and TCP/IP protocols.

The Driver is able to read/write using the Data Table structure employed by all Logix Series PLC's.

PCCC support is also provided for legacy devices that do not fully support CIP encapsulation. EIP PCCC Encapsulation was tested at the FST factory using a PLC5 I785 ENET card. The following data types were tested:

- N
- F
- S

The Driver also supports PCCC communication on SLC and MicroLogix (Tested on MicroLogix 1400 Device)

Fragmented Services (0x52) is supported for data_table read and write operations.

The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer.

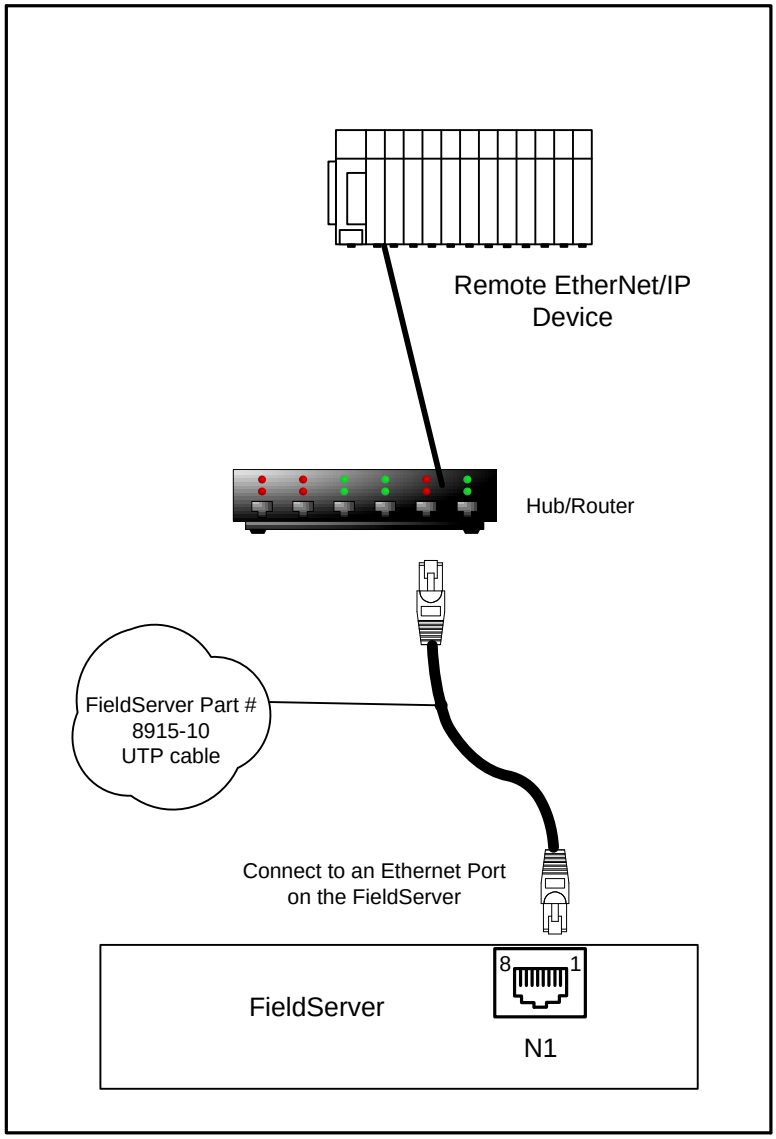
2 DRIVER SCOPE OF SUPPLY

2.1 Supplied by FieldServer Technologies for this driver

FieldServer Technologies PART #	Description
FS-8915-10	UTP cable (7 foot) for Ethernet connection

3 HARDWARE CONNECTIONS

It is possible to connect an EtherNet/IP device to either port N1 or N2[1] on the FieldServer. These ports must just be configured to use EtherNet/IP in the configuration file.



[1] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

4 DATA ARRAY PARAMETERS

Data Arrays are "protocol neutral" data buffers for storage of data to be passed between protocols. It is necessary to declare the data format of each of the Data Arrays to facilitate correct storage of the relevant data.

Section Title
Data_Arrays

Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	FLOAT, BIT, UInt16, SInt16, Byte.
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	1-10, 000

Example

```
//  Data Arrays
Data_Arrays
Data_Array_Name   , Data_Array_Format   , Data_Array_Length
DA_AI_01          , UInt16              , 200
DA_AO_01          , UInt16              , 200
DA_DI_01          , Bit                 , 200
DA_DO_01          , Bit                 , 200
DA_OUTPUTS        , UInt16              , 3
DA_INPUTS         , UInt16              , 8
DA_Config         , UInt16              , 1
```

5 CONFIGURING THE FIELDSERVER AS AN ETHERNET/IP CLIENT

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer.

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an EtherNet/IP Server.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for EtherNet/IP communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination device addresses need to be declared in the "Client Side Nodes" section, and the data required from the servers needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

5.1 Client Side Connection Parameters

Section Title		
Adapter		
Column Title	Function	Legal Values
Adapter	Adapter Name	N1, N2[2]
Protocol	Specify protocol used	EtherNet/IP

Example

```
// Client Side Connections

Adapters
Adapter                    , Protocol
N1                         , EtherNet/IP
```

5.2 Client Side Node Parameters [3]

Section Title		
Nodes		
Column Title	**Function**	**Legal Values**
Node_Name	Provide name for node	Up to 32 alphanumeric characters
IP_Address	Address of Server	Any valid address on subnet
Protocol	Specify protocol used	EtherNet/IP
Adapter	Specify port Adapter used	N1, N2[2]
Plc_Type*	Specify the type of remote PLC. Only required for PCCC Services	MicroLogix, SLC5, **PLC5**

[2] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

[3] Only one explicit connection is created per node. All explicit Map Descriptors attached to that node will use the same explicit connection.

EIP_Cache_Connection*	Specify if connection to remote server should be reused or closed after every data transfer transaction.	**YES**, NO
EIP_Skip_Cmd*	Some devices do not support all 'under the hood' commands specified by the EIP driver. Specify the command that should not be issued by the FieldServer.	**-,** LIST_IDENTIFY

Example

```
//  Client Side Nodes

Nodes
Node_Name          , IP_Address       , Adapter   , Protocol
PLC 1              , 192.168.1.174    , N1        , EtherNet/IP
```

5.3 Client Side Map Descriptor

5.3.1 FieldServer Related Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from Section 4
Data_Array_Offset	Starting location in Data Array	0 to (Data_Array_Length -1) as specified in Section 4.
Function	Function of Client Map Descriptor	Rdbc, Wrbc, Wrbx Note: It is possible with Data Table Read on the Client side to read and write to the same tag by using the "Write Thru" property of the Rdbc function.

5.3.2 Driver Related Map Descriptor Parameters – Unconnected Messages

Column Title	Function	Legal Values
EIP_Service	The action to be performed.	Get_Attrib, Set_Attrib
EIP_Class	Class to be polled.	One of the classes supported by the driver. Refer to 0
EIP_Attribute	Attribute associated with the class given.	See particular attributes of each class. Refer to 0
EIP_Con_Typ	The type of data transfer required. Also referred to as the “Transport Method”	Unconnected Explicit
EIP_Path*	Used to stipulate the path to the CPU in certain PLC’s. Paths	Any space delimited numerical

Column Title	Function	Legal Values
	vary and are dependent on the structure of the network.	value. Refer to vendor's device documentation. Also see Appendix A.3, **0 0**
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to Appendix D for further information.	For any given Map Descriptor there can be 200 Floats, 400 Integers or 800 Bytes
Address	Instance of the class to be polled.	Depends on the supported instances for each class.

