PositionServo

Dynamic Link Library (DLL)

Communication Reference Guide

P94DLL01A

About These Instructions

This documentation applies to the implementation of DLL with 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. This manual describes the use of DLL with the PositionServo drives. It contains information for anyone who participates in the evaluation of or design of a distributed motion control system. The user should have prior knowledge of motion control, networks and software development.

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.

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 standards documentation.

	**DANGER**! • After the controller has been disconnected from the supply voltage, do not touch live components or power connection until capacitors can discharge. Wait at least 3 minutes before servicing 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

- PositionServo Programming Manual: PM94P01
- PositionServo User Manual: S94P01C
- MotionView Software Manual: IM94MV01A

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

2. PositionServo DLL Overview

This reference guide assumes that the reader has a working knowledge of DLL protocol and familiarity with the programming and operation of motion control equipment. This guide is intended as a reference only.

PositionServo Communication Dynamic-link Library (DLL) provides a set of functions to control, configure and monitor PositionServo drives over Ethernet, RS-485 or RS-232 interfaces. PositionServo drives support the Remote Procedure Call (RPC) protocol to transmit data to and from a master communication unit (PC or controller) and a client-server communication model can be effectively implemented.

All technical details about open standard ONC RPC protocols can be found in RFC-1831 and RFC-1832 documents. PositionServo uses UDP transport (UDP/IP) to send or receive data over network. In case of serial communications RS232 or RS485, the IP data packets are additionally encapsulated by PPP frame and can be transmitted over the serial interfaces. All encapsulation and data preparation are done by the PositionServo Communication Dynamic-link Library (DLL). See the communication flowchart in Section 4 for detailed steps when writing communication software.

3 Files in the DLL Library

The following DLL library files can be found in the MotionView help folder: "…\Help\940 Communication DLL Library". The sample codes to demonstrate their usage can be found in Section 7. See the detailed descriptions of the functions in the DLL library in Section 8.

SS940Control.dll	Main Control DLL Library
oncrpc.dll	RPC protocol library
SS940Control.lib	Static Links
SS940API.h	DLL API
Paramid.h	PositionServo 940 Parameter Definitions
AD485DLL.dll	RS232 or RS485 Communication Support Files
940ControlDeclares.bas	DLL functions declarations for Visual Basic (VB)

NOTE:

Note 1: SS940Control.dll must be kept in the project root directory or in the environment declared path.

Note 2: SS940Control.dll and oncrpc.dll must also be in %SysPath%\SYSTEM directory.

Note 3: If RS232 or RS485 communication is used, AD485DLL.dll must also be kept in the project root directory or in the environment declared path.

4 Communication Flowchart

The flowchart in Figure 1 provides the instructions of how to use DLL functions for communication over Ethernet or serial communications ports.

