



PositionServo (MVCD)
Users Manual

Copyright ©2005 by Lenze AC Tech Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from Lenze AC Tech Corporation. The information and technical data in this manual are subject to change without notice. Lenze AC Tech makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. Lenze AC Tech assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView®, PositionServo®, and all related indicia are either registered trademarks or trademarks of Lenze AG in the United States and other countries.

This document printed in the United States of America.



1	Introduction	5
1.1	About These Instructions	6
1.2	Scope of Supply	6
1.3	Legal Regulations	6
1.4	Part Number Designation	7
2	Technical Data	9
2.1	Electrical Characteristics	9
2.2	Power Ratings	10
2.3	Fuse Recommendations	10
2.4	Digital I/O Ratings	11
2.5	Environment	11
2.6	Operating Modes	11
2.7	Connections and I/O	11
2.8	PositionServo Dimensions	12
2.9	Clearance for Cooling Air Circulation	13
3	Installation	14
3.1	Wiring	15
3.2	Shielding and Grounding	15
3.2.1	General Guidelines	15
3.2.2	EMI Protection	16
3.2.3	Enclosure	16
3.3	Line Filtering	17
3.4	Heat Sinking	17
3.5	Line (Mains) Fusing	17
4	Interface	18
4.1	External Connectors	18
4.1.1	P1 & P7 - Input Power and Output Power Connections	18
4.1.2	P2 - Ethernet Communications Port	19
4.1.3	P3 - Controller Interface	20
4.1.4	P4 - Motor Feedback / Second Loop Encoder Input	21
4.1.5	P5 - 24 VDC Back-up Power Input	22
4.1.6	P6 - Braking Resistor and DC Bus	22
4.1.7	Connector and Wiring Notes	23
4.1.8	P11 - Resolver Interface Module (option)	24
4.1.9	P12 - Second Encoder Interface Module (option)	25
4.2	Digital I/O Details	26
4.2.1	Step & Direction / Master Encoder Inputs (P3, pins 1-4)	26
4.2.2	Buffered Encoder Output (P3, pins 7-12)	27
4.2.3	Digital Outputs	27
4.2.4	Digital Inputs	28
4.3	Analog I/O Details	29
4.3.1	Analog Reference Input	29
4.3.2	Analog Output	30
4.4	Communication Interfaces	30
4.4.1	Ethernet Interface (standard)	30
4.4.2	RS485 Interface (option)	30
4.4.3	RS485 Communication Setup	31
4.4.4	MODBUS RTU Support	31
4.4.5	CAN Interface (option)	31
4.5	Motor Selection	31
4.5.1	Motor Connection	31
4.5.2	Motor Over-temperature Protection	32
4.5.3	Motor Setup	32



Contents

4.6	Using a Custom Motor	33
4.6.1	Creating Custom Motor Parameters	33
4.6.2	Autophasing	34
4.6.3	Custom Motor Data Entry	34
5	Parameters	39
5.1	Parameter Storage and EPM Operation	39
5.1.1	Parameter Storage	39
5.1.2	EPM Operation	39
5.1.3	EPM Fault	39
5.2	Motor Group	40
5.3	Parameters	40
5.3.1	Drive Operating Modes	40
5.3.2	Drive PWM frequency	41
5.3.3	Current Limit	41
5.3.4	Peak Current Limit (8 kHz and 16 kHz)	41
5.3.5	Analog Input Scale (current scale)	41
5.3.6	Analog Input Scale (velocity scale)	41
5.3.7	ACCEL/DECEL Limits (velocity mode only)	42
5.3.8	Reference	42
5.3.9	Step Input Type (position mode only)	42
5.3.10	Fault Reset Option	42
5.3.11	Motor Temperature Sensor	42
5.3.12	Motor PTC Cut-off Resistance	42
5.3.13	Second Encoder	42
5.3.14	Regeneration Duty Cycle	43
5.3.15	Encoder Repeat Source	44
5.3.16	System to Master Ratio	44
5.3.17	Second to Prime Encoder Ratio	44
5.3.18	Autoboot	44
5.3.19	Group ID	44
5.3.20	Enable Switch Function	44
5.3.21	User Units	44
5.3.22	Resolver Track	44
5.3.23	Current Limit Max Overwrite	45
5.4	Communication	45
5.4.1	Ethernet Interface	45
5.4.2	RS-485 Configuration	48
5.4.3	Modbus Baud Rate	48
5.4.4	Modbus Reply Delay	48
5.5	Analog I/O	48
5.5.1	Analog Output	48
5.5.2	Analog Output Current Scale (Volt / amps)	49
5.5.3	Analog Output Current Scale (mV/RPM)	49
5.5.4	Analog Input Dead Band	49
5.5.5	Analog Input Offset Parameter	49
5.5.6	Adjust Analog Input Zero Offset	49
5.6	Digital I/O	49
5.6.1	Digital Input De-bounce Time	49
5.6.2	Hard Limit Switch Action	50
5.7	Velocity Limits	50
5.7.1	Zero Speed	50
5.7.2	Speed Window	50
5.7.3	At Speed	50
5.8	Position Limits	50
5.8.1	Position Error	50
5.8.2	Max Error Time	50
5.8.3	Second Encoder Position Error	50
5.8.4	Second Encoder Max Error Time	50



5.9	Compensation	.51
5.9.1	Velocity P-gain (proportional)	.51
5.9.2	Velocity I-gain (integral)	.51
5.9.3	Position P-gain (proportional)	.51
5.9.4	Position I-gain (integral)	.51
5.9.5	Position D-gain (differential)	.51
5.9.6	Position I-limit	.51
5.9.7	Gain Scaling Window	.52
5.10	Tools	.52
5.10.1	Oscilloscope Tool	.52
5.10.2	Run Panels	.52
5.11	Faults	.52
6	Operation	53
6.1	Minimum Connections	.53
6.2	Configuration of the PositionServo	.53
6.3	Position Mode Operation (gearing)	.55
6.4	Dual-loop Feedback	.55
6.5	Enabling the PositionServo	.56
6.6	Drive Tuning	.56
6.6.1	Tuning the Drive in Velocity Mode	.57
6.6.2	Tuning the Drive in Position Mode	.62
7	Quick Start Reference	68
7.1	Quick Start - External Torque Mode	.68
7.2	Quick Start - External Velocity Mode	.69
7.3	Quick Start - External Positioning Mode	.71
8	Diagnostics	73
8.1	Display	.73
8.2	LEDs	.74
8.3	Faults	.74
8.3.1	Fault Codes	.74
8.3.2	Fault Event	.76
8.3.3	Fault Reset	.76
8.4	Troubleshooting	.76

Safety Information





All safety information given in these Operating Instructions has a similar layout:



Signal Word! (Characteristics the severity of the danger)

Note (describes the danger and informs on how to proceed)

Pictographs used in these instructions:

Icon		Signal Words	
	Warning of hazardous electrical voltage	DANGER!	Warns of impending danger . Consequences if disregarded: Death or severe injuries.
	Warning of a general danger	WARNING!	Warns of potential, very hazardous situations . Consequences if disregarded: Death or severe injuries.
	Warning of damage to equipment	STOP!	Warns of potential damage to material and equipment . Consequences if disregarded: Damage to the controller/drive or its environment.
	Information	NOTE	Designates a general, useful note. If you observe it, handling the controller/drive system is made easier.



1 Introduction

The PositionServo line of advanced general purpose servo drives utilizes the latest technology in power semiconductors and packaging. The PositionServo uses Field Oriented control to enable high quality motion.

The PositionServo is available in four mains (input power) configurations:

1. **400/480V** (nominal) three phase input. An external input mains (line) filter is available. Actual voltage can range from 320 - 528 VAC.
2. **120/240V** (nominal) Single Phase input with integrated input mains (line) filter, Actual input voltage can range from 80VAC to 264VAC. The maximum output voltage is approximately equal to the input voltage.
3. **120V or 240V** (nominal) Single or Three Phase input. Actual input voltage can range from 80VAC to 264VAC. The maximum output voltage is approximately equal to the input voltage. An external input mains (line) filter is available.
4. **120V or 240V** (nominal) single phase input. When wired for **Doubler mode** (L1-N), the input is for 120V nominal only and can range from 45VAC to 132 VAC and the maximum output voltage is double the input voltage. When wired to terminals L1-L2/N, the input can range from 80 VAC to 264 VAC and the maximum output voltage is equal to the input voltage.

The PositionServo drive can operate in one of three mode settings, torque (current), velocity, or position (step & direction or master encoder). The drive's command or reference signal can come from one of three sources. The first is an external reference. An external reference can be an analog input signal, a step and direction input or an input from a master encoder. The second reference is an internal reference. An internal reference is when the commanded move is derived from the drive's user program. The third reference is when the commanded move is done via a host device over a communications network. This Host device can be an external motion controller, PLC, HMI or PC. The communication network can be RS485 (Point-to-Point or Modbus RTU), Ethernet (using MotionView DLL's), Modbus over TCP/IP, or CANopen (DS301).

Depending on the primary feedback, there are two types of drives: the Model 940 PositionServo encoder-based drive which accepts an incremental encoder with Hall channel inputs and the Model 941 PositionServo resolver-based drive which accepts resolver inputs. The feedback signal is brought back to the P4 connector on the drive. This connector will be a 15 pin D-sub for the encoder version and a 9 pin D-sub for the resolver version. A second encoder can be used in position and velocity modes.

The MotionView software is the setup and management tool for PositionServo drives. All parameters can be set and monitored via this user-friendly tool. It has a real-time oscilloscope tool for analysis and optimum tuning. The users program written with SimpleMotion Programming Language (SML) can be utilized to command motion and handle the drive's inputs/outputs (I/O). The programming language is designed to be very intuitive and easy to implement. For programming details, refer to the PositionServo Programming Manual. All PositionServo related manuals can be downloaded from the Technical Library on the AC Tech website (<http://www.lenze-actech.com>).

On each PositionServo drive, there is an Electronic Programming Module (EPM), which stores all drive setup and tuning information. This module can be removed from the drive and reinstalled into another drive, making the field replacement of the drive extremely easy. This also makes it easy to duplicate the settings for several drives.

The PositionServo drive supports a variety of communication protocols, including Point-to-Point (PPP), Modbus RTU over RS485, Ethernet TCP/IP, Modbus over TCP/IP and CANopen (DS301).





Introduction

1.1 About These Instructions

These Operating Instructions are provided to assist the user in connecting and commissioning the PositionServo drive with model number ending in "EX" or "RX". Read this manual in its entirety and observe all safety instructions contained in this document.

All persons working on or with the controller must have the Operating Instructions available and must observe the information and notes relevant for their work.

A	B	C	D	E	F
Lenze AC Tech Made in USA POSITIONServo Model 940		Type: E94P120Y2NEX ID-No: 13014745  IND. CONT. EQ.	INPUT: 1(3)/PE 120/240 V 24.0 (13.9) A 50-60 HZ 	OUTPUT: 3/PE 0 - 230 V 12.0 A	For detailed information refer to instruction Manual: S94P01 SN 13014745012345678 E94P120Y2NEX0XX###

A	B	C	D	E	F
Certifications	Type	Input Ratings	Output Ratings	Hardware Version	Software Version

1.2 Scope of Supply

Scope of Supply	Important
<ul style="list-style-type: none"> 1 Model PositionServo type E94P or E94R. 1 Users Manual (English) 1 MotionView CD ROM including: <ul style="list-style-type: none"> - configuration software - documentation (Adobe Acrobat) 	<p>After reception of the delivery, check immediately whether the scope of supply matches the accompanying papers. Lenze- AC Tech does not accept any liability for deficiencies claimed subsequently.</p> <p>Claim</p> <ul style="list-style-type: none"> visible transport damage immediately to the forwarder visible deficiencies / incompleteness immediately to your Lenze representative.

1.3 Legal Regulations

Identification	Nameplate	CE Identification	Manufacturer
	Lenze controllers are unambiguously designated by the contents of the nameplate	In compliance with the EC Low-Voltage Directive	AC Technology Corp. member of the Lenze Group 630 Douglas Street Uxbridge, MA 01569 USA
Application as directed	<p>E94P or E94R servo controller</p> <ul style="list-style-type: none"> must only be operated under the conditions prescribed in these Instructions. are components for: <ul style="list-style-type: none"> - closed loop control of variable speed/torque applications with PM synchronous motors. - installation in a machine. - assembly with other components to form a machine. are electric units for installation in control cabinets or similarly enclosed housing. comply with the requirements of the Low-Voltage Directive. are not machines for the purpose of the Machinery Directive. are not to be used as domestic appliances, but only for industrial purposes. <p>Drive systems with E94P or E94R servo inverters</p> <ul style="list-style-type: none"> comply with the EMC Directive if they are installed according to the guidelines of CE-typical drive systems. can be used for: <ul style="list-style-type: none"> - for operation on public and non-public mains - for operation in industrial premises and residential areas. The user is responsible for the compliance of his application with the EC directives. <p>Any other use shall be deemed as inappropriate!</p>		



Liability	<ul style="list-style-type: none"> The information, data, and notes in these instructions met the state of the art at the time of publication. Claims on modifications referring to controllers that have already been supplied cannot be derived from the information, illustrations, and descriptions. The specifications, processes and circuitry described in these instructions are for guidance only and must be adapted to your own specific application. Lenze does not take responsibility for the suitability of the process and circuit proposals. The specifications in these Instructions describe the product features without guaranteeing them. Lenze does not accept any liability for damage and operating interference caused by: <ul style="list-style-type: none"> - Disregarding the operating instructions - Unauthorized modifications to the controller - Operating errors - Improper working on and with the controller 		
Warranty	<ul style="list-style-type: none"> Warranty conditions: see Sales and Delivery Conditions of Lenze Drive Systems GmbH. Warranty claims must be made to Lenze immediately after detecting the deficiency or fault. The warranty is void in all cases where liability claims cannot be made. 		
Disposal	Material	Recycle	Dispose
	Metal	•	-
	Plastic	•	-
	Assembled PCB's	-	•

1.4 Part Number Designation

The table herein describes the part number designation for the PositionServo drive. The available filter and communication options are detailed in separate tables.

	E94	P	020	S	1	N	E	X
Electrical Products in the 94 Series								
P = PositionServo Model 940 with Encoder Feedback								
R = PositionServo Model 941 with Resolver Feedback								
Drive Rating in Amps:								
020 = 2 Amps 090 = 9 Amps								
040 = 4 Amps 100 = 10 Amps								
050 = 5 Amps 120 = 12 Amps								
060 = 6 Amps 180 = 18 Amps								
080 = 8 Amps								
Input Phase:								
S = Single Phase Input only								
Y = Single or Three Phase Input								
T = Three Phase Input only								
Input Voltage:								
1 = 120 VAC Doubler (120V, 1~ in/ 240V, 3~ out)								
2 = 200/240 VAC								
4 = 400/480 VAC								
Line Filter								
N = No Line Filter								
F = Integrated Line Filter								
Secondary Feedback								
E = Incremental Encoder								
R = Standard Resolver								
EN954-1 Safety Circuit								
X = No EN954 Safety Circuit								



Introduction

Filter Part Number Designation

	E94Z	F	4	T	4	A1
Electrical Option in the 94 Series						
F = EMC Filter						
Filter Current Rating in Amps:						
04 = 4.4 Amps		12 = 12 Amps				
07 = 6.9 Amps		15 = 15 Amps				
10 = 10 Amps		24 = 24 Amps				
Input Phase:						
S = Single Phase						
T = Three Phase						
Max Voltage:						
2 = 240 VAC						
4 = 400/480 VAC						
Degree of Filtering/Variation						
A1 = Industrial/1st Variation						
A2 = Industrial/2nd Variation						

Servo Option Part Number Designation

	E94Z	A	CAN	1
Electrical Option in the 94 Series				
A = COMM, Feedback or Breakout Module				
Module Type:				
CAN = CANopen COMM Module		ENC = 2nd Encoder Feedback Module		
RS4 = RS485 COMM Module		RSV = Resolver Feedback Module		
ETH = Ethernet COMM Module		HBK = Motor Brake Terminal Module		
		TBO = Terminal Block I/O Module		
		SCA = Panel Saver I/O Module		
Variations				
1 = 1st Variation				
2 = 2nd Variation				
3 = 3rd Variation				



2 Technical Data

2.1 Electrical Characteristics

Single-Phase Models					
Type ⁽¹⁾	Mains Voltage ⁽²⁾	1~ Mains Current (doubler)	1~ Mains Current (Std.)	Rated Output Current ⁽⁵⁾	Peak Output Current ⁽⁶⁾
E94_020S1N_X	120V ⁽³⁾ or 240V ⁽⁴⁾	9.7	5.0	2.0	6
E94_040S1N_X		15	8.6	4.0	12
E94_020S2F_X	120 / 240V ⁽⁴⁾ (80 V -0%...264 V +0%)	--	5.0	2.0	6
E94_040S2F_X		--	8.6	4.0	12
E94_080S2F_X		--	15.0	8.0	24
E94_100S2F_X		--	18.8	10.0	30
Single/Three-Phase Models					
Type ⁽¹⁾	Mains Voltage ⁽²⁾	1~ Mains Current	3~ Mains Current	Rated Output Current ⁽⁵⁾	Peak Output Current ⁽⁶⁾
E94_020Y2N_X	120 / 240V ⁽⁴⁾ 1~ or 3~ (80 V -0%...264 V +0%)	5.0	3.0	2.0	6
E94_040Y2N_X		8.6	5.0	4.0	12
E94_080Y2N_X		15.0	8.7	8.0	24
E94_100Y2N_X		18.8	10.9	10.0	30
E94_120Y2~_X		24.0	13.9	12.0	36
E94_180T2~_X	240V 3~ (180 V -0%...264 V +0%)	--	19.6	18.0	54
E94_020T4N_X	400 / 480V 3~ (320 V -0%...528 V +0%)	--	2.7	2.0	6
E94_040T4N_X		--	5.5	4.0	12
E94_050T4N_X		--	6.9	5.0	15
E94_060T4~_X		--	7.9	6.0	18
E94_090T4~_X		--	12.0	9.0	27

- (1) The first "_" equals "P" for the 940 encoder based drive or "R" for the 941 resolver based drive.
When the 10th digit is marked by "-", "N" = No line filter or "F" = Integrated line filter
The second "_" equals "E" for incremental encoder (must have E94P drive) or "R" for the standard resolver (must have E94R drive).
- (2) Mains voltage for operation on 50/60 Hz AC supplies (48 Hz -0% ... 62Hz +0%).
- (3) Connection of 120VAC (45 V ... 132 V) to input power terminals L1 and N on these models doubles the voltage on motor output terminals U-V-W for use with 230VAC motors.
- (4) Connection of 240VAC or 120VAC to input power terminals L1 and L2 on these models delivers an equal voltage as maximum to motor output terminals U-V-W allowing operation with either 120VAC or 230VAC motors.
- (5) Drive rated at 8kHz Carrier Frequency. Derate Continuous current by 17% at 16kHz.
- (6) Peak RMS current allowed for up to 2 seconds. Peak current rated at 8kHz. Derate by 17% at 16kHz.

