



A Sierra Monitor Company



## SlotServer Configuration Manual

### APPLICABILITY & EFFECTIVITY

This manual provides instructions for the following FieldServer products:

#### Description

PS56-BAS-xxx

PS56-FIR-xxx

PS56-PRO-407

The instructions are effective for the above as of December 2009

Kernel Version:	5.16
Document Revision:	2

---

**TABLE OF CONTENTS**

**1 Introduction.....4**

1.1 About this product..... 4

1.2 Required configuration for the SlotServer ..... 4

**2 Steps for implementation of a SlotServer Project .....5**

2.1 Install the FieldServer Utilities..... 5

2.2 Load the SlotServer Configuration into the SlotServer ..... 5

2.3 Program the ControlLogix CPU to communicate with the SlotServer..... 5

2.4 Commission the third party network. .... 5

**3 Configuring the CPU interface to the SlotServer .....6**

3.1 CPU interface Description ..... 6

3.2 Data Array Parameters ..... 7

3.3 Configuring the SlotServer as a Logix I/O Server..... 7

3.4 Client Side Connection Parameters..... 8

3.5 Client Side Node Parameters..... 8

3.6 Client Side Map Descriptor Parameters ..... 9

    3.6.1 SlotServer Specific Map Descriptor Parameters..... 9

    3.6.2 Driver Specific Map Descriptor Parameters ..... 9

    3.6.3 Map Descriptor Example..... 10

**4 Programming the ControlLogix CPU for a small SlotServer Interface .....11**

4.1 Step 1: Establish an RSLogix project..... 11

4.2 Step 2: Add and configure the SlotServer as an IO Module ..... 11

4.3 Step 3: Write Ladder Program to get Input Data from Data Arrays..... 13

4.4 Step 4: Write Ladder Program to Send Output Data to Data Arrays ..... 16

4.5 Step 5: Download the RSLogix Program and Run..... 16

4.6 Step 6: Set up the third party connection ..... 16

**5 Programming the ControlLogix CPU for larger SlotServer Interfaces.....17**

5.1 Multiple Input Data Arrays ..... 17

5.2 Accessing Multiple Output Data Arrays ..... 17

**6 Timing Parameters .....19**

---

<b>Appendix A. Useful Features .....</b>	<b>22</b>
Appendix A.1. How to obtain Node Status from the SlotServer .....	22
<b>Appendix B. Troubleshooting .....</b>	<b>23</b>
Appendix B.1. SLOTSERVER : block number [ 0 ] out of range!.....	23
Appendix B.2. Driver Error screen returns “Illegal Protocol” Error.....	23
<b>Appendix C. Vendor Information .....</b>	<b>24</b>
Appendix C.1. Installing SlotServer on a Remote Rack using CNB Cards .....	24
<i>Appendix C.1.1. RSLogix configuration .....</i>	<i>24</i>
<i>Appendix C.1.2. RSNetWorx configuration .....</i>	<i>25</i>
<i>Appendix C.1.3. Testing.....</i>	<i>26</i>
<i>Appendix C.1.4. Connection limitations -Controlling the SlotServer using ControlNet.....</i>	<i>27</i>
Appendix C.2. Dealing with ControlLogix RPI Settings .....	28
Appendix C.3. Rules for Naming Logix driver Data Arrays .....	29
<b>Appendix D. Reference.....</b>	<b>31</b>
Appendix D.1. Description of Data Transfer Process .....	31
Appendix D.2. The IO image header .....	33

## 1 INTRODUCTION

### 1.1 ABOUT THIS PRODUCT

The SlotServer Configuration Manual provides the information necessary to configure the SlotServer, allowing an Allen Bradley ControlLogix platform to pass data between a ControlLogix CPU and other third party communications protocols supported by the SlotServer. **The SlotServer uses implicit communications between the CPU and the SlotServer and is consequently treated as an I/O Server in RSLogix.**

The SlotServer Start-up guide covers information for installing the SlotServer, the Configuration Manual provides information on configuring the module to transfer data with the CPU on the ControlLogix Rack. Depending on the SlotServer Module ordered, supplementary driver manuals are provided for information on how to configure the third party protocols residing in the SlotServer.

### 1.2 REQUIRED CONFIGURATION FOR THE SLOTSERVER

To achieve data transfer between CPU tags and the SlotServer third party protocols, it is necessary to write and load a configuration into the SlotServer that tells the SlotServer how to map the ControlLogix Tags to the required protocol addresses. This configuration is written in a Comma Separated Variable (csv) file, and any text editor or spreadsheet program that supports csv format can be used for this purpose. FieldServer Technologies provides an example configuration file so that the configuration does not need to be written from scratch. Configuration parameters needed to exchange data between the CPU and the SlotServer data images (Data Arrays) are presented in Section 3. The appropriate driver manual supplement will describe how to map the data in and out of the Data Arrays for the relevant protocol.

The SlotServer configuration manual details basic and advanced techniques for the configuration of the SlotServer, and it is strongly advised that this manual is read before attempting to write the SlotServer Configuration.

FieldServer Technologies provides SlotServer configuration services if the user does not wish to perform the configuration themselves.

---

## 2 STEPS FOR IMPLEMENTATION OF A SLOTSERVER PROJECT

### 2.1 INSTALL THE FIELDSEVER UTILITIES

Refer to the Start-up guide for further information..

### 2.2 LOAD THE SLOTSERVER CONFIGURATION INTO THE SLOTSERVER

Refer to the FieldServer Utilities manual and SlotServer Start-up Guide for information on the FieldServer Remote User Interface.

### 2.3 PROGRAM THE CONTROLLOGIX CPU TO COMMUNICATE WITH THE SLOTSERVER.

The sections that follow highlight a few examples on how this is done.

### 2.4 COMMISSION THE THIRD PARTY NETWORK.

Refer to standard commissioning guidelines for the protocol in question.

### 3 CONFIGURING THE CPU INTERFACE TO THE SLOTSERVER

#### 3.1 CPU INTERFACE DESCRIPTION

The SlotServer Data Images (Data Arrays) share data between the ControlLogix CPU tags (using the backplane for communication) and the FieldServer Logix driver. To map the Logix Driver, the Driver needs to be configured in the SlotServer Configuration. The information that follows details the configuration parameters that can be used for this driver. The driver can only act as an I/O Server (Adapter) to a ControlLogix CPU. The SlotServer presently works with ENBT Ethernet cards, but not EN2T or EN2F

#### **Max Nodes Supported**

SlotServer Mode	Nodes	Comments
Server	1	Only one IO image connection supported

```
// Common Information
FieldServer
Title                , System_Station_Address
SlotServer Server v1.03d , 1
```

### 3.2 DATA ARRAY PARAMETERS

The configuration file tells the SlotServer about its interfaces, and the routing of data required. In order to enable the SlotServer for Logix communications, the driver independent SlotServer buffers need to be declared in the “Data Arrays” section.

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, Packed_Bit, Byte, Packed_Byte, Swapped_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.		For IO_Data_Type: Real 1-120 UInt 1-244 Sint 1-492 Dint - 1-120

**Example**

```
// Data Arrays

Data_Arrays
Data_Array_Name ,Data_Array_Format ,Data_Array_Length
TestOut_1      , UInt16           , 244
TestOut_2      , UInt16           , 244
TestIn_1       , UInt16           , 244
TestIn_2       , UInt16           , 244
```

### 3.3 CONFIGURING THE SLOTSERVER AS A LOGIX I/O SERVER

The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the SlotServer (See “.csv” sample files provided with the SlotServer).

The configuration file tells the SlotServer about its interfaces, and the routing of data required. In order to enable the SlotServer for Logix communications, the driver independent SlotServer buffers need to be declared in the “Data Arrays” section, the SlotServer virtual node(s) needs to be declared in the “Server Side Nodes” section, and the data to be provided to the Clients 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.

### 3.4 CLIENT SIDE CONNECTION PARAMETERS

Section Title		
Connections		
Column Title	Function	Legal Values
Adapter	Adapter Name	ControlNet

**Example**

```
// Server Side Connections

Connections
Adapter
ControlNet
```

### 3.5 CLIENT SIDE NODE PARAMETERS

Section Title		
Nodes		
Column Title	Function	Legal Values
Node_Name	Provide name for Node	Up to 32 alphanumeric characters
Node_ID	Virtual Node ID	0-15
Protocol	Specify protocol used	Logix_SS

**Example**

```
// Server Side Nodes

Nodes
Node_Name      , Node_ID  , Protocol
SlotServer_CPU , 1      , Logix_SS
```



### 3.6 CLIENT SIDE MAP DESCRIPTOR PARAMETERS

#### 3.6.1 SlotServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Function	Function of Server Map Descriptor	WRBC-to write into Input Image Data buffer RDBC-to read from Output Image Data buffer

#### 3.6.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in "Client Node Descriptor" above
IO_Data_Type	Data type of IO image	Use same as used in RSLogix to add the module INT (16-bit integer) SINT (8-bit signed) DINT (32-bit double) REAL (32-bit float)
DA_Name_Start <sup>1</sup>	Name of Data Array to include in IO image data	One of the Data Array names from Section 3.2. Data Arrays must be named as follows: DaName_x where x is a positive integer value e.g. DaName_1
DA_Count	Number of Data Arrays to include in IO image data	1-200 <sup>2</sup>
Scan_Interval	Rate at which IO image data is updated.	Use twice the rate of RPI, e.g: RPI = 0.4s, Scan_Interval=0.2s

<sup>1</sup> Refer to Appendix C.3 for information specific to naming Logix driver Data Arrays.

