

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.

© 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



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
Reference Documents.....	2
Technical Support	3
ACR9000	3
Ethernet/IP	3
Chapter 2 I/O Data Connections.....	4
Overview.....	5
Modes of Communication.....	5
Class 1 I/O (UDP)	5
Class 3 (TCP) CIP Messages (connected and unconnected).....	6
Explicit Messaging.....	6
Object Model	7
Multiple Attributes—0x0A	7
Block Read Table—0x4C	7
Block Write Table—0x4D.....	9
NAND and OR Masks—0x4E	11
Binary Move Block—0x34.....	13
Float Read—0x48.....	15
Generic Binary Packets.....	17

CHAPTER ONE

Introduction

IN THIS CHAPTER

• Introduction	2
• Compatible Parker Hannifin Products.....	2
• Assumptions of Technical Experience	2
• Reference Documents	2
• Technical Support	3

Introduction

CIP (Control and Information Protocol) is designed for industrial controls devices, which includes DeviceNet, ControlNet, and EtherNet/IP.

This document describes the implementation of EtherNet/IP for the ACR series motion controllers.

This specification applies to the following:

ACR Series Controllers..... Operating System revision 1.18.15
or greater

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.

Reference Documents

The following documents will prove helpful when implementing EtherNet/IP. We assume you are familiar with the following materials. You can download them at the ODVA website (www.odva.org/).

- *EtherNet/IP Specification—Volume 1: CIP Common Specification*
- *EtherNet/IP Specification—Volume 2: EtherNet/IP*
- *EtherNet/IP Terms*

If you are using a ControlLogix System from Rockwell Automation, consider downloading the following reference documents (www.ab.com/)

- *Communicating with RA Products Using EtherNet/IP Explicit Messaging*
- *Establishing I/O Communications with RA ControlLogix Systems on EtherNet/IP*
- *Logix5000 Data Access Reference Manual*
Publication 1759-RM005A-EN-E (March 2000)

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.

CHAPTER TWO

I/O Data Connections

IN THIS CHAPTER

- Overview 5
- Modes of Communication 5
- Explicit Messaging 6
- Object Model 7

Overview

CIP is an industrial controls protocol that is structurally designed to allow TCP/IP communication between many different industrial devices.

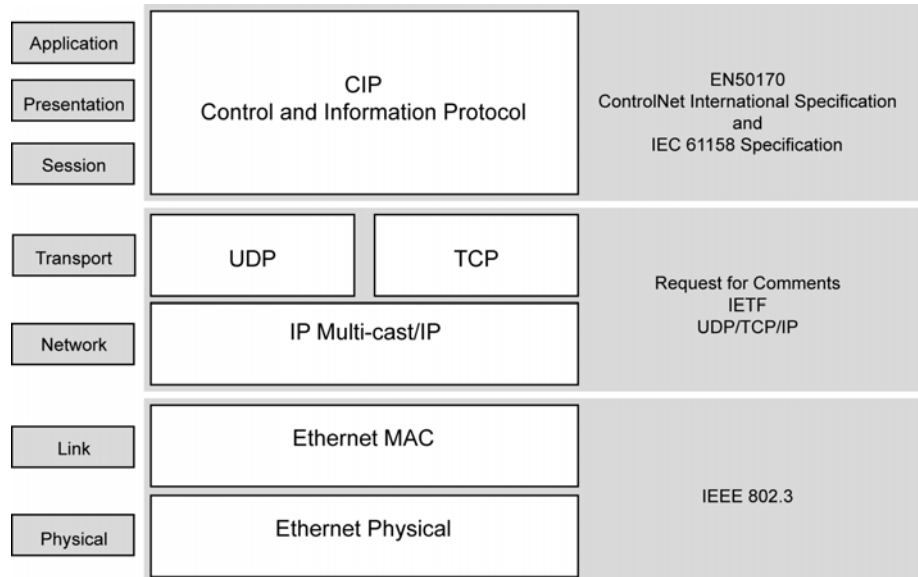
CIP uses a connection-based design that allows a variety of devices to communicate between each other; a CIP connection provides a communication pathway between multiple end-points.

The CIP specification describes the features available: message protocol, object modeling, messaging protocol, communication objects, general object library, device profiles, electronic data sheets, service, and data management.

