



p/n 88-025607-01A

EtherNet/IP: Using ACR Series Controllers with ControlLogix

Effective: March 2005



IMPORTANT

User Information



Warning!



ACR Series products are used to control electrical and mechanical components of motion control systems. You should test your motion system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

ACR series products and the information in this guide are the proprietary property of Parker Hannifin Corporation or its licensors, and may not be copied, disclosed, or used for any purpose not expressly authorized by the owner thereof.

Since Parker Hannifin constantly strives to improve all of its products, we reserve the right to change this guide, and software and hardware mentioned therein, at any time without notice.

In no event will the provider of the equipment be liable for any incidental, consequential, or special damages of any kind or nature whatsoever, including but not limited to lost profits arising from or in any way connected with the use of the equipment or this guide.

ControlLogix and RSLogix are trademarks of Rockwell Automation, Inc. and Rockwell Software, Inc.

© 2003 Parker Hannifin Corporation
All Rights Reserved

Technical Assistance

Contact your local automation technology center (ATC) or distributor.

North America and Asia

Parker Hannifin
5500 Business Park Drive
Rohnert Park, CA 94928
Telephone: (800) 358-9070 or (707) 584-7558
Fax: (707) 584-3793
Email: emn_support@parker.com
Internet: <http://www.parkermotion.com>

Europe (non-German speaking)

Parker Hannifin
21 Balena Close
Poole, Dorset
England BH17 7DX
Telephone: +44 (0)1202 69 9000
Fax: +44 (0)1202 69 5750
Email: eme_application@parker.com

Germany, Austria, Switzerland

Parker Hannifin
Postfach: 77607-1720
Robert-Bosch-Str. 22
D-77656 Offenburg
Telephone: +49 (0) 781 509-0
Fax: +49 (0) 781 509-176
Email: eme_application@parker.com

Italy

Parker Hannifin
20092 Cinisello Balsamo
Milan, Italy via Gounod, 1
Telephone: +49 (0) 781 509-0
Fax: +49 (0) 781 509-176
Email: sales.sbc@parker.com



Automation

Technical Support E-mail

emn_support@parker.com

Table of Contents

Chapter 1 Introduction	1
Introduction	2
Compatible Parker Hannifin Products	2
Assumptions of Technical Experience	2
Technical Support	2
ACR9000	2
Ethernet/IP	2
ControlLogix and RsLogix.....	2
Chapter 2 Tutorial	3
Overview.....	4
Class 1 I/O (UDP)	4
Class 3 (TCP) CIP Messages (connected and unconnected).....	4
Requirements.....	4
Objective	5
Demonstration Program.....	6
I/O configurations	2
Limits.....	2
General Purpose I/O	2
Fast Status—A Special Case	4
Messages Class3 Explicit.....	6
Multiple Attributes—0x0A	6
Block Read Table—0x4C	6
Block Write Table—0x4D.....	7
NAND and OR Masks—0x4E	8
Binary Move Block—0x34.....	9
Float Read—0x48	11
General Flow of Messages.....	13
RSLogix I/O Configuration.....	14
Adding ACR9000 as an I/O module	14
Request Packet Interval (RPI).....	15
RSLinx Driver configuration.....	16
Driver Setup.....	16
Driver Diagnostics.....	17

CHAPTER ONE

Introduction

IN THIS CHAPTER

• Introduction	2
• Compatible Parker Hannifin Products.....	2
• Assumptions of Technical Experience	2
• Technical Support	2

Introduction

The purpose of this guide is to help you set up the ACR9000 Stand-Alone Controller for use with RsLogix software using EtherNet/IP.

Compatible Parker Hannifin Products

Controllers..... ACR9000

Assumptions of Technical Experience

To install and troubleshoot the ACR9000 Stand-Alone Controller, you should have a fundamental understanding of the following:

- Electronic concepts such as voltage, current, and switches.
- Mechanical motion control concepts such as inertia, torque, velocity, distance, and force.

Before setting up an EtherNet/IP network, you should have a fundamental understanding of the following:

- CIP (Control and Information Protocol) object models for devices.
- CIP object classes for connected and unconnected messaging.
- EtherNet/IP adaptation of CIP.

Technical Support

ACR9000

For solutions to your questions about implementing the ACR9000 Stand-Alone Controller, refer to the following documents:

- *ACR9000 Hardware Installation Guide* (this document)
- *ACR User's Guide (Online Help System in the ACR-View software)*

If you cannot find the answer in these documents, contact your local Automation Technology Center (ATC) or distributor for assistance.

If you need to talk to our in-house Application Engineers, please contact us at the numbers listed in "Technical Assistance" on the inside cover, page ii.

Ethernet/IP

For technical questions regarding EtherNet/IP or CIP, contact your local EtherNet/IP user group. You can find information at www.odva.org.