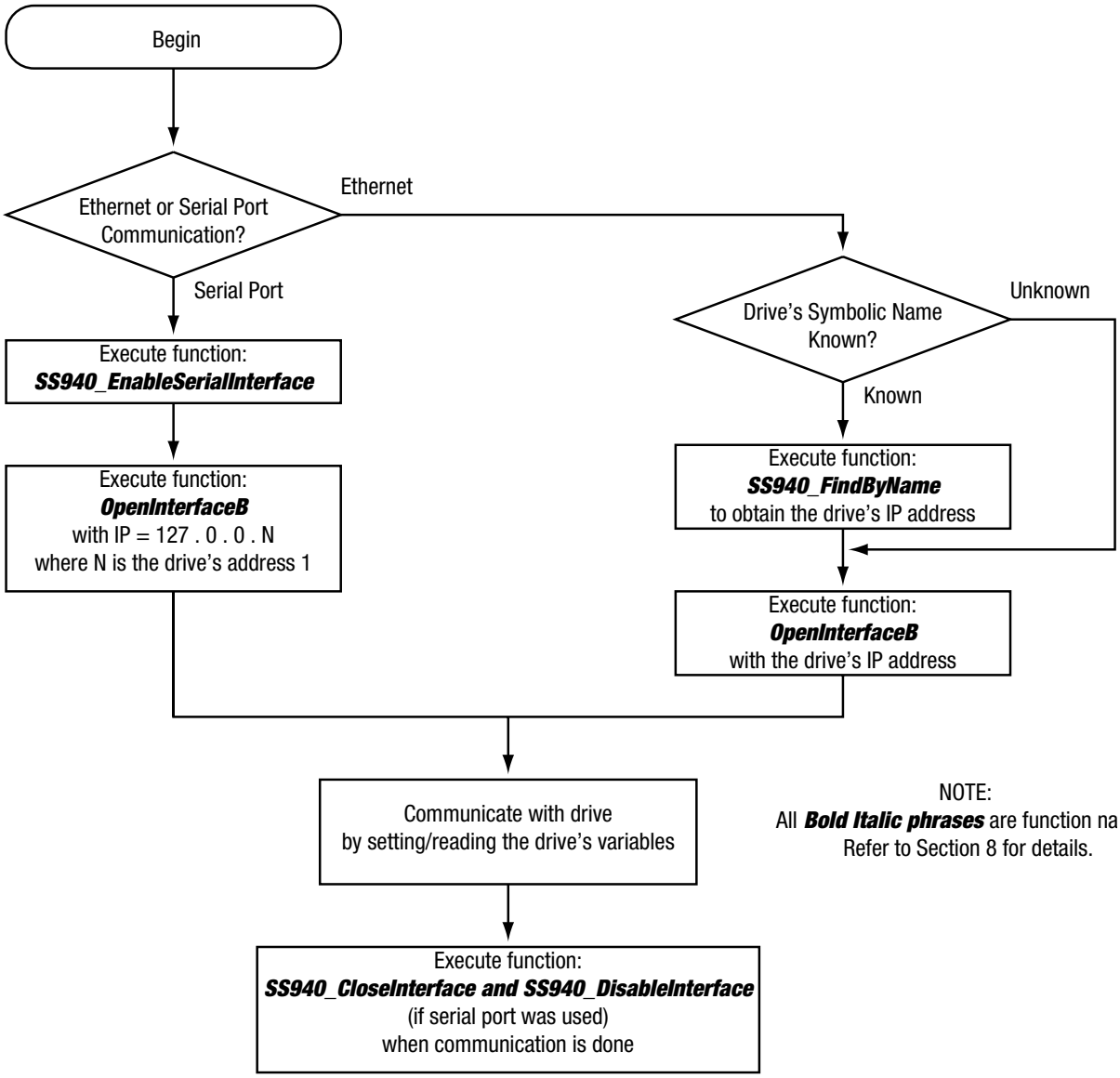

Figure 1 Communication Flowchart

5 DLL Functions Overview

Every aspect of the PositionServo drive can be manipulated by writing or reading the variable(s) inside the drive. All variables are addressable by their respective index number. See the full list of variables in the Appendix A “Complete list of variables” in PositionServo Model Programming Manual (Document No PM94P01).

Every variable can be interpreted as a 32 bit integer or as DOUBLE. For Read/Write, and Set/Get functions, there are two function versions to read/write variables as integer or double precision types. Each variable has its native format inside the drive, regardless of how the value was sent. The received value will be cast to its natural format by the drive.

All variables are located in RAM but some of them have a non-volatile copy in the EPM (Electronic Programming Module) memory. The DLL library provides two types of functions for writing such variables. One type (SET) changes only RAM (run time) copy of the variables when the other (WRITE) changes both – the run time copy in RAM and the non-volatile copy in EPM. At the drive reboot, the variables which have non-volatile copies will be initialized with the values stored in EPM.

Two types of functions are also provided for reading. One type (GET) reads the RAM (run time) value of the variables. The other (READ) reads the non-volatile variable value from EPM.

SET and GET function types also include LIST function versions. The corresponding LIST functions write or read a sequential list of the variables up to 10 variables in one function call.

6 Return Codes

Table 1 lists the return codes, their abbreviation and description.

Table 1: PositionServo DLL Return Codes

Abbreviation	Code	Description
EC_CMD_NS	0	Command is not specified
EC_OK	1	Command is performed OK
EC_VALUE_TOOSMALL	-1	Value for variable is too small. Value is boosted and accepted
EC_VALUE_TOOBIG	2	Value for variable is too big. Truncated and accepted
EC_INVALID_HANDLE	100	Invalid handle
EC_TIMEOUT	101	Request is timed out
EC_COMPORT_ENABLED	200	COM port is enabled
EC_COMPORT_FAILED	201	Failed to open/operate the specified COM port

7 DLL Functions Usage Examples

Sample projects with the complete source codes are provided for demonstration purposes of DLL function usage. Two sets of source codes written in Visual Basic and Visual C++ are available under the MotionView help folder “…\Help\940 Communication DLL Library\940 Communication DLL Source Codes”.

8 DLL Functions

All functions in the DLL library are described in detail in paragraphs 8.1 “Connection Services Functions” and 8.2 “Data Manipulation Functions”.

8.1 Connection Services Functions

There are six DLL Connection Service functions applicable to the PositionServo drive. The function names are identified by ***bold italic*** text.

long GetDllVersion( )

Purpose: To obtain the DLL version

Inputs: none

Returns: DLL version as an integer number

long SS940_EnableSerialInterface (long port, long baudrate)

Purpose: To open the specified serial port for communication and redirect all communication requests to the serial port.

Inputs:	port	serial port number	(example: 1 selects COM port 1)
	baudrate	port baudrate	(example: 115200)
Returns:	error code.	Value 1 indicates OK.	

long SS940_DisableSerialInterface ( )

Purpose: To close the serial port for communication and stops redirecting all communication to the serial port.

Inputs: none

Returns: error code

SHANDLE SS940_OpenInterfaceB (BYTE* address, int timeout)

Purpose: To open communication interface to device with the IP address supplied

Inputs: address byte array containing 4 bytes IP address of the drive.

