Lenze
AC Tech

PositionServo

Modbus Communication Module

Communications Interface Reference Guide

P94MOD01B

About These Instructions

This documentation applies to Modbus RTU and Modbus TCP/IP communications for the PositionServo drive and should be used in conjunction with the PositionServo User Manual (S94P01) that shipped with the drive. These documents should be read carefully as they contain important technical data and describe the installation and operation of the drive and this option module.

This document printed in the United States of America

Table of Contents

1. Safety Information

1.1 Warnings, Cautions & Notes

General

Some parts of Lenze controllers (frequency inverters, servo inverters, DC controllers) can be live, with the potential to cause attached motors to move or rotate. Some surfaces can be hot.

Non-authorized removal of the required cover, inappropriate use, and incorrect installation or operation creates the risk of severe injury to personnel or damage to equipment.

All operations concerning transport, installation, and commissioning as well as maintenance must be carried out by qualified, skilled personnel (IEC 364 and CENELEC HD 384 or DIN VDE 0100 and IEC report 664 or DIN VDE0110 and national regulations for the prevention of accidents must be observed).

According to this basic safety information, qualified skilled personnel are persons who are familiar with the installation, assembly, commissioning, and operation of the product and who have the qualifications necessary for their occupation.

Application as directed

Drive controllers are components which are designed for installation in electrical systems or machinery. They are not to be used as appliances. They are intended exclusively for professional and commercial purposes according to EN 61000-3-2. The documentation includes information on compliance with the EN 61000-3-2.

When installing the drive controllers in machines, commissioning (i.e. the starting of operation as directed) is prohibited until it is proven that the machine complies with the regulations of the EC Directive 98/37/EC (Machinery Directive); EN 60204 must be observed.

Commissioning (i.e. starting of operation as directed) is only allowed when there is compliance with the EMC Directive (89/336/EEC).

The drive controllers meet the requirements of the Low Voltage Directive 73/23/EEC. The harmonised standards of the series EN 50178/DIN VDE 0160 apply to the controllers.

The availability of controllers is restricted according to EN 61800-3. These products can cause radio interference in residential areas.

Installation

Ensure proper handling and avoid excessive mechanical stress. Do not bend any components and do not change any insulation distances during transport or handling. Do not touch any electronic components and contacts.

Controllers contain electrostatically sensitive components, which can easily be damaged by inappropriate handling. Do not damage or destroy any electrical components since this might endanger your health!

Electrical connection

When working on live drive controllers, applicable national regulations for the prevention of accidents (e.g. VBG 4) must be observed.

The electrical installation must be carried out according to the appropriate regulations (e.g. cable cross-sections, fuses, PE connection). Additional information can be obtained from the national regulation documentation. In the United States, electrical installation is regulated by the National Electric Code (nec) and NFPA 70 along with state and local regulations.

The documentation contains information about installation in compliance with EMC (shielding, grounding, filters and cables). These notes must also be observed for CE-marked controllers.

The manufacturer of the system or machine is responsible for compliance with the required limit values demanded by EMC legislation.

Operation

Systems including controllers must be equipped with additional monitoring and protection devices according to the corresponding standards (e.g. technical equipment, regulations for prevention of accidents, etc.). You are allowed to adapt the controller to your application as described in the documentation.

	**DANGER**! • After the controller has been disconnected from the supply voltage, live components and power connection must not be touched immediately, since capacitors could be charged. Wait at least 60 seconds before servic-ing the drive Please observe the corresponding notes on the controller. • Do not continuously cycle input power to the controller more than once every three minutes. • Please close all protective covers and doors during operation.
	**WARNING!** Network control permits automatic operation of the inverter drive. The system design must incorporate adequate protection to prevent personnel from accessing moving equipment while power is applied to the drive system.

Pictographs used in these instructions:

Pictograph	Signal Word	Meaning	Consequence if Ignored
	**DANGER!**	Warning of Hazardous Electrical Voltage.	Reference to an imminent danger that may result in death or serious personal injury if the corresponding measures are not taken.
	**WARNING!**	Impending or possible danger to personnel	Death or injury
	**STOP!**	Possible damage to equipment	Damage to drive system or its surroundings
	**NOTE**	Useful tip: If note is observed, it will make using the drive easier	

1.2 Reference Documents

- Modbus Application Protocol Specification V1.1

  To view this specification, visit: http://www.modbus.org/tech.php

  or: http://www.modbus-ida.org/specs.php
- Modbus Over Serial Line Specification & Implementation Guide V1.0.