5.3.3 Driver Related Map Descriptor Parameters – Data Table Read/Write.

Column Title	Function	Legal Values
EIP_Service	The action to be performed.	Data_Table_Read, Data_Table_Write
EIP_Con_Typ	The type of data transfer required.	Explicit
EIP_Path*	Used to stipulate the path to the CPU in certain PLC's. Paths vary and are dependent on the structure of the network.	Any space delimited numerical value. Refer to vendor's device documentation. Also see Appendix A.3, **0 0**
EIP_Tag_Name	Tag name expressed in PLC program. The data type of this parameter is used to set the data format of the Data Array if the EIP_DATA_TYPE parameter is not specified.	Maximum length 48 characters.
EIP_Data_Type*	This parameter can be used to force the data type of the tag to match the data type used in the remote device. If this parameter is not specified the Data Type of the Data Array will be used. Refer to 0 for more information. Data Types can be specified in either FieldServer or Rockwell Data Type. Refer to Appendix C.4 for more information.	Value Alias (As used in PLC) Float REAL Uint32 DINT Uint16 INT BYTE SINT BIT BOOL & BOOLEAN
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. See Appendix D for further information.	0 to (Data_Array_Length -1) as specified in Section 4.
EIP_Structure_Handle*	This parameter is required to read/write structures. The driver supports read/write structures having members of same type, i.e. all members are of type Byte, UINT16, UINT32 or Float etc When this parameter is defined, the number of structure members must be specified as the length of the Map Descriptor. Refer to Appendix A.7 for more information.	Any 16bit Integer number (e.g. 59592), **0**

5.3.4 Driver Related Map Descriptor Parameters – PCCC

Column Title	Function	Legal Values
EIP_Service	Action to be performed	Exec_PCCC (Encapsulation using Allen Bradley PCCC)
EIP_Con_Typ	The type of data transfer required	Explicit
EIP_Path*	Used to stipulate the path to the CPU in certain PLC's. Paths vary and are dependent on the structure of the network.	Any space delimited numerical value. Refer to vendor's device documentation. Also see Appendix A.3, **0 0**
File_Type	Allen Bradley file type	N Integer F Float O Output B Boolean I Input S Status
File_Number	Allen Bradley file number	Any valid numerical value
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to Appendix D for further information.	For any given Map Descriptor there can be 200 Floats, 400 Integers or 800 Bytes
Address	Address in the file	Any valid numerical value between 0 to 255

5.3.5 Driver Related Map Descriptor Parameters – Implicit IO Messages

Column Title	Function	Legal Values
EIP_Con_Typ	The type of data transfer required. Also referred to as the "Transport Method"	Implicit
EIP_Class	EIP class to be polled.	Integer value. In most cases it will be the assembly class i.e. value will be 4
Address	Production/Consumption connection point or Instance of the class.	Integer value depending upon server configuration
Length	Number of data items to be polled	For any given Map Descriptor there can be 125 Floats/32bit Integers, 250 Integers or 500 Bytes
Parent_Map_Descriptor	Specify the name of previously created Map Descriptor to which this Map Descriptor should be linked.	Use – for no setting, or specify name of other Map Descriptor

5.3.6 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is polled	≥0.001s

5.3.7 Map Descriptor Example 1: Unconnected Messages

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Scan_Interval , Data_Array_Name , Data_Array_Offset , Function , EIP_Con_Typ , Node_Name , EIP_Class , Address , EIP_Attribute , EIP_Service , Length
CMD_PRO_03            , 0s            , DA_AI_01        , 0                 , Rdbc     , Unconnected  , EIP_01    , 10         , 1       , 3             , Get_Attrib   , 1
CMD_PRO_02            , 0s            , DA_AI_01        , 1                 , Rdbc     , Unconnected  , EIP_01    , 10         ,         , 2             , 3            , Get_Attrib , 1
```

5.3.8 Map Descriptor Example 2: Data Table Messages

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Scan_Interval , Data_Array_Name , Data_Array_Offset , EIP_Con_Typ , Node_Name , Function , EIP_Service    , EIP_Path , EIP_Tag_Name , Length
Cmd_Pro_09            , 0s            , DA_AI_05        , 0                 , Explicit     , EIP_01    , Rdbc     , Data_Table_Read , 1 1      , analog_in_3  , 2
Cmd_Pro_10            , 0s            , DA_AI_06        , 0                 , Explicit     , EIP_01    , Rdbc     , Data_Table_Read , 1 1      , analog_in_4  , 2
```

5.3.9 Map Descriptor Example 3: PCCC Messages

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name , Data_Array_Offset, , Function , EIP_Con_Typ , Node_Name , EIP_Service , EIP_Path , File_Type , File_Number , Address , Length
CMD_01                , DA_F_01         , 0,                , Rdbc     , Explicit     , EIP_01    , Exec_PCCC   , 1 0      , F         , 8,           , 30      , 10
```

5.3.10 Map Descriptor Example 4: Implicit IO Messages

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Scan_Interval , Data_Array_Name , Data_Array_Offset , Function , EIP_Con_Typ , Node_Name , EIP_Class , Address , Length , Parent_Map_Descriptor
CMD_Producer          , 1s            , DA_OUTPUTS      , 0                 , Wrbc     , Implicit     , EIP_01    , 4         , 101     , 3      , -
CMD_Consumer          , 1s            , DA_INPUTS       , 0                 , Passive  , Implicit     , EIP_01    , 4         , 102     , 8      , CMD_Producer
CMD_Config            , -             , DA_Config       , 0                 , Passive  , Implicit     , EIP_01    , 4 ,       , 103     , 1      , CMD_Producer
```

The config Map Descriptor (Rockwell calls it Configuration) is optional for FieldServer (as per EIP specs), but it is used by Controllogix and technically it is an instance of Assembly Class.

The address numbers are random, but should be according to Server information; at what numbers server will consume and produce data.

To open an implicit connection, both Producer and Consumer connection information is required. Producer is an active Map Descriptor, so consumer should be linked to producer.

6 CONFIGURING THE FIELDSERVER AS AN ETHERNET/IP SERVER

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer.