Applies to all models:

Acceleration Time Range (Zero to Max Speed)	0.1 ... 5x10 ⁶ RPM/sec
Deceleration Time Range (Max Speed to Zero)	0.1 ... 5x10 ⁶ RPM/sec
Speed Regulation (typical)	± 1 RPM
Input Impedance (AIN+ to COM and AIN+ to AIN-)	47 kΩ
Power Device Carrier Frequency (sinusoidal commutation)	8.16 kHz
Encoder Power Supply (max)	+5 VDC @ 300 mA
Maximum Encoder Feedback Frequency	2.1 MHz (per channel)
Maximum Output Frequency (to motor)	400Hz
Resolver Carrier Frequency	4.5 - 5.5kHz (5kHz nominal)
Resolver Turns Ratio between Reference and SIN/COS signal	2:1



2.2 Power Ratings

Type ⁽¹⁾	Output kVA at Rated Output Current (8kHz) ⁽²⁾	Leakage Current	Power Loss at Rated Output Current (8kHz)	Power Loss at Rated Output Current (16 kHz) ⁽³⁾
Units	kVA	mA	Watts	Watts
E94_020S1N_X	0.8	Typically >3.5 mA. Consult factory for applications requiring <3.5 mA.	19	21
E94_040S1N_X	1.7		29	30
E94_020S2F_X	0.8		19	21
E94_040S2F_X	1.7		29	30
E94_080S2F_X	3.3		61	63
E94_100S2F_X	4.2		80	85
E94_020Y2N_X	0.8		19	21
E94_040Y2N_X	1.7		29	30
E94_080Y2N_X	3.3		61	63
E94_120Y2--_X	5.0		114	129
E94_180T2--_X	7.5		171	195
E94_020T4N_X	1.7		31	41
E94_040T4N_X	3.3		50	73
E94_050T4N_X	4.2		70	90
E94_060T4--_X	5.0		93	122
E94_090T4--_X	7.5		138	182

2.3 Fuse Recommendations

Type ⁽¹⁾	AC Line Input Fuse (1ø/3ø)	Miniature Circuit Breaker ⁽⁶⁾ (1ø/3ø)	AC Line Input Fuse ⁽⁴⁾ or Breaker ⁽⁵⁾ (N. America)	DC Bus Input Fuse ⁽⁷⁾
Amp Ratings				
E94_020S1N_X	M20/M10	C20/C10	20/10	10
E94_040S1N_X	M32/M20	C32/C20	30/20	20
E94_020S2F_X	M20	C20	20	15
E94_040S2F_X	M20	C20	20	20
E94_080S2F_X	M32	C32	32	40
E94_100S2F_X	M40	C40	40	45
E94_020Y2N_X	M20/M16	C20/C16	20/15	15
E94_040Y2N_X	M20/M16	C20/C16	20/15	20
E94_080Y2N_X	M32/M20	C32/C20	30/20	40
E94_120Y2--_X	M50/M32	C50/C32	50/30	55
E94_180T2--_X	M40	C40	40	80
E94_020T4N_X	M10	C10	10	10
E94_040T4N_X	M10	C10	10	20
E94_050T4N_X	M16	C16	15	25
E94_060T4--_X	M20	C20	20	30
E94_090T4--_X	M25	C25	25	40

- (1) The first "_" equals "P" for the Model 940 encoder based drive or "R" for the Model 941 resolver based drive. When the 10th digit is marked by "--", "N" = No line filter or "F" = Integrated line filter. The second "_" equals "E" for incremental encoder (must have E94P drive) or "R" for the standard resolver (must have E94R drive).
- (2) At 240 VAC line input for drives with suffixes "S1N", "S2F", "Y2N". At 480 VAC line input for drives with suffixes "T4N".
 - a. The output power is calculated from the formula: output kVA = $[(\sqrt{3}) \times U_L \times I_{rated}] / 1000$
 - b. The actual output power (kW) depends on the motor in use due to variations in motor rated voltage, rated speed and power factor, as well as actual max operating speed and desired overload capacity.
 - c. Typical max continuous power (kW) for PM servo motors runs 50-70% of the kVA ratings listed.
- (3) At 16 kHz, de-rate continuous current by 17%
- (4) Installations with high fault current due to large supply mains may require a type D circuit breaker.
- (5) UL Class CC or T fast-acting current-limiting type fuses, 200,000 AIC, preferred. Bussman KTK-R, JLN, JJS or equivalent.
- (6) Thermal-magnetic type breakers preferred.
- (7) DC-rated fuses, rated for the applied voltage. Examples Bussman KTM or JLN as appropriate.



2.4 Digital I/O Ratings

	Scan Times	Linearity	Temperature Drift	Offset	Current	Input Impedance	Voltage Range
Units	μs	%	%	%	mA	Ohm	VDC
Digital Inputs⁽¹⁾	512				Depend on load	2.2 k	5-24
Digital Outputs	512				15 max	N/A	30 max
Analog Inputs	512	± 0.013	0.1% per °C rise	± 0 adjustable	Depend on load	47 k	± 18
Analog Outputs	512		0.1% per °C rise	± 0 adjustable	10 max	N/A	± 10

(1) Inputs do not have scan time. Their values are read directly by indexer program statement.
De-bounce time is programmable and can be set as low as 0. Propagation delay is typical 20 us

2.5 Environment

Vibration	2 g (10 - 2000 Hz)
Ambient Operating Temperature Range	0 to 40°C
Ambient Storage Temperature Range	-10 to 70°C
Temperature Drift	0.1% per °C rise
Humidity	5 - 90% non-condensing
Altitude	1500m/5000ft [derate by 1% per 300m (1000 ft) above 1500m (5000 ft)]

2.6 Operating Modes

Torque

Reference	± 10 VDC 16-bit; scalable
Torque Range	100:1
Current-Loop Bandwidth	Up to 1.5 kHz*

Velocity

Reference	± 10 VDC or 0...10 VDC; scalable
Regulation	± 1 RPM
Velocity-Loop Bandwidth	Up to 200 Hz*
Speed Range	5000:1 with 5000 ppr encoder

Position

Reference	0...2 MHz Step & Direction or 2 channels quadrature input; scalable
Minimum Pulse Width	500 nanoseconds
Loop Bandwidth	Up to 200 Hz*
Accuracy	± 1 encoder count for encoder feedback ± 1.32 arc-minutes for resolver feedback (14-bit resolution)

* = motor and application dependent

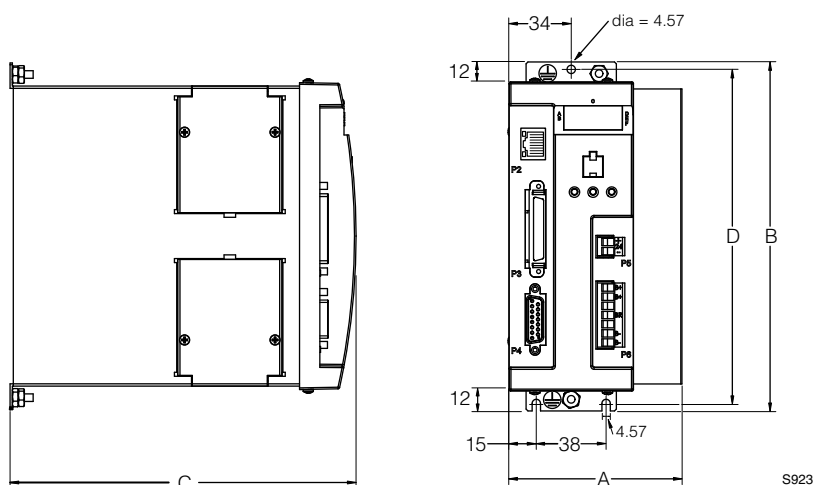
2.7 Connections and I/O

Mains Power	4-pin removable terminal block	(P1)
Ethernet Port	Standard RJ45 Connector	(P2)
I/O Connector	Standard 50-pin SCSI.	(P3)
- Buffered Encoder Output	A, B & Z channels with compliments (5V @ 20mA)	(P3)
- Digital Inputs	11 programmable, 1 dedicated (5-24V)	(P3)
- Digital Outputs	4 programmable, 1 dedicated (5-24V @ 15mA)	(P3)
- Analog Input	2 differential; ± 10 VDC (one 16 bit, one 10 bit)	(P3)
- Analog Output	1 single ended; ± 10 VDC (10-bit)	(P3)
Encoder Feedback (E94P drive)	Feedback connector, 15-pin D-shell	(P4)
Resolver Feedback (E94R drive)	Feedback connector, 9-pin D-shell	(P4)
24VDC Power "Keep Alive"	2-pin removable terminal block	(P5)
Regen and Bus Power	5-pin removable terminal block	(P6)
Motor Power	6-pin pin removable terminal block	(P7)
Resolver Feedback (option bay)	Option module with standard 9-pin D-shell	(P11)
Encoder Feedback (option bay)	Option module with standard 9-pin D-shell	(P12)
Comm Option Bay	Optional Comm Modules (CAN, RS485)	(P21)
Windows® Software:	MotionView (Windows 98, NT, 2000, XP)	



Technical Data

2.8 PositionServo Dimensions



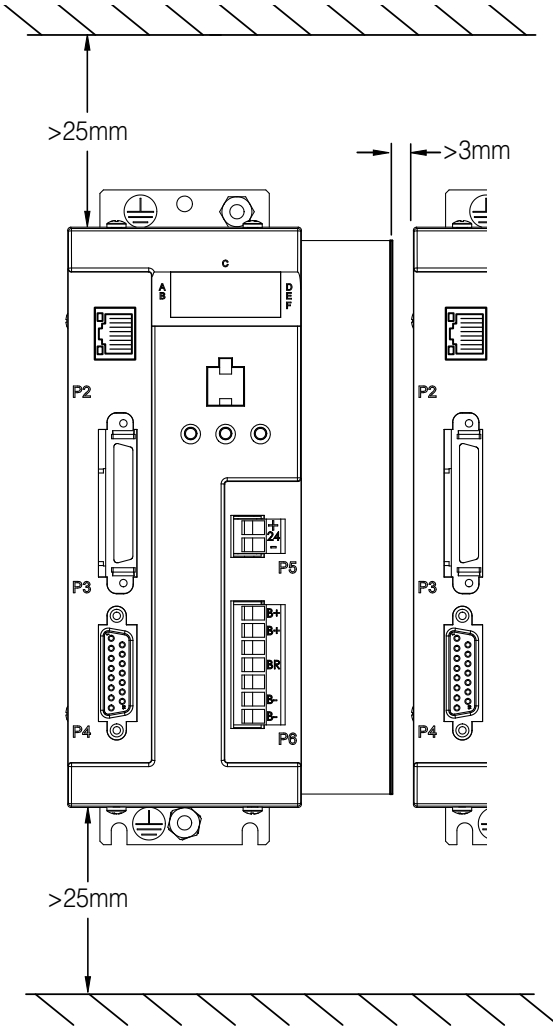
S923

Type ⁽¹⁾	A (mm)	B (mm)	C (mm)	D (mm)	Weight (kg)
E94_020S1N_X	68	190	190	182	1.1
E94_040S1N_X	69	190	190	182	1.2
E94_020S2F_X	68	190	235	182	1.3
E94_040S2F_X	69	190	235	182	1.5
E94_080S2F_X	87	190	235	182	1.9
E94_100S2F_X	102	190	235	182	2.2
E94_020Y2N_X	68	190	190	182	1.3
E94_040Y2N_X	69	190	190	182	1.5
E94_080Y2N_X	95	190	190	182	1.9
E94_100Y2N_X	114	190	190	182	2.2
E94_120Y2~_X	68	190	235	182	1.5
E94_180T2~_X	68	242	235	233	2.0
E94_020T4N_X	68	190	190	182	1.5
E94_040T4N_X	95	190	190	182	1.9
E94_050T4N_X	114	190	190	182	2.2
E94_060T4~_X	68	190	235	182	1.4
E94_090T4~_X	68	242	235	233	2.0

- (1) The first "_" equals "P" for the Model 940 encoder based drive or "R" for the Model 941 resolver based drive. When the 10th digit is marked by "-", "N" = No line filter, "F" = Integrated line filter or "C" = Cold plate drive. The second "_" equals "E" for incremental encoder (must have E94P drive) or "R" for the standard resolver (must have E94R drive).



2.9 Clearance for Cooling Air Circulation



S924



Installation

3 Installation

Perform the minimum system connection. Please refer to section 6.1 for minimum connection requirements. Observe the rules and warnings below carefully:



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.



STOP!

- The PositionServo must be mounted vertically for safe operation and to ensure enough cooling air circulation.
- Printed circuit board components are sensitive to electrostatic fields. Avoid contact with the printed circuit board directly. Hold the PositionServo by its case only.
- Protect the drive from dirt, filings, airborne particles, moisture, and accidental contact. Provide sufficient room for access to the terminal block.
- Mount the drive away from any and all heat sources. Operate within the specified ambient operating temperature range. Additional cooling with an external fan may be recommended in certain applications.
- Avoid excessive vibration to prevent intermittent connections
- DO NOT connect incoming (mains) power to the output motor terminals (U, V, W)! Severe damage to the drive will result.
- Do not disconnect any of the motor leads from the PositionServo drive unless (mains) power is removed. Opening any one motor lead may cause failure.
- Control Terminals provide basic isolation (insulation per EN 61800-5-1). Protection against contact can only be ensured by additional measures, e.g., supplemental insulation.
- Do not cycle mains power more than once every 2 minutes. Otherwise damage to the drive may result.



WARNING!

For compliance with EN 61800-5-1, the following warning applies.

This product can cause a d.c. current in the protective earthing conductor. Where a residual current-operated protective (RCD) or monitoring (RCM) device is used for protection in case of direct or indirect contact, only an RCD or RCM of Type B is allowed on the supply side of this product.



UL INSTALLATION INFORMATION

- Suitable for use on a circuit capable of delivering not more than 200,000 rms symmetrical amperes, at the maximum voltage rating marked on the drive.
- Use Class 1 wiring with minimum of 75°C copper wire only.
- Shall be installed in a pollution degree 2 macro-environment.



3.1 Wiring



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing the drive. Capacitors retain charge after power is removed.



WARNING!

Leakage current may exceed 3.5mA AC. Minimum size of the protective earth conductor shall comply with local safety regulations for high leakage current equipment.



STOP!

Under no circumstances should power and control wiring be bundled together. Induced voltage can cause unpredictable behavior in any electronic device, including motor controls.

Refer to section 4.1.1 for power wiring specifications.

3.2 Shielding and Grounding

3.2.1 General Guidelines

Lenze recommends the use of single-point grounding (SPG) for panel-mounted controls. Serial grounding (a “daisy chain”) is not recommended. The SPG for all enclosures must be tied to earth ground at the same point. The system ground and equipment grounds for all panel-mounted enclosures must be individually connected to the SPG for that panel using 14 AWG (2.5 mm²) or larger wire.

In order to minimize EMI, the chassis must be grounded to the mounting. Use 14 AWG (2.5 mm²) or larger wire to join the enclosure to earth ground. A lock washer must be installed between the enclosure and ground terminal. To ensure maximum contact between the terminal and enclosure, remove paint in a minimum radius of 0.25 in (6 mm) around the screw hole of the enclosure.

Lenze recommends the use of the special PositionServo drive cables provided by Lenze. If you specify cables other than those provided by Lenze, please make certain all cables are shielded and properly grounded.

It may be necessary to earth ground the shielded cable. Ground the shield at both the drive end and at the motor end.

If the PositionServo drive continues to pick up noise after grounding the shield, it may be necessary to add an AC line filtering device and/or an output filter (between the drive and servo motor).



Installation

EMC

Compliance with EN 61800-3:2004

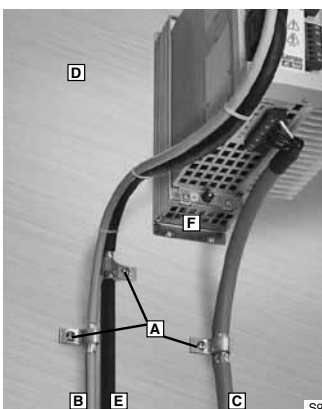
In a domestic environment this product may cause radio interference. The user may be required to take adequate measures

Noise emission

Drive Models ending in the suffix "2F" are in compliance with class A limits according to EN 55011 if installed in a control cabinet and the motor cable length does not exceed 10m. Models ending in "N" will require an appropriate line filter.