ControlLogix and RsLogix

For technical questions regarding RsLogix software, contact Rockwell Automation. You can find information at <http://support.rockwellautomation.com/>

CHAPTER TWO

Tutorial

IN THIS CHAPTER

• Overview	4
• I/O configurations	2
• General Purpose I/O	2
• Fast Status—A Special Case.....	4
• Messages Class3 Explicit	6
• General Flow of Messages	13
• RSLogix I/O Configuration	14
• Request Packet Interval (RPI)	15
• RSLinx Driver configuration	16

Overview

The tutorial demonstrates the functionality of ACR9000 Ethernet/IP with ControlLogix software. Throughout the examples, the ACR9000 acts as a server (also known as adapter or slave), and ControlLogix acts as the client (also known as scanner or master). Communication between the two devices can consist of the following:

- Class 1 I/O (UDP) (Implicit Messages)
- Class 3 (TCP) (Explicit Messages)

Class 1 I/O (UDP)

With Class 1 I/O communication, the scanner (RSLogix) subscribes to the adapter (ACR9000). When ControlLogix establishes a connection with the ACR9000 controller, it configures the ACR9000 data for cyclical production and consumption. When configuration is completed, the two devices can share I/O data in both directions.

Class 3 (TCP) CIP Messages (connected and unconnected)

With Class 3 communication, the scanner (ControlLogix) initiates the connection. If data needs repeated transmission, this connection should be cached. This reduces the overhead related to establishing and closing connections.

ControlLogix uses Request/Reply message format to communicate with ACR9000. In addition, it can transmit other generic or vendor specific CIP objects over this connection.

Requirements

The following hardware is necessary for this demonstration:

- ACR9000
- 1756-L1/A Logix5550 (or comparable Rockwell product)
- 1756-ENET/B Ethernet Bridge (or comparable Rockwell product)

The following software is necessary:

- ACR-View
- RSLogix
- RSLinx

Objective

For this tutorial, you must enter and download the demonstration program to the ACR9000—see the subsequent section for the necessary AcroBasic code.

When run, the program performs three tests concurrently: motion, PLC, and I/O. Each test is independent of the others, but when run concurrently they will affect the overall communication throughput.

Motion Test

ControlLogix runs an AcroBasic motion program—PROG0 in the ACR 9000—that performs 2-axis coordinated motion. The test runs in an endless loop, and does the following:

- ControlLogix gets the current position of the axes. If the current position of the axes is equal to the target position, then ControlLogix downloads new target points for the next move.
- ControlLogix compares the anticipated and actual time to complete these moves. The variation in time is stored in an accumulator for the checking the performance of the Ethernet/IP connection.

PLC Test

ControlLogix runs an AcroBasic PLC program—PLC0 in the ACR 9000—that starts a 10 second timer. The test runs in an endless loop, and does the following:

- When the timer has run out, it resets and starts again.
- ControlLogix continuously monitors the timer and checks that it is expiring every 10 second. It captures the variation-errors in the timer expirations.

I/O Test

ControlLogix runs an AcroBasic motion program—PROG1 in the ACR 9000—that performs a short-circuit test on the ACR9000. The test requires the adapter and scanner to produce and consume data cyclically, every 20 milliseconds. The test runs in an endless loop, and does the following:

- ControlLogix writes a random number to the expansion I/O input of the ACR9000.
- ACR9000 copies the input data to the Output of ACR9000
- ControlLogix reads the ACR9000 output data, and compare the received and sent data. If equal, ControlLogix sends the next incremental move Target.
- ControlLogix tracks the number of writes and any errors that might have occurred.

Demonstration Program

Use the following program for the tutorial.

```
SYS
  HALT ALL
  DETACH ALL
  NEW ALL
  CLEAR

  PERIOD 0.001

  DIM PROG0 2000
  DIM PROG1 1000
  DIM PLC0 1000

  PROG0

  ATTACH MASTER0
  ATTACH SLAVE0 AXIS0 "X"
  ATTACH SLAVE1 AXIS1 "Y"

  VEL 707.107
  FOV 0.5

  STP 0 ACC0 DEC 0
  CLR 8467
  CLR 8499
  CLR 522

  1 PBOOT
  2 RUN PLC0
  3 RUN PROG1
  4 DWL 1
  5 SET 32

  10 P8248=0
  12 P8249=0

  15 CLR 522
  20 PRINT "CONTROLLOGIX HALTED THE
  MOTION"
  22 DWL 1
  23 PRINT P8249,P8250
  25 IF (P8249=0) THEN GOTO 20
  30 P8250 =0
  40 SET 32
  50 X (P8504) Y (P8505)
  55 IF (!BIT516) THEN GOTO 60
  56 DWL .5
  57 PRINT P8504,P8505, P4104
  60 IF (P8250=0) THEN GOTO 50
  70 GOTO 12

PROG1

  10 IF (BIT1552=0) GOTO 10
  15 DWL 2
  20 CLR 32
  30 DWL 1
  40 SET 32
  50 GOTO 10

PLC0
  CLR 32

  10 LD 32
  20 TMO 1000
  30 OUT 34

PROG0

  DWL 1

  RUN PROG1
  DWL 1

  SET 32
  P8249=0
  P8250=0
  LRUN
```


I/O configurations

To establish a Class 1 connection, the scanner sends a `ForwardOpen` request to the adapter (ControlLogix to ACR9000). The `ForwardOpen` request contains device specific configuration information along with the type and number of parameters to access. The connection can be multicast or peer-to-peer.

Limits

This I/O is cyclic, and the update rate of this I/O is user defined. The update rate limits are as follows:

Min RPI 100 ms
 Max RPI 3200 ms
 Timeout 4 times the cyclic rate (RPI)

The Maximum limit on the size of I/O is as follows:

Total parameter 100
 Total I/O bits 3200
 Max group 16
 Max parameter in each group 8

General Purpose I/O

Following table details the information to set up for general purpose I/O. For an example how the information is entered in the RSLogix software, see below.