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an EtherNet/IP Client.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for EtherNet/IP communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the FieldServer virtual node(s) needs to be declared in the "Server Side Nodes" section, and the data to be provided to the Client needs to be mapped in the "Server Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

6.1 Server Side Connection Parameters

Section Title
Connections

Column Title	Function	Legal Values
Adapter	Adapter Name	N1, N2[4]
Protocol	Specify protocol used	EtherNet/IP

Example

```
// Server Side Connections

Adapters
Adapter                    , Protocol
N1                         , EtherNet/IP
```

6.2 Server Side Node Parameters [3]

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for Node	Up to 32 alphanumeric characters
Protocol	Specify protocol used	EtherNet/IP
Server_Hold_Timeout*	Specifies time FieldServer will reserve server side connection while waiting for the Client side to update data.	>1.0s
IP_Address	Specify multicast IP_Address to which the production data should be sent. It will be used only if client requested multicast connection. This parameter is only required for Implicit IO Messages.	IP Address

[3] Only one explicit connection is created per node. All explicit Map Descriptors attached to that node will use the same explicit connection.
[4] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

Example

```
//  Server Side Nodes

Nodes
Node_Name          , Protocol
EIP_01             , EtherNet/IP
```

6.3 Server Side Map Descriptor

6.3.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from Section 4
Data_Array_Offset	Starting location in Data Array	0 to (Data_Array_Length -1) as specified in Section 4.
Function	Function of Server Map Descriptor	Passive
Server_Hold_Timeout*	Specifies the length of time that the FieldServer will reserve the Server side connection while waiting for the Client side to update data in Data Array (if necessary)	>1.0s

6.3.2 Server Specific Map Descriptor Parameters – Unconnected Messages

Column Title	Function	Legal Values
EIP_Service	The action to be performed.	Get_Attrib, Set_Attrib
EIP_Class	Class to be served.	One of the classes supported by the driver. Refer to Appendix D.2
EIP_Attribute	Attribute associated with the class served.	See particular attributes of each class. Refer to Appendix D.2
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to Appendix D for further information.	For any given Map Descriptor there can be 200 Floats, 400 Integers or 800 Bytes
Address	Instance of the class to be served.	Depends on the supported instances for each class.

6.3.3 Server Specific Map Descriptor Parameters – Data Table Read/Write.

Column Title	Function	Legal Values
EIP_Service	The action to be performed.	Data_Table_Read, Data_Table_Write Note: Separate Map Descriptors need to be configured for each service.
EIP_Tag_Name	Tag name expressed in PLC program. The data type of this parameter is used to set the data format of the Data Array if the EIP_Data_Type parameter is not specified.	Maximum length 48 characters.
EIP_Data_Type*	If set, the data will be stored in the specified format which may be different to the format of the tag being polled. If the parameter is not set, the data type of the Data Array will be used. This is only applicable to Data Table Read when FieldServer is the Server. Refer to 0 for further information.	Float, Uint16, Uint32, Bit, Byte, Boolean, **-**
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to Appendix D for further information.	0 to (Data_Array_Length - 1) as specified in Section 4.
EIP_Structure_Handle*	This parameter is required only for read structures i.e. where EIP_Service is Data_Table_Read. The driver supports read structures having members of same type, i.e. all members are of type Byte, Uint16, Uint32 or Float etc. When this parameter is defined, the number of structure members must be specified as the length of the Map Descriptor.	Any 16bit Integer number e.g. 59592, **0**

6.3.4 Driver Related Map Descriptor Parameters – PCCC

Column Title	Function	Legal Values
EIP_Con_Typ	The type of data transfer required	Explicit
EIP_Service	Action to be performed	EXEC_PCCC (Encapsulation using Allen Bradley PCCC)
File_Type	Allen Bradley file type	N Integer F Float O Output B Boolean I Input S Status
File_Number	Allen Bradley file number	Any valid numerical value
Length	Number of data elements to be mapped. If the number of data elements exceeds the Map Descriptor length, the list of data elements will be truncated and an error message will be printed once per Map Descriptor. Refer to Appendix D for further information.	For any given Map Descriptor there can be 61 Floats, 122 Integers or 244 Bytes. .
Address	Address in the file	Any valid numerical value between 0 to 255

6.3.5 Server Specific Map Descriptor Parameters – Implicit IO Messages

Column Title	Function	Legal Values
EIP_Con_Typ	The type of data transfer required. Also referred to as the "Transport Method"	Implicit
EIP_Class	Class to be served.	Integer value In most cases it will be the assembly class i.e. 4
EIP_Attribute	Attribute associated with the class served.	Integer value. In most cases it will be data attribute 3. Refer to Appendix D.2 for legal attribute values
Length	Number of data elements to be mapped.	1 to Data Array length
Address	Production or Consumption connection point address.	Any integer number

6.3.6 Map Descriptor Example 1: Unconnected Messages

```
// Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name , EIP_Class , Address , EIP_Attribute , EIP_Service , Length
SMD_PRO_01          , DA_AI_01        , 0                 , Passive  , EIP_01    , 10        , 1       , 3             , Get_Attrib   , 1
SMD_PRO_02          , DA_AI_01        , 1                 , Passive  , EIP_01    , 10        , 2       , 3             , Get_Attrib   , 1
```

6.3.7 Map Descriptor Example 2: Data Table Messages

```
// Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Node_Name , Function , EIP_Service     EIP_Tag_Name , Length
SMD_PRO_09          , DA_AI_05        , 0                 , EIP_01    , Passive  , Data_Table_Read , Analog_in_3 , 2
SMD_PRO_10          , DA_AI_06        , 0                 , EIP_01    , Passive  , Data_Table_Read , Analog_in_4 , 2
```

6.3.8 Map Descriptor Example 3: PCCC Messages

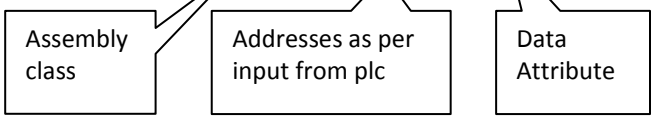
```
// Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , EIP_Con_Typ , Node_Name , EIP_Service , File_Type , File_Number , Address , Length
SRV_AI_01           , DA_F_01         , 0                 , Passive  , Explicit     , EIP_01    , Exec_PCCC   , F         , 8           , 30      , 20
```

6.3.9 Map Descriptor Example 4: Implicit IO Messages

```
// Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name , EIP_Class , Address , EIP_Attribute , Length
SMD_Consumer        , DA_INPUTS       , 0                 , Server   , EIP_01    , 4         , 101     , 3             , 3
SMD_Producer        , DA_OUTPUTS      , 0                 , Server   , EIP_01    , 4         , 102     , 3             , 8
```



Appendix A. Useful Features

Appendix A.1. General Notes

- The connection type does not need to be specified in the Server side Map Descriptor, but must be stipulated on the Client side of the driver.
- Data_Table_Read as a service can only be used when creating an explicit connection.

Appendix A.2. FieldServer as an Adapter and Scanner.

It is possible for the FieldServer to act as a scanner and an adapter at the same time so long as the scanner and adapter are configured on different ports. Consequently this functionality is not possible on an FS-X20 platform.

Appendix A.3. Common Paths

Device	Typical Path
Direct AB	1 0
AB ENI module	3 1
AB ControlLogix 1756-L55 (With network card 1756-ENBT/A)	1 1 or 1 0
CompactLogix ENI (1769-L31 using the 1761-NET-ENI)	3 1
CompactLogix Direct Connection (P/N 1769-L35E)	1 1 or 1 0

Appendix A.4. Setting the Data Type for stored data.

The default Data Type of stored data is determined by the Data Type of the Data Array. It is possible to configure the driver to store the data as a different type. This can be achieved by specifying the data type under the parameter EIP_Data_Type. Note that the EIP_DATA_TYPE parameter has meaning only for DATA_TABLE_WRITE where the FieldServer is the Client and for DATA_TABLE_READ where the FieldServer is the Server.