  Refer to: http://www.modbus.org/
- PositionServo Programming Manual: PM94P01

  Refer to: http://www.actech.com

	NOTE: The complete list of variables can be found in Appendix A of the PositionServo Programming Manual (PM94P01).

2. Introduction

This reference guide assumes that the reader has a working knowledge of Modbus RTU or Modbus TCP/IP Protocol and familiarity with the programming and operation of motion control equipment. This guide is intended as a reference only. The PositionServo drive supports both Modbus TCP/IP and Modbus RTU communication protocols. Modbus TCP/IP communications can be established via the PositionServo drive's Ethernet port. When using the RS485 option module (P/N E94ZARS41), Modbus RTU is the standard communication protocol. Refer to Section 3 for installation of the RS485 option module.

2.1 Modbus Overview

Modbus is an internationally accepted asynchronous serial protocol designed for commercial and industrial automation applications. The Modbus RTU architecture is based upon a Master-Slave orientation. In Modbus RTU, the PositionServo drive always acts as the slave, responding to reads and writes from the Master. In Modbus TCP/IP, the PositionServo drive can act as the Master or a slave. While the Modbus RTU protocol does not specify the physical layer, the E94ZARS41 module uses a 2-wire RS-485 physical interface which is quite common and well suited for the industrial environment. The E94ZARS41 module provides both galvanic and optical isolation of the physical interface.

2.2 Ethernet TCP/IP Modbus Configuration

The PositionServo Drive is a standard ModbusTCP slave device and its IP address, subnet mask and default gateway can be configured using MotionView software. It supports up to two (2) simultaneous connections on the standard Modbus TCP port 502. On open connections with no activity for more then 75 seconds, the PositionServo Drive sends a TCP keep-alive message every 75 seconds to check the connection status.

2.3 Modbus RTU Implementation Specification

Typical communications between master and slave would be:

- Write commands from Master
  - Run command
  - Reference values
  - Modification of Drive operating parameters
- Read requests from Master
  - Reporting of drive status
  - Fault status (and fault history)

The PositionServo drive most nearly conforms to the Modicon® Micro 84 in capabilities. This may be of importance when configuring networks for DDE Servers.The following table identifies the Modbus serial communication specifications. If the specification is fixed (non-adjustable) the value is shown under “Range”, if the specification is selectable, the table identifies the Parameter and available range of selections.

Table 1: ModBus Communications Module Parameters

Description	Type	Range
Baud Rate	Selectable	19200, 38400, 57600, 115200
Data Bits	Fixed	8
Parity/Stop Bits	Selectable	Even/1, Odd/1, None1, None2
Network Address	Selectable	1 - 247

The following communication parameters are available:

RS485 configuration – If the value of this parameter is ‘Modbus slave’, the modbus slave protocol is enabled on the RS485 port. If the value is ‘Normal’, the RS485 uses PPP (Point-to-Point Protocol). In Modbus ‘Slave’ mode the drive is configured to receive commands from the Modbus Master.

Modbus Baud Rate – the RS485 baud rate in Modbus mode. The default configuration uses 2 stop bits and no parity. Stop and parity bits are configurable from the same RS485 configuration menu.

Modbus Reply Delay – The delay introduced after receiving a Modbus request and before sending a reply. Note that this delay will always be >= 3.5 characters as required by the Modbus specification. Some Modbus master devices are slow and an increase of the ‘Modbus reply delay’ value may be required to successfully work with these devices.

Each slave device in the Modbus network must have its own unique network address. The ‘Addr’ submenu on the drive display and the front panel buttons can be used to set the Modbus network address. The Modbus Master has no address.

The PositionServo is a Modbus slave device in a single master, multiple slave network. Sometimes slave devices are named servers because they wait for requests. Therefore the PositionServo drive can be one of many slaves communicating to the Modbus master. Most terminals are configured as masters with a Modbus generic driver.

The PositionServo drive uses the required delay of 3.5 characters time between packets to determine the end of a packet.

The following Modbus functions are supported:

1. 03 (0x03) Read Multiple Holding Registers
2. 16 (0x10) Write Multiple Holding Registers

Note that all parameters in the PositionServo drive are 32 bit values and they cannot be accessed by Write Single Register functions or Read Holding Register functions with a length of 1 (one 16 bit register). The user has to use two 16 bit registers to read/write one 32 bit register and for this purpose should reserve 2 consecutive addresses per 32 bit register.