- A** Screen clamps
- B** Control cable
- C** Low-capacitance motor cable
(core/core < 75 pF/m, core/screen < 150 pF/m)
- D** Earth grounded conductive mounting plate
- E** Encoder Feedback Cable
- F** Footprint or Sidemount Filter (optional)

Installation according to EMC Requirements



3.2.2 EMI Protection

Electromagnetic interference (EMI) is an important concern for users of digital servo control systems. EMI will cause control systems to behave in unexpected and sometimes dangerous ways. Therefore, reducing EMI is of primary concern not only for servo control manufacturers such as Lenze, but the user as well. Proper shielding, grounding and installation practices are critical to EMI reduction.

3.2.3 Enclosure

The panel in which the PositionServo is mounted must be made of metal, and must be grounded using the SPG method outlined in section 3.2.1.

Proper wire routing inside the panel is critical; power and logic leads must be routed in different avenues inside the panel.

You must ensure that the panel contains sufficient clearance around the drive. Refer to section 2.9 for the recommended cooling air clearance.



3.3 Line Filtering

In addition to EMI/RFI safeguards inherent in the PositionServo design, external filtering may be required. High frequency energy can be coupled between the circuits via radiation or conduction. The AC power wiring is one of the most important paths for both types of coupling mechanisms. In order to comply with IEC61800-3:2004, an appropriate filter must be installed within 20cm of the drive power inputs.

Line filters should be placed inside the shielded panel. Connect the filter to the incoming power lines immediately after the safety mains and before any critical control components. Wire the AC line filter as close as possible to the PositionServo drive.



NOTE

The ground connection from the filter must be wired to solid earth ground, not machine ground.

If the end-user is using a CE-approved motor, the AC filter combined with the recommended motor and encoder cables, is all that is necessary to meet the EMC directives listed herein. The end user must use the compatible filter to comply with CE specifications. The OEM may choose to provide alternative filtering that encompasses the PositionServo drive and other electronics within the same panel. The OEM has this liberty because CE requirements are for the total system.

3.4 Heat Sinking

The PositionServo drive contains sufficient heat sinking within the specified ambient operating temperature in its basic configuration. There is no need for additional heat sinking. However, the user must ensure that there is sufficient clearance for proper air circulation. As a minimum, an air gap of 25 mm above and below the drive is necessary.

3.5 Line (Mains) Fusing

External line fuses must be installed on all PositionServo drives. Connect the external line fuse in series with the AC line voltage input. Use fast-acting fuses rated for 250 VAC or 600 VAC (depending on model), and approximately 200% of the maximum RMS phase current. Refer to section 2.3 for fuse recommendations.



4 Interface

The standard PositionServo drive contains seven connectors: four quick-connect terminal blocks, one SCSI connector and one subminiature type “D” connector. These connectors provide communications from a PLC or host controller, power to the drive, and feedback from the motor. Prefabricated cable assemblies may be purchased from Lenze to facilitate wiring the drive, motor and host computer. Contact your Lenze Sales Representative for assistance.

As this manual makes reference to specific pins on specific connectors, we will use the convention PX.Y where X is the connector number and Y is the pin number.

4.1 External Connectors

4.1.1 P1 & P7 - Input Power and Output Power Connections

P1 is a 3 or 4-pin quick-connect terminal block used for input (mains) power. P7 is a 6-pin quick-connect terminal block used for output power to the motor. P7 also has a thermistor (PTC) input for motor over-temperature protection. The tables in this section identify the connector pin assignments.

**DANGER!**

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.

**STOP!**

DO NOT connect incoming power to the output motor terminals (U, V, W)! Severe damage to the PositionServo will result.

Check phase wiring (U, V, W) and thermal input (T1, T2) before powering up drive. If miswired, severe damage to the PositionServo will result.

All conductors must be enclosed in one shield with a jacket around them. The shield on the drive end of the motor power cable should be terminated to the conductive machine panel using screen clamps as shown in section 3.2. The other end should be properly terminated at the motor shield. Feedback cable shields should be terminated in a like manner. Lenze recommends Lenze cables for both the motor power and feedback. These are available with appropriate connectors and in various lengths. Contact your Lenze representative for assistance.

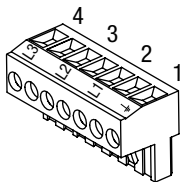
Wire Size

Current A (rms)	Terminal Torque (lb-in)	Wire Size
$I \leq 8$	4.5	16 AWG (1.5mm ²) or 14 AWG (2.5mm ²)
$8 < I \leq 12$	4.5	14 AWG (2.5mm ²) or 12 AWG (4.0mm ²)
$12 < I \leq 15$	4.5	12 AWG (4.0mm ²)
$15 < I \leq 20$	5.0 - 7.0	10 AWG (6.0mm ²)
$20 < I \leq 24$	11.0 - 15.0	10 AWG (6.0mm ²)



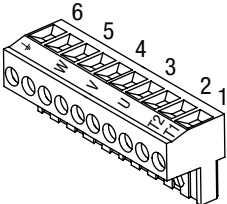
P1 Pin Assignments (Input Power)

Pin	Standard Models		Doubler Models	
	Name	Function	Name	Function
1	PE	Protective Earth (Ground)	PE	Protective Earth (Ground)
2	L1	AC Power in	N	AC Power Neutral (120V Doubler only)
3	L2	AC Power in	L1	AC Power in
4	L3	AC Power in (3~ models only)	L2/N	AC Power in (non-doubler operation)



P7 Pin Assignments (Output Power)

Pin	Terminal	Function
1	T1	Thermistor (PTC) Input
2	T2	Thermistor (PTC) Input
3	U	Motor Power Out
4	V	Motor Power Out
5	W	Motor Power Out
6	PE	Protective Earth (Chassis Ground)

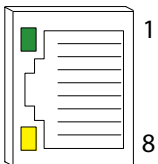


4.1.2 P2 - Ethernet Communications Port

P2 is a RJ45 Standard Ethernet connector that is used to communicate with a host computer via Ethernet TCP/IP.

P2 Pin Assignments (Communications)

Pin	Name	Function
1	+ TX	Transmit Port (+) Data Terminal
2	- TX	Transmit Port (-) Data Terminal
3	+ RX	Receive Port (+) Data Terminal
4	N.C.	
5	N.C.	
6	- RX	Receive Port (-) Data Terminal
7	N.C.	
8	N.C.	




NOTE

To communicate from the PC directly to the drive a crossover cable is required. If using a hub or switch, use a regular patch cable.



Interface

4.1.3 P3 - Controller Interface

P3 is a 50-pin SCSI connector for interfacing to the front-end of the controllers. It is strongly recommended that you use OEM cables to aid in satisfying CE requirements. Contact your Lenze representative for assistance.

P3 Pin Assignments (Controller Interface)

Pin	Name	Function	Connector
1	MA+	Master Encoder A+ / Step+ input ⁽²⁾	
2	MA-	Master Encoder A- / Step- input ⁽²⁾	
3	MB+	Master Encoder B+ / Direction+ input ⁽²⁾	
4	MB-	Master Encoder B- / Direction- input ⁽²⁾	
5	GND	Drive Logic Common	
6	5+	+5V output (max 100mA)	
7	BA+	Buffered Encoder Output: Channel A+ ⁽¹⁾	
8	BA-	Buffered Encoder Output: Channel A- ⁽¹⁾	
9	BB+	Buffered Encoder Output: Channel B+ ⁽¹⁾	
10	BB-	Buffered Encoder Output: Channel B- ⁽¹⁾	
11	BZ+	Buffered Encoder Output: Channel Z+ ⁽¹⁾	
12	BZ-	Buffered Encoder Output: Channel Z- ⁽¹⁾	
13-19		Empty	
20	AIN2+	Positive (+) of Analog signal input	
21	AIN2-	Negative (-) of Analog signal input	
22	ACOM	Analog common	
23	AO	Analog output (max 10 mA)	
24	AIN1+	Positive (+) of Analog signal input	
25	AIN1 -	Negative (-) of Analog signal input	
26	IN_A_COM	Digital input group ACOM terminal ⁽³⁾	
27	IN_A1	Digital input A1	
28	IN_A2	Digital input A2	
29	IN_A3	Digital input A3 ⁽³⁾	
30	IN_A4	Digital input A4	
31	IN_B_COM	Digital input group BCOM terminal	
32	IN_B1	Digital input B1	
33	IN_B2	Digital input B2	
34	IN_B3	Digital input B3	
35	IN_B4	Digital input B4	
36	IN_C_COM	Digital input group CCOM terminal	
37	IN_C1	Digital input C1	
38	IN_C2	Digital input C2	
39	IN_C3	Digital input C3	
40	IN_C4	Digital input C4	
41	RDY+	Ready output Collector	
42	RDY-	Ready output Emitter	
43	OUT1-C	Programmable output #1 Collector	
44	OUT1-E	Programmable output #1 Emitter	
45	OUT2-C	Programmable output #2 Collector	
46	OUT2-E	Programmable output #2 Emitter	
47	OUT3-C	Programmable output #3 Collector	
48	OUT3-E	Programmable output #3 Emitter	
49	OUT4-C	Programmable output #4 Collector	
50	OUT4-E	Programmable output #4 Emitter	

⁽¹⁾ See Note 1, Section 4.1.7 - Connector and Wiring Notes

⁽²⁾ See Note 2, Section 4.1.7 - Connector and Wiring Notes

⁽³⁾ See Note 3, Section 4.1.7 - Connector and Wiring Notes



4.1.4 P4 - Motor Feedback / Second Loop Encoder Input

For encoder-based 940 drives, P4 is a 15-pin DB connector that contains connections for an incremental encoder with Hall emulation tracks or Hall sensors. For synchronous servo motors, Hall sensors or Hall emulation tracks are necessary for commutation. If an asynchronous servo motor is used, it is not necessary to connect Hall sensor inputs. Encoder inputs on P4 have 26LS32 or compatible differential receivers for increased noise immunity. Inputs have all necessary filtering and line balancing components so no external noise suppression networks are needed.

For resolver-based 941 drives, P4 is a 9-pin DB connector for connecting resolver feedback and thermal sensor. For pin assignments, refer to the table P4B. The resolver feedback is translated to 65,536 counts per revolution.

All conductors must be enclosed in one shield with a jacket around them. Lenze recommends that each and every pair (for example, EA+ and EA-) be twisted. In order to satisfy CE requirements, use of an OEM cable is recommended.

The PositionServo buffers encoder/resolver feedback from P4 to P3. For example, when encoder feedback is used, channel A on P4, is Buffered Encoder Output channel A on P3. For more information on this refer to section 4.2.2 "Buffered Encoder Outputs".



STOP!

Use only +5 VDC encoders. Do not connect any other type of encoder to the PositionServo reference voltage terminals. When using a front-end controller, it is critical that the +5 VDC supply on the front-end controller NOT be connected to the PositionServo's +5 VDC supply, as this will result in damage to the PositionServo.



NOTE

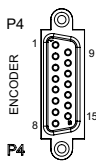
- The PositionServo encoder inputs are designed to accept differentially driven hall signals. Single-ended or open-collector type hall signals are also acceptable by connecting "HA+", "HB+", "HC+" and leaving "HA-, HB-, HC-" inputs unconnected. The user does not need to supply pull-up resistors for open-collector hall sensors. The necessary pull-up circuits are already provided.
- Encoder connections (A, B and Z) must be full differential. The PositionServo does not support single-ended or open-collector type outputs from the encoder.
- An encoder resolution of 2000 PPR (pre-quadrature) or higher is recommended.

Using P4 as second encoder input for dual-loop operation:

P4 can be used as a second loop encoder input in situations where the motor is equipped with a resolver as the primary feedback. If such a motor is used, the drive must have a resolver feedback option module installed. A second encoder can then be connected to the A and B lines of the P4 connector for dual loop operation. Refer to section 6.4 ("Dual-loop Feedback Operation") for details.

P4A Pin Assignments (Encoder Feedback - E94P Drives)

Pin	Name	Function	Pin	Name	Function
1	EA+	Encoder Channel A+ Input ⁽¹⁾	9	PWR	Encoder supply (+5VDC)
2	EA-	Encoder Channel A- Input ⁽¹⁾	10	HA-	Hall Sensor A- Input ⁽²⁾
3	EB+	Encoder Channel B+ Input ⁽¹⁾	11	HA+	Hall Sensor A+ Input ⁽²⁾
4	EB-	Encoder Channel B- Input ⁽¹⁾	12	HB+	Hall Sensor B+ Input ⁽²⁾
5	EZ+	Encoder Channel Z+ Input ⁽¹⁾	13	HC+	Hall Sensor C+ Input ⁽²⁾
6	EZ-	Encoder Channel Z- Input ⁽¹⁾	14	HB-	Hall Sensor B- Input ⁽²⁾
7	GND	Drive Logic Common/Encoder GND	15	HC-	Hall Sensor C- Input ⁽²⁾
8	SHLD	Shield			



(1) See Note 1, Section 4.1.7 - Connector and Wiring Notes

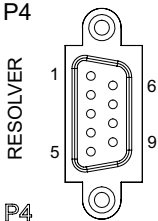
(2) For asynchronous servo motor, an incremental encoder without Hall effect sensors (commutation tracks) can be used.



Interface

P4B Pin Assignments (Resolver Feedback - E94R Drives)

Pin	Name	Function
1	Ref +	Resolver reference connection
2	Ref -	
3	N/C	No Connection
4	Cos+	Resolver Cosine connections
5	Cos-	
6	Sin+	Resolver Sine connections
7	Sin-	
8	PTC+	Motor PTC Temperature Sensor
9	PTC-	

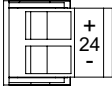


4.1.5 P5 - 24 VDC Back-up Power Input

P5 is a 2-pin quick-connect terminal block that can be used with an external 24 VDC (500mA) power supply to provide “Keep Alive” capability: during a power loss, the logic and communications will remain active. Applied voltage must be greater than 20VDC.

P5 Pin Assignments (Back-up Power)

Pin	Name	Function
1	+24 VDC	Positive 24 VDC Input
2	Return	24V power supply return




WARNING!

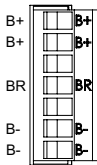
Hazard of unintended operation! The “Keep Alive” circuit will restart the motor upon restoration of mains power when the enable input remains asserted. If this action is not desired, then the enable input must be removed prior to re-application of input power.

4.1.6 P6 - Braking Resistor and DC Bus

P6 is a 5-pin quick-connect terminal block that can be used with an external braking resistor (the PositionServo has the regen circuitry built-in). The Brake Resistor connects between the Positive DC Bus (either P6.1 or 2) and P6.3.

P6 Terminal Assignments (Brake Resistor and DC Bus)

Pin	Terminal	Function
1	B+	Positive DC Bus / Brake Resistor
2	B+	
3	BR	Brake Resistor
4	B-	Negative DC Bus
5	B-	




DANGER!

Hazard of electrical shock! Circuit potentials are up to 680 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing the drive. Capacitors retain charge after power is removed.



4.1.7 Connector and Wiring Notes

Note 1 - Buffered Encoder Inputs

Each of the encoder output pins on P3 is a buffered pass-through of the corresponding input signal on P4. Refer to section 4.2.2 “Buffered Encoder Outputs”. This can be either from a motor mounted encoder/resolver, (primary feedback), or from an auxiliary encoder/resolver when an optional feedback module is used.

Via software, these pins can be re-programmed to be a buffered pass through of the signals from a feedback option card. This can be either the second encoder option module (E94ZAENC1) or an encoder emulation of the resolver connected to the resolver option module (E94ZARSV2 or E94ZARSV3).

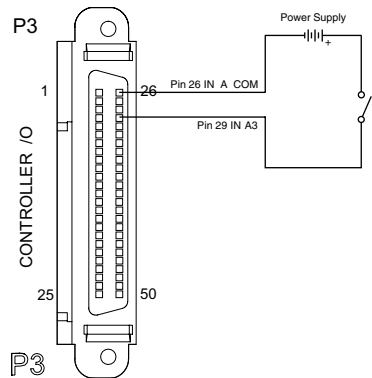
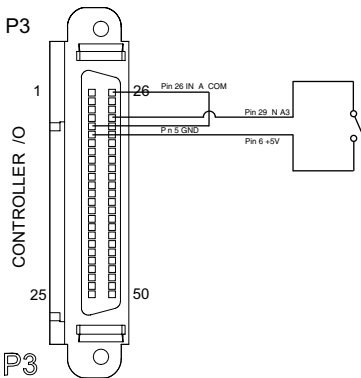
Note 2 - Master Encoder Inputs or Step/Direction Inputs

An external pulse train signal (“step”) supplied by an external device, such as a PLC or stepper indexer, can control the speed and position of the servomotor. The speed of the motor is controlled by the frequency of the “step” signal, while the number of pulses that are supplied to the PositionServo determines the position of the servomotor. Direction input controls direction of the motion.

Note 3 - Digital Input A3

For the drive to function, an ENABLE input must be wired to the drive, and should be connected to IN_A3, (P3.29), which is, by the default the ENABLE input on the drive. This triggering mechanism can either be a switch or an input from an external PLC or motion controller. The input can be wired either sinking or sourcing (section 4.2.3). The Enable circuit will accept 5-24V control voltage.

Wiring the Enable Input:





Interface

4.1.8 P11 - Resolver Interface Module (option)

PositionServo drives can operate motors equipped with resolvers from either the (P4) connection, for a resolver-based (E94R) drive, or from the Resolver option module for an encoder-based (E94P) drive. The option module connections are made to a 9 pin D-shell female connector (P11) on the resolver option module E94ZARSV2 (scalable) or E94ZARSV3 (standard). When the motor profile is loaded from the motor database or from a custom motor file, the drive will select the primary feedback source based on the motor data entry.

The E94ZARSV3 has a fixed resolution of 1024 PPR prequadrature or 4096 postquadrature. The E94ZARSV2 has a selectable set of 15 resolutions. The resolution refers to the pulses per revolution (PPR) of the Buffered Encoder Outputs (P3-7 to P3-12) if the Encoder Repeat Source is set as “Optional Feedback Input” in MotionView.