Description	Value	Size
Connection Type	33 (Always)	DINT
Number of Parameter	1-16 (min – max)	DINT
Parameter List [0]. Offset	Any P-parameter number	DINT
Parameter List [0]. Length	Block size; consecutive P-parameters to read	DINT
Parameter List [0]. Direction	1 = INPUT to ACR9000 0 = OUTPUT from ACR9000	DINT
Parameter List [1]. Offset	:	:
Parameter List [1]. Length	:	:
Parameter List [1]. Direction	:	:
Parameter List [2]. Offset	:	:
Parameter List [2]. Length	:	:
Parameter List [2]. Direction	:	:
:	:	:
Parameter List [15]. Offset	:	:
Parameter List [15]. Length	:	:
Parameter List [15]. Direction	:	:

Example

In this example, configure the general purpose I/O for five P-parameters on the ACR9000 (P8504 and P8505 are inputs to ACR9000) (P12288, P12544, and P4144 are outputs from ACR9000).

The following data is entered in **ConfigData** tag:

1. In the **Value** cell of the **ConfigData.ConnType** row, type 33.
2. In the **Value** cell of the **ConfigData.NumParms** row, type 4.
This represents the total number of parameter groups being set up.
3. Under **ParmList[0]**, enter the following:
 - a. In the **Value** cell of the **Offset** row, type 8504.
This is the P-parameter to access.
 - b. In the **Value** cell of the **Length** row, type 2.
This is the number of consecutive parameters to access.
 - c. In the **Value** cell of the **Direction** row, type 1.
This sets the data as an input for ACR9000 (output for ControlLogix).
4. Under **ParmList[1]**, enter the following:
 - a. In the **Value** cell of the **Offset** row, type 12288.
This is the P-parameter to access.
 - b. In the **Value** cell of the **Length** row, type 1.
This is the number of consecutive parameters to access.
 - c. In the **Value** cell of the **Direction** row, type 0.
This sets the data as an output from ACR9000 (input for ControlLogix).
5. Under **ParmList[2]**, enter the following:
 - a. In the **Value** cell of the **Offset** row, type 12544.
This is the P-parameter to access.
 - b. In the **Value** cell of the **Length** row, type 1.
This is the number of consecutive parameters to access.
 - c. In the **Value** cell of the **Direction** row, type 0.
This sets the data as an output from ACR9000 (input for ControlLogix).
6. Under **ParmList[3]**, enter the following:
 - a. In the **Value** cell of the **Offset** row, type 4144.
This is the P-parameter to access.
 - b. In the **Value** cell of the **Length** row, type 1.
This is the number of consecutive parameters to access.
 - c. In the **Value** cell of the **Direction** row, type 0.
This sets the data as an output from ACR9000 (input for ControlLogix).

Tag Name	Value	Force Mask	Style	Type
ConfigData	{...}	{...}		ConfigData
ConfigData.ConnType	33		Decimal	DINT
ConfigData.NumParams	4		Decimal	DINT
ConfigData.ParamList	{...}	{...}		ParamCtg[16]
ConfigData.ParamList[0]	{...}	{...}		ParamCtg
ConfigData.ParamList[0].Offset	8504		Decimal	DINT
ConfigData.ParamList[0].Length	2		Decimal	DINT
ConfigData.ParamList[0].Direction	1		Decimal	DINT
ConfigData.ParamList[1]	{...}	{...}		ParamCtg
ConfigData.ParamList[1].Offset	12288		Decimal	DINT
ConfigData.ParamList[1].Length	1		Decimal	DINT
ConfigData.ParamList[1].Direction	0		Decimal	DINT
ConfigData.ParamList[2]	{...}	{...}		ParamCtg
ConfigData.ParamList[2].Offset	12544		Decimal	DINT
ConfigData.ParamList[2].Length	1		Decimal	DINT
ConfigData.ParamList[2].Direction	0		Decimal	DINT
ConfigData.ParamList[3]	{...}	{...}		ParamCtg
ConfigData.ParamList[3].Offset	4144		Decimal	DINT
ConfigData.ParamList[3].Length	1		Decimal	DINT
ConfigData.ParamList[3].Direction	0		Decimal	DINT
ConfigData.ParamList[4]	{...}	{...}		ParamCtg
ConfigData.ParamList[5]	{...}	{...}		ParamCtg
ConfigData.ParamList[6]	{...}	{...}		ParamCtg
ConfigData.ParamList[7]	{...}	{...}		ParamCtg

Fast Status – A Special Case

Instead of subscribing to specific parameters, as shown in the previous section, you use fast status. Fast status allows you to read a series of parameters. For more information, see the `FSTAT` command in the ACR User's Guide.

The size of the fast status on the ControlLogix is fixed: 80 DINT (320 bytes). You can read long or float variables from the ACR9000. All these values are received in the ControlLogix input buffer. In the case of a float parameter you must copy the float values to the RsLogix Tag with real data type.