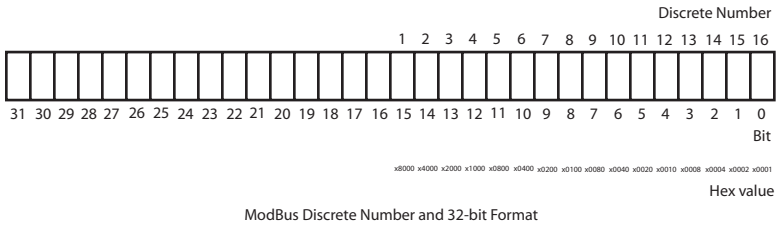

Figure 1: 32 bit Pack Format Discrete Data

The user doesn't have any discrete functions but can use holding registers with DWORD (integer 32 bit) access to write discrete values.

When the PositionServo drive receives a Modbus message, the requested function is executed and the drive can reply with a 'Normal' message, an 'Exception' Message or a 'No' message. These messages are described in the Modbus Application Protocol Specification V1.1.

The PositionServo drive also supports a Modbus broadcast address. In this case the PositionServo drive receives a correct Modbus request and executes it without sending any reply back.

The DWORD (double word) and FLOAT numbers are sent with the LOW WORD FIRST convention by the PositionServo drive.

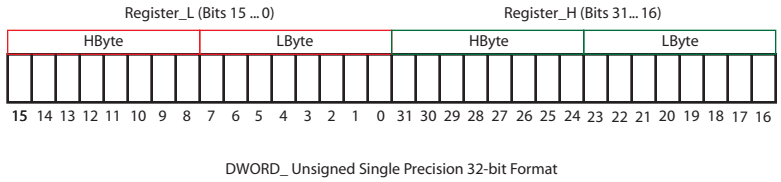

Figure 2: DWORD Format

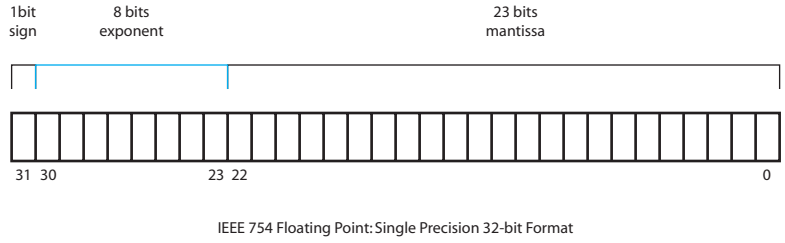

Figure 3: FLOAT Format

Note: Some terminals may have to be configured appropriately in order to accept such a WORD order.

3. Installation

This section is only applicable to Modbus RTU communication with the RS485 option module, E94ZARS41.

3.1 Mechanical Installation

Install the RS485 Communications Module as illustrated in Figure 4.

Disconnect power from drive and wait 3 minutes.

Remove the two COMM module screws that secure Option Bay 1.

With a flat head screwdriver, pry up the Option Bay 1 cover plate.

Install the RS485 COMM Module (E94ZARS41) in Option Bay 1.

Secure with two COMM module screws (max torque: 0.3Nm/3lb-in).