Appendix A.4.1. Transfer of Binary Values using EIP

The standard FieldServer data types do not follow the same naming convention as Rockwell. However, it is possible to enter the data types in the EIP_Data_Type field in either the FieldServer or the Rockwell format.

EIP_DATA_TYPE should be specified as Bit, BOOL or BOOLEAN.

Set the Map Descriptor length to 1 for the driver to transfer 1 bit. If a length other than 1 is specified then bits will be transferred in 32 bit words.

Appendix A.5. Configuring a PLC to read and write data to and from FieldServer using Explicit messaging

This example makes use of the Data Table Read/Write method for passing data between the FieldServer and an Allen Bradley PLC. The example shows configuration of a ControlLogix PLC, but all Rockwell PLC's that support Ethernet IP communications and Data Table Read/Write operations in Message blocks should be able to communicate this way. The Map Descriptors create an explicit connection to the Server and then transfer data in the data table format. The EIP_Tag_Name field contains the tag name polled from the client. DATA_TABLE_READ and DATA_TABLE_WRITE are the only legal values for EIP_SERVICE.

Appendix A.5.1. FieldServer Configuration File

The configuration file used for this example is configured with the following Connection, Node and Map Descriptor Parameters:

```
//  Data Arrays
//
Data_Arrays
Data_Array_Name    , Data_Format[5]    , Data_Array_Length
DA_Read            , Float             , 100
DA_Write           , Float             , 100
```

```
//  Server Side Connections
//
Connections
Adapter        , Protocol       , Turnaround_delay
N1             , Ethernet/IP    , 0.01s
```

```
//  Server Side Nodes
//
Nodes
Node_Name              , Protocol
EIP_01                 , Ethernet/IP
```

```
//  Server Side Map Descriptors
//
Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Node_Name   , Function   , EIP_SERVICE          , EIP_TAG_NAME   , Length
FS_TO_PLC_DATA        , DA_Read           , 0                  , EIP_01      , Passive    , DATA_TABLE_READ     , Read_Data      , 20
PLC_TO_FS_DATA        , DA_Write          , 0                  , EIP_01      , Passive    , DATA_TABLE_WRITE    , Write_Data     , 20
```

"FieldServer "Tag names that will be called in the PLC Message Block. The names must match what is written in the Message block in the PLC exactly.

Note: The corresponding PLC Tag Name can be different and probably will be. See Message Block Below.

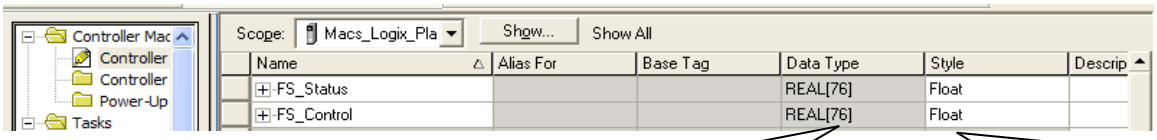
Number of data points made available for reading or writing within the Tag.

[5] The Data Format should match the Data Type used in the PLC program and tags to prevent a mismatch in the display of the Data Values.

Appendix A.5.2. The PLC Program

The PLC program example below shows the minimum steps necessary to program communications with the FieldServer. Depending on the real intended application, additional steps may be necessary for completeness.

Step 1: Configure Tags in the PLC for storing FieldServer read and write data:



Tag Length Must be equal to or greater than the number of points being written/read

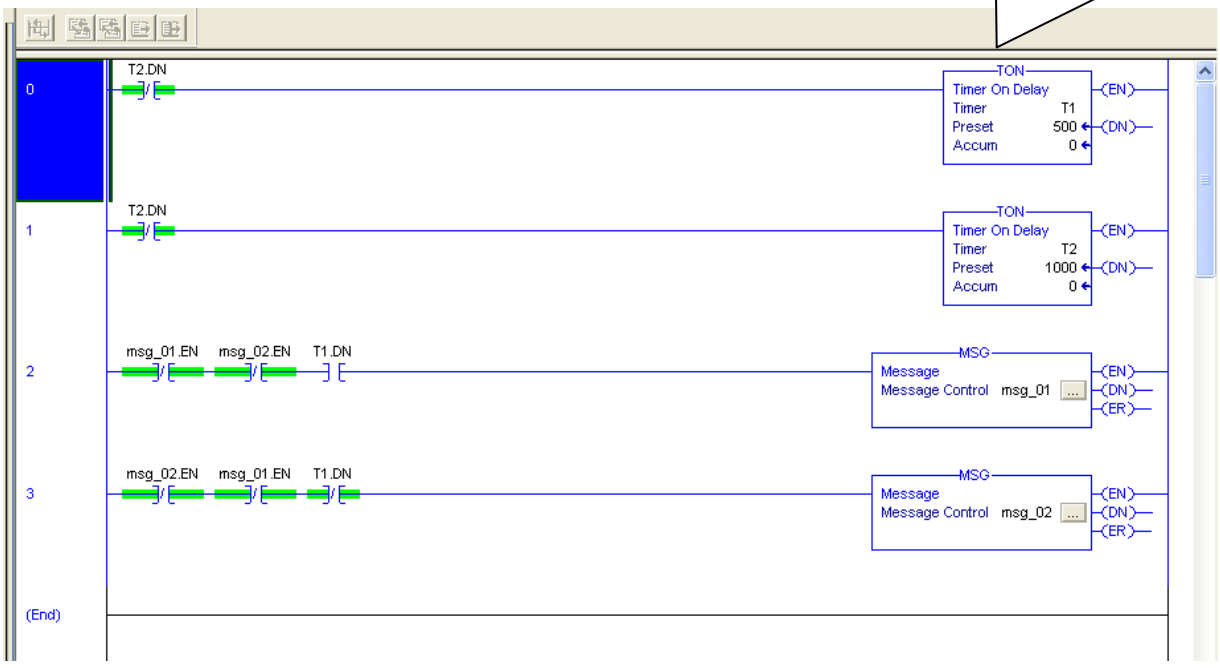
Avoid Using UDT Types. The Data will be read but the exact placement of the data in the Tags and Arrays will be hard to determine

Step 2: Configure Message Tags for storage of Message Block data:

MESSAGE Data Type must be used.

Note that this logic shown will cause the PLC to poll the FieldServer at a very high speed. This may overload network traffic - logic that schedules the communication at a slower rate is generally more advisable.