As CIP uses object modeling to express data, you can find a library of standard objects in the “CIP Common Specification”. In addition, the specification contains a library of device descriptions (or *device profiles*) for common industrial control devices.

Modes of Communication

The EtherNet/IP network is designed to use standard Ethernet and TCP/IP equipment for the industrial environment. The application layer protocol is an open standard—CIP (Control and Information Protocol). CIP is the same protocol used by DeviceNet and ControlNet networks, allowing interoperability between various industrial devices.



Class 1 I/O (UDP)

Implicit messaging is a “Class 1” connection type, providing point-to-point or multicast messaging over a UDP connection. Your application uses implicit messaging for I/O data transfer. Data is sent cyclically based on a user-defined duration.

Implicit Message Cycle

Minimum..... 100 milliseconds

Maximum..... 3 seconds

With Class 1 I/O communication, the adapter (ACR9000) subscribes to the scanner (ControlLogix). 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)

Explicit messaging is a “Class 3” connection type, providing point-to-point, one-time-only messaging over a TCP connection.

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

ControlLogix uses read-table and write-table messages to communicate with ACR9000. In addition, it can transmit other generic CIP objects this connection.

Explicit Messaging

The following table shows the general structure of an explicit message.

Termination Request Packet Data	
Function	Description
bService	Service code
iClass	Target object class
iInstance	Target instance
iAttribute	Attribute parameter
iMember	Optional member Id
iTagSize	Some targets, support named tags. In this case, iTagSize contains the tag string size.
RequestData	Request data.
iDataSize	Data size.
IExplicitMessageTimeout	Determines how fast the request should time out in milliseconds.

Object Model

The following Object Classes are supported in the ACR series motion controllers.

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.

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.

4C Request

In the following example, EtherNet/IP reads P4105 on the ACR9000. EtherNet/IP sends the following data:

Source: 172.25.8.29 (672, 0)
Destination: 172.25.8.23 (672, 0)
Item Count: 2 (672, 16)
Item: Connection-based (688, 16)
Item Length: 4 (704, 16)
Connection Identifier: 0x80020000 (720, 32)
Item: Connected Transport packet (752, 16)
Item Length: 14 (768, 16)
Sequence: 40 (784, 16)
Request/Reply: Request (800, 1)
Service Code: 0x4c (801, 7)
Path Size (Words): 4 (808, 8)

Path: 0x 91 05 50 34 31 30 35 00 (816, 64)

Segment: Symbolic, one byte version (816, 8)

Length: 5 (824, 8)

Symbolic Data: P4105 (832, 40)

Pad Byte: 0x00 (872, 8)

The following physical frame is the generated Block Read Request:

```
00 90 55 00 16 41 00 00 bc 03 6a d8 08 00 45 00
00 62 81 c8 00 00 40 06 90 67 ac 19 08 1d ac 19
08 17 af 13 af 12 a6 8b c5 2e 59 2f 12 24 50 18
11 1c e4 2e 00 00 70 00 22 00 00 05 65 b7 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 00 00 02 80 b1 00
0e 00 28 00 4c 04 91 05 50 34 31 30 35 00 01 00
43 52 43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

4C	Service code Read Table	
04	Path Length in words	
91	Symbolic Tag= string	Path
05	Data Length=5	
50 34 31 30 35	String Data= P4505	
00	Byte Padding	
01	Length= 1	
00	Byte Padding	

4C Response

In the following example, EtherNet/IP reads P4105 on the ACR9000. EtherNet/IP sends the following data:

Source: 172.25.8.23 (672, 0)

Destination: 172.25.8.29 (672, 0)

Item Count: 2 (672, 16)

Item: Connection-based (688, 16)

Item Length: 4 (704, 16)

Connection Identifier: 0x65007404 (720, 32)

Item: Connected Transport packet (752, 16)

Item Length: 12 (768, 16)

Sequence: 40 (784, 16)

Request/Reply: Reply (800, 1)

Service Code: 0x4c (801, 7)

General Status: Success (816, 8)

Additional Status Size (Words): 0 (824, 8)

The following physical frame is the generated Block Read response:

```
00 00 bc 03 6a d8 00 90 55 00 16 41 08 00 45 00
00 60 00 2d 00 00 40 06 12 05 ac 19 08 17 ac 19
08 1d af 12 af 13 59 2f 12 24 a6 8b c5 68 50 10
3e 80 e3 a8 00 00 70 00 20 00 00 05 65 b7 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 04 74 00 65 b1 00
0c 00 28 00 cc 00 00 00 c4 00 d9 05 00 00 43 52
43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

cc 00	Service code, Read Table Response
00	Standard Error Code = Success
00 c4	Data type Long
00	Padding
d9 05 00 00	Long Data Read = 1497

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.