Description	Value	Size
Connection Type	33 always	DINT
Number of Parameter	1-16 (min – max)	DINT
Parameter List [0]. Offset	Any P-parameter number	DINT
Parameter List [0]. Length	Block size; consecutive P-parameters to read	DINT
Parameter List [0]. Direction	1 = INPUT to ACR9000 0 = OUTPUT from ACR9000	DINT
Parameter List [1]. Offset	:	DINT
Parameter List [1]. Length	:	DINT
Parameter List [1]. Direction	:	DINT
Parameter List [j]. Offset	Mask BIT-n is set if FSTAT-n is a float type and vice versa	DINT
Parameter List [j]. Length	80 always fixed size	DINT
Parameter List [j]. Direction	55 type is FSTAT	DINT

Example

The example is similar to the previous. However, in this case ControlLogix accesses (reads) the fast status of the ACR9000 controller. Use the `FSTAT` command on the ACR9000 to set up the fast status.

In RSLogix, the difference is in how you set up `ParmList[1]`. Because you can read long and float values, you need to specify what those variables are. You can do this through a binary mask— a set bit indicates the `FSTAT` type real, and a clear bit indicates long. The mask is then converted to decimal notation and supplied as the offset.

Suppose you are setting up the following:

FSTAT0 long

FSTAT1 long

FSTAT2 real

FSTAT3 real

FSTAT4 long

FSTAT5 real

With each set bit representing real and each clear bit representing long, the binary mask becomes 101100. Converted to decimal notation, we can write

ConfigData.Offset =Mask= 44.

The following data is entered in **ConfigData** tag:

1. In the **Value** cell of the **ConfigData.ConnType** row, type 33. This is necessary for an I/O connection.
2. In the **Value** cell of the **ConfigData.NumParms** row, type 2. This represents the total number of parameter groups being set up.
3. Under **ParmList[0]**, enter the following:
 - a. In the **Value** cell of the **Offset** row, type 8504. This is the P-parameter to access.
 - b. In the **Value** cell of the **Length** row, type 2. This is the number of consecutive parameters to access.
 - c. In the **Value** cell of the **Direction** row, type 1. This sets the data as an input for ACR9000 (output for ControlLogix).
4. Under **ParmList[1]**, enter the following:
 - a. In the **Value** cell of the **Offset** row, type 44. This mask indicates which fast status indexes use data type real or long.
 - b. In the **Value** cell of the **Length** row, type 80. For fast status, this is fixed. It is always 80.
 - c. In the **Value** cell of the **Direction** row, type 55. This specifies the request is fast status.

Note: You can mix fast status with ControlLogix output. In the example above, the ControlLogix is also writing to P8204 and P8205.

Note: When using fast status, do not set up other inputs to the ControlLogix.

Note: Make sure the ConfigData.ParmList for fast status is last.

Tag Name	Value	Force Mask	Style	Type
[-] ConfigData	{...}	{...}		ConfigData
[+] ConfigData.ConnType	33		Decimal	DINT
[+] ConfigData.NumParams	2		Decimal	DINT
[-] ConfigData.ParamList	{...}	{...}		ParamCtg[16]
[-] ConfigData.ParamList[0]	{...}	{...}		ParamCtg
[+] ConfigData.ParamList[0].Offset	8504		Decimal	DINT
[+] ConfigData.ParamList[0].Length	2		Decimal	DINT
[+] ConfigData.ParamList[0].Direction	1		Decimal	DINT
[-] ConfigData.ParamList[1]	{...}	{...}		ParamCtg
[+] ConfigData.ParamList[1].Offset	44		Decimal	DINT
[+] ConfigData.ParamList[1].Length	80		Decimal	DINT
[+] ConfigData.ParamList[1].Direction	55		Decimal	DINT
[+] ConfigData.ParamList[2]	{...}	{...}		ParamCtg
[+] ConfigData.ParamList[3]	{...}	{...}		ParamCtg
[+] ConfigData.ParamList[4]	{...}	{...}		ParamCtg
[+] ConfigData.ParamList[5]	{...}	{...}		ParamCtg
[+] ConfigData.ParamList[6]	{...}	{...}		ParamCtg
[+] ConfigData.ParamList[7]	{...}	{...}		ParamCtg

Messages Class3 Explicit

Following is the list of ACR9000 device specific objects.

Description	Service Code	Class Hex	Instance Hex	Attribute Hex
Multiple Attribute	0x0A	FFFF	FFFF	FFFF
Read Tag	0x4C	FFFF	FFFF	FFFF
Write Tag	0x4D	FFFF	FFFF	FFFF
Mask NAND OR	0x4E	4	4	4
Vendor Move	0x34	4	4	4
Vendor Float	0x48	Parameter Number	Number of Elements	1

Multiple Attributes – 0x0A

The 0x0A service code allows Interact software (from Parker Hannifin) to communicate with an ACR9000. This service wraps around other service codes such as 4C, 4D, and 4E. It allows you to pack multiple service codes inside 0X0A and efficiently send across the network.

For more information, see the Rockwell documentation.

Block Read Table – 0x4C

The 0x4C service code allows you to read certain 32-bit registers. The object can read any long P-parameter in ACR9000.

You can also use it for multiple-block reads. This is an efficient method to read a contiguous block of P-parameters. You can read 1-16 elements.

For more information, see the Rockwell documentation.

Block Write Table – 0x4D

The 0x4D service code allows you to write to any 32-bit Register. The object can write to any P-parameter in ACR9000 with the data type long or float. You can also use it for multiple-block writes. This is an efficient method to write a contiguous block of P-parameters.