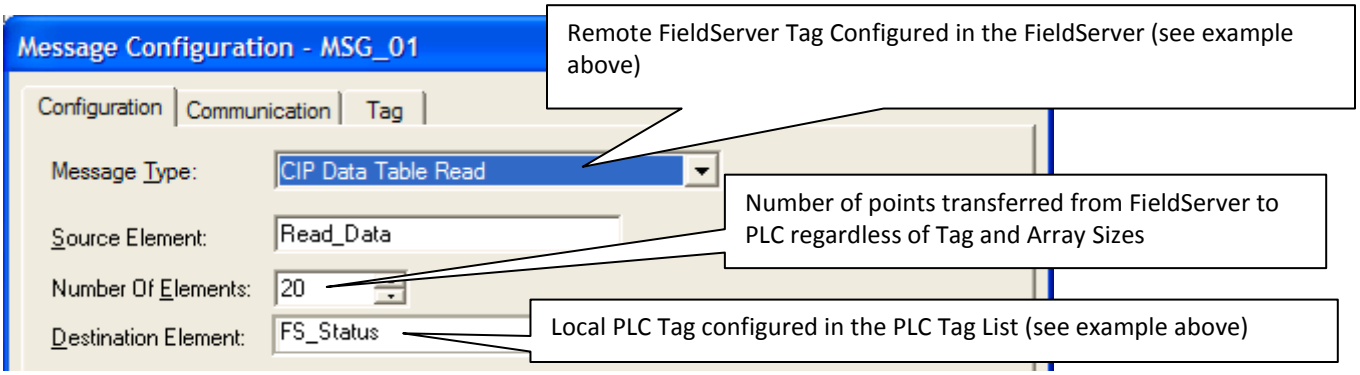
Step 3: Write Ladder Logic to exercise a Read and Write Message Block



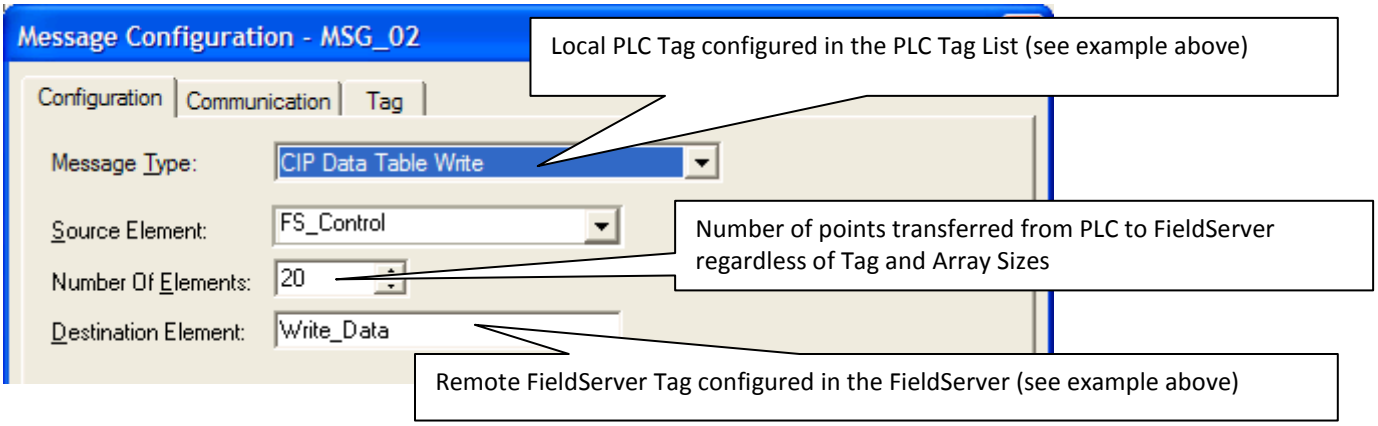
NOTE: It is important to use the Ladder logic to ensure that only one Message block can be executing at any time.

Step 4: Configure the Properties for the two Message Blocks by clicking on the "…" button:

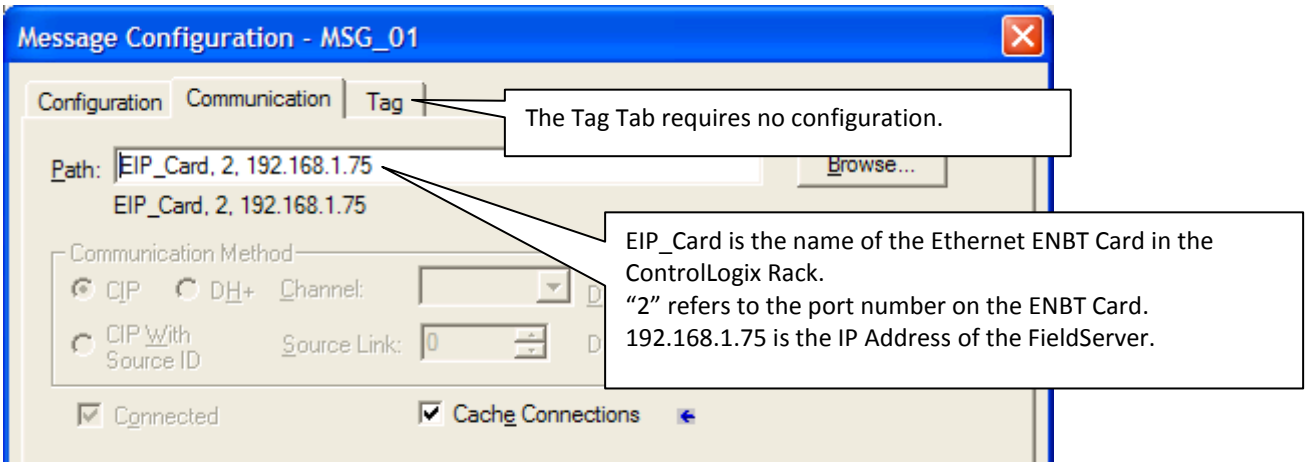
Read Message Block:



Write Message Block:



Communication Tab for both Message Blocks:



Notes:

1) A message block only executes on the leading edge of a rung condition

2) No more than one message block executes at a time.

Step 5: Download the program and set the PLC to Run Mode.

Appendix A.6. Configuring a FieldServer to read and write Data to and from a PLC using Explicit messaging

This example makes use of the Data Table Read/Write method for passing data between the FieldServer and an Allen Bradley PLC. The example shows configuration of a ControlLogix PLC, but all RockWell PLC's that support Ethernet IP communications and Data Table Read/Write operations in Message blocks should be able to communicate this way. . These map descriptors will create an explicit connection to the server and will then transfer data in the data table format. The EIP_Tag_Name field contains the tag name referenced in the server and the EIP_Path field represents the path (through different ports) to the server. Each port jump is separated by a space. This field generally holds a backplane/cpu slot combination. DATA_TABLE_READ and DATA_TABLE_WRITE are the only legal values for EIP_SERVICE.

Note that this is by far the preferred method for communicating with Allen Bradley PLC's due to it's ease of configuration, quantity of data that can be transferred and speed of transfer.

When the FieldServer is the active component as shown below (i.e: the FieldServer Polls the PLC and not the other way around), then very little programming is needed in the PLC, other than the Tag Creation and setting the PLC IP Address.

Appendix A.6.1. FieldServer Configuration File

```
//  Data Arrays
//
Data_Arrays
Data_Array_Name    , Data_Format    , Data_Array_Length
DA_Read            , Float          , 50
DA_Write           , Float          , 50
```

```
//  Client Side Connections
//
Connections
Adapter                    , Protocol
N1                         , Ethernet/IP
```

```
//  Client Side Nodes
//
Nodes
Node_Name          , IP_Address     , Protocol       , Adapter
EIP_01             , 192.168.1.9    , Ethernet/IP    , N1
```

These are the PLC Tag names that will be accessed in the PLC. The names must match the PLC tag name exactly.