<sup>2</sup> The update rate decreases as the number of blocks go up. For 25 blocks the update rate is  $25 * RPI = 25 * 0.1 = 2.5$  seconds.

### 3.6.3 Map Descriptor Example

Only two Map Descriptors are permitted, one to transfer data to the Logix CPU and one to accept data from the Logix CPU.

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name , Scan_Interval , Function , Node_Name , IO_Data_Type , DA_Name_Start3 , DA_Count
Input_BP_Image , 0s , Wrbc , SlotServer_CPU , Int , TestOut_1 , 2
Output_BP_Image , 0s , Rdbc , SlotServer_CPU , Int , TestIn_1 , 2
```

---

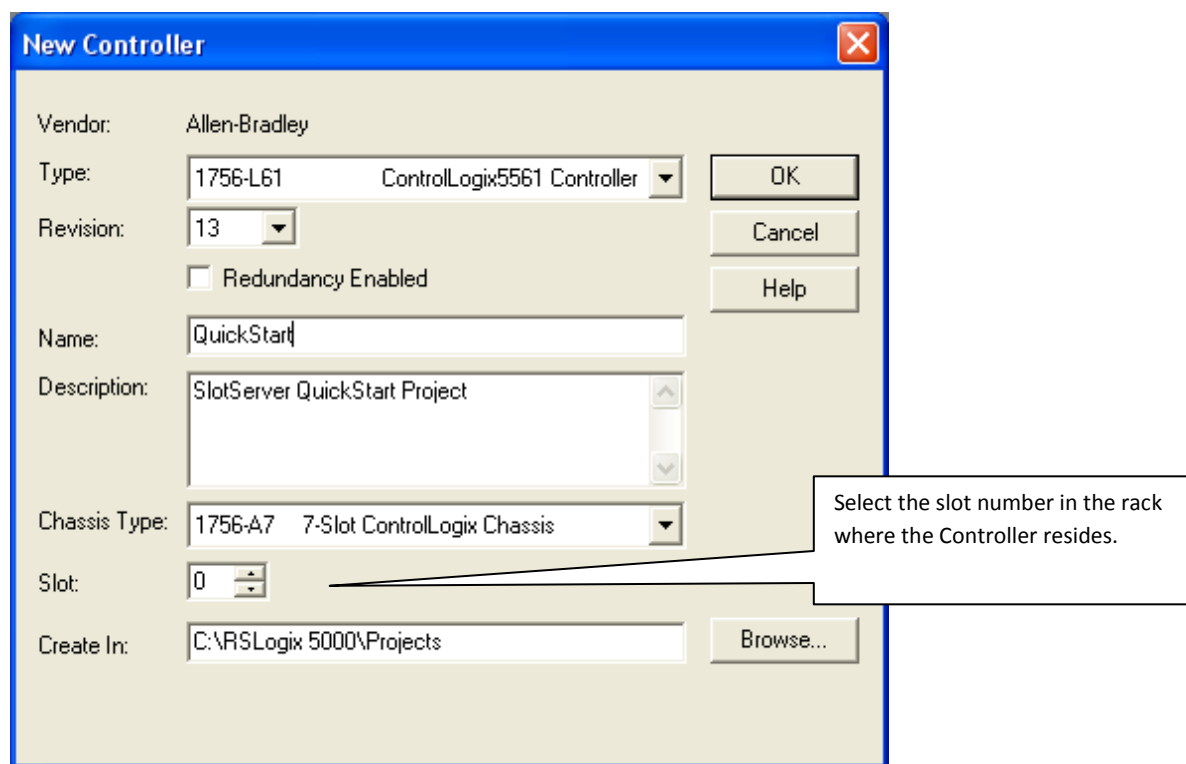
<sup>3</sup> Refer to Appendix C.3 for information specific to naming Logix driver Data Arrays.

## 4 PROGRAMMING THE CONTROLLOGIX CPU FOR A SMALL SLOTSERVER INTERFACE

The discussion that follows describes the basic steps to set up and test the system for transferring data between CPU tags and the SlotServer using the I/O image method. The quick Start example uses LonWorks as the example 3<sup>rd</sup> Party Protocol. A hardcoded template filled with Lon variables is created. Each item uses a different amount of bytes and the total adds up to 104 Lon Network Variables. This limit of 104 does not apply when using customized data items – the actual limit is 496 bytes. Refer to the Advanced Project to access more than 104 Network Variables.

### 4.1 STEP 1: ESTABLISH AN RSLOGIX PROJECT<sup>4</sup>

Use File/New to create a new project or File/Open to open an existing project.

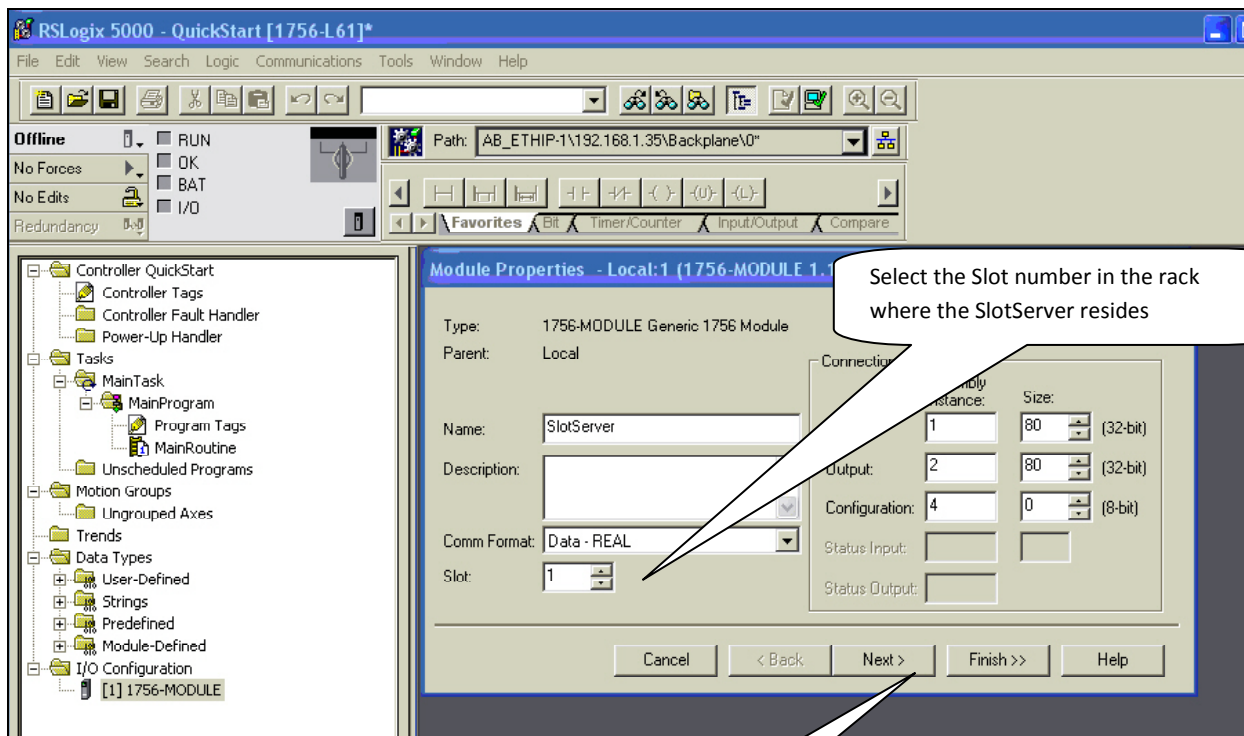
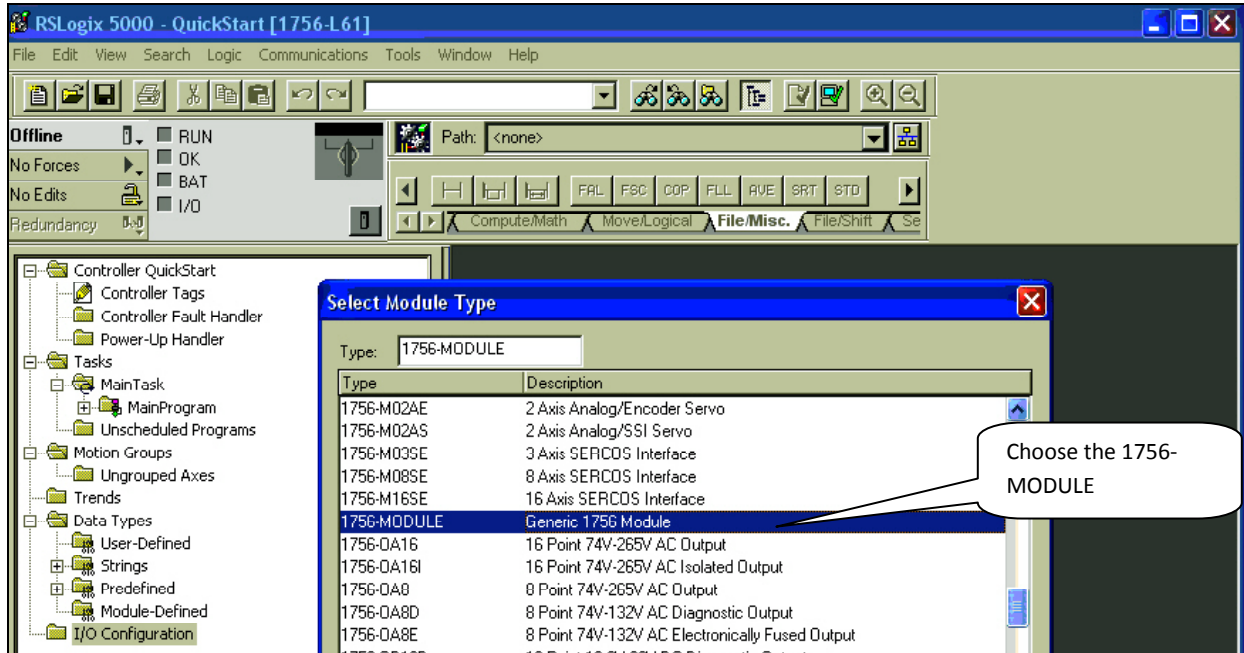


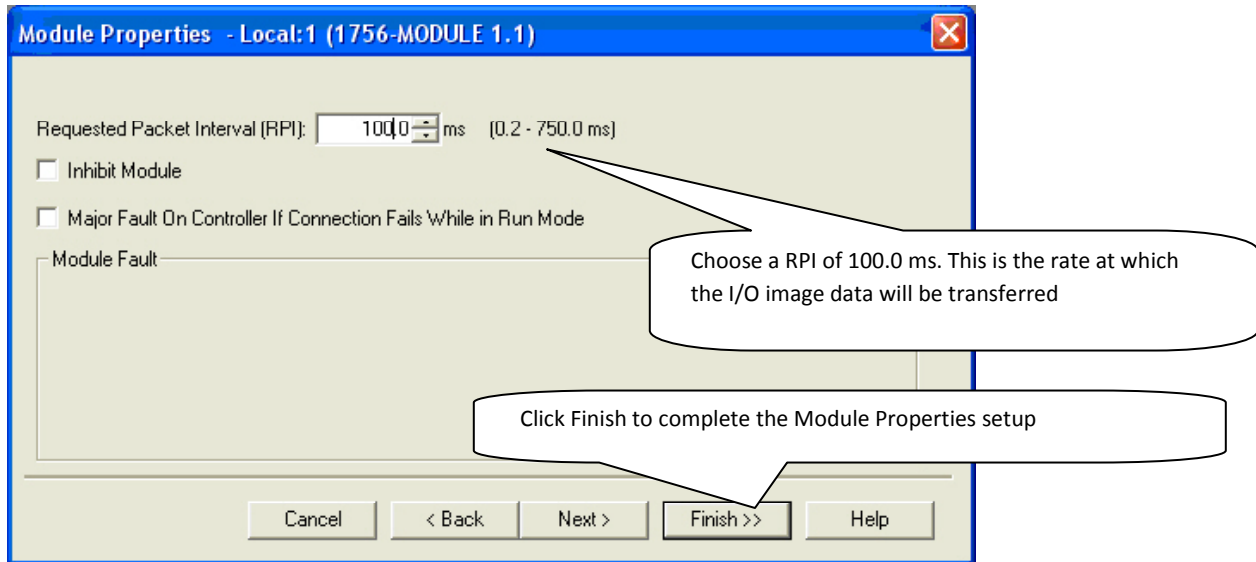
### 4.2 STEP 2: ADD AND CONFIGURE THE SLOTSERVER AS AN IO MODULE

Right-click on I/O Configuration and select “New Module”.

Choose the 1756-MODULE

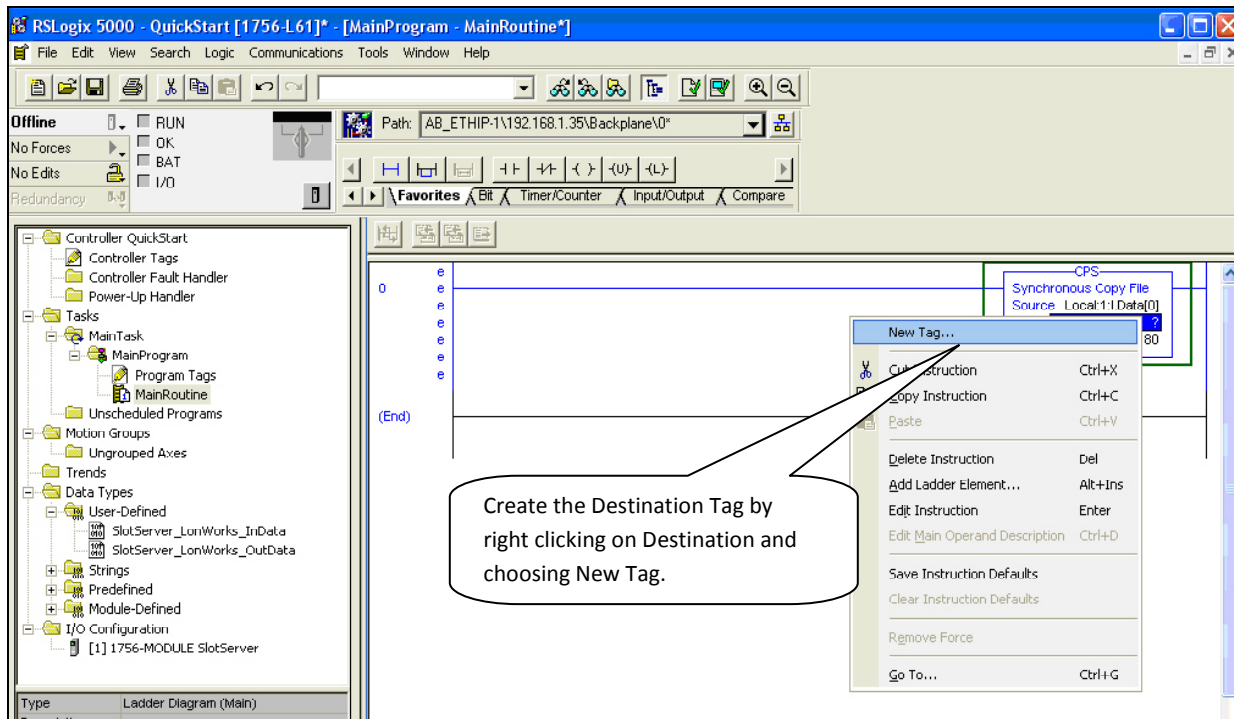
<sup>4</sup> Your Controller may be of a different type to the one shown in the example.

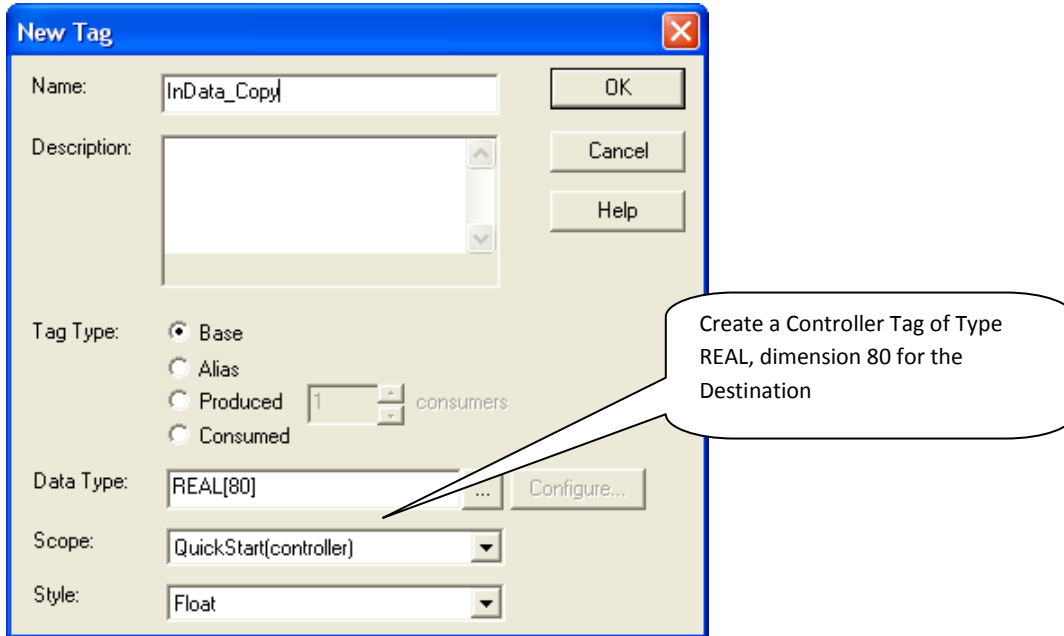