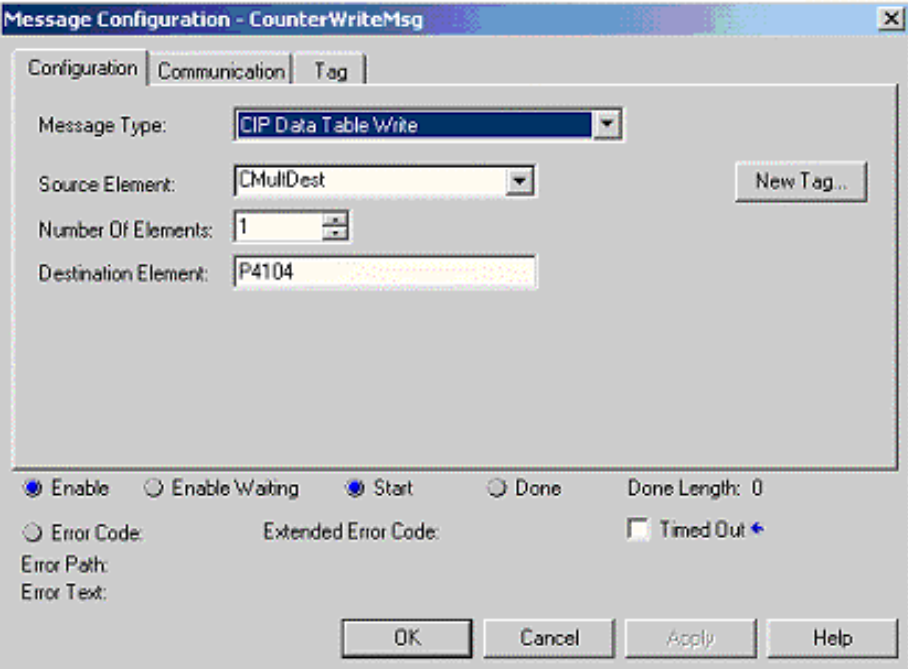
You can write 1-16 elements. Make sure that the data type and size of the source and destination elements are the same.

For more information, see the Rockwell documentation.

Example

In the following illustration, the value of Source Element “CmultDest” is written to parameter P4104 in the ACR9000.

The number of elements is set to 1—ControlLogix writes to only one register.



The screenshot shows the "Message Configuration - CounterWriteMsg" dialog box. It has three tabs: "Configuration", "Communication", and "Tag". The "Configuration" tab is active. The "Message Type" is set to "CIP Data Table Write". The "Source Element" is "CmultDest" and the "Destination Element" is "P4104". The "Number Of Elements" is set to 1. There are radio buttons for "Enable", "Enable Waiting", "Start", and "Done", with "Start" selected. The "Done Length" is 0. There are also radio buttons for "Error Code" and "Extended Error Code", and a checkbox for "Timed Out". At the bottom, there are buttons for "OK", "Cancel", "Apply", and "Help".

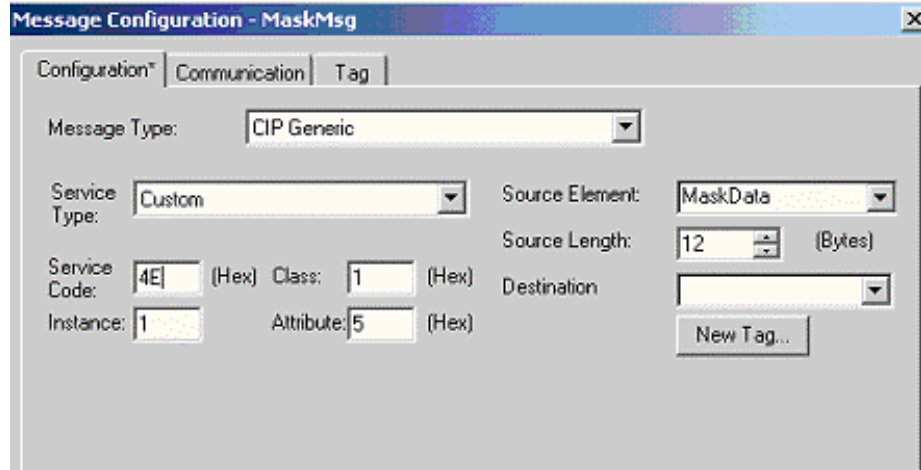
NAND and OR Masks – 0x4E

The 0x4E service code allows you to set and clear individual bits. To do this, send CIP Generic messages to the ACR9000.

To set or clear the appropriate bits in a 32-bit word, use the NANDmask and ORmask with the write P-parameter. The masking works the same way as for I/O.

Example

In the following illustration, the MaskData is defined as follows: the lower 16 bits of P4111 are cleared and then the second bit is set.



The NAND mask clears bits, and the OR mask sets bits. The data is modified as follows: $\text{data} = (\text{data AND nandmask}) \text{ OR ormask}$

[-] MaskData	{...}	{...}	Decimal	DINT[3]
[+] MaskData[0]	4111		Decimal	DINT
[+] MaskData[1]	16#0000_ffff		Hex	DINT
[+] MaskData[2]	16#0000_0002		Hex	DINT