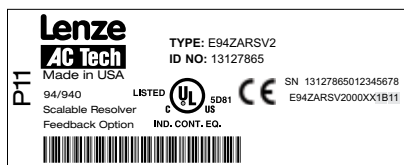
When using the E94ZARSV2, the default resolution is 1024 PPR prequadrature. Depending on the hardware/software revision of the E94ZARSV2 module, the available PPRs are different. Refer to the table below for the Dip Switch settings for SW1 and the different resolutions.

SW1 DIP Switch Settings

Dip Switch SW1				PPR prequadrature ⁽¹⁾	
Position 1	Position 2	Position 3	Position 4	Hardware/Software Revision ⁽²⁾ 1A10, 1A11, 1B11, 1C11	Hardware/Software Revision ⁽²⁾ 1C12 and higher
OFF	OFF	OFF	OFF	250	1024 (default)
OFF	OFF	OFF	ON	256	256
OFF	OFF	ON	OFF	360	360
OFF	OFF	ON	ON	400	400
OFF	ON	OFF	OFF	500	500
OFF	ON	OFF	ON	512	512
OFF	ON	ON	OFF	720	720
OFF	ON	ON	ON	800	800
ON	OFF	OFF	OFF	1000	1000
ON	OFF	OFF	ON	1024 (default)	1024 (default)
ON	OFF	ON	OFF	2000	2000
ON	OFF	ON	ON	2048	2048
ON	ON	OFF	OFF	2500	2500
ON	ON	OFF	ON	2880	2880
ON	ON	ON	OFF	4096	250
ON	ON	ON	ON	4096	4096

(1) For PPR postquadrature, multiply by 4.

(2) Hardware/Software Revision can be found on the dataplate label attached to the plastic cover of the module. For example, the revision in the example below is 1B11.





Setting the Dip Switches

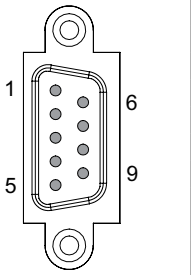
To change the DIP SWITCH SETTING

1. Loosen the three set screws on the module
2. Detach the PCB board from the plastic cover
3. Change the SW1 positions according to the table above
4. Put the PCB board back in the plastic cover
5. Tighten the three set screws

When using a Lenze motor with resolver feedback and a Lenze resolver cable, the pins are already configured for operation. If a non-Lenze motor is used, the resolver connections are made as follows:

P11 Pin Assignments (Resolver Feedback)

Pin	Name	Function
1	Ref +	Resolver reference connection
2	Ref -	
3	N/C	No Connection
4	Cos+	Resolver Cosine connections
5	Cos-	
6	Sin+	Resolver Sine connections
7	Sin-	
8	PTC+	Motor PTC Temperature Sensor
9	PTC-	




STOP!

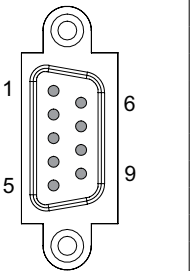
Use only 10 V (peak to peak) or less resolvers. Use of higher voltage resolvers may result in feedback failure and damage to the resolver option module.

4.1.9 P12 - Second Encoder Interface Module (option)

PositionServo drives can support a second incremental encoder interface for dual-loop systems. Regardless of whether the motor's primary feedback type is an encoder or resolver, a 2nd Encoder Option Module, E94ZAENC1, can be installed at Option Bay 2, (P12). Once installed the optional feedback card can be selected as the primary encoder repeat source from the "Parameter" folder in MotionView. The 2nd Encoder Option Module includes a 9 pin D-shell male connector. When using a Lenze motor with encoder feedback and a Lenze encoder cable, the pins are already configured for operation. If a non-Lenze motor is used, the encoder connections are made as follows:

P12 Pin Assignments (Second Encoder Feedback)

Pin	Name	Function
1	E2B+	Second Encoder Channel B+ Input
2	E2A-	Second Encoder Channel A- Input
3	E2A+	Second Encoder Channel A+ Input
4	+5v	Supply voltage for Second Encoder
5	COM	Supply common
6	E2Z-	Second Encoder Channel Z- Input
7	E2Z+	Second Encoder Channel Z+ Input
8	N/C	No Connection
9	E2B-	Second Encoder Channel B- Input



The second encoder needs to be enabled using MotionView software. (section 6.4).



STOP!

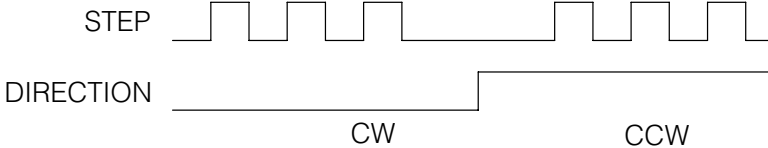
Use only +5 VDC encoders. Do not connect any other type of encoder to the option module. Otherwise, damage to drive's circuitry may result.



4.2 Digital I/O Details

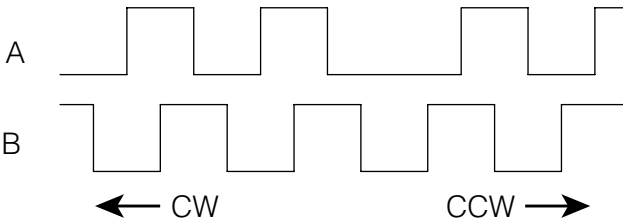
4.2.1 Step & Direction / Master Encoder Inputs (P3, pins 1-4)

You can connect a master encoder with quadrature outputs or a step and direction pair of signals to control position in step / direction operating mode (stepper motor emulation). These inputs are optically isolated from the rest of the drive circuits and from each other. Both inputs can operate from any voltage source in the range of 5 to 24 VDC and do not require additional series resistors for normal operation.



Timing characteristics for Step And Direction signals

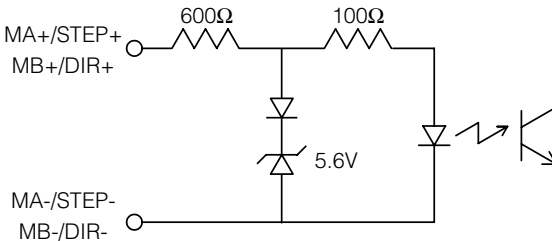
S904



S905

Timing characteristics for Master Encoder signals

Input type/ output compatibility	Insulated, compatible with Single-ended or differential outputs (5-24 VDC)
Max frequency (per input)	2 MHz
Min pulse width (negative or positive)	500nS
Input impedance	700 Ω (approx)



S906

Master encoder/step and direction input circuit

Differential signal inputs are preferred when using Step and Direction. Single ended inputs can be used but are not recommended. Sinking or sourcing outputs may also be connected to these inputs. The function of these inputs “Master Encoder” or “Step and Direction” is software selectable. Use the MotionView set up program to choose the desirable function.



4.2.2 Buffered Encoder Output (P3, pins 7-12)

There are many applications where it is desired to close the feedback loop to an external device. This feature is built into the PositionServo drive and is referred to as the “Buffered Encoder Output”. If a motor with encoder feedback is being used, the A+, A-, B+, B-, Z+ and Z- signals are directly passed through the drive through pins 7-12 with no delays, up to a speed of 25MHz. If a motor with resolver feedback is being used a simulated encoder feedback is transmitted. The default resolution of the simulated encoder is 1024 pulses per revolution, pre-quad. If a different resolution is desired refer to section 5.3.22 “Resolver Tracks”.

4.2.3 Digital Outputs

There are a total of five digital outputs (“OUT1” - “OUT4” and “RDY”) available on the PositionServo drive. These outputs are accessible from the P3 connector. Outputs are open collector type that are fully isolated from the rest of the drive circuits. See the following figure for the electrical diagram. These outputs can be either used via the drives internal User Program or they can be configured as Special Purpose outputs. When used as Special Purpose, each output (OUT1-OUT4) can be assigned to one of the following functions:

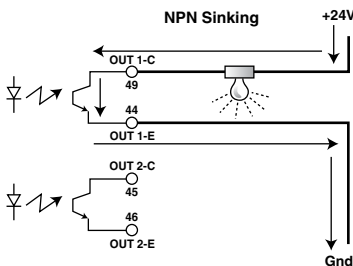
- Not assigned
- Zero speed
- In-speed window
- Current limit
- Run-time fault
- Ready
- Brake (motor brake release)

Please note that if you assign an output as a Special Purpose Output then that output can not be utilized by the User Program. The “RDY” Output has a fixed function, “ENABLE”, which will become active when the drive is enabled and the output power transistors becomes energized.

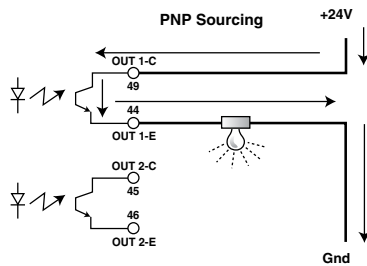
Digital outputs electrical characteristics

Circuit type	Isolated Open Collector
Digital outputs load capability	15mA
Digital outputs Collector-Emitter max voltage	30V

The inputs on drive can be wired as either sinking (NPN) or sourcing (PNP), as illustrated in wiring examples mb101 and mb102.



mb101



mb102



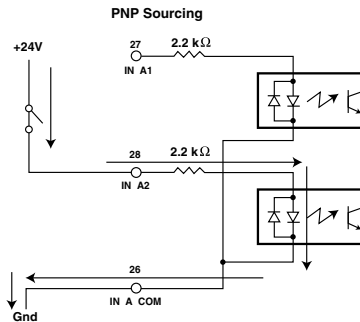
Interface

4.2.4 Digital Inputs

IN_Ax, IN_Bx, IN_Cx (P3.26-30, P3.31-35, P3.36-40)

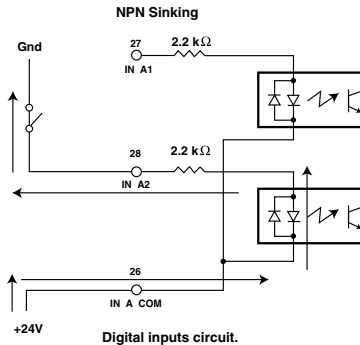
The PositionServo Drive has 12 optically isolated inputs. These inputs are compatible with a 5 - 24V voltage source. No additional series resistors are needed for circuit operation. The 12 inputs are segmented into three groups of 4, Inputs A1 - A4, Inputs B1 - B4, and Inputs C1 - C4. Each group, (A, B and C) have their own corresponding shared common, (ACOM, BCOM and CCOM). Each group or bank can be wired as sinking or sourcing. Refer to wiring examples mb103 and mb104. All inputs have a separate software adjustable de-bounce time. Some of the inputs can be set up as Special Purpose Inputs. For example inputs A1 and A2 can be configured as limit inputs, input A3 is always set up as an Enable input and input C3 can be used as a registration input. Refer to the PositionServo Programming Manual (PM94P01) for more detail.

For the registration input (C3), the registration time is $3\mu\text{s}$ for an encoder and $7\mu\text{s}$ for a resolver.



Digital inputs circuit.

mb103



Digital inputs circuit.

mb104



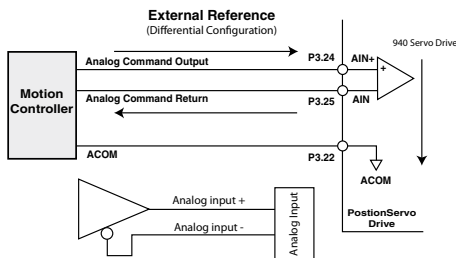
4.3 Analog I/O Details

4.3.1 Analog Reference Input

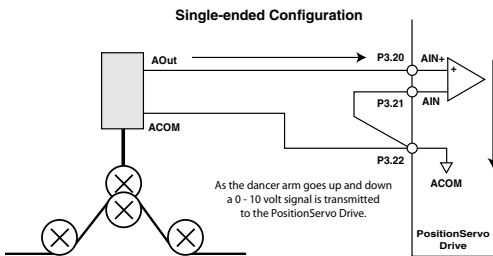
AIN1+, AIN1- (P3.24 and P3.25)

The analog reference input can accept up to a $\pm 10\text{V}$ analog signal across AIN1+ and AIN1-. The maximum limit with respect to analog common (ACOM) on each input is $\pm 18\text{VDC}$. The analog signal will be converted to a digital value with 16 bit resolution (15 bit plus sign). This input is used to control speed or torque of the motor in velocity or torque mode. The total reference voltage as seen by the drive is the voltage difference between AIN1+ and AIN1-. If used in single-ended mode, one of the inputs must be connected to Analog Common (ACOM). If used in differential mode, the voltage source is connected across AIN1+ and AIN1- and the driving circuit common (if any) needs to be connected to the drive Analog Common (ACOM) terminal. Refer to wiring examples mb105 and mb106.

Reference as seen by drive: $V_{ref} = (AIN1+) - (AIN1-)$ and $-10\text{V} \leq V_{ref} \leq +10\text{V}$



mb105



mb106

AIN2+, AIN2- (P3.20 and P3.21)

The analog reference input can accept up to a $\pm 10\text{V}$ analog signal across AIN2+ and AIN2-. The maximum limit with respect to analog common (ACOM) on each input is $\pm 18\text{VDC}$. The analog signal will be converted to a digital value with 10 bit resolution (9 bit plus sign). This input is available to the User's program. This input does not have a predefined function. Scaling of this input is identical to AIN1.



Interface

4.3.2 Analog Output

AO (P3.23)

The analog output is a single-ended signal (with reference to Analog Common (ACOM)) which can represent the following motor data:

- Not Assigned
- Phase R Current
- Phase S Current
- Phase T Current
- Motor Velocity
- Iq Current
- Id Current

Motor phase U, V and W corresponds to R, S and T respectively.

MotionView Setup program can be used to select the signal source for the analog output as well as its scaling.

If the output function is set to “Not Assigned” then the output can be controlled directly from user’s program. Refer to the PositionServo Programming Manual (PM94P01) for programming details.



STOP!

Upon application of power to the PositionServo, the Analog Output supplies -10VDC until bootup is complete. Once bootup is complete, the Analog Output will supply the commanded voltage.

4.4 Communication Interfaces

4.4.1 Ethernet Interface (standard)

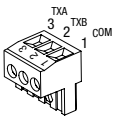
Programming and diagnostics of the drive are performed over the standard Ethernet communication port. The Drives IP address can be displayed from the drive’s front panel display. The last octet of the drive’s IP address can be set from the drive’s display. Changes do not take affect until after a power cycle. The interface supports both 100 BASE-TX as well as 10 BASE-T. This configuration allows the user to monitor and program multiple drives from MotionView. Refer to section 5.4.1 for PC configuration information.

4.4.2 RS485 Interface (option)

PositionServo drives can be equipped with an RS485 communication interface option module (E94ZARS41) which is optically isolated from the rest of the drive’s circuitry. This option module can be used for two functions: drive programming and diagnostics using MotionView from a PC (with RS485 port) or as a Modbus RTU slave. The PositionServo drives support 5 different baud rates, ranging from 9600 to 115200. Drives are addressable with up to 32 addresses from 0-31. The factory setting for the baud rate is 38,400 with a node address of “1”. The drives address must be set from the front panel display of the drive. When used with MotionView software, the communication speed is also set from the front panel display. If used for Modbus RTU communications, the Modbus baud rate is set as a parameter within MotionView.

Pin Assignments (RS485 interface)

Pin	Name	Function
1	ICOM	Isolated Common
2	TXB	Transmit B(+)
3	TXA	Transmit A(-)





4.4.3 RS485 Communication Setup

When establishing communication between MotionView and a PositionServo drive, a communication method must be selected. The connection choice can be either “UPP over RS485/RS232” or “Ethernet”. The “UPP over RS485/RS232” selection establishes a RS485 connection between MotionView and the first drive on the network. Multiple drives can then be added to the network via RS485. Each drive on the network must have a different Node Address. When setting up communications the node address of the target drive must be set. MotionView will then send out a communications packet to the drives on the network, via the RS485 connection. The message, “Device with address # not present in the network” will appear if the target node could not be found.

4.4.4 MODBUS RTU Support

As a default, the Ethernet and RS485 interfaces are configured to support MotionView program operations. In addition, the Ethernet port can support MODBUS TCP/IP slave protocols and the RS485 interface can be configured to support the MODBUS RTU slave protocol. These interfaces are configured through the MotionView program environment. When configured for MODBUS operation, the baud rate for RS485 is set by the parameter “Modbus baud rate” in MotionView. MODBUS RTU requires 8 data bits. The MODBUS RTU slave interface protocol definitions can be found in the MotionView help menu under “Product Manuals”.

4.4.5 CAN Interface (option)

An optional CANopen communication module (E94ZACAN1) is available for the PositionServo drive. Installed in Option Bay 1 as P21, the CANopen module is optically isolated from the rest of the drive’s circuitry. The 3-pin CANopen module is for HW/SW 1A10 and the 5-pin CANopen module is for HW/SW 1B10 or higher. Refer to the PS CANopen Reference Guide (P94CAN01) for more information.

CANopen Interface Pin Assignments

3-Terminal	Pin	Name	Function	Pin	Name	Function	5-Terminal
	1	ICOM	Isolated Common	1	ICOM	Isolated Common	
	2	CAN L	CAN Bus Low	2	CAN L	CAN Bus Low	
	3	CAN H	CAN Bus High	3	Shield		
				4	CAN H	CAN Bus High	
				5	NC	No connection	

4.5 Motor Selection

The PositionServo drive is compatible with many 3-phase AC synchronous servo motors as well as 3-phase AC asynchronous servo motors. MotionView is equipped with a motor database that contains over 600 motors for use with the PositionServo drive. If the desired motor is in the database, no data to set it up is needed. Just select the motor and click “OK”. However, if your motor is not in the database, it can still be used, but some electrical and mechanical data must be provided to create a custom motor profile. The auto-phasing feature of the PositionServo drive allows the user to correctly determine the relationship between phase voltage and hall sensor signals, eliminating the need to use a multi-channel oscilloscope.