#### 4.3 STEP 3: WRITE LADDER PROGRAM TO GET INPUT DATA FROM DATA ARRAYS

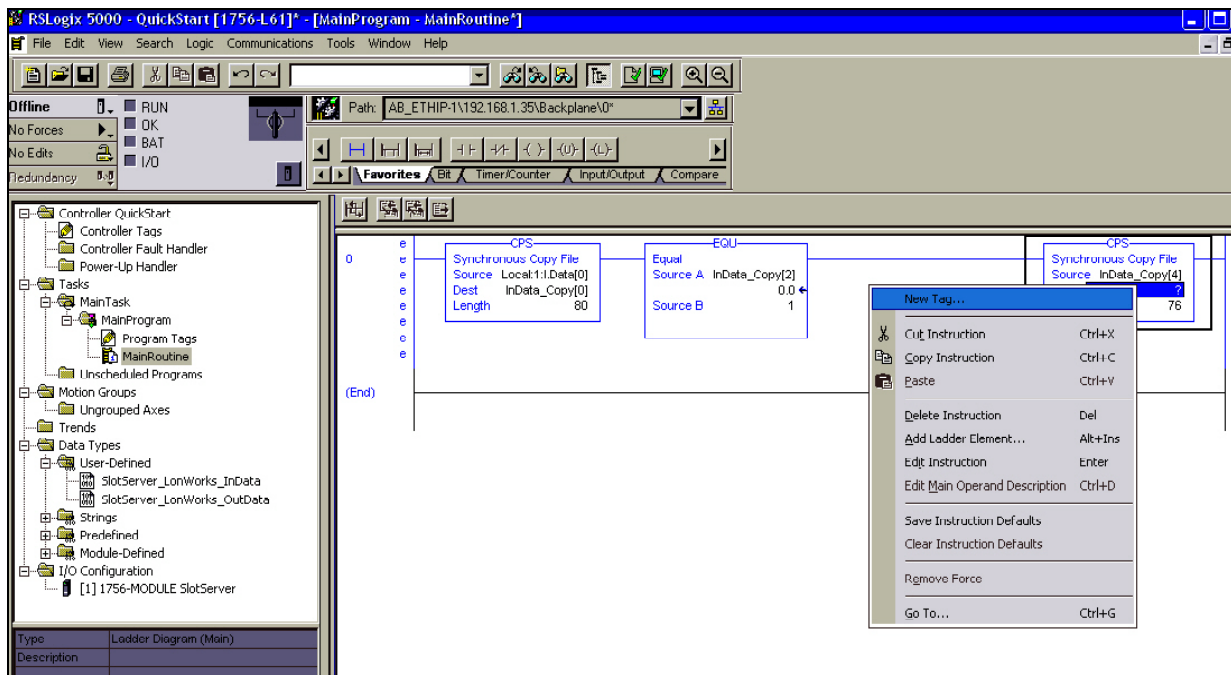
Add a CPS (Synchronous Copy File) Ladder element to synchronize the incoming Data from the Input Data Arrays. Use the Input Image Data as Source.

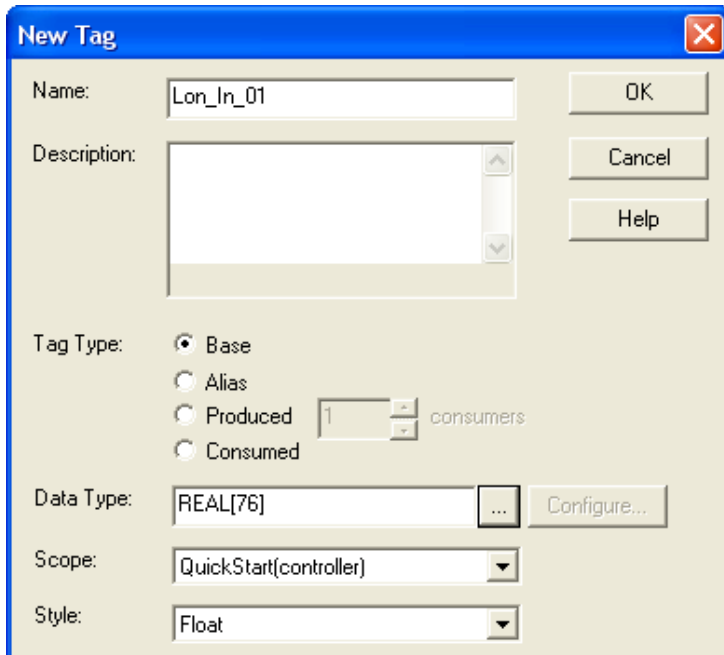




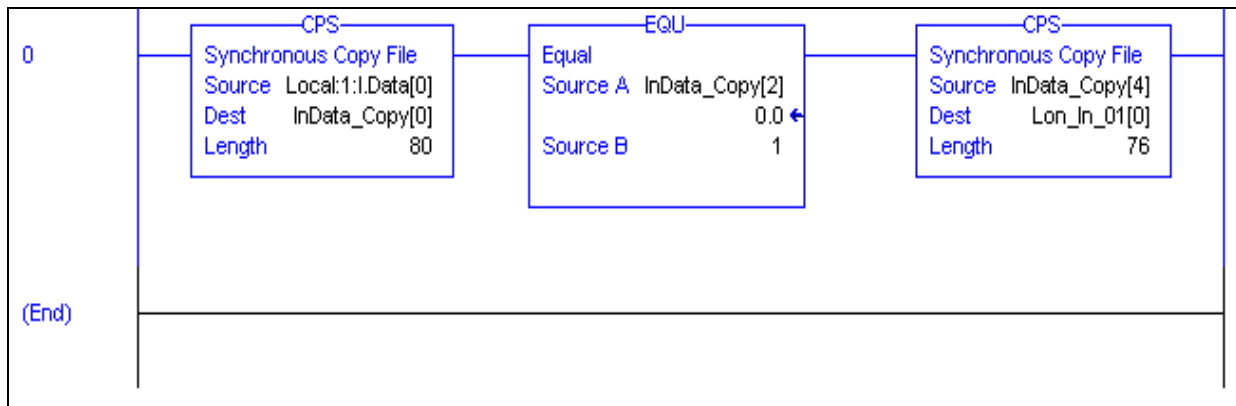
Add an EQU (Compare if equal) ladder element to check when the first Data Array has been received. The block number is at offset 2 of the input image.

Finally, add another CPS ladder element to copy the LonWorks Data from the InData\_Copy Tag to a new Controller Tag, called Lon\_In\_01. Also create the Tag by right clicking on Destination and choosing New Tag. The New Tag must be of type REAL and a dimension of 76.





Below is the final ladder program to access data from LonWorks Function Block In[0]



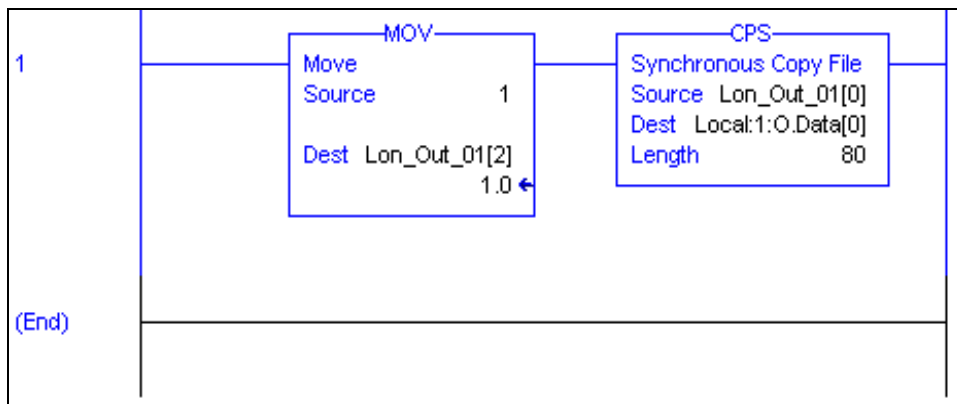
**Very Important Note!**

It is very important to first make a synchronous copy of the input image data before using it. If this is not done, the input data cannot be guaranteed to be from a specific SlotServer Data Array.

#### 4.4 STEP 4: WRITE LADDER PROGRAM TO SEND OUTPUT DATA TO DATA ARRAYS

This step demonstrates how to write data to the Data Array Out[0]

- Create a Controller Tag called Lon\_Out\_01 of type REAL[80].
- Add a new rung to the Ladder program and add a MOV element to move a block number value of 1 into Lon\_Out\_01[2].
- Finally add a CPS (Synchronous Copy File) element to copy the full Lon\_Out\_01 tag into the Output Image Tag.



The LonWorks Data are present from Lon\_Out\_01[4] to Lon\_Out\_01[79]

You can create a User Defined Data Type to replace the type of Lon\_Out\_01 mapping the points to LonWorks point names.

#### Very Important Note!

It is very important to only update all the data of the Output Image Tag once using a Synchronous File Copy element. It is not permissible to update the block number into the Output Image Tag and then the data as this will cause an asynchronous transfer of data.