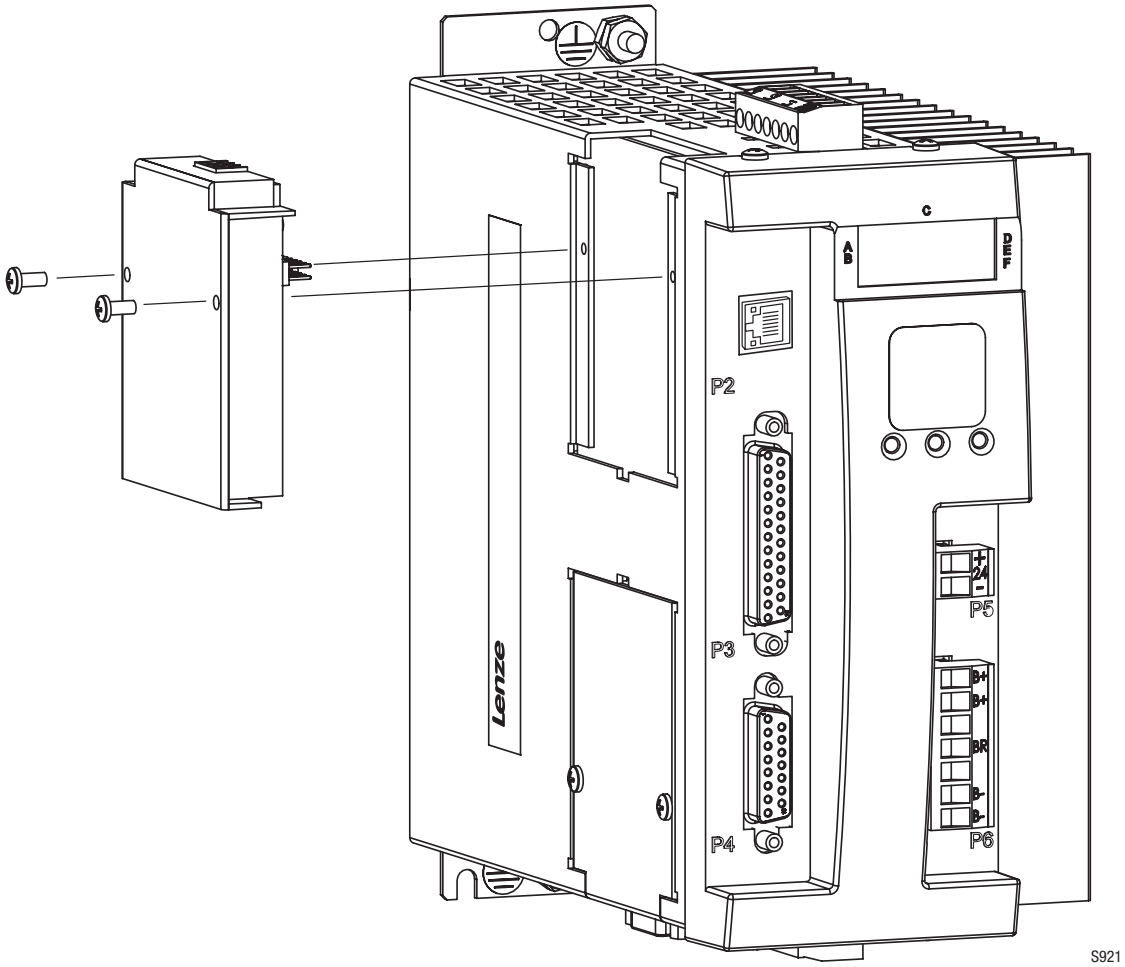

Figure 4: Installation of RS485 Communications Module

3.2 Electrical Installation

Table 2 and Figure 5 illustrate the pinout of the PositionServo RS485 Option Module E94ZARS41. The 3-pin connector provides 2-wire plus isolated ground connection to the network.

Table 2: RS485 Interface Pin Designation

Terminal	Name	Description
1	ICOM	Isolated Common
2	TxB (+)	Transmit B (+)
3	TxA (-)	Transmit A (-)

Figure 5: RS485 Interface Pin Designation

Connections and Shielding

Figure 6 illustrates the connection of the cables for a PositionServo drive in a Modbus master/slave network. A 120ohm (1%) termination is necessary between the high and low terminals of the first and last slaves in the network. Note that while ModBus serial protocol supports up to 247 slave addresses, the maximum number of slave nodes for a network is dictated by the Modbus Master

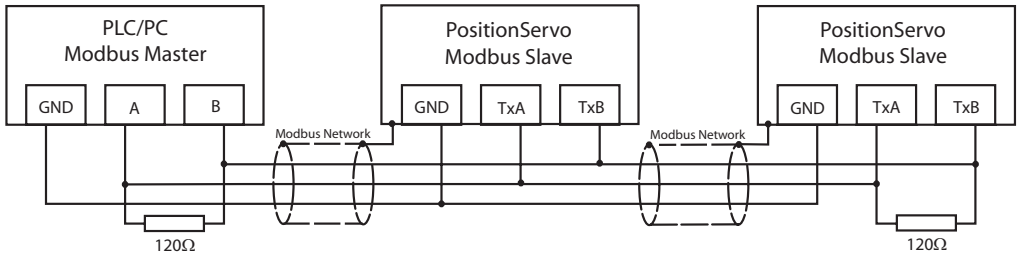

Figure 6: 120Ω (1%) Termination in RS485 Network

4 Modbus Protocol Details

The Modbus protocol defines a simple protocol data unit **(PDU)** independent of the underlying communication layers. There are some additional application data unit **(ADU)** fields introduced by the network layer