4.5.1 Motor Connection

Motor phase U, V, W (or R, S, T) are connected to terminal P7. It is very important that motor cable shield is connected to Earth ground terminal (PE) or the drive’s case. The motor’s encoder/resolver feedback cable must be connected to terminal P4. If a resolver option module is used, connect to terminal P11, and if a second encoder option module is used, connect to terminal P12.



Interface

4.5.2 Motor Over-temperature Protection

If using a motor equipped with an encoder and PTC thermal sensor, the encoder feedback cable will have flying leads exiting the P4 connector to be wired to the P7.1 (T1) and P7.2 (T2) terminals. If using a motor equipped with a Resolver and a PTC sensor, the thermal feedback is passed directly to the drive via the resolver 9-pin D shell connector.

Use parameter “Motor PTC cut-off resistance” (refer to section 5.3.12) to set the resistance that corresponds to maximum motor allowed temperature. The parameter “Motor temperature sensor” must also be set to ENABLE. If the motor doesn’t have a PTC sensor, set this parameter to DISABLE. This input will also work with N.C. thermal switches which have only two states; Open or Closed. In this case “Motor PTC cut-off resistance” parameter can be set to the default value.

4.5.3 Motor Setup

Once you are connected to the PositionServo via MotionView a “Parameter Tree” will appear in the “Parameter Tree Window”. The various parameters of the drive are shown here as folders and files. If the “Motor” folder is selected, all motor parameters can be viewed in the “Parameter View Window”. To view selected motor parameters or to select a new motor click the section marked “CLICK HERE TO CHANGE”.



S911

MotionView’s “Motor Group” folder and its contents



NOTE

If the drive is ENABLED, a new motor cannot be set. You can only set a new motor when the drive is DISABLED.

To View selected motor parameters or to make a new motor selection:

- Click “Click here to change the motor” from the Parameter View Window (see figure above). If just viewing the motor parameters click Cancel on the Motor Parameters dialog box when done to dismiss the box.
- Select motor Vendor from the right list box and desired motor from the left list box.
- If you will be using a “custom” motor (not listed in our motor database) go to “Using a custom motor” topic in the next section.
- Finally, click the OK button to dismiss the dialog box and return to MotionView’s main program.



NOTE

To help prevent the motor from drawing too much current and possibility overheating it is recommended that the drives “Current Limit” be checked against the motors “Nominal Phase Current” and set accordingly.



4.6 Using a Custom Motor

You can load a custom motor from a file or you can create a new custom motor.

- To create a custom motor click “CREATE CUSTOM” and follow the instructions in the next section “Creating custom motor parameters”.
- To load a custom motor click “OPEN CUSTOM” button then select the motor file and click the “OPEN” button to select or click the “CANCEL” button to return to the previous dialog box.
- Click OK to load the motor data and return to the main MotionView menu or Cancel to abandon changes. When clicking OK for a custom motor, a dialog box will appear asking if you want to execute “Autophasing” (refer to section 4.6.2).

4.6.1 Creating Custom Motor Parameters



STOP!

Use extreme caution when entering custom parameters! Incorrect settings may cause damage to the drive or motor! If you are unsure of the settings, refer to the materials that were distributed with your motor, or contact the motor manufacturer for assistance.

1. Enter custom motor data in the Motor Parameters dialog fields. Complete all sections of dialog: Electrical, Mechanical, Feedback. Refer to section 4.6.3 for explanation of motor parameters and how to enter them.



NOTE

If unsure of the motor halls order and encoder channels A and B relationship, leave “B leads A for CW”, “Halls order” and “inverted” fields as they are. You can execute autophasing (refer to section 4.6.2) to set them correctly.

2. Enter motor model and vendor in the top edit boxes. Motor ID cannot be entered, this is set to 0 for custom motors.
3. Click “Save File” button and enter filename without extension. Default extension .cmt will be given when you click OK on file dialog box.



NOTE

Saving the file is necessary even if the autophasing feature will be used and some of the final parameters are not known. After autophasing is completed the corrected motor file can be updated before loading it to memory.

4. Click OK to exit from the Motor Parameters dialog.
5. MotionView will ask if you want to autophase your custom motor. If you answer “No”, the motor data will be loaded immediately to the drive’s memory. If you answer “Yes”, the motor dialog will be dismissed and the drive will start the autophasing sequence. Refer to section 4.6.2 for autophasing information.
6. If you answered “Yes” for autophasing, you will be returned to the same motor selection dialog box after autophasing is complete. For motors with incremental encoders, the fields “B leads A for CW”, “Halls order” and “inverted” will be assigned correct values. For motors with resolvers, the fields “Offset in degree” and “CW for positive” will be assigned correct values.
7. Click “Save File” to save the custom motor file and then click “OK” to exit the dialog box and load the data to the drive.



Interface

4.6.2 Autophasing

The Autophasing feature determines important motor parameters when using a motor that is not in MotionView's database. For motors equipped with incremental encoders, Autophasing will determine the Hall order sequence, Hall sensor polarity and encoder channel relationship (B leads A or A leads B for CW rotation). For motors equipped with resolvers, Autophasing will determine resolver angle offset and angle increment direction ("CW for positive").

To perform autophasing:

1. Complete the steps in the previous section "Setting custom motor parameters". If the motor file you are trying to autophase already exists, simply load it as described under "Using a custom motor" at the beginning of this section.
2. Make sure that the motor's shaft is not connected to any mechanical load and can freely rotate.



STOP!

Autophasing will energize the motor and will rotate the shaft. Make sure that the motor's shaft is not connected to any mechanical load and can freely and safely rotate.

3. Make sure that the drive is not enabled.

4. It is not necessary to edit the field "Hall order" and check boxes "inverted" and "B leads A for CW" as these values are ignored for autophasing.
5. Click OK to dismiss motor selection dialog. MotionView responds with the question "Do you want to perform autophasing?"
6. Click OK. A safety reminder dialog appears. Verify that it is safe to run the motor then click "Proceed" and wait until autophasing is completed.



NOTE

If there is a problem with the motor connection, hall sensor connection or resolver connection, MotionView will respond with an error message. Problems commonly occur with power, shield and ground terminations or when an improper cable is used. Correct the wiring problem(s) and repeat steps 1 - 6.

If the error message repeats, exchange motor phases U and V (R and S) and repeat. If problems persist, contact the factory.

7. If autophasing is completed with no error then MotionView will return to the motor dialog box. For motors with incremental encoders, the parameter field "Hall order" and the check boxes "inverted", "B leads A for CW" will be filled in with correct values. For resolver equipped motors, fields "Offset" and "CW for positive" will be correctly set.
8. Click "Save File" to save the completed motor file (you can use the same filename as you use to save initial data in step 1) and click OK to load the motor data to the drive.

4.6.3 Custom Motor Data Entry

A Custom Motor file is created by entering motor data into the "Motor Parameters" dialog box. This box is divided up into the following three sections, or frames:

- Electrical constants
- Mechanical constants
- Feedback

When creating a custom motor you must supply all parameters listed in these sections. All entries are mandatory except the motor inertia (Jm) parameter. A value of 0 may be entered for the motor inertia if the actual value is unknown.



4.6.3.1 Electrical constants

Motor Torque Constant (Kt)

Enter the value and select proper units from the drop-down list.



NOTE

Round the calculated result to 3 significant places.

Motor Voltage Constant (Ke)

The program expects Ke to be entered as a phase-to-phase Peak voltage. If you have Ke as an RMS value, multiply this value by 1.414 for the correct Ke Peak value.

Phase-to-phase winding Resistance (R) in Ohms

This is also listed as the terminal resistance (Rt). The phase-to-phase winding Resistance (R) will typically be between 0.05 and 200 Ohms.

Phase-to-phase winding Inductance (L)

This must be set in millihenries (mH). The phase-to-phase winding Inductance (L) will typically be between 0.1 and 200.0 mH.



NOTE

If the units for the phase-to-phase winding Inductance (L) are given in micro-henries (μH), then divide by 1000 to get mH.

Nominal phase current (RMS Amps)

Nominal continuous phase current rating (I_n) in Amps RMS. Do not use the peak current rating.



NOTE

Sometimes the phase current rating will not be given. The equation below may be used to obtain the nominal continuous phase-to-phase winding current from other variables.

$$I_n = \text{Continuous Stall Torque} / \text{Motor Torque Constant (Kt)}$$

The same force x distance units must be used in the numerator and denominator in the equation above. If torque (T) is expressed in units of pound-inches (lb-in), then Kt must be expressed in pound-inches per Amp (lb-in/A). Likewise, if T is expressed in units of Newton-meters (N-m), then units for Kt must be expressed in Newton-meters per Amp (N-m/A).

Example:

Suppose that the nominal continuous phase to phase winding current (I_n) is not given. Instead, we look up and obtain the following:

Continuous stall torque T = 3.0 lb-in

Motor torque constant Kt = 0.69 lb-in/A

Dividing, we obtain:

$$I_n = 3.0 \text{ lb-in} / 0.69 \text{ lb-in/A} = 4.35 \text{ (A)}$$

Our entry for (I_n) would be 4.35.

Note that the torque (lb-in) units are cancelled in the equation above leaving just Amps (A). We would have to use another conversion factor if the numerator and denominator had different force x distance units.



Interface

Nominal Bus Voltage (Vbus)

The Nominal Bus Voltage can be calculated by multiplying the Nominal AC mains voltage supplied by 1.41. When using a model with the suffix "S1N" where the mains are wired to the "Doubler" connection, the Nominal Bus Voltage will be doubled.

Example:

If the mains voltage is 230VAC, $V_{bus} = 230 \times 1.41 = 325V$

This value is the initial voltage for the drive and the correct voltage will be calculated dynamically depending on the drive's incoming voltage value.

Rotor Moment of Inertia (Jm)

From motor manufacturer or nameplate.



NOTE

Round the calculated result to 3 significant places.

Maximum Motor Speed in RPM

This is also listed as "Speed @ Vt" (motor speed at the terminal voltage rating). The maximum motor speed will typically be a round even value between 1000 - 6000 RPM.

Number of Poles

This is a positive integer number that represents the number of motor poles, normally 2, 4, 6 or 8.

4.6.3.2 For Incremental Encoder - Equipped Motors Only

Encoder Line Count

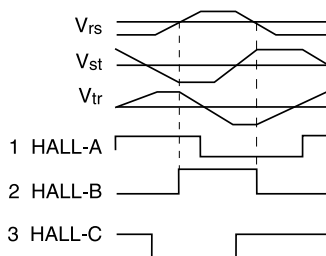
The Encoders for servomotors normally have Line Counts of 1000, 1024, 2000, 2048, 4000, or 4096. The Encoder Line Count must be a positive integer and must be pre-quadrature.

Index pulse offset. Enter 0 (zero)

Index marker pulse position. This field is reserved for backward compatibility. All PositionServo drives determine actual marker pulse position automatically.

Halls Order

Each hall signal is in phase with one of the three phase-phase voltages from the motor windings. Hall order number defines which hall sensor matches which phase-phase voltage. Motor phases are usually called R-S-T or U-V-W or A-B-C. Phase-Phase voltages are called V_{rs} , V_{st} , V_{tr} . Halls are usually called HALL-A, HALL-B, HALL-C or just Halls 1, 2, 3. A motor's phase diagram is supplied by motor vendor and usually can be found in the motor data sheet or by making a request to the motor manufacturer. A sample phase diagram is illustrated in Figure S912.



S912



The Halls Order is obtained as follows:

1. Look at the “Vrs” Output Voltage and determine the Hall Voltage that is lined up with (or in phase with) this voltage. To determine which Hall Voltage is in phase with the Vrs Output Voltage draw vertical lines at those points where it crosses the horizontal line (zero). The dashed lines at the zero crossings (above) indicate that Hall B output is lined up with (and in phase with) the Vrs Output Voltage.
2. Look at the “Vst” Output Voltage. Determine which Hall Voltage is in phase with this Voltage. Per Figure S912, the Hall C output is in phase with the Vst Output Voltage.
3. Look at the “Vtr” Output Voltage. Determine which Hall Voltage is in phase with this Voltage. Per Figure S912, the Hall A output is in phase with the Vtr Output Voltage.



NOTE

If hall sensors are in phase with the corresponding phase voltage but are inverted 180 degrees (hall sensor waveform edge aligns with the phase-phase voltage waveform but the positive hall sensor cycle matches the negative phase-phase waveform or visa-versa), you must check the “Inverted” check box.

4. The phases that correspond to the Vrs, Vst and Vtr voltages are Hall B then Hall C then Hall A or Halls number 2 then 3 then 1. Referring to the following table, we find that 2-3-1 sequence is Halls Order number 3. We would then enter 3 for the Halls Order field in the motor dialog box.

Hall Order Numbers for Different Hall Sequences

Halls Order	Hall Sequence
0	1-2-3
1	1-3-2
2	2-1-3
3	2-3-1
4	3-1-2
5	3-2-1



NOTE

Each Hall Voltage is in phase with one and only one Output Voltage.

B leads A for CW

This is the encoder phase relationship for CW/CCW shaft rotation. When you obtain the diagram for your motor phasing similar to shown above, it's assumed by the software that the motor shaft rotates CW when looking at the mounting face of the motor. For that rotation Encoder phase A must lead phase B. If it does leave the check box unchecked. Otherwise (if B leads A), check B leads A in the CW box.



NOTE

Lenze convention references the shaft direction of rotation from the front (shaft end) of the motor. Some manufacturer's timing diagrams are CW when viewed from the “rear” of the motor.



Interface

4.6.3.3 For Resolver Equipped Motors Only

If parameter “Resolver” is checked, following parameters appear on the form:

Offset in degree (electrical)

This parameter represents offset between resolver’s “0 degree” and motor’s windings “0 degree”.

CW for positive

This parameter sets the direction for positive angle increment.

“Offset in degree” and “CW for positive” will be set during Auto-Phasing of the motor.

4.6.3.4 For Asynchronous Servo Motors Only

Four additional parameters need to be defined for asynchronous motors:

Power Factor Cos Phi ($\cos \phi$)

The power factor is defined as the ratio of the active (true or real) power to apparent power. The power factor range is from 0 to 1.

Base Frequency in Hz

The motor base frequency defines the output frequency, when operating at rated voltage, rated current, rated speed, and rated temperature.

Velocity Nominal in RPM

Also called rated velocity or speed, velocity nominal is obtained when the motor is operated at the base frequency, rated current, rated voltage, and rated temperature.

Velocity Max in RPM

This is the maximum speed of the motor. The maximum velocity is usually limited by mechanical construction.



5 Parameters

The PositionServo drive is configured through an RS485 or Ethernet interface. The drive has many programmable features and parameters accessible via a universal software called MotionView. Refer to the MotionView Manual for details on how to make a connection to the drive and change parameter values. This chapter covers the PositionServo's programmable features and parameters in the order they appear in the Parameter Tree of MotionView. Programmable parameters are divided into groups. Each group holds one or more user adjustable parameters.

All drives can execute a User Program in parallel with motion. Motion can be specified by variety of sources and in three different modes: Torque, Velocity and Position.

In Torque and Velocity mode Reference can be taken from Analog Input AIN1 or from the User Program by setting a particular variable (digital reference). In Position mode, the reference could be taken from MA/MB master encoder/step and directions inputs (available in terminal P3) or from trajectory generator. Access to the trajectory generator is provided through the User Program's motion statements, MOVEx and MDV. Refer to the PositionServo Programming Manual for details on programming.

Whether the reference comes from an external device, (AIN1 or MA/MB) or from the drives internal variables (digital reference and trajectory generator) will depend on the parameter settings. Refer to "Parameters" group in MotionView.

5.1 Parameter Storage and EPM Operation

5.1.1 Parameter Storage

All settable parameters are stored in the drive's internal non-volatile memory. Parameters are saved automatically when they are changed and are copied to the EPM memory module located on the drive's front panel. In the unlikely event of drive failure, the EPM can be removed and inserted into the replacement drive, thus making an exact copy of the drive being replaced. This shortens down time by eliminating the configuration procedure. The EPM can also be used for replication of the drive's settings.

5.1.2 EPM Operation

When the drive is powered up it first checks for a white EPM in the EPM Port. If the EPM Port is empty, no further operation is possible until a white EPM is installed into the EPM Port. The drive will display "--EP--" until an EPM is inserted. Never install or remove the EPM module while the drive is powered.

If a different color EPM is inserted the drive may appear to function however, some operations will not be correct and the drive may hang. The white EPM is the only acceptable EPM for the PositionServo drive. If a white EPM is detected, the drive compares data in the EPM to that in its internal memory. In order for the drive to operate, the contents of the drive's memory and EPM must be the same. If "FEP?" is displayed press the enter button to load the EPM's data to the drive. Wait. The drive will display "BUSY" during loading and will return to normal display once the update is completed.

**STOP!**

If the EPM contains any data from an inverter drive, that data will be overwritten during this procedure.

5.1.3 EPM Fault

If the EPM fails during operation or the EPM is removed from the EPM Port, the drive will generate a fault and will be disabled (if enabled). The fault is logged to the drives fault history. Further operation is not possible until the EPM is replaced (inserted) and the drive's power is cycled. The fault log on the display shows "F_EP" fault.



5.2 Motor Group

The motor group shows the data for the currently selected motor. Refer to section 4.5 for details on how to select another motor from the motor database or to configure a custom motor.

5.3 Parameters

5.3.1 Drive Operating Modes

The PositionServo has 3 operating mode selections: Torque, Velocity and Position.

For Torque and Velocity modes the drive will accept an analog input voltage on the AIN1+ and AIN1- pins of P3 (refer to section 4.3.1). This voltage is used to provide a torque or speed reference.

For Position mode the drive will accept step and direction logic signals or a quadrature pulse train on pins P3.1- P3.4.