This example sets the second in the parameter P4111

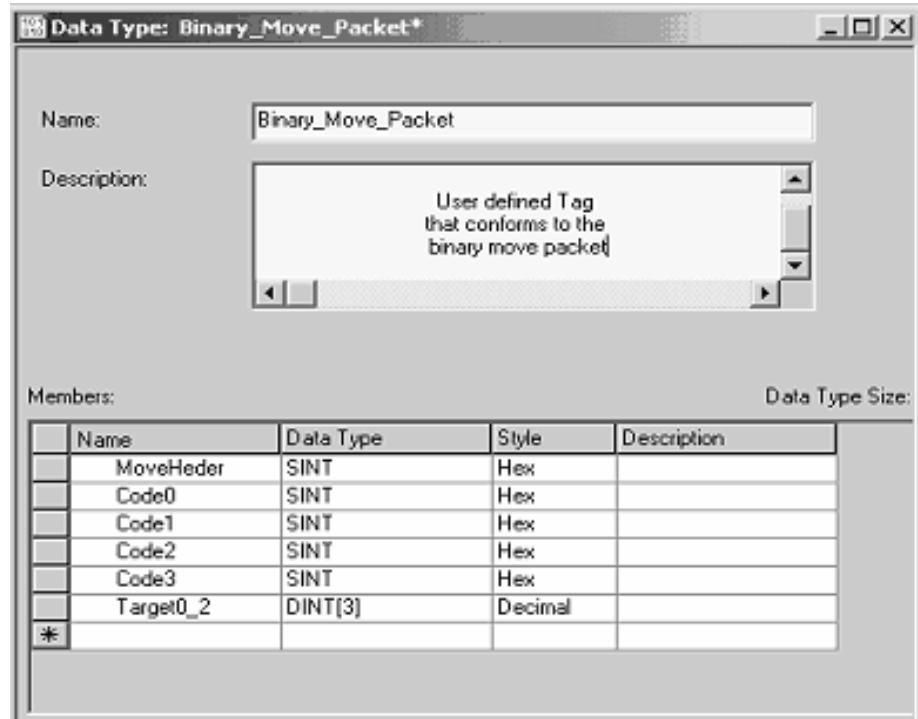
Parameter	NAND Mask	OR Mask
4111	00 00 FF FF	00 00 00 02

Binary Move Block – 0x34

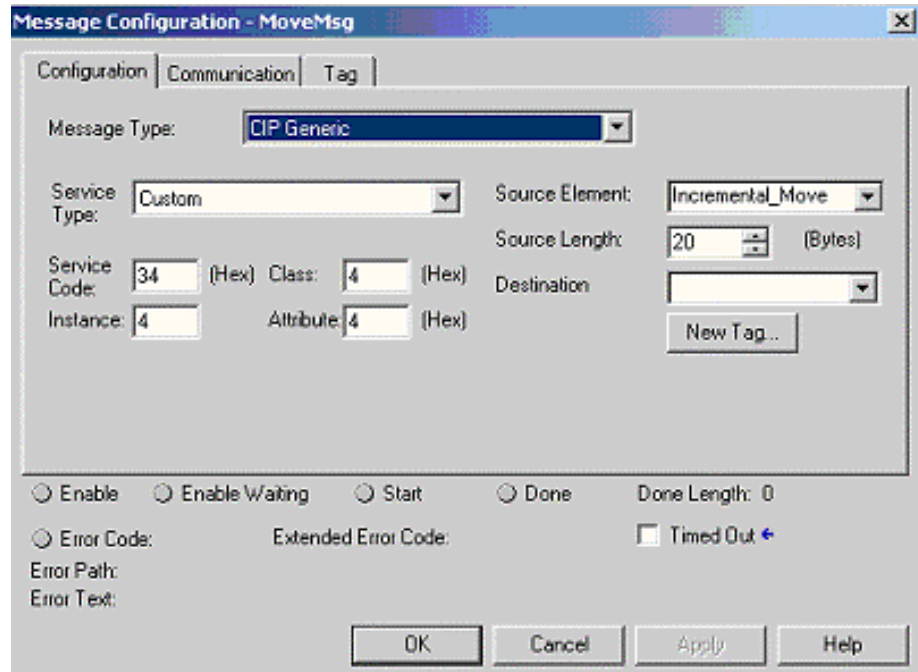
The 0x34 service code lets you send binary moves. To do this, send CIP Generic messages to the ACR9000.

Example

The following illustration shows a user defined tag that sends target positions (DINT) to three axes.



The Incremental_Move (below) is the user-defined tag (above). The Source Length is the size of the Incremental_Move defined as bytes.



Float Read – 0x48

The 0x48 service code lets you access 32-bit register with the data type real. This object can read any P-parameter in ACR9000 with the data type float. You can also use it for multiple-block reads. This is an efficient method to read a contiguous block of P-parameters. You can write 1-16 elements.