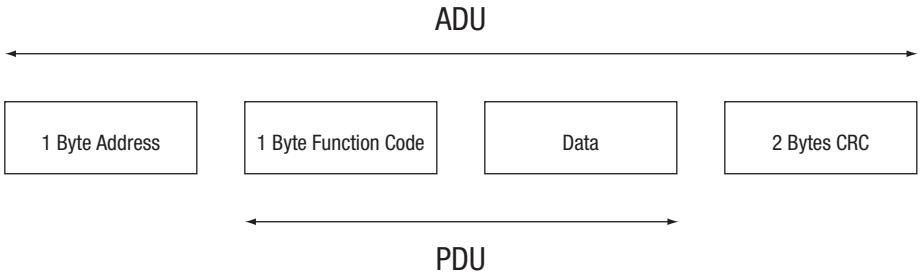

Figure 7: RS485 Network Modbus Frame

The Master that initiates a Modbus transaction builds the Modbus application data unit. The function indicates to the server what kind of action to perform. The Modbus application protocol establishes the format of a request initiated by a client.

The function code field of a MODBUS data unit is coded in one byte. Valid codes are in the range of 1 ... 255 decimal (128 – 255 reserved for exception responses). When a message is sent from a Master to a Server device the function code field tells the server what kind of action to perform. Sub-function codes are added to some function codes to define multiple actions.

The data field of messages sent from a master to server devices contains additional information that the server uses to take the action defined by the function code. This can include items like discrete and register addresses, the quantity of the items to be handled, and the count of the actual data bytes in the field.

The data field may be nonexistent (of zero length) in certain kinds of requests. In this case the server does not require any additional information. The function code alone specifies the action. If no error occurs related to the Modbus function requested in a properly received Modbus ADU, the data field of a response from a server to a master contains the data requested. If an error related to the Modbus function requested occurs, the field contains an exception code that the server application can use to determine the next action to be taken.

For example a master can read the ON / OFF states of a group of discrete outputs or inputs or it can read/write the data contents of a group of registers.

When the server responds to the master, it uses the function code field to indicate either a normal (error-free) response or that some kind of error occurred (called an exception response). For a normal response, the server simply echoes the original function code.

4.1 Data Transmission

The PositionServo drive uses the RTU (Remote Terminal Unit) transmission mode of the Modbus Protocol and operates as a Slave device on the network. All devices communicating with the drive(s) must be a Modbus Master.

The default baud rate for the PositionServo drive is 19200. The default data format is 1 start bit, 8 data bits, no parity and two stop bits as illustrated in Figure 8.

DATA										
Start Bit	1	2	3	4	5	6	7	8	Stop bit	Stop bit

Figure 8: Default Data Format

4.2 Register Numbering

Modbus 4X register numbers are always **one greater** than the actual drive register numbers. For example: drive register #24 would correspond to Modbus 4X register #25.

4.3 Supported Modbus Function Codes

The function codes supported by the drive are:

1. 03 (0x03) Read Multiple 4X Holding Registers
2. 16 (0x10) Write Multiple 4X Holding Registers

5 Drive Memory Access

The PositionServo drive uses one type of register, (4X Holding Registers) when communicating via Modbus. Either the RAM copy or the EPM (non-volatile copy) and RAM copy of each register can be accessed at one time. Each register can be accessed as a 32 bit integer (called DWORD throughout this document) or as a Float value.

5.1 Mapping from Drive Variable to the Register Address

The memory address ranges are divided into six ranges according to the variable type as shown in Table 3.

Table 3: Memory Address Ranges

Variable Type	Memory Range
RAM Integer	0 - 511
RAM Float	512 - 1023
EPM Integer	1024 - 1535
EPM Float	1556 - 2047
RAM 16-bit Integer	2068 - 2303
EPM 16-bit Integer	2304 - 2560

Table 4: Hex to Decimal Number Reference

Hex	0x00	0x1FF	0x200	0x3FF	0x400	0x5FF	0x600	0x7FF	0x800	0x8FF	0x900	0xA00
Decimal	0	511	512	1023	1024	1535	1536	2047	2048	2303	2304	2560

The register address of a drive variable can be calculated by one of the following methods:

NOTE: All values in decimal notation

To access the <variable index> as a RAM-integer, use the following formula to calculate this register address (maximum address allowed is 511):