5.3.1.1 Torque Mode

In torque mode, the servo control provides a current output proportional to the analog input signal at input AIN1, if parameter "Reference" is set to "External". Otherwise the reference is taken from the drive's internal variable, IREF. (Refer to the PositionServo Programming Manual for details). For analog reference "Set Current", (current the drive will try to provide), is calculated using the following formula:

$$\text{Set Current(A)} = \text{Vinput(Volt)} \times \text{Iscale (A/Volt)}$$

where: Vinput is the voltage at analog input

Iscale is the current scale factor (input sensitivity) set by the Analog input (Current Scale) parameter (section 5.3.5).

5.3.1.2 Velocity Mode

In velocity mode, the servo controller regulates motor shaft speed (velocity) proportional to the analog input voltage at input AIN1, if parameter "Reference" is set to "External". Otherwise the reference is taken from the drive's internal variable, IREF. (Refer to the PositionServo Programming Manual for details). For analog reference, Target speed (set speed) is calculated using the following formula:

$$\text{Set Velocity (RPM)} = \text{Vinput (Volt)} \times \text{Vscale (RPM/Volt)}$$

where: Vinput is the voltage at analog input (AIN1+ and AIN1-)

Vscale is the velocity scale factor (input sensitivity) set by the Analog input (Velocity scale) parameter (section 5.3.6).

5.3.1.3 Position Mode

In this mode the drive reference is a pulse-train applied to P3.1-4 terminals, if the parameter "Reference" is set to "External". Otherwise the reference is taken from the drive's internal variables. (Refer to the PositionServo Programming Manual for details).

P3.1-4 inputs can be configured for two types of signals: step and direction and Master encoder quadrature signal. Refer to section 4.2.1 for details on these inputs connections. Refer to section 6.3 for details about positioning and gearing.

When the Reference is set to Internal, the drives reference position, (theoretical or Target position), is generated by trajectory generator. Access to the trajectory generator is provided by motion statements, MOVEx and MDV, from the User Program.



5.3.2 Drive PWM frequency

This parameter sets the PWM carrier frequency. Frequency can be changed only when the drive is disabled. Maximum overload current is 300% of the drive rated current when the carrier is set to 8kHz. It is limited to 250% at 16kHz.

5.3.3 Current Limit

The CURRENT LIMIT setting determines the nominal currents, in amps RMS per phase, which output to the motor phases. To prevent the motor from overloading, this parameter is usually set equal to the motor nominal (or rated) phase current. If MotionView (6.04) or higher is used, the Current Limit is set equal to the nominal motor phase current by default when a motor model is selected. To modify this parameter, refer to section 5.3.23.

5.3.4 Peak Current Limit (8 kHz and 16 kHz)

Peak Current Limit sets the motor RMS phase current that is allowed for up to 2 seconds. After this two second limit, the drive output current to motor will be reduced to the value set by the Current Limit parameter. When the motor current drops below nominal current for two seconds, the drive will automatically re-enable the peak current level. This technique allows for high peak torque on demanding fast moves and fast start/stop operations with high regulation bandwidth. If 8 kHz is used for Drive PWM frequency, use the parameter 8 kHz Peak Current Limit, otherwise, use 16 kHz Peak Current Limit.

If MotionView (6.04) or higher is used, the Peak Current Limit is set equal to 2.5 times the nominal motor phase current by default when a motor model is selected. To prevent motor from overloading, the Peak Current Limit shall be set no higher than the maximum motor current. Otherwise, the motor may be damaged due to overheating. To modify this parameter, refer to section 5.3.23.

5.3.5 Analog Input Scale (current scale)

This parameter sets the analog input sensitivity for current reference used when the drive operates in torque mode. Units for this parameter are A/Volt. To calculate this value use the following formula:

$$I_{scale} = I_{max} / V_{in\ max}$$

I_{max} maximum desired output current (motor phase current RMS)

$V_{in\ max}$ max voltage fed to analog input at I_{max}

Example:

$$I_{max} = 5A \text{ (phase RMS)}$$

$$V_{in\ max} = 10V$$

$$I_{scale} = I_{max} / V_{in\ max}$$

$$= 5A / 10V = 0.5 A / Volt \text{ (value to enter)}$$

5.3.6 Analog Input Scale (velocity scale)

This parameter sets the analog input sensitivity for the velocity reference used when the drive operates in velocity mode. Units for this parameter are RPM/Volt. To calculate this value use the following formula:

$$V_{scale} = VELOCITY_{max} / V_{in\ max}$$

$VELOCITY_{max}$ maximum desired velocity in RPM

$V_{in\ max}$ max voltage fed to analog input at $Velocity_{max}$

Example:

$$VELOCITY_{max} = 2000 \text{ RPM}$$

$$V_{in\ max} = 10V$$

$$V_{scale} = VELOCITY_{max} / V_{in\ max}$$

$$= 2000 / 10V$$

$$= 200 \text{ RPM / Volt (value to enter)}$$



5.3.7 ACCEL/DECEL Limits (velocity mode only)

The ACCEL setting determines the time the motor takes to ramp to a higher speed. The DECEL setting determines the time the motor takes to ramp to a lower speed. If the ENABLE ACCEL/DECEL LIMITS is set to DISABLE, the drive will automatically accelerate and decelerate at maximum acceleration limited only by the current limit established by the PEAK CURRENT LIMIT and CURRENT LIMIT settings.

5.3.8 Reference

The REFERENCE setting selects the reference signal being used by the drive. This reference signal can be either External or Internal. An External Reference can be one of three types, an Analog Input signal, a Step and Direction Input or an Input from an external Master Encoder. The Analog Input reference is used when the drive is either in torque or velocity mode. The Master Encoder and Step and Direction reference is used when the drive is in position mode. An Internal Reference is used when the motion being generated is derived from drive's internal variable(s), i.e., User Program, (Refer to the PositionServo Programming Manual).

5.3.9 Step Input Type (position mode only)

This parameter sets the type of input for position reference the drive expects to see. Signal type can be step and direction (S/D) type or quadrature pulse-train (Master Encoder / Electronic Gearing). Refer to section 4.2.1 for details on these inputs.

5.3.10 Fault Reset Option

The FAULT RESET OPTION selects the type of action required to reset the drive after a FAULT signal has been generated by the drive. ON DISABLE clears the fault when the drive is disabled. This is useful if you have a single drive and motor connected in a single drive system. The ON ENABLE option clears the fault when the drive is re-enabled. Choose ON ENABLE if you have a complex servo system with multiple drives connected to an external controller. This makes troubleshooting easier since the fault will not be reset until the drive is re-enabled. Thus, a technician can more easily determine which component of a complex servo system has caused the fault.

5.3.11 Motor Temperature Sensor

This parameter enables / disables motor over-temperature detection. It must be disabled if the motor PTC sensor is not wired to either P7.1-2 or to the resolver option module (P11).

5.3.12 Motor PTC Cut-off Resistance

This parameter sets the cut-off resistance of the PTC which defines when the motor reaches the maximum allowable temperature. See section 4.5.2 for details how to connect motor's PTC.

5.3.13 Second Encoder

Disables or enables second encoder. Effectively selects single-loop or double-loop configuration in position mode. The second encoder connects to the Encoder Option Module (E94ZAENC1) connector P12, refer to section 6.4 for details on dual loop operation.



5.3.14 Regeneration Duty Cycle

This parameter sets the maximum duty cycle for the brake (regeneration) resistor. This parameter can be used to prevent brake resistor overload. Use the following formula to calculate the maximum value for this parameter. If this parameter is set equal to the calculated value, the regeneration resistor is most effective without overload. One may set this parameter with a value smaller than the calculated one if the drive will not experience over voltage fault during regeneration.

$$D = P * R / (U_{max})^2 * (1/D_{application}) * 100\%$$

Where:

D (%) regeneration duty cycle

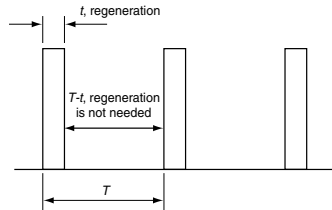
U_{max} (VDC) bus voltage at regeneration conditions

U_{max} = 390 VDC for 120/240 VAC drives and 770 VDC for 400/480 VAC drives.

R (Ohm) regeneration resistor value

P (W) regeneration resistor rated power

D_{application} (%) application duty cycle. For the continuous regeneration applications, use D_{application} = 1. For the intermittent regeneration applications, use D_{application} = t/T, where t is the duration when regeneration is needed and T is the time interval between two regenerations. Both t and T must use the same time unit, e.g., seconds



If calculation of D is greater than 100% set it to 100% value. If calculation of D is less than 10% then resistor power rating is too low. For more information refer to the PositionServo Dynamic Braking Manual (G94BR01).

Minimum Required Dynamic Braking Resistance

Drive Model	DB Resistor Minimum Resistance (Ω)
E94_180T2N~~	15
E94_080S2F~~, E94_080Y2N~~, E94_100S2F~~, E94_100Y2N~~	20
E94_120Y2N	30
E94_020S1N~~, E94_020S2F~~, E94_020Y2N~~, E94_040S1N~~, E94_040S2F~~, E94_040Y4N~~	40
E94_090T4N~~	45
E94_040T4N~~, E94_050T4N~~, E94_060T4N~~	75
E94_020T4N~~	150



Parameters

5.3.15 Encoder Repeat Source

This parameter sets the feedback source signal for the buffered encoder outputs (P3.7 -P3.12). The source can be the drive's feedback input (P4) or an optional feedback module (resolver, second encoder etc.)

5.3.16 System to Master Ratio

This parameter is used to set the scale between the reference pulse train (when operating in position mode) and the system feedback device. In a single loop configuration, the system feedback device is the motor encoder or resolver. In a dual-loop system the system encoder is the second encoder. See sections 6.3 and 6.4 for details.

5.3.17 Second to Prime Encoder Ratio

This parameter sets the ratio between the secondary encoder and the primary feedback device when the drive is configured to operate in dual-loop mode. When the primary feedback device is a resolver, the pulse count is fixed at 65,536. The resolutions of encoders are "post quadrature" (PPR x 4). See section 6.4.



NOTE

Post quadrature pulse count is 4X the pulses-per-revolution (PPR) of the encoder.

5.3.18 Autoboot

When set to "Enabled" the drive will start to execute the user's program immediately after cold boot (reset). Otherwise the user program has to be started from MotionView or from the Host interface.

5.3.19 Group ID

Refer to the PositionServo Programming Manual for details. This parameter is only needed for operations over Ethernet network.

5.3.20 Enable Switch Function

If set to "Run", input IN_A3 (P3.29) acts as an "Enable" input when the user program is not executing. If the user program is executing, the function will always be "Inhibit" regardless of the setting. This parameter is needed so the drive can be Enabled/Disabled without running a user's program.

5.3.21 User Units

This parameter sets up the relationship between User Units and motor revolutions. From here you can determine how many User Units there is in one motor revolution. This parameter allows the user to scale motion moves to represent a desired unit of measure, (inches, meters, in/sec, meters/sec, etc). For example: A linear actuator allows a displacement of 2.5" with every revolution of the motor's shaft.

Units = Units / Revolutions

Units = 2.5 Inches / Revolution

Units = 2.5

5.3.22 Resolver Track

The Resolver Track parameter is used in conjunction with the resolver motors and Buffered Encoder Outputs, (section 4.2.2). If a motor with resolver feedback is being used a simulated encoder feedback is transmitted out the Buffered Encoder Outputs, P3.7 to P3.12. The default resolution of this feedback is 1024 pulses per revolution, pre quad. If a different resolution is required then the Resolver Track parameter is utilized.



The number entered into this field, 0-15, directly correlates to a different encoder resolution. Please reference the table herein.

Resolver Track Configuration

Resolver Track	Resolution Before Quad	Resolver Track	Resolution Before Quad
0	1024	8	1000
1	256	9	1024
2	360	10	2000
3	400	11	2048
4	500	12	2500
5	512	13	2880
6	720	14	250
7	800	15	4096

5.3.23 Current Limit Max Overwrite

If this parameter is set to “Disable”, the parameters “Current limit”, “8 kHz peak current limit” and “16 kHz peak current limit” cannot be overwritten. If you want to overwrite the above three current limit parameters, this parameter must be set as “Enable”. To prevent the motor from overloading, the “current Limit”, “8 kHz peak current limit” and “16 kHz peak current limit” shall be set to values no higher than the corresponding current limits of the motor in use. Note that this parameter applies to firmware version (3.06) or higher.

5.4 Communication

5.4.1 Ethernet Interface

Programming and diagnostics of the PositionServo drive are done over the standard 10/100 Mbps Ethernet communication port. All devices on an Ethernet network have an IP address. Before connecting MotionView software to the PositionServo drive, set up the IP address of the drive and configure the PC as well.

The IP address of the PositionServo drive is composed of four sub-octets that are separated by three dots. This conforms to the Class C Subnet structure. The sub-octets IP_1, IP_2, IP_3 and IP_4 can be found by using “UP” and “DOWN” buttons of the LED panel and are organized in the following order:

IP_1.IP_2.IP_3.IP_4

where each sub-octet IP_x can be any number between 1-254. On the LED display, only IP_4 can be changed. IP_1, IP_2 and IP_3 can be changed once the PositionServo drive is connected to the MotionView software. As shipped from the factory the default IP address is 192.168.124.120.

If using the default PC Ethernet port on your computer for internal use (email, web browsing, etc.) AC Tech recommends that you add an additional Ethernet port to your PC. The most common and cost effective way to do this is by using a USB / Ethernet dongle or a PCMCIA Ethernet card. Then configure this Ethernet port to the PositionServo Subnet address and leave your local connection for your internal use.

There are two modes to obtain the IP address of the PositionServo drive by setting DHCP equal to either 0 or 1. These modes are described herein. It is important to know that the drive must be rebooted after changing any Ethernet settings such as IP address and DHCP.



Parameters

5.4.1.1 Manually Obtain the PositionServo Drive's IP Address

The PositionServo drive can be connected to a local PC or a private network if setting DHCP=0. In this mode, make sure to set DHCP = 0 via the diagnostic display LED, refer to section 7.1 for details. One can also verify the IP address of the drive via the display LED. When shipped from the factory the default IP address of the PositionServo drive is 192.168.124.120. Before MotionView can establish communications to the drive, both the PC and the PositionServo drive must be on the same subnet, but have different addresses. That is, both the PC and PositionServo drive shall have the same sub-octets IP_1, IP_2 and IP_3 and different IP_4. When connecting MotionView to a brand-new PositionServo drive out of box, set the PC's IP address as 192.168.124.1. Refer to section 6.4.1.3 on how to set up your PC IP address. Every time dHCP, or any IP sub-octet IP_x is changed, one must reboot the PositionServo drive so that the change can take effect.

Once the MotionView software is connected to the PositionServo drive, one can change the DHCP setting and the drive IP address via the communication option "Ethernet" – "IP setup" in MotionView. If one wants to configure the PositionServo drive's IP address under a specific subnet, for example, 10.135.110.xxx as shown below. One can pick an available IP_4, e.g., 246 is used below, then click "OK" to confirm. After this change, make sure to reboot the drive. After the drive reboot, the IP address stored in the EPM before last power-off will be the drive's IP address. In the meantime, one needs to configure the PC's IP address under the same subnet. In case, one may choose "Obtain an IP Address Automatically" for the PC or pick up an available IP address, refer to section 5.4.1.3 for details.



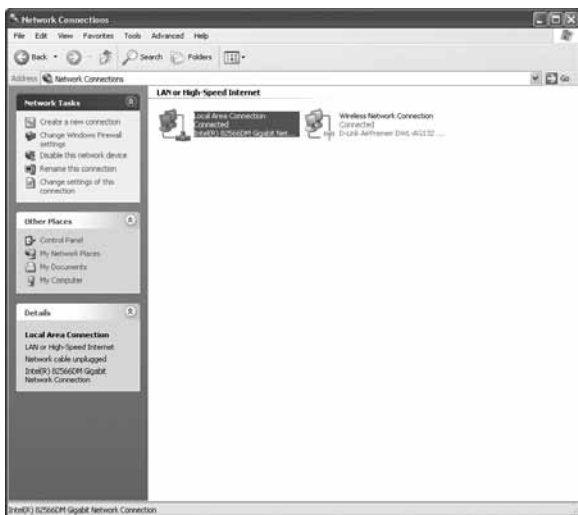
5.4.1.2 Automatically Obtain the PositionServo's IP Address

To use this mode set dHCP = 1 via the diagnostic display LED (refer to section 7.1 for details). After setting this parameter, cycle the input power to the PositionServo drive so that the setting can take effect. The LED display will be "----" if one checks the IP address octets IP_1, IP_2, IP_3 and IP_4. This means that the drive is still trying to acquire an IP address from the dHCP server. To obtain the PositionServo drive's IP address automatically, there must be a dHCP server available.

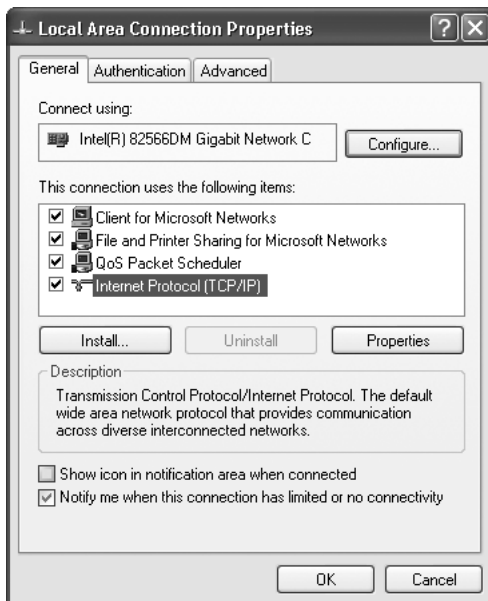
5.4.1.3 Set the PC's IP Address

Follow these steps to set up the PC's IP address:

To display the IP address of your PC, from the Start menu, select "Control Panel" and then select "Network Connections".