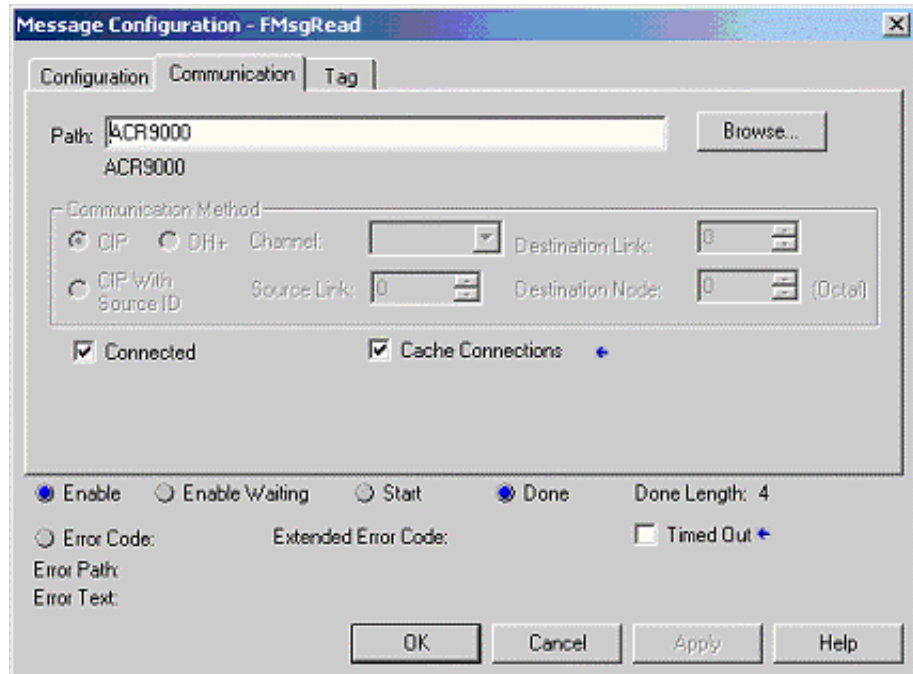
Example

In the following illustration, the value of Source Element (0x2026) P8230.

The screenshot shows the 'Message Configuration - FMsgRead' dialog box with the following settings:

- Message Type: CIP Generic
- Service Type: Custom
- Service Code: 68 (Hex)
- Class: 2026 (Hex)
- Instance: 1
- Attribute: 1 (Hex)
- Source Element: (empty)
- Source Length: 0 (Bytes)
- Destination: FloatR
- Done Length: 4
- Radio buttons: Enable (selected), Enable/Waiting, Start, Done (selected)
- Timed Out:

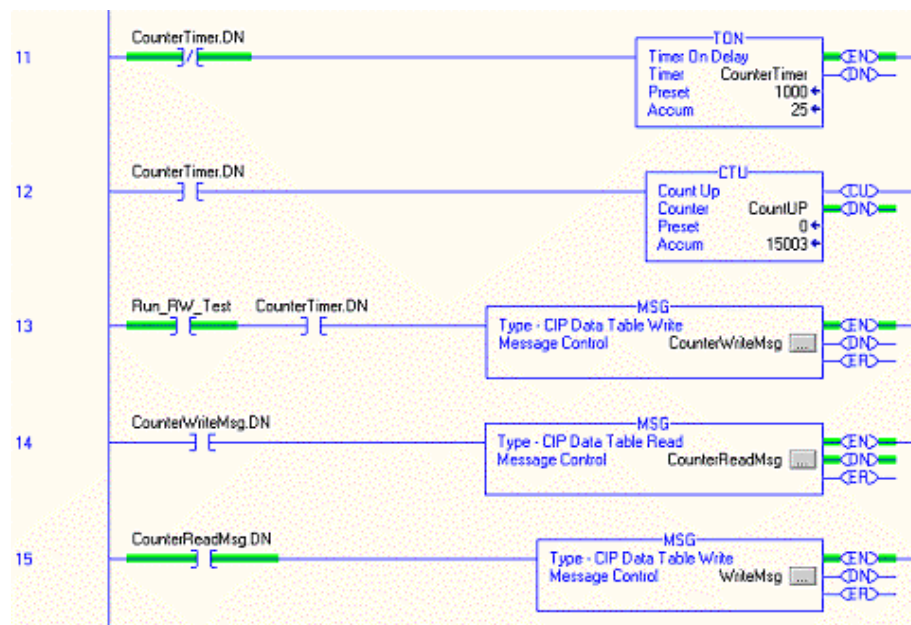
For the purposes of the tutorial, all messages are connected and cached. On the Communication tab, select the **Connected** and **Cache Connections** check boxes.



General Flow of Messages

The PLC diagram (below) illustrates an important aspect of setting up the structure for Class3 Messages. It is recommended that you follow the same sequence for all messages.

- The message must be triggered each time before it is sent to the ACR9000.
- A one second timer counter has been created to trigger the message block.
- CounterTimer.DN triggers the CounterWriteMsg.
- CounterWriteMsg.DN triggers the CounterReadMsg.
- CounterReadMsg.DN will trigger the WriteMsg.



RSLogix I/O Configuration

Adding ACR9000 as an I/O module

1. RSLogix should be in offline mode.
2. Under **I/O Configuration**, right-click the **1756-ENET/B EnetBridge** node and select **Add Module**.
3. Select module type **Generic Ethernet Module**, and then click **OK**.
4. In the **Name** box, enter the module name (for example, ACR9000).
5. Under **Address/Host Name**, click **IP Address**. Then enter the IP address of the ACR9000.
6. Under Connection Parameters, do the following:
 - a. Input: In the **Assembly Instance** box, type 101. Then in the **Size** box, type 80.
 - b. Output: In the **Assembly Instance** box, type 102. Then in the **Size** box, type 2.
 - c. Configuration: In the **Assembly Instance** box, type 3. Then in the **Size** box, type 64.

The Assembly instances are fixed for ACR9000 device. However, you can change the size of the input, output and configuration to meet requirements.