<register address> = 0 + 2 * <variable index> + 1;

To access the <variable index> as a RAM-float, use the following formula to calculate this register address (maximum address allowed is 1023):

<register address> = 512 + 2 * <variable index> + 1;

To access the<variable index> as a EPM-integer, use the following formula to calculate this register address (maximum address allowed is 1535):

<register address> = 1024 + 2 * <variable index> + 1;

To access the <variable index> as EPM-float, use the following formula to calculate this register address (maximum address allowed is 2047):

<register address> = 1536 + 2 * <variable index> + 1;

Two special methods are created for those terminals that can omly handle 16-bit registers:

To access the <variable index> as a RAM- 16 bit integer register (the RAM copy of a variable that is represented as a 16 bit integer) use the following formula to calculate this register address (maximum address allowed is 2303):

<register address> = 2048 + <variable index> + 1;

For these terminals the values are represented only as integers. The variable index is not multiplied by 2 because one variable is mapped to one register only. If the variable, which is represented as a 32 bit value internally, is out of range (lower than minimum or higher than maximum value for 16 bit integers), then the return value is truncated to the closest value supported by the 16 bit signed number. The access to a variable using this register address range will only read/write the RAM copy of a variable.

To access the <variable index> as an EPM -16 bit signed integer register (the EPM copy of a variable that is represented as a 16 bit integer) use the following formula to calculate this register address (maximum address allowed is 2560):

<register address> = 2304 + <variable index> + 1;

For these terminals the values are represented only as integers. The variable index is not multiplied by 2 because one variable is mapped to one register only. If the variable, which is represented as a 32 bit value internally, is out of range (lower than minimum or higher than maximum value for 16 bit integers), then the return value is truncated to the closest value supported by the 16 bit register. The access to a variable using this register address range will read only the RAM copy of a variable and write both the RAM and EPM copies of a variable.

Register and variable addresses and their mapping based on the type of access are illustrated in Figure 9.

5.2 Register Reading

Use the function code “03 (0x03) Read 4X Holding Registers” to read an adjoining block of holding registers in a remote device. The request PDU (protocol data unit) specifies the starting register address and the number of registers. PDU registers are addressed beginning with 0 (i.e. Register #1-16 would be numbered 0-15). The response is packed two bytes to a register with the binary contents right justified in each byte.

5.3 Register Writing

No discrete register access is provided for PositionServo Drive. Use the “16 (0x10) Write Multiple Registers” function to write binary values. This requires the user programming to pack bits into user registers.

The function code “16 (0x10) Write Multiple Registers” is used to write a block of adjoining registers (1-123, Master device dependent) in a remote device. Specify the the requested written values in the request data field. Data is packed two bytes to a register. A normal response returns the function code, starting address and number of registers written.