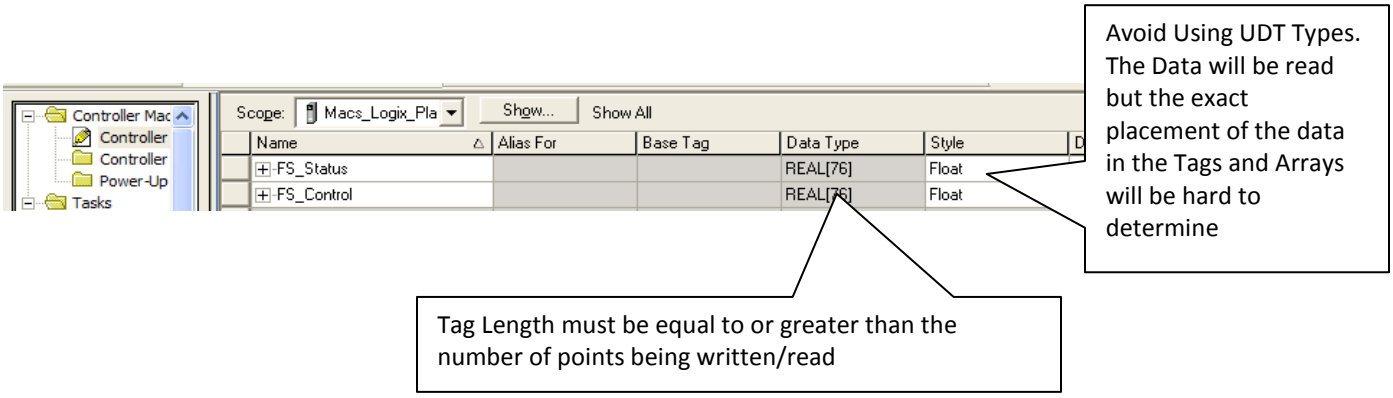
Appendix A.3 lists the paths for specific devices

```
//  Client Side Map Descriptors
//
Map_Descriptors
Map_Descriptor_Name  , Scan_Interval  , Data_Array_Name  , Data_Array_Offset  , EIP_CON_TYP  , Node_Name  , Function  , EIP_SERVICE          , EIP_PATH  , EIP_TAG_NAME  , Length
PLC_TO_FIELDSERVER   , 0.1            , DA_Read          , 0                   , EXPLICIT      , EIP_01     , Rdbc      , DATA_TABLE_READ     , 1 0       , FS_Status     , 20
FIELDSERVER_TO_PLC   , 0.1            , DA_Write         , 0                   , EXPLICIT      , EIP_01     , Wrbc      , DATA_TABLE_WRITE    , 1 0       , FS_Control    , 20
```

Appendix A.6.2. The PLC Program

The PLC program example below shows the minimum steps necessary to program communications with the FieldServer. Depending on the real intended application, additional steps may be necessary for completeness.

When the FieldServer is polling the PLC, all that is needed is to configure the tags being accessed:



Note that providing dedicated (long), flat structured (not UDT) tags for communicatons interface to the FieldServer is preferable to polling single length tags in the PLC as it allows for much more efficient communications and reduces complexity when mapping data in the FieldServer. It is better to think of these tags as a "I/O Buffer" Interface that the real tags in the PLC get mapped to.

Appendix A.7. Read/write structures and value of EIP_Structure_Handle :

Some devices require that a specific value be used for the EIP_Structure _Handle field while writing the structure to them. This value may be specified in the vendor documentation. If any non-zero integer is used in a Read Map Descriptor for EIP_Structure_Handle, the value will be updated internally. The Map Descriptor can then be browsed to obtain this value. Other devices do not validate this field when the structure is written by the third-party device. A summary of the procedures to obtain this value is presented below:

FieldServer as a Client:

- Read:
  - Use a value of 1, the driver will automatically update the field when a response is received from the device. The Map Descriptor can then be browsed to obtain the value if required.
- Write:
  - Use the value supplied by the vendor OR
  - Use the value obtained in the Read Map Descriptor above OR
  - Use any non-zero value if the other device doesn't validate it.

FieldServer as a Server:

- Read:
  - Use the value supplied by the vendor OR
  - Use the value obtained in the Read Map Descriptor above OR
  - Use any value if the other device doesn't validate it.
- Write:
  - Not required – use any non-zero value if the parameter is specified.

Example: Consider a situation where a customer defines a type in RSlogix SSS_SINT3 with 3 members of each type SINT.

SSS_SINT3
- SINT room1_temp
- SINT room2_temp
- SINT room3_temp

Now he has his own type SSS_SINT3 and he can define tags: TAG_3ROOM_TEMPS of type SSS_SINT3

```
// Read/write structures

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , EIP_CON_TYP   , Node_Name   , EIP_SERVICE          , EIP_Path   , EIP_TAG_NAME          , EIP_Structure_Handle   , Length   , Scan_Interval
CMD_Struct_SINT3      , DA_STRUCT_R       , 0                  , Rdbc       , EXPLICIT       , EIP_01      , DATA_TABLE_READ    , 1 0        , TAG_3ROOM_TEMPS       , 59592                  , 3          , 1.0s
```

Appendix A.8. Configuring a PLC to read and write data to and from FieldServer using Implicit messaging

Implicit messaging uses the User Datagram Protocol (UDP/IP) to produce/consume data over the EtherNet/IP network. Implicit messages are not regular poll/response messages. Implicit connections are time critical, scheduled and use a requested packet interval (RPI) to specify the rate at which data updates.

This example makes use of the Implicit messaging (I/O) for passing data between the FieldServer and an Allen Bradley PLC. The example shows configuration of a ControlLogix PLC, but all Rockwell PLC's that support Ethernet IP communications and implicit (I/O) messaging should be able to communicate this way. When the PLC is configured, it will open a connection to the FieldServer.

The Map Descriptors accept a connection from the PLC to consume and produce data at the rate requested by the PLC. The data produced by the PLC will be consumed by Map Descriptor 'MD_CONSUMER' and data produced by Map Descriptor 'MD_PRODUCER' will be consumed by the PLC.

Appendix A.8.1. FieldServer Configuration File

The configuration file used for this example is configured with the following Connection, Node and Map Descriptor Parameters:

```
//   Data Arrays
//
Data_Arrays
Data_Array_Name    , Data_Format    , Data_Array_Length
DA_INPUTS          , Uint16         , 100
DA_OUTPUTS         , Uint16         , 100
```

```
//   Server Side Connections
//
Connections
Adapter          , Protocol
N1               , Ethernet/IP
```

IP address is multicast IP address and will be used only if EIP client requests FS to create multicast production connection.

```
//   Server Side Nodes
//
Nodes
Node_Name              , Node_ID    , Protocol       , IP_Address
EIP_01                 , 1          , Ethernet/IP    , 192.168.1.60
```

```
//   Server Side Map Descriptors
//
Map_Descriptors
Map_Descriptor_Name    , Data_Array_Name    , Data_Array_Offset    , Node_Name    , EIP_CON_TYP    , Function    , EIP_CLASS    , Address    , EIP_ATTRIBUTE    , Length
MD_CONSUMER            , DA_INPUTS          , 0                    , EIP_01       , IMPLICIT       , Server      , 4            , 101        , 3                , 100
MD_PRODUCER            , DA_OUTPUTS         , 0                    , EIP_01       , IMPLICIT       , Server      , 4            , 102        , 3                , 100
```

Connection Type set to Implicit.