#### 4.5 STEP 5: DOWNLOAD THE RSLOGIX PROGRAM AND RUN

Use the Who Active or Communications Path directly to Download and Run the Program on the Controller / CPU.

#### 4.6 STEP 6: SET UP THE THIRD PARTY CONNECTION

In this example, this step would involve binding the LonWorks variables using a LonWorks Network Manager.



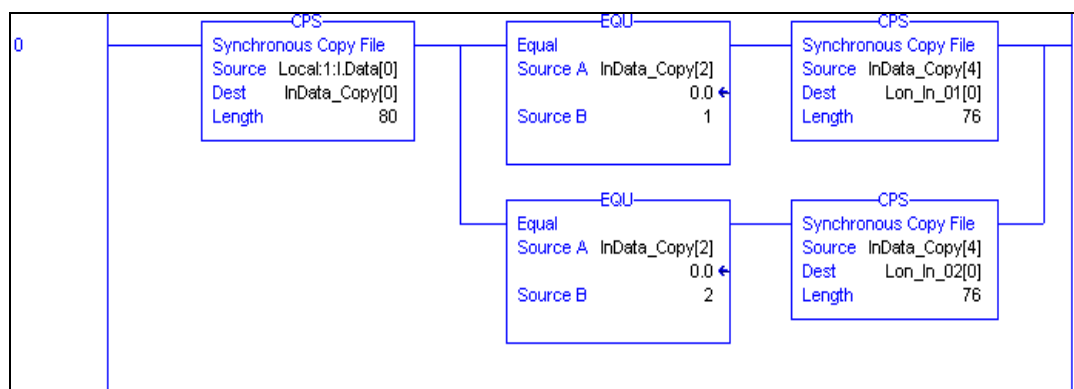
## 5 PROGRAMMING THE CONTROLLOGIX CPU FOR LARGER SLOTSERVER INTERFACES

The previous example is for accessing only one Data Array. The following steps describe how to access multiple Data Arrays.

### 5.1 MULTIPLE INPUT DATA ARRAYS

In this example, we access Input Data Arrays In[1] up to In[24]. We simply add to the existing ladder program as shown in the Quickstart example. Add a branch after the CPS element that copies the input image Tag and copy and paste EQU and CPS elements from the first rung. Create a new Input Tag for Lon\_In\_02 of type REAL and dimension 76. Finally, remember to set the EQU Source B value to 2 to compare for incoming data from the 2<sup>nd</sup> LonWorks functional block which is In[1].

See the ladder program below how to add In[1].



#### Very Important Note!

It is very important to first make a synchronous copy of the input image data before using it. If this is not done, the input data cannot be guaranteed to be from a specific LonWorks Function Block.

### 5.2 ACCESSING MULTIPLE OUTPUT DATA ARRAYS

To access more output Data Arrays it is necessary to create a Multiplexer in Ladder. The Flash drive supplied with the product includes an ACD file with an example of multiplexing use.

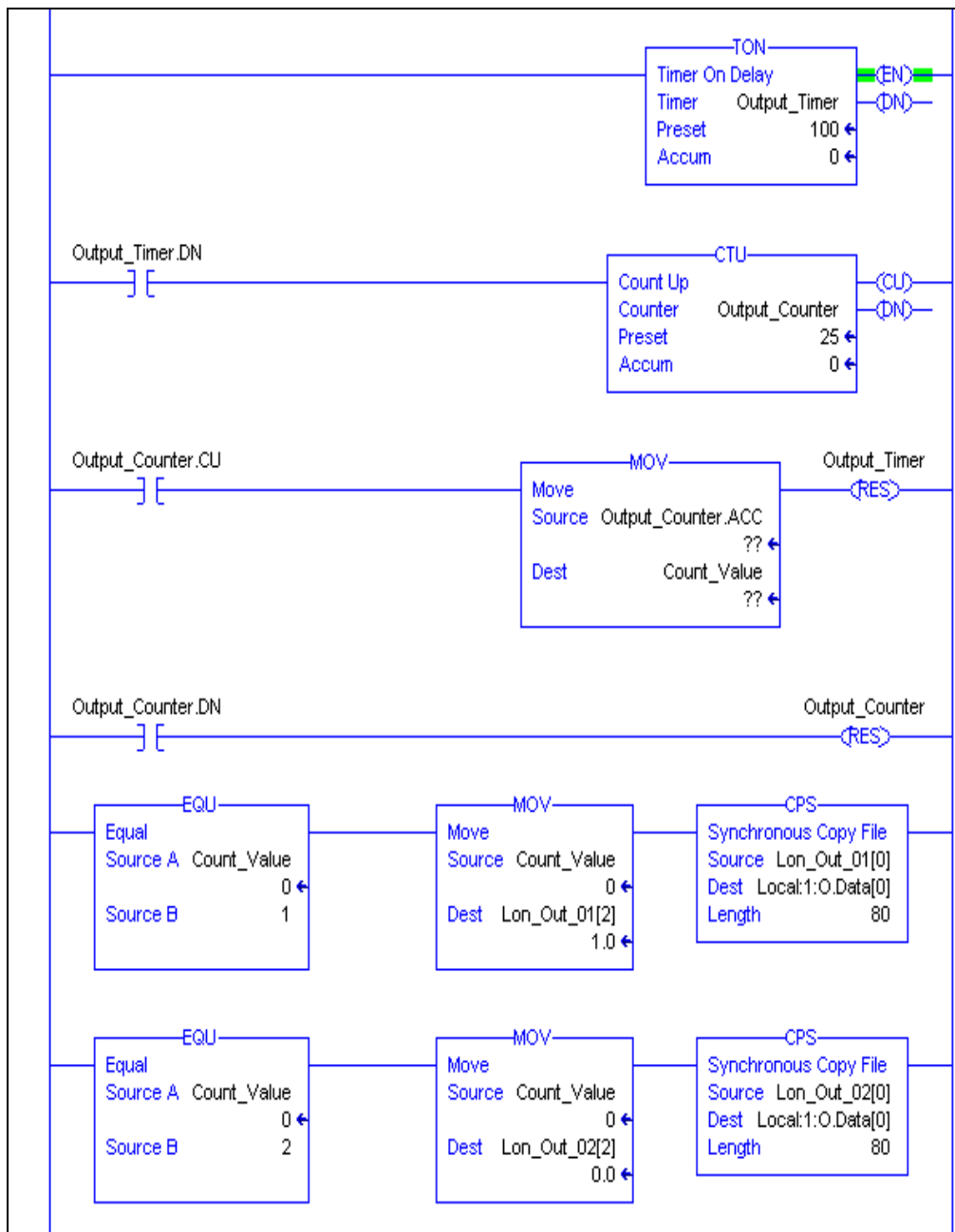
The basic steps include:

- Create a Counter which counts up every 100ms.
- Place the counter value into the Lon\_Out\_xx Tag at offset 2.