4D Request

In the following example, EtherNet/IP writes the value 1862 (0x746) to P4203 on the ACR9000. EtherNet/IP sends the following data:

```
Source: 172.25.8.29 (672, 0)
Destination: 172.25.8.23 (672, 0)
Item Count: 2 (672, 16)
Item: Connection-based (688, 16)
    Item Length: 4 (704, 16)
    Connection Identifier: 0x80020000 (720, 32)
Item: Connected Transport packet (752, 16)
    Item Length: 20 (768, 16)
    Sequence: 775 (784, 16)
    Request/Reply: Request (800, 1)
    Service Code: 0x4d (801, 7)
    Path Size (Words): 4 (808, 8)
    Path: 0x 91 05 50 34 32 30 33 00 (816, 64)
        Segment: Symbolic, one byte version (816, 8)
            Length: 5 (824, 8)
```

Symbolic Data: P4203 (832, 40)

Pad Byte: 0x00 (872, 8)

The following physical frame is the generated Block Write Request:

```
00 90 55 00 16 41 00 00 bc 03 6a d8 08 00 45 00
00 68 88 71 00 00 40 06 89 b8 ac 19 08 1d ac 19
08 17 af 13 af 12 cf 0d 3b 7d 95 83 7c 7b 50 18
11 1c 24 55 00 00 70 00 28 00 00 00 eb 08 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 00 00 02 80 b1 00
14 00 07 03 4d 04 91 05 50 34 32 30 33 00 c4 00
01 00 46 07 00 00 43 52 43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

4d	Service code	
04	Path Length in words	
91	Symbolic Tag, string	Path
05	Data Length in bytes	
50 34 32 30 33	String Data = P4203	
00	Padding	
c4 00	Data Type= long	
01	Length= 1	
00	Padding	
46 07 00 00	Data = parameter data being written = 1862	

4D Response

In the following example, EtherNet/IP writes the value 1862 (0x746) to P4203 on the ACR9000. EtherNet/IP sends the following data:

Source: 172.25.8.23

Destination: 172.25.8.29

Item Count: 2

Item: Connection-based

Item Length: 4

Connection Identifier: 0x49007204

Item: Connected Transport packet

Item Length: 6

Sequence: 775

Request/Reply: Reply

Service Code: 0x4d

General Status: Success

Additional Status Size (Words): 0

The following physical frame is the generated Block Write Response:

```
00 00 bc 03 6a d8 00 90 55 00 16 41 08 00 45 00
00 5a 7c da 00 00 40 06 95 5d ac 19 08 17 ac 19
08 1d af 12 af 13 95 83 7c 7b cf 0d 3b bd 50 10
3e 80 e2 01 00 00 70 00 1a 00 00 00 eb 08 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 04 72 00 49 b1 00
06 00 07 03 cd 00 00 00 43 52 43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

cd 00	Service code, Write Table Response
00	Standard Error Code = Success
00	Padding

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.

This mask command consists of three long words: the P-parameter, the NANDmask and the Ormask. The NAND mask clears bits and the OR mask sets bits. The P-parameter is modified as follows:

$$P = (P \text{ AND } \text{nandmask}) \text{ OR } \text{ormask}$$

4E Request

In the following example, the value of P4111 is modified as follows :

$$P4111 = (P4111 \& 0x223D) | 0x10D6$$

Where

$$\text{NANDMASK} = 8765 = 0x223D \text{ (bits being cleared)}$$

$$\text{ORMASK} = 4310 = 0x10D6 \text{ (bits being set)}$$

Following packet is by the client to the ACR9000:

Source: 172.25.8.29 (672, 0)

Destination: 172.25.8.23 (672, 0)

Item Count: 2 (672, 16)

Item: Connection-based (688, 16)

Item Length: 4 (704, 16)