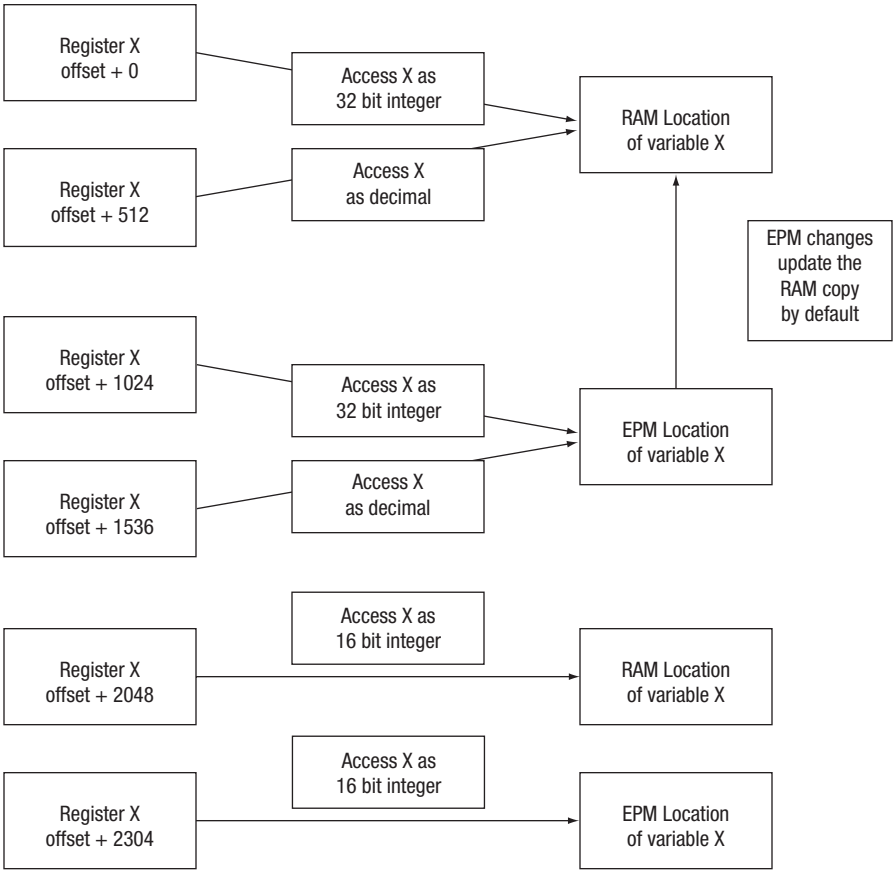

Figure 9: Mapping Register and Variable Addresses based on Access Type

5.4 No Response Conditions

The PositionServo Drive will not respond to any message that:

- contains one or more parity errors
- has an invalid CRC value
- was not directed to the drive's network address
- is not at least 8 bytes long (minimum required for the supported functions)
- is more than 18 bytes long (maximum allowed before input buffer overflow occurs)

5.5 Exception Responses

Per Modbus Application Protocol Specification V1.1b, a Modbus Exception response provides the client with the relevant information concerning the error detected during the processing. The exception function code = the request function code + 0x80 (an exception code is provided to indicate the reason of the error).

If you try to read a 32 bit register using a read holding register command for one 16 bit register and using the base address of the register, then it will return the least significant word of the 32 bit register because the PositionServo drive uses least-significant-word-first rule.

If you try to read a 32 bit register using a read holding register command for one 16 bit register and using the base address of the register plus one, then it will return the most significant word of the 32 bit register.

Notice that this kind of usage is not useful unless you retrieve both halves of a register and re-build the initial value on the client/user side. Request for LSW of 32 Bit value as described above gets MSW to be cached. When MSW is requested, the cached value is returned to allow automatic operation.

If you try to write a 32 bit register using a write multiple register command that writes only one 16 bit register at a time, then the write command is completed only after you write the second word (MSW). No partial writes are executed internally to allow automatic operation and ensure 32 bit value integrity.

Modbus as a protocol does not define the formatting of the data in any one or block of registers. The Modbus Master and slave must both be programmed to interpret the data in the same format.

Exception Codes

If you try to access a function that is not supported by PositionServo drive, the return value is “Illegal Function” (as defined by “Modbus application Protocol specification V1.1”) which is exception code 0x01.

If you try to access an illegal register address the return code value is “Illegal data address” (as defined by “Modbus application Protocol specification V1.1”) which is exception code 0x02.

If you try to set a value that is not allowed for a certain variable, the return value is “Illegal Data Value” (as defined by “Modbus application Protocol specification V1.1”) which is exception code 0x03.

6 Commissioning

6.1 Drive Monitoring

The network will read the drive parameters as long as Modbus communications are enabled. Set the variable #175=1 (Modbus); configure variables #172-173 correctly and program the Modbus Master to poll the appropriate register.

	NOTE: The complete list of variables can be found in Appendix A of the PositionServo Programming Manual (PM94P01).

6.2 Controlling the Drive

Controlling the drive over Modbus is essentially identical to controlling the drive from the User's program. The list of variables and their functionality is identical for both User's program and Modbus control. Refer to the variable table in the PositionServo Programming Manual (PM94P01) for the functionality of the drive's variables.

6.3 Changing Drive Parameters

To change drive parameters, simply write to the appropriate register as listed in the PositionServo Programming Manual (PM94P01)

7 Programming Parameters

	NOTE: The complete list of variables can be found in Appendix A of the PositionServo Programming Manual (PM94P01).

Drive variables #172-176 are RS-485 communication programming parameters specifically for configuration of the RS-485 interface.

7.1 Negative Number Transmission

Drive variables 51, 60, 79, 81 and 90 are signed integer values and could be negative (refer to drive User and Programming manual for details on these parameters). These registers are sent over the modbus communications in signed internal units.

AC Technology Corporation
www.actech.com
630 Douglas Street
Uxbridge, MA 01569
Telephone: (508) 278-9100
Facsimile: (508) 278-7873

P94MOD01B