- Copy the whole Tag into the output Data Image Tag for transferring to the LonWorks network.

The example program following shows an output counter that can count up to 25 which allows the transfer of data into 25 Output Function Blocks. Only 2 rungs are shown to transfer data for blocks 1 and 2. Add more rungs with more Lon\_Out\_xx tags to transfer data to other output Function Blocks.

It is possible to add up to 65535 blocks. The update rate decreases as the number of blocks goes up. For 25 blocks the update rate is  $25 * RPI = 25 * 0.1 = 2.5$  seconds.

The outputcount may be restricted to a certain value, e.g.2 by changing the Preset value of the CTU element.



**Very Important Note!**

It is very important to only update all the data of the Output Image Tag once using a Synchronous File Copy element. It is not permissible to update the block number into the Output Image Tag and then the data as this will cause an asynchronous transfer of data.

## 6 TIMING PARAMETERS

Under normal operation, the FieldServer will send a poll request to a Server device and that device will reply with a response. The amount of time between successive poll requests is called the **Scan\_Interval**. The time between receiving a response from a Server device and the next poll request is called the **Poll\_Delay**.

If the FieldServer sends a poll request, and the Server device does not send a response, it is considered a timeout. The time the FieldServer waits before declaring a timeout can be adjusted by the **Timeout** parameter. If a timeout occurs, then the FieldServer will retry the poll request a few times (number of times tried is specified by the **retries** parameter). The interval between **retries** is specified by the **Retry\_Interval**. The FieldServer will send poll requests at the end of each **Retry\_Interval**. Once the specified numbers of **Retries** have been sent, the FieldServer will mark the Node offline. Once a Node has been marked offline, it will wait for a period specified by **Recovery\_Interval** before sending another poll request.