Number of data points to be transferred.

Appendix A.8.2. The PLC Setup

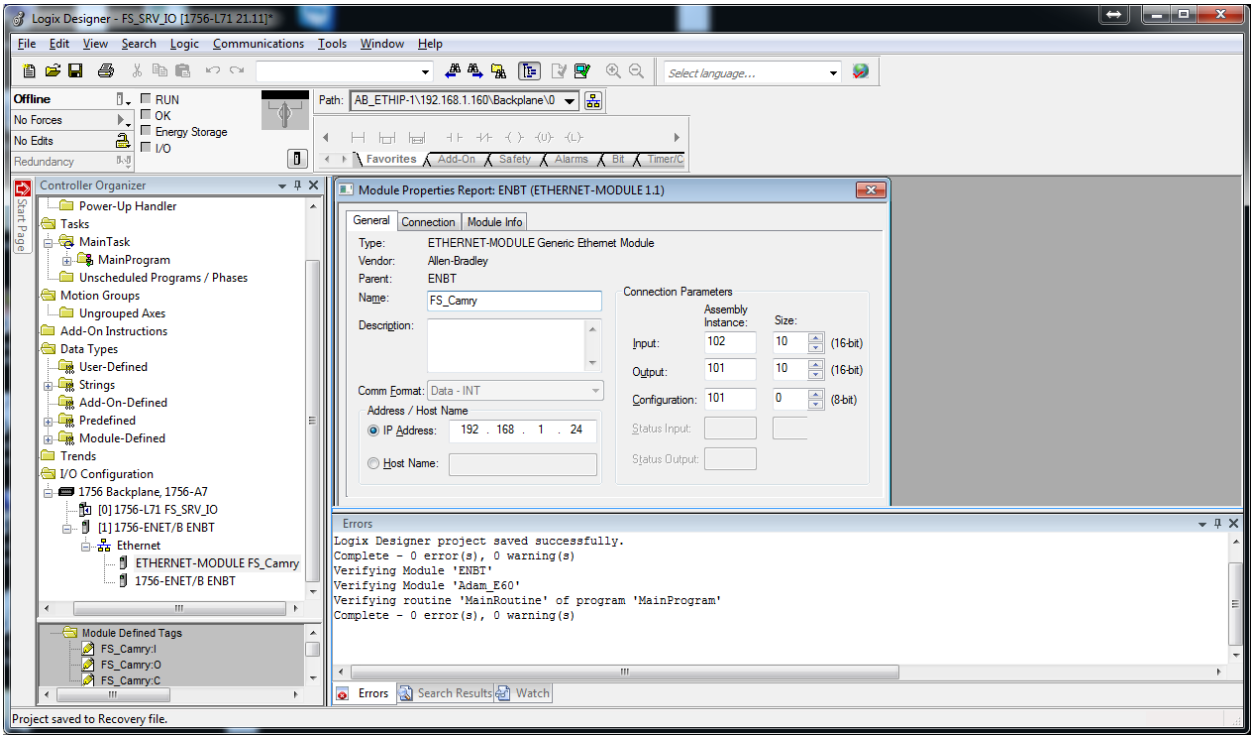
The PLC (192.168.1.60) should be connected to the FieldServer (192.168.1.24).

To start Implicit Communications, use the following 3 steps:

- Add a “Generic Ethernet Module”.
- Configure the Connection Parameters as per the screenshot below.
- Enter the FieldServer’s IP address.

The PLC will produce and consume 10 16bit words, but the FieldServer is configured for 100 words in case more data is needed. The FieldServer will produce only as much data as requested by the PLC during session set up.

No further PLC programming is needed.



Appendix B. Troubleshooting

Appendix B.1. Firmware Update Downloading

If you are trying to update firmware and continuously get failed messages it might be due to the traffic on the N1 or N2 ports. EtherNet/IP is a high traffic protocol and once a connection is created continuous data transfer occurs. In this situation the best way to download new firmware would be to manually disconnect the scanner or adapter that the FieldServer is connected to.

Appendix C. Vendor Information

Appendix C.1. ControlLogix

If EIP Error Code 100 is experienced when using a ControlLogix PLC Message block to read and write data to and from the FieldServer, it may be possible to resolve this error by choosing to cache the connection in the message block settings

Appendix C.2. Connection information – Allen Bradley Message Blocks

The Driver is not able to split data between 2 Data Arrays when writing, or to read a Server mapping that is discontinuous.

For example, on the Server Side:
If:
Server map 1: N21: 0-31
Server map 2: N21: 32-100

This will panic the FieldServer and crash RS view as the DH+ will attempt to map N21: 0-100. If set up as Server Map 1: N21: 0-100, no problems are experienced.

Similarly on the Client Side, it is not possible to read sections e.g. 32bits to one array and then 16bits from the same file type to another array etc. Read Blocks must be contiguous.

Appendix C.3. FieldServer not recognised by RSlinx

If RSlinx does not recognise the FieldServer (message "? Unrecognized Device"), load the Ethernet IP EDS file into RSLinx. This file is available at: http://www.fieldserver.com/techsupport/utility/utility.php

- Press Start|all programs|Rockwell Software|RSlinx tools|EDS Hardware Installation tool|add|register a single file and browse to the location of the Ethernet IP EDS file.
- Run RSlinx, press communication|RSwho and all EIP devices on the network should be visible.

Appendix C.4. Using EIP Data_Types with RSLogix

The EIP_Data_Types defined for the FieldServer are not all supported by the PLC when working with RSLogix. The following aliases have been created so that data can be transferred in the required format.

FieldServer Data Type	Rockwell Data Type	Data Format
BYTE	SINT	8 bit integers
UINT16	INT	16 bit integers
UINT32	DINT	32 bit integers
FLOAT	REAL	32bit floats
BIT	BOOL	Single 1 bit value
BIT_ARRAY, BOOLEAN	BOOL_ARRAY	Multiple 1 bit values

Appendix D. Reference

Appendix D.1. Error Messages

Message	Description
"EIP:#01 FYI. %d out of %d data elements will be stored" "MD=%s, data_type=0x%04X, raw bytes=%d"	If the number of data elements exceeds the Map Descriptor length only the number of data elements corresponding to the Map Descriptor will be stored. This message will print once per Map Descriptor.

Appendix D.2. Classes and Attributes Supported

EtherNet/IP is an object orientated protocol. The Object Oriented structure therefore allows for classes, instances, attributes and services. The 'data types' listed below are to be considered as the objects supported in the protocol. Each of these has attributes that have been supported to differing degrees.