Select the connection you wish to set: “Local Area Connection”, the PC Default Port or “Local Area Connection x” your additional Ethernet port. Then double-click the icon to open the [Connection Status] details. To view the connection properties click the [Properties] button.



Select [Internet Protocol (TCP/IP)] and click the [Properties] button.



Parameters



Select “Use the following IP address” and enter [192.168.124.1] for the IP address. Now enter the subnet mask [255.255.255.0], and then click the [OK] button. Note that one can use “Obtain an IP address automatically” after the PositionServo drive’s IP address has been configured under the same subnet to which the PC is connected.

5.4.2 RS-485 Configuration

This parameter sets how the optional RS485 interface will function. The RS485 interface can be configured for normal operation (programming and diagnostics using MotionView software) or as a Modbus RTU slave. Refer to section 4.4 for details on communication interfaces.

5.4.3 Modbus Baud Rate

This parameter sets the baud rate for RS485 interface in Modbus RTU mode. When the drive is operating in normal mode the baud rate is set to the same setting as the RS232 interface.

5.4.4 Modbus Reply Delay

This parameter sets the time delay between the drives reply to the Modbus RTU master. This delay is needed for some types of Modbus masters to function correctly.

5.5 Analog I/O

5.5.1 Analog Output

The PositionServo has one analog output with 10-bit resolution on P3 pin 23. The signal is scaled to $\pm 10V$. The analog output can be assigned to following functions:

- Not Assigned
- Phase current RMS
- Phase current Peak
- Motor Velocity
- Phase R current
- Phase S current
- Phase T current
- Iq current (Torque component)
- Id current (Direct component)



5.5.2 Analog Output Current Scale (Volt / amps)

Applies scaling to all functions representing CURRENT values.

5.5.3 Analog Output Current Scale (mV/RPM)

Applies scaling to all functions representing VELOCITY values. (Note: that mV/RPM scaling units are numerically equivalent to volts/kRPM).

5.5.4 Analog Input Dead Band

Allows the setting of a voltage window (in mV) at the reference input AIN1+ and AIN1- (P3 pins 24 and 25) such that any voltage within that window will be treated as zero volts. This is useful if the analog input voltage drifts resulting in motor rotation when commanded to zero.

5.5.5 Analog Input Offset Parameter

Allows you to adjust the offset voltage at AIN1+ and AIN1- (P3 pins 24 and 25). This function is equivalent to the balance trim potentiometer found in analog drives. Lenze recommends that this adjustment be made automatically using the "Adjust analog voltage offset" button while the external analog reference signal commands zero speed.

5.5.6 Adjust Analog Input Zero Offset

This control button is useful to allow the drive to automatically adjust the analog input voltage offset. To use it, command the external reference source input at AIN1+ and AIN1- (P3 pins 24 and 25) to zero volts and then click this button. Any offset voltage at the analog input will be adjusted out and the adjustment value will be stored in the "Analog input offset" parameter.

5.6 Digital I/O

The PositionServo has four digital outputs. These outputs can be either assigned to one of the following functions, or be used by the drives internal User Program

- **Not Assigned** No special function assigned. Output can be used by the User Program.
- **Zero Speed** Output activated when drive is at zero speed, refer to "Velocity Limits" (section 5.7) for settings.
- **In Speed Window** Output activated when drive is in set speed window, refer to "Velocity Limits" (section 5.7) for settings.
- **Current Limit** Output activated when drive detects current limit.
- **Run Time Fault** A fault has occurred. Refer to section 8.3 for details on faults.
- **Ready** Drive is enabled.
- **Brake** Command for the holding brake option (E94ZAHBK2) for control of a motor with a holding brake. This output is active 10ms after the drive is enabled and deactivates 10ms before the drive is disabled.
- **In position** Position mode only. Refer to the Programming Manual.

5.6.1 Digital Input De-bounce Time

Sets de-bounce time for the digital inputs to compensate for bouncing of the switch or relay contacts. This is the time during an input transition that the signal must be stable before it is recognized by the drive.



5.6.2 Hard Limit Switch Action

Digital inputs IN_A1 and IN_A2 can be used as limit switches if their function is set to “Fault” or “Stop and Fault”. Activation of these inputs while the drive is enabled will cause the drive to Disable and go to a Fault state. The “Stop and Fault” action is available only in Position mode when the “Reference” parameter is set to “Internal”, i.e., when the source for the motion is the Trajectory generator. Refer to the PositionServo Programming Manual for details on “Stop and Fault” behavior. IN_A1 is the negative limit switch. IN_A2 is the positive limit switch. Both are treated as normally open.

5.7 Velocity Limits

These parameters are active in Velocity Mode Only.

5.7.1 Zero Speed

Specifies the upper threshold for motor zero speed in RPM. When the motor shaft speed is at or below the specified value the zero speed condition is set to true in the internal controller logic. The zero speed condition can also trigger a programmable digital output, if selected.

5.7.2 Speed Window

Specifies the speed window width used with the “In speed window” output.

5.7.3 At Speed

Specifies the speed window center used with the “In speed window” output.

These last two parameters specify speed limits. If motor shaft speed is within these limits then the condition AT SPEED is set to TRUE in the internal controller logic. The AT SPEED condition can also trigger a programmable digital output, if selected. For example if “AT SPEED” is set for 1000 RPM, and the “SPEED WINDOW” is set for 100, then “AT SPEED” will be true when the motor velocity is between 950 -1050 RPM.

5.8 Position Limits

5.8.1 Position Error

Specifies the maximum allowable position error in the primary (motor mounted) feedback device before enabling the “Max error time” clock. When using an encoder, the position error is in post-quadrature encoder counts. When using a resolver, position error is measured at a fixed resolution of 65,536 counts per motor revolution.

5.8.2 Max Error Time

Specifies maximum allowable time (in mS) during which a position error can exceed the value set for the “Position error” parameter before a Position Error Excess fault is generated.

5.8.3 Second Encoder Position Error

Specifies the maximum allowable error of the second encoder in post quadrature encoder counts before enabling the “Second encoder max error time” clock.

5.8.4 Second Encoder Max Error Time

Specifies maximum allowable time (in mS) during which the second encoder’s position error can exceed the value set for the “Second encoder position error” parameter before a Position Error Excess fault is generated.



5.9 Compensation

5.9.1 Velocity P-gain (proportional)

Proportional gain adjusts the system's overall response to a velocity error. The velocity error is the difference between the commanded velocity of a motor shaft and the actual shaft velocity as measured by the primary feedback device. By adjusting the proportional gain, the bandwidth of the drive is more closely matched to the bandwidth of the control signal, ensuring more precise response of the servo loop to the input signal.

5.9.2 Velocity I-gain (integral)

The output of the velocity integral gain compensator is proportional to the accumulative error over cycle time, with I-gain controlling how fast the error accumulates. Integral gain also increases the overall loop gain at the lower frequencies, minimizing total error. Thus, its greatest effect is on a system running at low speed, or in a steady state without rapid or frequent changes in velocity.



NOTE

The following four position gain settings are only active if the drive is operating in Position mode. They have no effect in Velocity or Torque modes.

5.9.3 Position P-gain (proportional)

Position P-gain adjusts the system's overall response to position error. Position error is the difference between the commanded position of the motor shaft and the actual shaft position. By adjusting the proportional gain, the bandwidth of the drive is more closely matched to the bandwidth of the control signal, ensuring more precise response of the servo loop to the input signal.

5.9.4 Position I-gain (integral)

The output of the Position I-gain compensator is proportional to accumulative error over cycle time, with I-gain controlling how fast the error accumulates. Integral gain also increases overall loop gain at the lower frequencies, minimizing total error. Thus, its greatest effect is on a system running at low speed, or in a steady state without rapid or frequent changes in position.

5.9.5 Position D-gain (differential)

The output of the Position D-gain compensator is proportional to the difference between the current position error and the position error measured in the previous servo cycle. D-gain decreases the bandwidth and increases the overall system stability. It is responsible for removing oscillations caused by load inertia and acts similar to a shock-absorber in a car.

5.9.6 Position I-limit

The Position I-limit will clamp the Position I-gain compensator to prevent excessive torque overshooting caused by an over accumulation of the I-gain. It is defined in terms of percent of maximum drive velocity. This is especially helpful when position error is integrated over a long period of time.



Parameters

5.9.7 Gain Scaling Window

Sets the total velocity loop gain multiplier (2^n) where n is the velocity regulation window. If, during motor tuning, the velocity gains become too small or too large, this parameter is used to adjust loop sensitivity. If the velocity gains are too small, decrease the total loop gain value, by decreasing this parameter. If gains are at their maximum setting and you need to increase them even more, use a larger value for this parameter.

5.10 Tools

5.10.1 Oscilloscope Tool

The oscilloscope tool gives real time representation of different signals inside the PositionServo drive and is helpful when debugging and tuning drives. Operation of the oscilloscope tool is described in greater detail in the MotionView User's Manual (IM94MV01). The following signals can be observed with the oscilloscope tool:

Phase Current (RMS):	Motor phase current
Phase Current (Peak):	Motor peak current
Iq Current:	Measures the motor Iq (torque producing) current
Motor Velocity:	Actual motor speed in RPM
Commanded Velocity:	Desired motor speed in RPM (velocity mode only)
Velocity Error:	Difference in RPM between actual and commanded motor speed
Position Error:	Difference between actual and commanded position (Step & Direction mode only)
Bus Voltage:	DC bus voltage
Analog Input:	Voltage at drive's analog input
Absolute Position:	Absolute (actual) position
Absolute Position Pulses:	Absolute position expressed in pulses of the primary feedback device
Secondary Abs Position:	Absolute (actual) position of secondary feedback device
Secondary Position Error:	Difference between actual and commanded position of secondary feedback device
Target Position:	Requested position
Target Position Pulses:	Requested position expressed in pulses of the primary feedback device
Position Increment:	Commanded position increment

5.10.2 Run Panels

Check Phasing

This button activates the Autophasing feature as described in section 4.6.2. However, in this panel only the motor phasing is checked, the motor data is not modified.

5.11 Faults

The Faults Group loads the fault history from the drive. The 8 most recent faults are displayed with the newer faults replacing the older faults in a first-in, first-out manner. In all cases fault # 0 is the most recent fault. To clear the faults history from the drive's memory click on the "Reset Fault history" button. Each fault has its code and explanation of the fault. Refer to section 8.3 for details on faults.



6 Operation

This section offers guidance on configuring the PositionServo drive for operations in torque, velocity or position modes without requiring a user program. To use advanced programming features of PositionServo please perform all steps below and then refer to the PositionServo Programming Manual for details on how to write motion programs.

6.1 Minimum Connections

For the most basic operation, connect the PositionServo to mains (line) power at terminal P1, the servomotor power at P7 and the motor feedback as appropriate.



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait at least 60 seconds before servicing drive. Capacitors retain charge after power is removed.

As a minimum these connections must be made:

- Connect an Ethernet crossover cable between PositionServo's P2 and your PC's Ethernet port. A straight patch cable can be used if using a hub or switch.
- Connect mains power to terminal P1. Mains power must be as defined on the drive's data label (section 2.1).
- When connecting to an encoder-based drive, take the encoder feedback cable and connect it to the 15 pin D-sub connector located at P4. When connecting to a resolver-based drive, take the resolver feedback cable and connect it to the 9 pin D-sub connector located at P4.
- Connect motor windings U, V, W (also known as R, S, T) to terminal P7 according to Section 4.1.1. Make sure that motor cable shield is connected as described in section 3.2.
- Provide an Enable switch (IN_A3) according to Section 6.5.
- Perform drive configuration as described in the next section.



Note

When using an encoder-based drive and operating with a resolver option module as the primary feedback, a second encoder can be connected to P4.

6.2 Configuration of the PositionServo

The PositionServo must first be configured for the specific motor that will be used, the mode of operation, and then any additional features that will be used.

Drive configuration consists of following steps:

- Motor Selection
- Mode of operation selection
- **Reference source selection (Very Important)**
- Drive parameters (i.e. current limit, acceleration / deceleration) setup
- Operational limits (velocity or position limits) setup
- Input / Output (I/O) setup
- Velocity / position compensator (gains) setup
- Optionally store drive settings in a PC file and exit the MotionView program.

Operation

To configure drive:

1. Ensure that the control is properly installed and mounted. Refer to section 4 for installation instructions.
2. Perform wiring to the motor and external equipment suitable for desired operating mode and your system requirements.
3. Connect the Ethernet port P2 on the drive to your PC Ethernet port. If connecting directly to the drive from the PC, a crossover cable is required.
4. Make sure that the drive is disabled.
5. Apply power to the drive and wait until “d rS” shows on the display. For anything other than this, refer to the chart below before proceeding.

Drive Display	Meaning
-EP-	EPM missing. Refer to 6.1.2
EPi	EPM data. Refer to 6.1.2
----	No valid firmware
----	Monitor mode

6. Confirm that the PC and the drive have the correct IP setting. Refer to section 5.4.1.1 - Setting Your PC IP Address.
7. Launch MotionView software on your computer.
8. From the MotionView menu, select <Project> <Connection setup>.
9. Select “Ethernet UDP”, then click the OK button.
10. From the MotionView menu, select <Node> <Connect Drive>.
11. Click the Discover button to ping the network for any drives. If a drive is located the address will appear on the screen. If no address appears then you can type the IP address in. The default address for the drive is 192.168.124.120. Click the Connect button to connect to the drive.
12. Once MotionView connects to the drive, its node icon will appear in the upper left-hand corner of the Parameter Tree Window. Refer to the PositionServo Programming Manual for more details.



Note

MotionView’s “Connection setup” properties need only be configured the first time MotionView is operated or if the port connection is changed. Refer to MotionView User’s Manual for details on how to make a connection to the drive.

13. Double-click on the drive’s icon to expand parameter group’s folders.
14. Select the motor to be used according to the section 4.5.
15. Expand the folder “Parameters” and choose the operating mode for the drive. Refer to section 5.3.1 for details on operating modes.
16. Click on the “Current limit” parameter, refer to section 5.3.3 and enter current limit (in Amp RMS per phase) appropriate for the motor.
17. Click on the appropriate “Peak current limit” parameter, refer to section 5.3.4, based on the “Drive PWM frequency” parameter, refer to section 5.3.2, used and enter the peak current limit (in Amps RMS per phase) appropriate for your motor.
18. Set up additional parameters suitable for the operating mode selected in step 17.
19. After you configure the drive, proceed to the tuning procedure if operating in “Velocity”, or “Position” mode. “Torque” mode doesn’t require additional tuning or calibration. Refer to section 6.6 for details on tuning.



6.3 Position Mode Operation (gearing)

In position mode the drive will follow the master reference signals at the 1-4 inputs of P3. The distance the motor shaft rotates per each master pulse is established by the ratio of the master signal pulses to motor encoder pulses (in single loop configuration). The ratio is set by "System to Master ratio" parameter (see section 5.3.16).

Example 1

Problem: Setup the drive to follow a master encoder output where 1 revolution of the master encoder results in 1 revolution of the motor

Given: Master encoder: 4000 pulses/revolution (post quadrature)
 Motor encoder: 8000 pulses/revolution (post quadrature)

Solution: Ratio of System (motor encoder) to Master Encoder is $8000/4000 = 2/1$
 Set parameter "System to master ratio" to 2:1

Example 2

Problem: Setup drive so motor can follow a master encoder wheel where 1 revolution of the master encoder results in 3 revolutions of the motor

Given: Motor encoder: 4000 pulses/revolution (post quadrature)
 Master encoder: 1000 pulses/revolution (post quadrature).
 Desired "gear ratio" is 3:1

Solution: Ratio is motor encoder PPR divided by master encoder PPR times the "gear ratio":
 $(\text{Motor PPR} / \text{Master PPR}) * (3/1) \Rightarrow (4000/1000) * (3/1) \Rightarrow 12/1$
 Set parameter "System to master ratio" to 12:1

6.4 Dual-loop Feedback

In dual-loop operation (position mode only) the relationship between the Master input and mechanical system movement requires that two parameters be set:

1. "System to master ratio" sets the ratio between the second encoder pulses (system encoder) and the master input pulses.
2. "Prime to second encoder ratio" sets the ratio between the second and primary (motor) encoder. If the motor is equipped with a resolver connected to the resolver option module, the primary encoder resolution of 65536 (post quadrature) must be used.

When operating in this mode the second encoder input is applied to integral portion of the position compensator. Therefore it is important that the Position I-gain and Position I-limit parameters are set to non 0 values. Always start from very small values of Position I-limit values.



Note

When using an encoder-based drive and operating with the Resolver Option Module as the primary feedback, a second encoder can be connected to P4.

6.5 Enabling the PositionServo

Regardless of the selected operating mode, the PositionServo must be enabled before it can operate. A voltage in the range of 5-24 VDC connected between P3 pins 26 and 29 (input IN_A3) is used to enable the drive, refer to section 4.1.7, note 3. The behavior of input IN_A3 differs depending on the setting of “Enable switch function”.



TIP!

If using the onboard +5VDC power supply for this purpose, wire your switch between pins P3.6 and P3.29. Jumper P3.5 to P3.26. If doing this, all inputs in group A must be powered by P3.6.

When the “Enable switch function” is set to “RUN”:

IN_A3 acts as positive logic ENABLE or negative logic INHIBIT input depending on:

If user program is not running: Activating IN_A3 enables the drive
 User program running: Activating IN_A3 acts as negative logic “Inhibit” and operates exactly as if parameter “Enable switch function” set to “Inhibit” (see below)

When the “Enable switch function” set to “Inhibit”:

IN_A3 acts as negative logic INHIBIT input regardless of mode or program status.

Activating input IN_A3 doesn't enable the drive. The drive can be enabled from the user's program or interface only when IN_A3 is active. Attempt to enable drive by executing the program statement “ENABLE” or from interface will cause the drive to generate a fault, F_36. Regardless of the mode of operation, if the input is deactivated while the drive is enabled, the drive will be disabled and will generate a fault, F_36.