Once the communications have been re-established, the FieldServer will wait for a period called **Probation\_Delay**, before marking the Node as online.

**Note 1:** The **lc\_Timeout** parameter monitors the time between characters in a response. If the time exceeds the **lc\_Timeout**, the response is discarded and is considered a Timeout.

**Note 2:** All parameters in **bold** above are configurable. See table below for where they are configured, and what the defaults will be if they are not configured.

Parameter	Default Value	Where Used
Scan_Interval	2 seconds	Map Descriptor, Node, Connection
Poll_Delay	0.05 seconds	Connection
Timeout	2 seconds	Map Descriptor, Node, Connection
Retry_Interval	10 seconds	Node
Retries	3 times	Node
Recovery_Interval	30 seconds	Node
Probation_Delay	1 minute	Node
lc_Timeout	0.5 seconds	Map Descriptor, Node, Connection

**Note 3:** A non-response from the remote Server device causes a Timeout. The driver does nothing until a response is received or the timeout period has expired. Thus if a connection has two Nodes and one Node is producing Timeouts this will have the effect of slowing down communication for the other Node in the sense that the driver does nothing while the timeout timer is counting up to its setpoint. Once there is a timeout on one Node, the driver will not retry any Map Descriptors on that Node until the **Retry\_Interval** has expired. Thus during the **Retry\_Interval** the other Node will get 100% of the service.

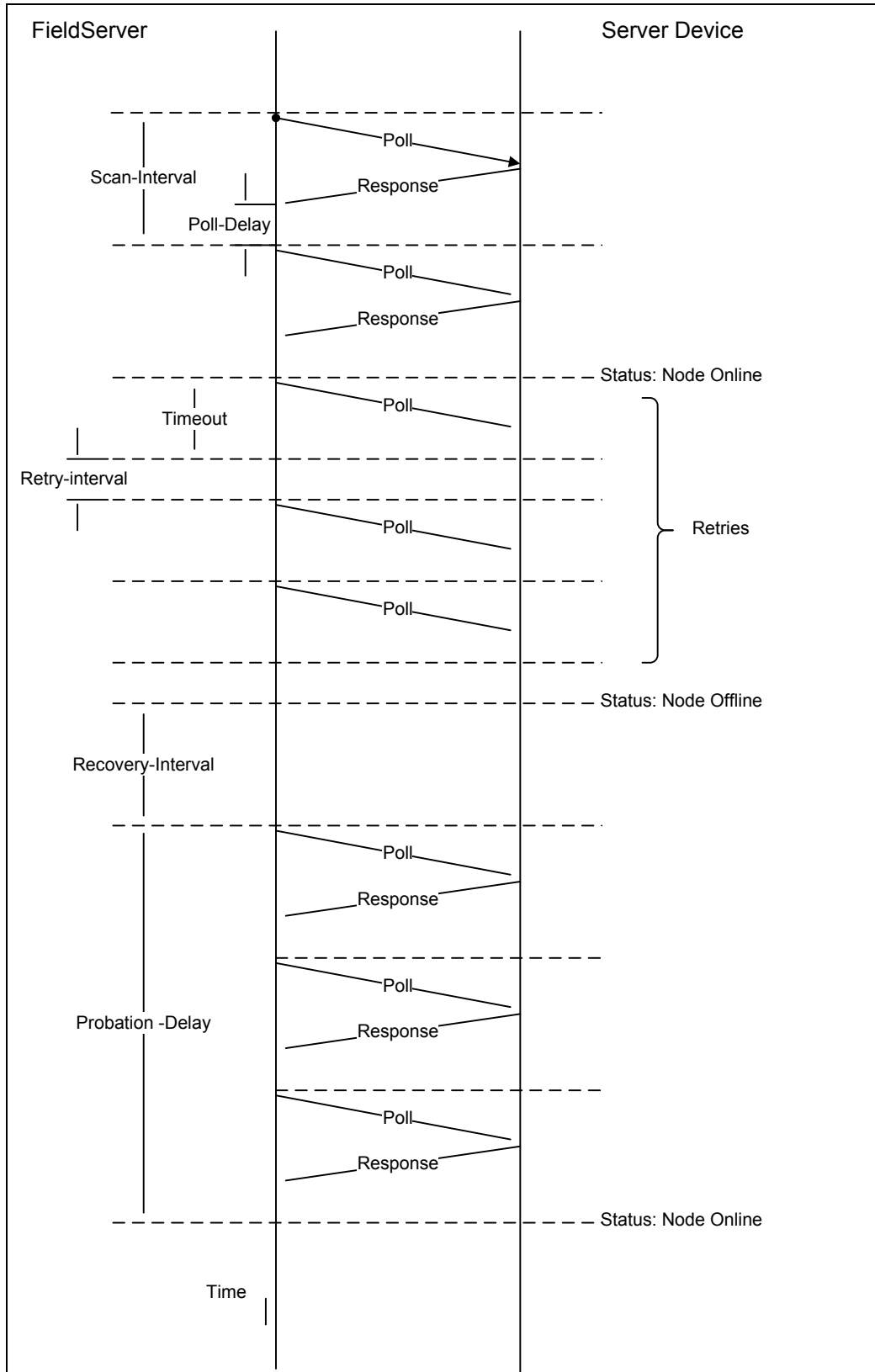


Figure I - SlotServer Timing Diagram



---

## Appendix A. Useful Features

### Appendix A.1. How to obtain Node Status from the SlotServer

By declaring the following Data Array, the Node Status field in the IO image header will be filled in with the Node Statuses of all Nodes declared on the SlotServer:

```
// Data Arrays  
  
Data_Arrays  
Data_Array_Name   , Data_Format   , Data_Array_Length   , Data_Array_Function  
SlotServerNodes  , Bit           , 256                , Node_Status
```

Note: The Data Array Name must be as shown for this function to work correctly.

---

## Appendix B. Troubleshooting

### Appendix B.1. SLOTSERVER : block number [ 0 ] out of range!

It is necessary to put a non-zero buffer number in offset 2 of the output buffer, or to remove the related Map Descriptor if output buffers are not being used. Refer also to Appendix B.2.

### Appendix B.2. Driver Error screen returns “Illegal Protocol” Error.

- Ensure that the configuration is not set to write to offset 0 of the output buffer. Offsets 0, 1 and 3 should be clear and the correct non-zero buffer number should be written into offset 2.
- If the output buffer is not being used then the corresponding Logix Map descriptor should be removed.

---

## Appendix C. Vendor Information

### Appendix C.1. Installing SlotServer on a Remote Rack using CNB Cards<sup>5</sup>

---

#### Appendix C.1.1. RSLogix configuration

In order to see the SlotServer from the CPU, the Hardware must be configured in RSLogix as follows:

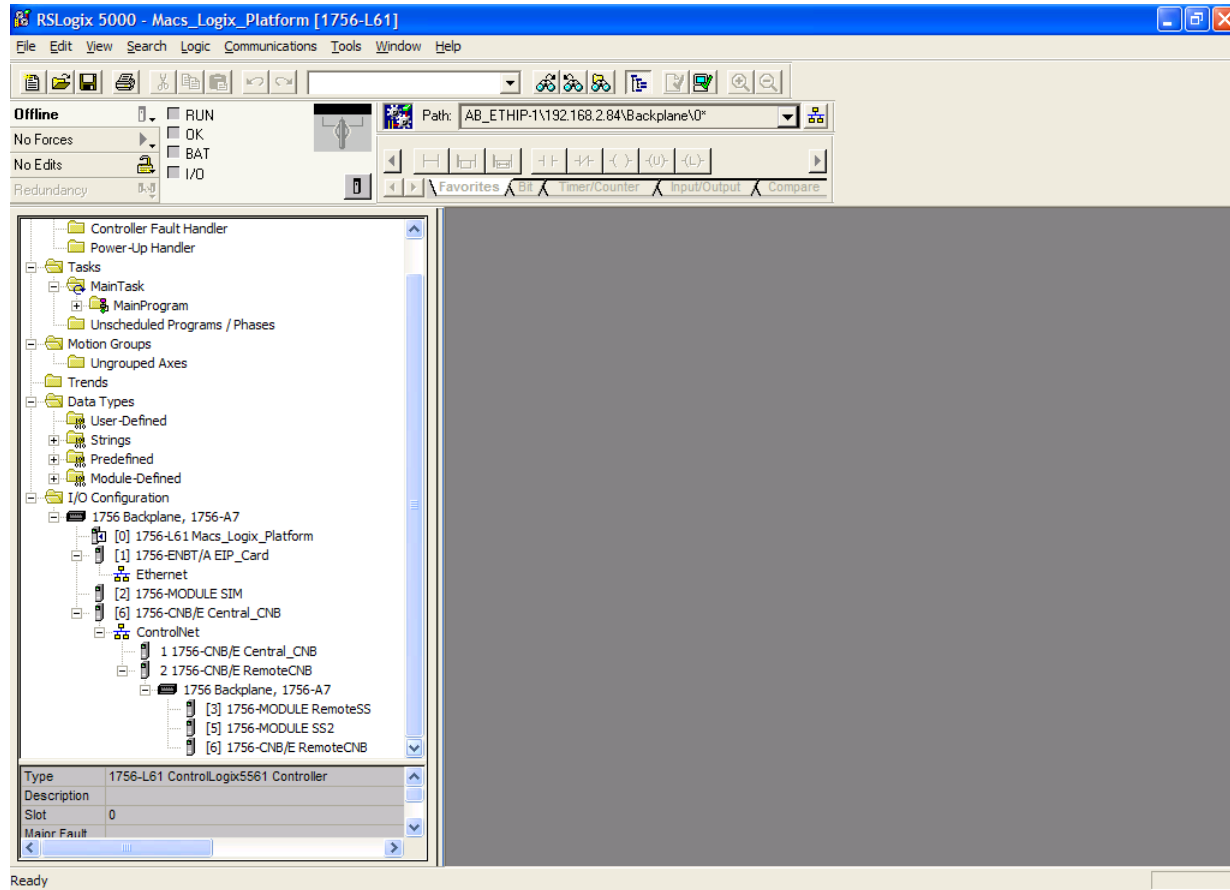
- Configure the cards in the local rack in the I/O Configuration section, including the CPU and the local CNB card.
- Right click on the local CNB card and add the remote CNB card using the “New Module” function.
- Right Click on the remote CNB card, and add the 1756 Backplane
- Right Click on the 1756 Backplane and add the cards that are present in the remote rack, including the SlotServer (As a generic I/O module-see earlier section on how to do this)
- Save the RSLogix configuration, and download it to the PLC.

---

<sup>5</sup> The principles for connecting other 1756 bridging cards to the SlotServer are similar.



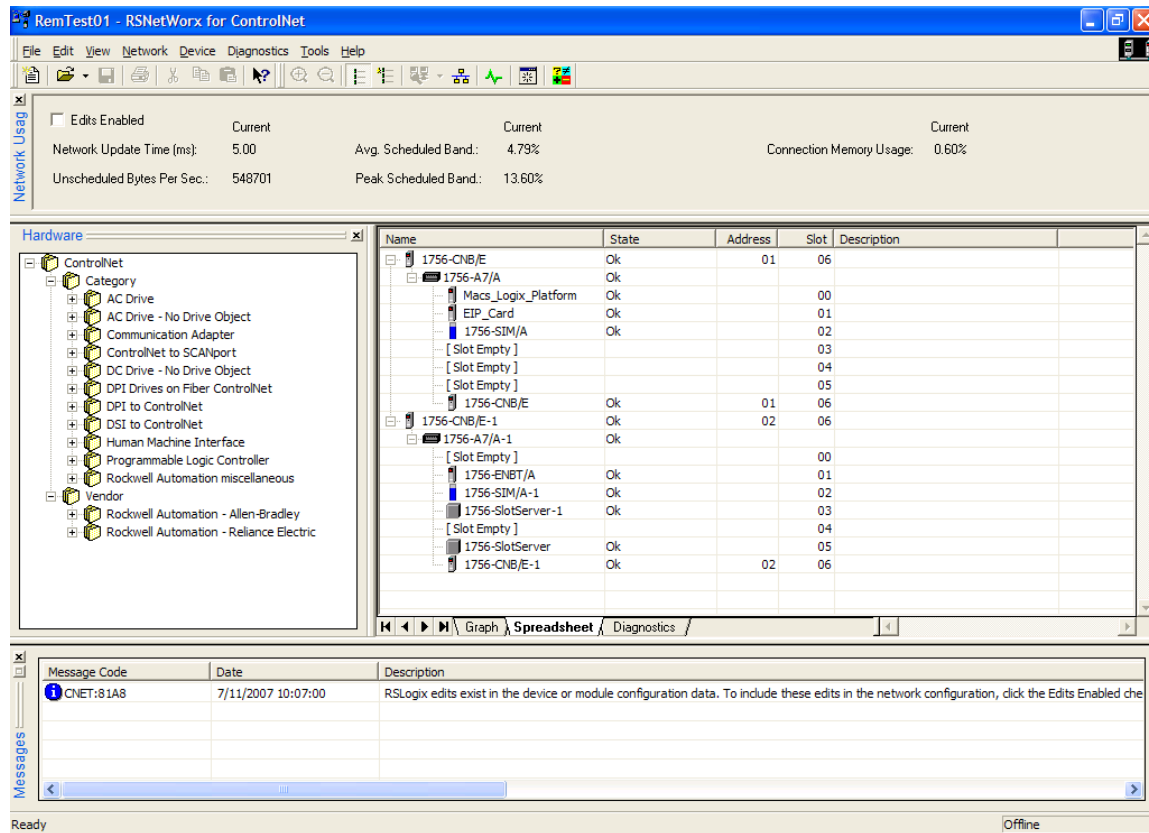
The finished I/O configuration should look similar to this:



#### Appendix C.1.2. RSNetWorx configuration

- Open up RSNetWorx and add the two CNB Cards to the Network by dragging them onto the Network in the Graph tab (Must be done with Edits Enabled). Follow the prompts on the screen to configure the Chassis being used, and the cards in each Chassis (Rack).
- Go Online with RSNetWorx, and then press "save". This will transfer the RSNetWorx Configuration to the Keeper.

The final RSNetworx Configuration should look similar to this:



### Appendix C.1.3. Testing

The SlotServer should now be visible to the CPU.

- Go Online with RSLogix and check the Input buffer of the SlotServer for data.
- Examine offset 2 of the input tag for a non-Zero value. If the SlotServer is multiplexing (DA\_Count >1), then this value will be cycling through the Buffer numbers; if DA\_Count=1, then offset 2 will be fixed at 1.
- If offset 2 is zero, then the SlotServer is not being seen by the CPU, and Diagnostics will need to be performed using RSNetWorx and RSLogix to determine the cause of the problem.

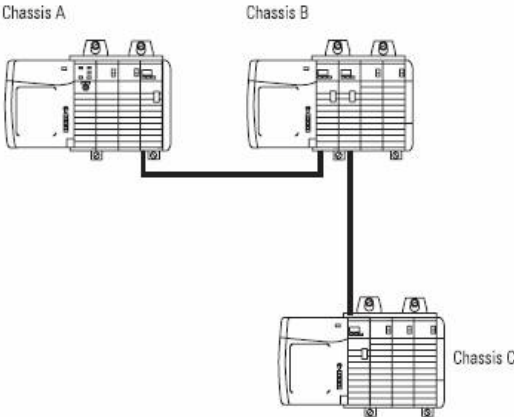
Appendix C.1.4. Connection limitations -Controlling the SlotServer using ControlNet

- Only one remote I/O rack is supported.
- I/O can only be added online using a direct connection.

The following Vendor information provides clarification:

For a ControlLogix controller to control 1756 I/O, the I/O must be:

- in the same chassis as the controller **or**
- on a ControlNet network that is local to that controller **or**
- on an Ethernet/IP network that is local to that controller



The diagram illustrates a network topology with three chassis labeled Chassis A, Chassis B, and Chassis C. Chassis A and Chassis B are connected to each other, and Chassis B is connected to Chassis C. Each chassis is shown with a rack of modules and a network port.

For example, assume that the network links in this example are either ControlNet or EtherNet/IP links. Both links can be the same, or one link can be a ControlNet link and the other can be an EtherNet/IP link. Chassis A can control the 1756 I/O modules in Chassis A and in Chassis B, but not in Chassis C. The ControlLogix controller in Chassis C can only send messages to the devices in Chassis C.

### Add I/O online

When online:

- You can add 1756 I/O modules to the local chassis, remotely via the unscheduled portion of a ControlNet network, and remotely via an EtherNet/IP network.
- The I/O modules you add online use direct connections (rack-optimized connections are not supported when adding I/O modules online).

## Appendix C.2. Dealing with ControlLogix RPI Settings

When setting up the SlotServer for ControlLogix, it is necessary to select the Request Packet Interval (RPI). The RPI is the rate at which data is transferred to and from the SlotServer IO buffers. The following factors need to be considered when deciding on an RPI:

- Minimum RPI setting for the SlotServer is 100ms.
- The Scan\_Interval parameter of the two Logix Map descriptors must be faster than the RPI to ensure smooth communications and prevent timeouts.
- The number of message blocks used does not affect the RPI setting, but will affect the effective update rate for any one message block. The effective update rate for data to/from the SlotServer's Data Arrays to the Logix CPU tags is the scan interval of the block since data is updated to the block every scan interval. Increasing the number of blocks will decrease the effective update rate per block. This update rate does not include the time taken to obtain data from the third party network, which is dependant entirely on the third party protocol involved.