FieldServer Data Type	Description (or Device Data Type)	
Identity – Class Code 0x01	**Attributes Supported:** *One instance supported (0x01)* Attributes List: • Vendor ID • Device Type • Product Code • Device Revision • Status • Serial Number • Device Description (text)	**Services Supported:** Get_Attribute_All; Get_Attribute_Single
Message Router – Class Code 0x02	**Attributes Supported:** *One instance supported (0x01)* Attributes List: • Max Connections	**Services Supported:** Get_Attribute_Single
Assembly – Class Code 0x04	Attributes Supported: *Class Instance Support (0x00)* Class Attributes: 0x02 (Max Instance) *Two instances supported (0x0100 and 0x0101)* Attributes List: • Member List • Data (0x03)	**Services Supported:** Get_Attribute_Single
Connection Manager – Class Code 0x06	Forward Open Service Forward Close Service	

FieldServer Data Type	Description (or Device Data Type)	
Register – Class Code 0x07	**Attributes Supported:** *Class Instance Support (0x00)* Class Attributes: 0x02 (Max Instance) *Two instances supported (0x01 and 0x02)* Attributes List: • Status Flag • Direction (read/write) • Size of Data (bits)	**Services Supported:** Get_Attribute_Single
Discrete Input Point – Class Code 0x08	No visible interface currently	
Discrete Output Point – Class Code 0x09	No visible interface currently	
Analog Input Point – Class Code 0x0A	**Attributes Supported:** *Class Instance Support (0x00)* Class Attributes: 0x02 (Max Instance) *Two instances supported (0x01 and 0x02)* Attributes List: • Number of Attributes • Analog value (UINT16) • Vendor ID	**Services Supported:** Get_Attribute_Single
Analog Output Point – Class Code 0x0B	**Attributes Supported:** *Class Instance Support (0x00)* Class Attributes: 0x02 (Max Instance) *Two instances supported (0x01 and 0x02)* Attributes List: • Number of Attributes • Analog value (UINT16) • Vendor ID	**Services Supported:** Set_Attribute_Single; Get_Attribute_Single
TCP/IP Interface Object – Class Code 0xF5	**Attributes Supported:** *One instance supported (0x01)* Attributes List: • Status • Configuration Capability • Configuration Control • Physical Link Object • Interface Configuration • Host Name	**Services Supported:** Get_Attribute_Single
EtherNet Link Object – Class Code 0xF6	**Attributes Supported:** *One instance supported (0x01)* Attributes List: • Interface Speed • Interface Flags • Physical Address • Interface Counters • Media Counters	**Services Supported:** Get_Attribute_Single

FieldServer Data Type	Description (or Device Data Type)	
Data Table Object – Private Object	**Attributes Supported:** This object does not support instances or attributes but uses the data table structure, and associated tags, in Logix5000 PLC's.	**Services Supported:** CIP Read Data

Appendix D.3. Error Codes

General Status	Extended Status	Description
0x01	0x0100	Connection in Use or Duplicate Forward Open.
0x01	0x0103	Transport Class and Trigger combination not supported
0x01	0x0106	Ownership conflict
0x01	0x0107	Connection not found at target application.
0x01	0x0108	Invalid Connection Type. Indicates a problem with either the Connection Type or Priority of the Connection.
0x01	0x0109	Invalid connection size
0x01	0110	Device not configured
0x01	0x0111	RPI not supported. May also indicate problem with connection time-out multiplier or production inhibit time.
0x01	0x0113	Connection Manager cannot support any more connections
0x01	0x0114	Either the Vendor Id or the Product Code in the key segment did not match the device
0x01	0x0115	Product Type in the key segment did not match the device
0x01	0x0116	Major or Minor Revision information in the key segment did not match the device
0x01	0x0117	Invalid Connection Point
0x01	0x0118	Invalid Configuration Format
0x01	0x0119	Connection request fails since there is no controlling connection currently open.
0x01	0x011A	Target Application cannot support any more connections
0x01	0x011B	RPI is smaller than the Production Inhibit Time
0x01	0x0203	Connection cannot be closed since the connection has timed out
0x01	0x0204	Unconnected Send timed out waiting for a response.
0x01	0x0205	Parameter Error in Unconnected Send Service
0x01	0x0206	Message too large for Unconnected message service
0x01	0x0207	Unconnected acknowledge without reply
0x01	0x0301	No buffer memory available
0x01	0x0302	Network Bandwidth not available for data
0x01	0x0303	No Tag filters available
0x01	0x0304	Not Configured to send real-time data
0x01	0x0311	Specified Port Not Available – the FieldServer is trying to open a connection but the target port is not available at the remote device. This can be resolved by using the correct EIP_Path parameter.
0x01	0x0312	Link Address specified in Port Segment Not Available
0x01	0x0315	Invalid Segment Type or Segment Value in Path
0x01	0x0316	Path and Connection not equal in close
0x01	0x0317	Either Segment not present or Encoded Value in Network Segment is invalid.
0x01	0x0318	Link Address to Self Invalid
0x01	0x0319	Resources on Secondary Unavailable
0x01	0x031A	Connection already established
0x01	0x031B	Direct connection already established
0x01	0x031C	Miscellaneous
0x01	0x031D	Redundant connection mismatch
0x01	0x031D	No more consumer resources available in the producing module
0x01	0x031E	No connection resources exist for target path
0x01	0x0320-	Vendor Specific

General Status	Extended Status	Description
	0x7FF	
0x02	n/a	Connection Manager resources are unavailable to handle service request
0x03	n/a	Invalid connection number specified by the Get_Connection_Data service. This is also returned by the Search_Connection_Data service if the specified connection is not found
0x04	Zero Based Word Offset	Segment Type in path is invalid. The Extended Status shall be the word offset (0 based) to the word in the path where the error occurred. The offset starts at the first word after the path size. This error shall not be returned if an error occurs when parsing the Connection Path. This error commonly occurs when tag names are incorrectly spelled, mismatches in case or do not exist. Verify Map Descriptor EIP_Tag_Name(s) spelling and case match PLC tag names.
0x05	Zero Based Word Offset	Destination in path is invalid. The Extended Status shall be the word offset (0 based) to the word in the path where the error occurred. The offset starts at the first word after the path size. This error shall not be returned if an error occurs when parsing the Connection Path.
0x07	n/a	Connection has been lost. This is used by the Get/Set Services when they are made through a connection
0x08	n/a	Connection Manager does not support the requested Service
0x09	Index to Element	Error in Data Segment. Extended Status shall be index to where the error was encountered in the Data Segment. The Configuration Revision Number if present in the Data Segment shall always be index 1. If the error occurs with the Get/Set Services, then the extended status indicates the attribute number that failed.
0x0C	Optional	Service cannot be performed while Object is in current state. The 1st word of Extended Status may optionally contain the object's current state.
0x10	Optional	Service cannot be performed while Device is in current state. The 1st word of Extended Status may optionally contain the device's current state.
0x11	n/a	Response data too large. This is used by the get services to indicate the amount of data requested was too large to fit into the response buffer.
0x13	n/a	Not enough data was received.
0x14	Attribute Id	Attribute specified in FIND service is not supported by Connection Manager
0x15	n/a	Too much data was received.
0x25	0x0114	Either the Vendor Id or the Product Code in the key segment did not match the device. Used if the Key Segment was contained in the path.
0x25	0x0115	Product Type in the key segment did not match the device. Used if the Key Segment was contained in the path.
0x25	0x0116	Major or Minor Revision information in the key segment did not match the device. Used if the Key Segment was contained in the path.
0x026	n/a	Invalid path size

NOTE:

- The word "n/a" in the Extended Status Column is used to signify that there is no additional Extended Status which is required to be returned for the particular General Status Code.
- The word "optional" in the Extended Status Column is used to signify that if Extended Status information is used, then the first word of that extended status is already defined