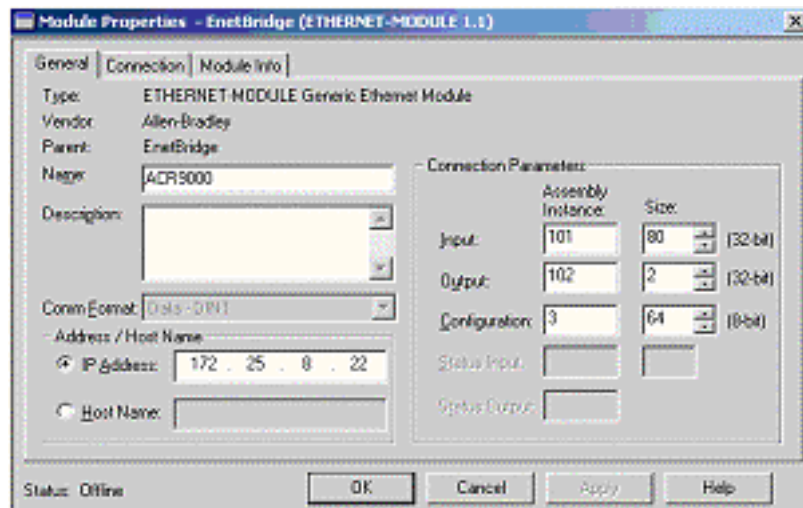
7. Click **OK**. The ACR9000 module should appear in the I/O Configuration view.

Note: The RSLinx adds 3 new global controller Tags.

ACR9000.I

ACR9000.O

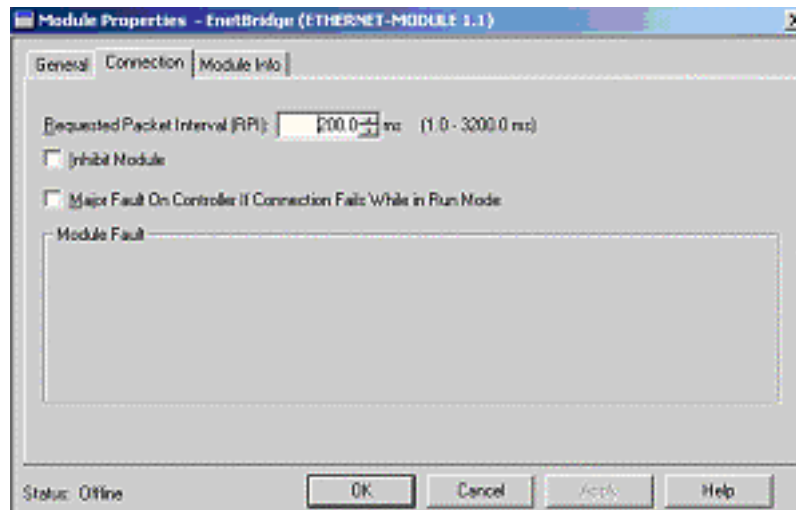
ACR9000.C



Request Packet Interval (RPI)

1. Double-click the ACR9000 module.
2. In the **Module Properties** box, select the **Connection** tab.
3. Set the interval for request packets in the **Requested Packet Interval** box.

Note: If you inhibit the module, ControlLogix closes the Class1 connection with ACR9000.

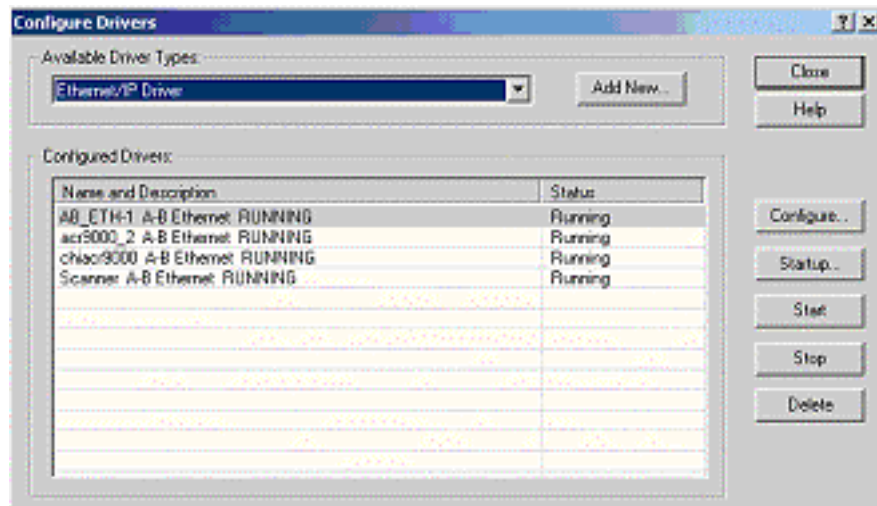


RSLinx Driver configuration

Driver Setup

Before ControlLogix can talk to ACR9000, you must configure the RSLinx drive.

1. Start RSLinx.
2. In the **Communications/Configure Drivers** menu, select **Add a New Driver**.
3. In the **Available Driver Types** list, select **EtherNet/IP Driver**.
4. Type the driver name ACR9000, and then click **OK**.
5. In the next dialog box, enter the IP address of 1756-ENET/B Ethernet bridge— 172.25.8.29



Driver Diagnostics

Having set up the drive, you should see the ACR9000 in the RSWho window. RSLinx identifies the ACR9000 as a motion controller. If a red cross appears on the device, then RSLinx cannot talk to ACR9000.

Note: To update the information, click the **Autobrowse** box.

- ▶ To check the ACR9000 status, right-click the ACR9000 device. Then select **Driver Diagnostic**.