Example: Form byte array for device IP 192.168.24.12

BYTE address[] = { 192,168,24,12 };

NOTE: For operation with a serial port use IP = 127.0.0.N, where N is the drive’s serial address in the range 0-31. The function ***SS940_EnableSerialInterface*** must be executed before communication is routed to the serial port

timeout request timeout in milliseconds. Set this value as 3000 in general case.

Returns: Handle to open interface. Valid handle is a non-zero number.

long SS940_CloseInterface (SHANDLE handle)

Purpose: To close the communication interface to device with the specific handle.

Inputs: handle to the previously opened interface by the function ***SS940_OpenInterfaceB.***

Return: error code

long SS940_FindByName (char* name, BYTE* ip_address, BYTE* ser_num, int timeout)

Purpose: To find PositionServo drives on the network by their names and retrieve their IP addresses and serial address numbers.

Inputs:	name	ASCII string containing the drive's name
	Timeout	request time out in ms. Set it in 3000 ms for general use
Return:	ip_address	4 bytes drive's IP address.
	ser_num	4 bytes serial number information

8.2 Data Manipulation Functions

There are twelve DLL Data Manipulation Functions that apply to the PositionServo drive.

long SS940_ReadParamAsDouble (SHANDLE h, int pid, double& param)

Purpose: To read the EPM copy of the variable as a DOUBLE type with the index specified by pid and return it in argument parameter.

Inputs:	h	handle to interface opened by ***SS940_OpenInterfaceB***
	pid	variable (parameter) index
Returns:	param	variable value in double format

long SS940_ReadParamAsInteger (SHANDLE h, int pid, int& param)

Purpose: To read EPM copy of the variable as a 32-bit INTEGER type with index specified by pid and return it in argument parameter.

Inputs:	h	handle to interface opened by ***SS940_OpenInterfaceB***
	pid	variable (parameter) index
Returns:	param	variable value in integer format

long SS940_GetParamAsDouble (SHANDLE h, int pid, double& param)

Purpose: To read a RAM (run time) variable as a DOUBLE type with index specified by pid and return it in argument parameter.

Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pid	variable (parameter) index
Returns:	param	variable value in double format

long SS940_GetParamAsInteger (SHANDLE h, int pid, int& param)

Purpose: To read the RAM (run time) variable as a 32-bit INTEGER type with index specified by pid and return it in argument parameter.

Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pid	variable (parameter) index
Returns:	param	variable value in integer format

long SS940_WriteParamAsDouble (SHANDLE h, int pid, double param)

Purpose: To write the RAM (run time) and EPM copies of the variable as a DOUBLE type with index specified by pid and return it in argument parameter.

Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pid	variable (parameter) index
Returns:	param	variable value in double format

long SS940_WriteParamAsInteger (SHANDLE h, int pid, int param)

Purpose: To write the RAM (run time) and EPM copies of the variable as a DOUBLE type with index specified by pid and return it in argument parameter.

Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pid	variable (parameter) index
Returns:	param	variable value in integer format

long SS940_SetParamAsDouble (SHANDLE h, int pid, double param)

Purpose:	To write the RAM (run time) variable as a DOUBLE type with index specified by pid and return it in argument parameter.	
Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pid	variable (parameter) index
Returns:	param	variable value in double format

long SS940_SetParamAsInteger (SHANDLE h, int pid, int param)

Purpose:	To write the RAM (run time) variable as a DOUBLE type with index specified by pid and return it in argument param.	
Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pid	variable (parameter) index
Returns:	param	variable value in integer format

long SS940_GetArrayAsDouble (SHANDLE h, BYTE pids, double* params, int count)*

Purpose:	To read the RAM (run time) array of variables (up to 10 variables) as a DOUBLE type with indexes specified by elements of array pids and return it in array parameter.	
Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pids	array of variable's indexes
	count	number of array elements. Maximum number of elements is 10.
Returns:	params	array of values of type DOUBLE.

long SS940_GetArrayAsInteger (SHANDLE h, BYTE pids, long* params, int count)*

Purpose:	To read the RAM (run time) array of variables (up to 10 variables) as a 32- bit INTEGER type with indexes specified by elements of array pids and return it in array parameter.	
Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pids	array of variable's indexes
	count	number of array elements. Maximum number of elements is 10.
Returns:	params	array of values of type DOUBLE.

long SS940_SetArrayAsDouble (SHANDLE h, long pids, double* params, int count)*

Purpose:	To write the RAM (run time) array of variables (up to 10 variables) as a DOUBLE type with indexes specified by elements of array pids and return it in array parameter.	
Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pids	array of variable's indexes
	count	number of array elements. Maximum number of elements is 10.
Returns:	params	array of error codes

long SS940_SetArrayAsInteger (SHANDLE h, long pids ,longt* params, int count)*

Purpose:	To write the RAM (run time) array of variables (up to 10 variables) as a type 32 bit INTEGER type with indexes specified by elements of array pids and return it in array parameter.	
Inputs:	h	handle to interface opened by the function ***SS940_OpenInterfaceB***
	pids	array of variable's indexes
	count	number of array elements. Maximum number of elements is 10.
Returns:	params	array of error codes

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

P94DLL01A