- The program scan rate should be set to run faster than the RPI. We recommend twice as fast. If the ladder program scan is slower than the RPI rate, it would be possible to miss some blocks altogether. It is further recommended that diagnostics be added to the ladder program to detect missed blocks

The effective update rate can be calculated using the following formula:

Effective update rate (EUR) = Scan interval \* (number of msg block pairs<sup>6</sup>)

e.g. (EUR) = 0.2s \* 1 = 0.2s  
when using only one input and output block

e.g. (EUR) = 0.2s \* 25 = 5s  
when using the full LonWorks Open Interface configuration.

---

<sup>6</sup> a message block pair consists of an output and input block

### Appendix C.3. Rules for Naming Logix driver Data Arrays

Unlike other FieldServer drivers, the Logix driver attaches significance to the name of the Data Array used in the Logix Driver Map Descriptor. This allows the user to declare a series of Data Arrays to be multiplexed through the input and output buffers of the SlotServer. In general, Data Arrays would be named In\_1 to In\_x for input buffer arrays, and Out\_1 to Out\_y for output buffer arrays (where x and y are numbers reflecting the maximum input and output Array numbers respectively). For example, an application that multiplexes 6 Data Arrays worth of data through the Input buffer would use Data Arrays named In\_1 through In\_6. In this example, DA\_Name\_Start is declared as In\_1, and DA\_Count is declared as 6.

If an alternative naming convention is required, the following restrictions apply:

- The Data Array name must end in \_x, where x is a positive integer number.
- The total length of the Data Array name (including \_x) must not exceed 15 characters.
- No leading zeros should be used in the \_x number (For example, use \_5, not \_05)
- The “x” part of the \_x in the data array name will be the number shown in offset 2 of the input buffer for the purposes of de-multiplexing in the CPU.
- There is only one Map Descriptor for linking Data Arrays to the input buffer (function Wrbc), and one for linking Data Arrays to the output buffer (function Rdbc). It is therefore not possible to map non-continuous Data Array number sequences. (For example, you can map numbers 5 through 25, but you cannot map numbers 1 through 3, and then 5 through 8 at the same time).
- All Data Arrays (not just the start Data Array) must be declared individually in the Data Arrays Section.

The following examples describe legal and illegal naming conventions respectively:

**Example 1: Legal Map Descriptors:**

Map_Descriptors						
Map_Descriptor_Name	,Scan_Interval	,Function	,Node_Name	,IO_Data_Type	,DA_Name_Start	,DA_Count
Input_BP_Image	,0s	,Wrbc	,CPU1	,INT	,Test_5	,2
Output_BP_Image	,0s	,Rdbc	,CPU1	,INT	,Test_3	,2

**Example 2: Illegal Map Descriptors:**

Map_Descriptors						
Map_Descriptor_Name	,Scan_Interval	,Function	,Node_Name	,IO_Data_Type	,DA_Name_Start	,DA_Count
Input_BP_Image	,0s	,Wrbc	,CPU1	,INT	,Test_05	,2
Output_BP_Image	,0s	,Rdbc	,CPU1	,INT	,Test	,2
Output_BP_Image2	,0s	,Rdbc	,CPU1	,INT	,Test_-3	,-5
Output_BP_Image3	,0s	,Rdbc	,CPU1	,INT	,Test6	,2

---

## Appendix D. Reference

### Appendix D.1. Description of Data Transfer Process

The data connection from the SlotServer to the Logix CPU consists of 496 bytes of input and 496 bytes of output data. Of the 2 Map Descriptors specified, the one with the WRBC function writes data to the Logix CPU filling its input data, and the one with the RDBC function accepts the Logix CPU's output data.

The Map Descriptor's IO\_Data\_Type field organizes the 496 bytes into either Bytes (SINT), Words (INT) or Double Words (DINT or REAL) reducing the number of elements that can be transferred. Of the resulting number of elements, the first 4 are reserved for the IO image header (Refer to Appendix D.2).

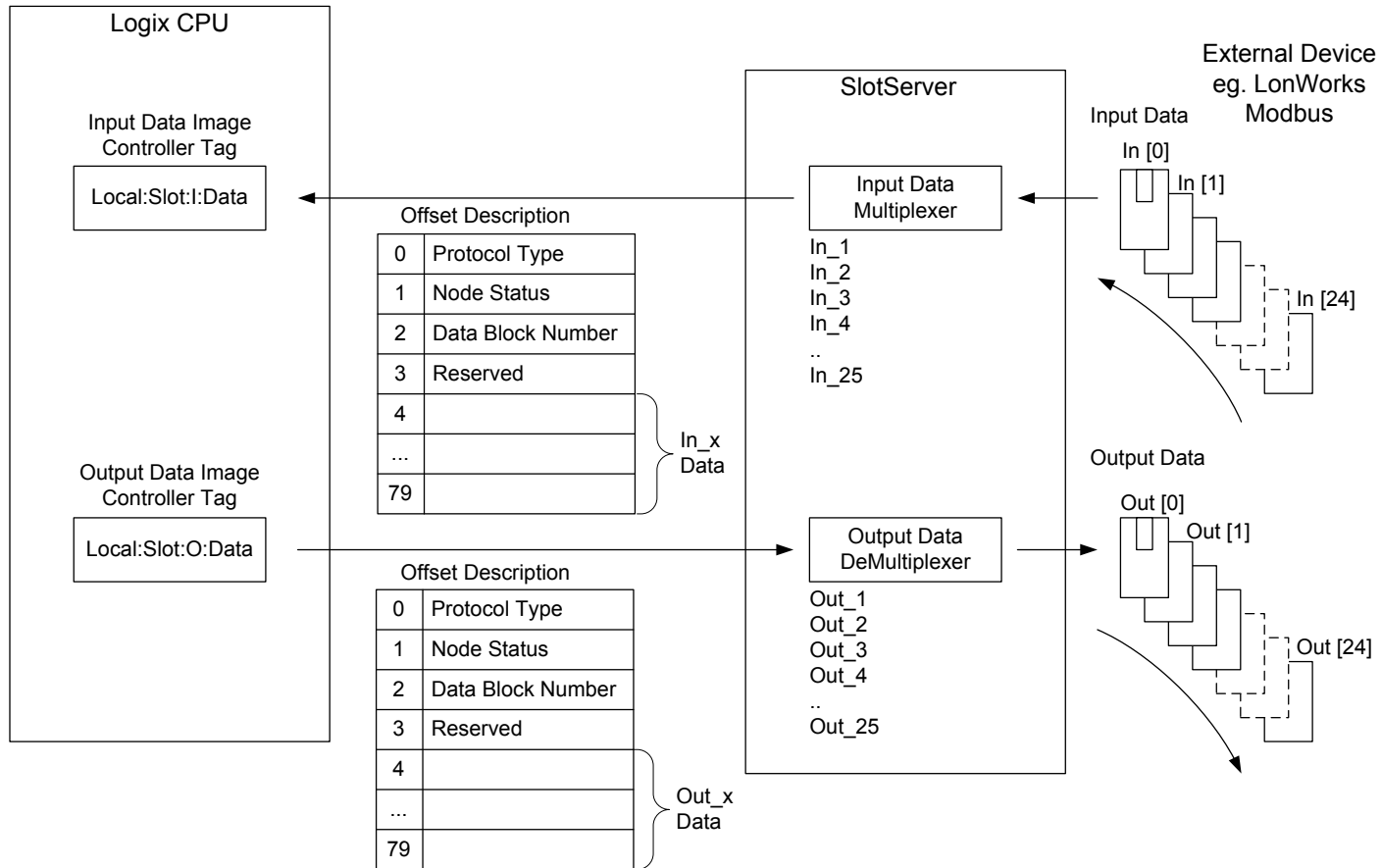
The SlotServer acts as a multiplexer when it sends data to the Logix-CPU and as a demultiplexer when it receives data from the Logix-CPU.

The diagram on the next page describes the SlotServer operation methodology:

For input data, the input data from the external device is placed into the 25 Data Arrays numbered In\_1 to In\_25. The SlotServer sends the data from these Data Arrays over the IO image connection one at a time by placing the block number at offset 2 of the image header and the data from offset 4. The reverse happens at the Logix-CPU where a demultiplexer is implemented in Ladder to route the data to each of the 25 CPU Tags.

For output data, the Logix-CPU has to place the data and block number into the Output Image Tag and send it to the SlotServer. The SlotServer then demultiplexes the data by placing it into the appropriate Out\_x Data Array depending on the block number specified in the IO image header.

SlotServer Data Transfer over IO Data Image





Appendix D.2. The IO image header

The IO image header appears at the start of every block of image data that is transferred to or from the SlotServer to the Logix CPU. It consists of 4 items of data:

Offset into image data block	Item	Description
0	Protocol Type	The value specified under the Map Descriptor's Protocol_Type_ID field is transferred to the Logix CPU and can be used to decode the protocol. The same value has to be transferred back to the SlotServer to indicate the protocol.
1	Node Status	This field is automatically filled in by the SlotServer if a Node Status Data Array with the name SlotServerNodes is declared. Its value can be used in the Logix CPU to check the status of Nodes connected to the SlotServer.
2	Block Number	The number of the Data Array for which the IO image data is valid for, e.g. a block number of 1 will indicate the data is to or from dataArray_1
3	Reserved	Not used