Connection Identifier: 0x80020000 (720, 32)

Item: Connected Transport packet (752, 16)

Item Length: 22 (768, 16)

Sequence: 117 (784, 16)

Request/Reply: Request (800, 1)

The following physical frame is the generated NAND/OR Mask Request:

```
00 90 55 00 16 41 00 00 bc 03 6a d8 08 00 45 00
00 6a 4d 53 00 00 40 06 c4 d4 ac 19 08 1d ac 19
08 17 af 13 af 12 ac 85 40 35 20 6b b7 21 50 18
11 1c 99 ca 00 00 70 00 2a 00 00 00 15 ff 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 00 00 02 80 b1 00
16 00 75 00 4e 03 20 01 24 01 30 05 0f 10 00 00
3d 22 00 00 d6 10 00 00 43 52 43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

4e	Service code Mask for bit set and clear	
03	Path Length in words	
20 01	Class = 1	Path
24 01	Instance = 1	
30 05	Attribute =5	
0f 10 00 00	Parameter Number = 4111	
3d 22 00 00	NAND MASK = 8765	
d6 10 00 00	OR MASK = 4310	

4E Response

A success response from ACR9000

Source: 172.25.8.23 (672, 0)

Destination: 172.25.8.29 (672, 0)

Item Count: 2 (672, 16)

Item: Connection-based (688, 16)

Item Length: 4 (704, 16)

Connection Identifier: 0x9b007504 (720, 32)

Item: Connected Transport packet (752, 16)

Item Length: 6 (768, 16)

Sequence: 117 (784, 16)

Request/Reply: Reply (800, 1)

Service Code: 0x4e (801, 7)

General Status: Success (816, 8)

Additional Status Size (Words): 0 (824, 8)

The following physical frame is the generated NAND/OR Mask Response:

```
00 00 bc 03 6a d8 00 90 55 00 16 41 08 00 45 00
00 5a 00 91 00 00 40 06 11 a7 ac 19 08 17 ac 19
08 1d af 12 af 13 20 6b b7 21 ac 85 40 77 50 10
3e 80 9f f9 00 00 70 00 1a 00 00 00 15 ff 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 04 75 00 9b b1 00
06 00 75 00 ce 00 00 00 43 52 43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

ce 00	Service code, MASK Response
00	Standard Error Code = Success
00	Padding

Binary Move Block – 0x34

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

34 Request

In the following example, EtherNet/IP sends the following moves to the ACR9000.

- binary move header = 04
- code 0 = 0F
- code 1 = 00
- code 2 = 07
- code 3 = 01
- axis0 = 100
- axis1 = 200
- axis3 = 300

EtherNet/IP sends the following data:

Source: 172.25.8.29 (672, 0)

Destination: 172.25.8.23 (672, 0)

Item Count: 2 (672, 16)

Item: Connection-based (688, 16)

Item Length: 4 (704, 16)

Connection Identifier: 0x80020000 (720, 32)

Item: Connected Transport packet (752, 16)

Item Length: 30 (768, 16)

Sequence: 4 (784, 16)

Request/Reply: Request (800, 1)

The following physical frame is the generated Binary Move Request:

```

00 90 55 00 16 41 00 00 bc 03 6a d8 08 00 45 00
00 72 a1 7e 00 00 40 06 70 a1 ac 19 08 1d ac 19
08 17 af 13 af 12 f2 b5 0a 82 59 29 c5 e3 50 18
11 1c 7a 56 00 00 70 00 32 00 00 00 1d 99 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 00 00 02 80 b1 00
1e 00 04 00 34 03 20 01 24 01 30 05 04 0f 00 07
01 00 00 00 64 00 00 00 c8 00 00 00 2c 01 00 00

```

The service for ACR9000 is underlined (above), and breaks down as follows:

34	Service code move packet	
03	Path Length in words	
20 01	Class = 1	Path
24 01	Instance = 1	
30 05	Attribute =5	
04	Binary move header	
0f	Header code 0	
00	Header code 1	
07	Header code 2	
01	Header code 3	
00 00 00	Byte Padding	
64 00 00 00	Axis0 Target Point	
c8 00 00 00	Axis1 Target Point	
2c 01 00 00	Axis2 Target Point	

34 Response

EtherNet/IP sends the following data:

Source: 172.25.8.23 (672, 0)

Destination: 172.25.8.29 (672, 0)

Item Count: 2 (672, 16)

Item: Connection-based (688, 16)

Item Length: 4 (704, 16)

Connection Identifier: 0x3800a604 (720, 32)

Item: Connected Transport packet (752, 16)

Item Length: 6 (768, 16)

Sequence: 4 (784, 16)

Request/Reply: Reply (800, 1)

Service Code: 0x34 (801, 7)

General Status: Success (816, 8)

Additional Status Size (Words): 0 (824, 8)

The following physical frame is the generated Binary Move Response:

```
00 00 bc 03 6a d8 00 90 55 00 16 41 08 00 45 00
00 5a 00 1e 00 00 40 06 12 1a ac 19 08 17 ac 19
08 1d af 12 af 13 59 29 c5 e3 f2 b5 0a cc 50 10
3e 80 cb 8c 00 00 70 00 1a 00 00 00 1d 99 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 04 a6 00 38 b1 00
06 00 04 00 b4 00 00 00 43 52 43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

b4 00	Service code move Response
00	Standard Error Code = Success
00	Padding

Float Read—0x48

The 0x48 service codes 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.

48 Request

In the following example, EtherNet/IP reads P6152 on the ACR9000. EtherNet/IP sends the following data:

Source: 172.25.8.61
Destination: 172.25.8.23
Item Count: 2
Item: Connection-based
Item Length: 4
Connection Identifier: 0x00000002
Item: Connected Transport packet
Item Length: 14
Sequence: 3
Request/Reply: Request
Service Code: 0x48
Path Size (Words): 4
Path: 0x 91 05 50 36 31 35 32 00
Segment: Symbolic, one byte version
Length: 5
Symbolic Data: P6152
Pad Byte: 0x00

Data:

Hex: 0x 01 00

ASCII:

The following physical frame is the generated Float Read Request:

```
00 90 55 00 16 41 00 20 ab 1b 02 45 08 00 45 00
00 62 01 c5 00 00 40 06 10 4b ac 19 08 3d ac 19
08 17 5c 5e af 12 02 3c b9 93 15 16 58 fa 50 18
16 d0 6e 55 00 00 70 00 22 00 00 45 01 d9 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 02 00 00 00 b1 00
0e 00 03 00 48 04 91 05 50 36 31 35 32 00 01 00
43 52 43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

48	Service code Read Float Parameter	
04	Path Length in words	
91	Symbolic Tag	Path
05	Tag Length in bytes	
50 36 31 35 32	Float Parameter number p6152	
00	Padding	
01	Length	
00	Padding	

48 Response

In the following example, EtherNet/IP reads P6152 on the ACR9000. EtherNet/IP sends the following data:

Source: 172.25.8.23

Destination: 172.25.8.61

Item Count: 2

Item: Connection-based

Item Length: 4

Connection Identifier: 0x00000001

Item: Connected Transport packet

Item Length: 10

Sequence: 3

Request/Reply: Reply

Service Code: 0x48

General Status: Success

Additional Status Size (Words): 0

Data:

Hex: 0x 00 00 00 00

ASCII:

The following physical frame is the generated Float Read Response:

```
00 20 ab 1b 02 45 00 90 55 00 16 41 08 00 45 00
00 5e 23 a2 00 00 40 06 ee 71 ac 19 08 17 ac 19
08 3d af 12 5c 5e 15 16 58 fa 02 3c b9 cd 50 10
3e 80 14 ec 00 00 70 00 1e 00 00 45 01 d9 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 02 00 a1 00 04 00 01 00 00 00 b1 00
0a 00 03 00 c8 00 00 00 00 08 42 45 43 52 43 21
```

The service for ACR9000 is underlined (above), and breaks down as follows:

c8 00	Service code, Read Table Response
00	Standard Error Code = Success
00	Padding
00 80 42 45	Float Data Read

Generic Binary Packets

You can use the standard binary packets, as defined in the ACR User Guide, to communicate. The following standard, ACR binary packets can be encapsulated in the generic Ethernet/IP service code:

- Write
- Set Long
- Set Float
- Mask
- Move
- Set/Clear
- FOV/ROV

For more information about the binary host interface for ACR series controllers, see the ACR Series User's Guide.