WARNING!

Enabling the servo drive allows the motor to operate depending on the reference command. The operator must ensure that the motor and machine are safe to operate prior to enabling the drive and that moving elements are appropriately guarded. Failure to comply could result in damage to equipment and/or injury to personnel!

6.6 Drive Tuning

The PositionServo Drive will likely require some tuning of its gains parameters in order to achieve best performance in the application in which it is being applied. Only when the drive is placed in Torque Mode are the gain values not required to be tuned. The table herein lists the gains parameters that should be adjusted for each of the drive operating modes. These parameters are found within the ‘Compensation’ folder.

MotionView Parameter	Torque Mode	Velocity Mode	Positioning Mode
Velocity P Gain	No	Yes	Yes
Velocity I Gain	No	Yes	Yes
Position P Gain	No	No	Yes
Position I Gain	No	No	Yes
Position D Gain	No	No	Yes
Position I-Limit	No	No	Yes
Gain Scaling	No	Yes	Yes

Before using the tuning procedures detailed in the next sections, ensure that the system is in a safe condition for tuning to be carried out. It is often beneficial to first tune the motor off-load to obtain approximate gains setting before fine tuning in the application.

Check that the drive output to the motor is disabled (via Input A3) and that the drive is powered up. Make sure any user program code previously entered into the [Indexer Program] folder in MotionView has been saved prior to tuning so it can be easily recalled after tuning is complete.



WARNING!

During both the Velocity and Position tuning procedures the PositionServo drive will perform rotation (motion) of the motor shaft in the forward and reverse directions at velocities based on the settings made by the user. Ensure that the motor and associated mechanics of the system are safe to operate in the way specified during these procedures.

6.6.1 Tuning the Drive in Velocity Mode

1) Parameter Setup

Set up the motor as per the instructions given in the relevant section of this manual. The motor must be configured correctly prior to tuning taking place.

The parameters Drive Mode, Reference and Enable Switch Function are configured automatically by the velocity tuning program. They are not required to be set at this stage.

2) Importing the Velocity Tuning Program

Before importing the Velocity Tuning Program, the example programs must be installed from the Documentation CD that shipped with the drive. If this has not been done then please do so now.

To load the TuneV program file to the drive, select [Indexer Program] in the MotionView Parameter Tree. Select [Import program from file] on the main toolbar. Navigate to [C:\Program Files\AC Technology\MotionView6.xx\Help\940Examples]. If during the installation of the Documentation CD files a different default directory was selected, then navigate to that directory. Click on the [TuneV.txt] file and select [Open].



3) Editing the Velocity Tuning Program

The Tune Velocity Program creates a step velocity demand in the forward and reverse directions that the drive will attempt to follow (based on its velocity gain settings). The drive will run for a set time in the forward direction and then reverse the reference and run for the same set time in the reverse direction, showing the acceleration, deceleration and steady state performance.

The speed and period (time for one complete cycle - forward and reverse) is set in the Indexer program with the following statements:

```
; Motion Parameters
Define SpeedReference 5 ; speed reference in Rps
Define Period 500 ; time in millisec
```

Adjust these parameters to values suitable to the application in which the drive is used before going to the next step.

Operation

4) Compile and Download Indexer Program to Drive

In the [Indexer program] folder in MotionView, select [Compile and Load with Source] from the pull down menu. The TuneV program will be compiled and sent to the drive. Select [Run] from the pull down menu to run the TuneV program. Do NOT enable the drive (via input A3) at this stage.

5) Oscilloscope Settings

Open the [Tools] folder in MotionView and select the [Oscilloscope] tool. Click the [Set on Top] box to place a checkmark in it and keep the scope on top.

In the Scope Tool Window make the following settings:

Channel 1: Signal = "Commanded Velocity"

Scale = appropriate to "SpeedReference" value set in Indexer Program

Channel 2: Signal = "Motor Velocity"

Scale = appropriate to "SpeedReference" value set in Indexer Program

Timebase: = as appropriate to "Period" value of Indexer Program

Trigger: = Channel 1, Rising Edge

Level: = 10 RPM

For better resolution, adjust these scaling factors during the tuning procedure.

6) Compensation Folder

In MotionView, open the [Compensation] folder for the drive. Set [Gain Scaling] to a relatively low value, e.g. -6 for Encoder motor and -8 for a Resolver Motor. Set the [Velocity P-gain] to a mid-value (16000) and set the [Velocity I-Gain] to 0.

7) Gain Tuning

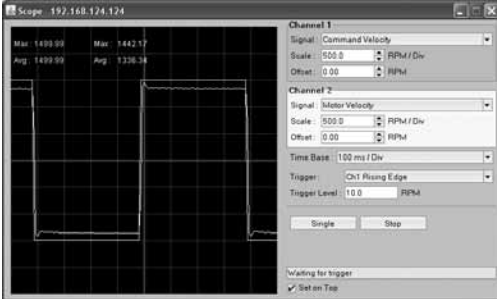
The system should now be ready to start tuning the velocity gains. Start the Oscilloscope by clicking [Run]. Apply the Enable input to Input A3 to enable the drive. At this point of the procedure it is desirable to have little to no motion until we start to increase the gain settings. If the motor vibrates uncontrollably disable the drive, lower the Gain Scaling parameter value and repeat the input enable.

Step 1: Setting the Gain Scaling Parameter

The gain scaling parameter is a 'course adjustment' of the other gain's parameter values. Steadily increase the value of the gain scaling parameter until a reasonable response is obtained from the motor (motor velocity starts to resemble the commanded velocity).



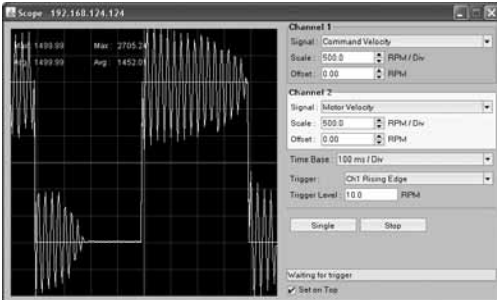
Gain Scaling set too LOW
Motor Velocity significantly different than
Commanded Velocity.



Gain Scaling set OK
Motor Velocity resembles Commanded Velocity. Motor Velocity is reasonably close with a slight overshoot.



Gain Scaling set too HIGH
Motor Velocity shows significant overshoot following the acceleration periods.



Gain Scaling set significantly too HIGH
Motor Velocity exhibits instability throughout the steady state Commanded Velocity.

Depending on the system begin tuned, the motor may go from stable operation (little to no overshoot with stable steady state velocity) to instability (continuous and pronounced oscillations during steady state command) very quickly as gains scaling is increased. The bandwidth for allowing some overshoot with a quick settle time may be very small and may only be achieved through adjustment of the Velocity P-Gain, as described in Step 2. Set the gain scaling parameter to the value preceding that where significant overshoot or continuous instability occurs. With the Gain scaling parameter set move onto tuning the velocity P and I gains.

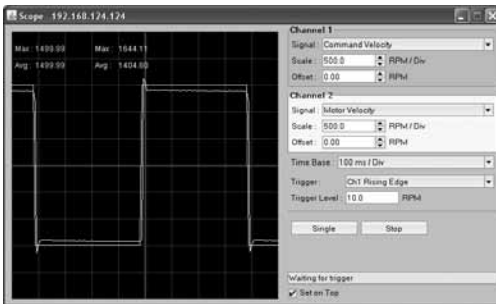
Operation

Step 2: Fine Tuning the Velocity P-Gain

Slowly alter the Velocity P-Gain (increase and decrease) and observe the motor velocity waveform on the oscilloscope. As the P-Gain increases the gradient of the velocity during acceleration and deceleration will also increase as will the final steady state velocity that is achieved. The application of too much P-Gain will eventually result in an overshoot in the motor velocity, and further increases will result in larger overshooting to the point that instability (continuous oscillation) occurs.

Increase the velocity P-gain until some overshoot occurs. Some overshoot is generally ok, and the objective is typically to achieve the shortest possible settle time (steady state velocity). When the system appears to have reached the shortest possible settle time, with acceptable overshoot, cease from increasing the P-Gain.

Scope traces will be similar to those shown in Step 1, however the P-gain will now be given a more precise adjustment in order to obtain the best possible tuning.

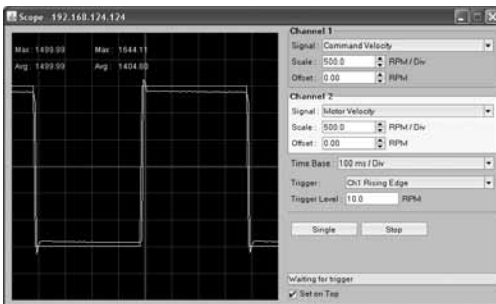


Good Fine Tuning of the P-Gain
Small overshoot with excellent settle time and steady state velocity regulation.

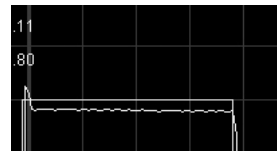
Step 3: Setting the Velocity I-Gain

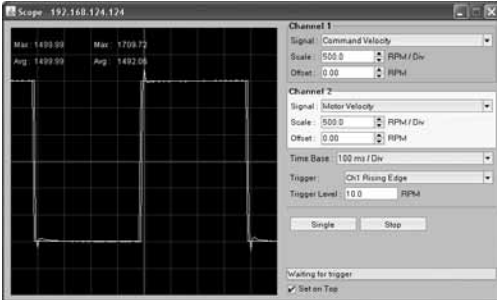
The purpose of the velocity I-gain is to correct any error that is present between the commanded velocity and the steady state velocity that could not be rectified by adjustment of the velocity P-gain. Adjustment of the velocity I-gain can also reduce the steady state ripple that may occur in the velocity waveform. Lastly, velocity I-gain has a positive effect on the holding torque produced by the motor.

Slowly increase the “Velocity I-Gain” and check for correction of the steady state error in the velocity waveform. Continuing to increase the velocity I-gain will eventually result in increased overshoot and instability in the motor velocity waveform. Stop increasing the I-Gain when additional overshoot or instability starts to occur.

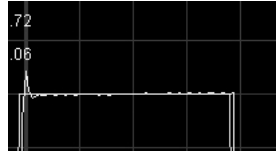


I-Gain set too LOW
Error exists between Commanded steady state velocity and Actual steady state velocity

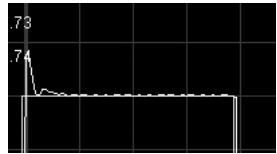
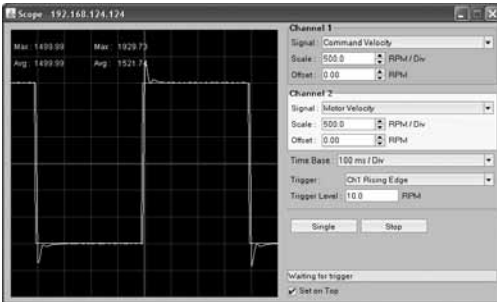




I-Gain set OK
No error between Commanded steady state velocity and Actual steady state velocity with excellent stability.



I-Gain set too HIGH
Additional overshoot and oscillations are starting to occur. Steady state velocity regulation



Step 4: Check Motor Currents

Finally check the motor currents on the Oscilloscope. Make the following settings to the oscilloscope.

Channel 1:

Signal = "Phase Current RMS"

Scale = as appropriate to peak current limit set in drive parameters (MotionView)

Timebase: = as appropriate to "Period" value of Indexer Program

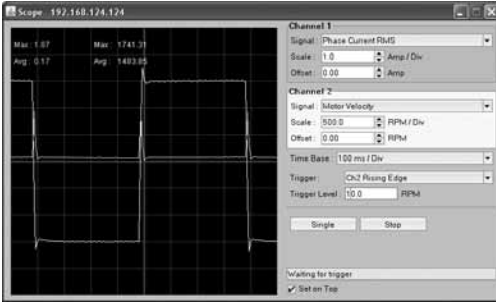
Trigger: = Channel 2, Rising Edge

Level: = 10 RPM

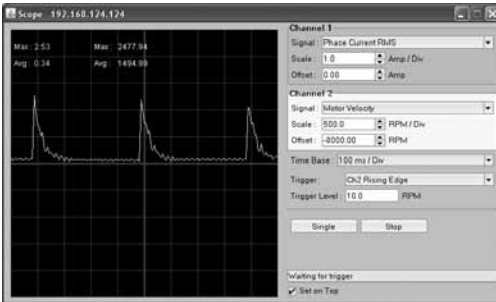
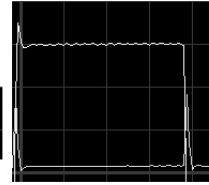
Observe the waveforms to insure there are no significant oscillations. Reduce the gains values if necessary.

The current waveform should be showing spikes of current during acceleration / deceleration and steady state current during any steady state velocity. The maximum value (peak value) of the current waveform is shown at the top of the oscilloscope screen. This maximum value can be compared to the drive nominal current and peak current settings to check how much of the motors potential performance is being used and if optimum performance is being achieved.

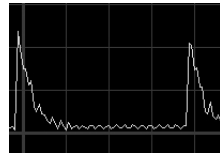
Operation



Good Current Trace
Uniform current pulses during accel/
deceleration and stable current during steady
state velocity.



Instability in Drive Output Current
(Note: Channel 2 trace has been removed for
clarity).



8) End Velocity Tuning

Remove the Enable Input from input A3 (disable the drive). In MotionView, click on the [Indexer] folder for the drive. Click [Reset] on the program toolbar. If the drive is to be run in just velocity mode then tuning is now complete. If the drive is to be used in Positioning mode continue with 'Tuning the Drive in Position Mode', section 6.6.2.

6.6.2 Tuning the Drive in Position Mode

Velocity Tuning should be carried out prior to the tuning of the position loop. Refer to the Velocity Tuning section, 6.6.1.

1) Parameter Set up

In MotionView, open the [Limits] folder and then the [Position Limits] sub-folder. Set the [Position Error] and [Max Error Time] parameters to their maximum values to effectively disable the position error trip while tuning takes place. Ensure the system is safe to operate in this manner.

Position Error = 32767

Max Error Time = 8000

The Drive Mode, Reference and Enable Switch Function parameters are automatically configured by the velocity tuning program. They do not require setting at this stage.



2) Importing the Position Tuning Program

Before importing the Position Tuning Program, the example programs must be installed from the Documentation CD that shipped with the drive. If this has not been done then please do so now.

To load the TuneP program file to the drive, select [Indexer Program] in MotionView. Select [Import program from file] on the main toolbar. Navigate to [C:\Program Files\AC Technology\MotionView6.x\Help\940Examples]. If during the installation of the Documentation CD files a different default directory was selected, then navigate to that directory. Click on the [TuneP.txt] file and select [Open].



3) Editing the Position Tuning Program

The Tune Position Program performs trapezoidal moves in the forward and reverse direction separated by a defined pause (or time delay).

The Accel, Decel, and MaxV variables within the TuneP program define the ramps and steady state velocity that will be used to execute the motion commands.

ACCEL = 500	;500 rps*s	Accel = Acceleration speed
DECCEL = 500	;500 rps*s	Decel = Deceleration speed
MAXV = 20	;20 Rps	MaxV = Maximum

The size of each move and the pause between the moves is defined in the following lines of code. There are two moves and pauses for the forward and reverse moves to be performed.

MOVED 0.25	;move 1 rev	MoveD = Move distance
wait time 200	;wait time to analyze 'standstill' stability	wait time = Delay period
MOVED -0.25	;move opposite direction 1 rev	
wait time 200	;wait time to analyze 'standstill' stability	

Adjust these parameters if required to best suit the application before going to the next step.

4) Compile and Download Indexer Program to Drive

In the [Indexer Program] folder in MotionView, select [Compile and Load with Source] from the pull down menu. The TuneP program will be compiled and sent to the drive. Select [Run] from the pull down menu to run the TuneP program. Do NOT enable the drive (via input A3) at this stage.

Operation

5) Oscilloscope Settings

Open the [Tools] folder in MotionView and select the [Oscilloscope] tool. Click the [Set on Top] box to place a checkmark in it and keep the scope on top.

In the Scope Tool Window, make the following settings:

Channel 1:

Signal = "Position Error"

Scale = as appropriate to the Error that results once the TuneP program is run.

Channel 2:

Signal = "Target Position"

Scale = as appropriate to the position move generated by the TuneP program

Timebase: = as appropriate to the "Period" of the moves being generated.

Trigger: = Channel 1, Rising Edge.

Level: = 10 Pulses

6) Compensation Folder

Open the [Compensation] folder in MotionView.

Leave the Velocity P-Gain and Velocity I Gain unchanged, as they should already have been setup during velocity tuning. Do not adjust the Gain Scaling Parameter during this procedure.

Set the [Position P-gain] to a low value (e.g. 100) and set the [Position I-Gain] and [Position D-Gain] to 0.

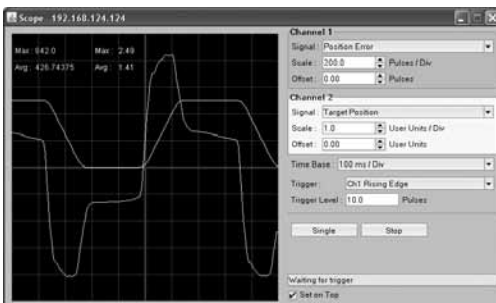
7) Gain Tuning

The system should now be ready to start tuning the position loop. Start the Oscilloscope by clicking [Run]. Apply the Enable input A3 to enable the drive.

The general goal in tuning the position loop is to achieve the minimum position error while maintaining system stability. Some experimentation with gain values will be required to achieve the best performance for the application.

Step 1: Setting the Position P-Gain

Slowly increase the Position P-Gain while watching the position error waveform on oscilloscope Channel 1. It is important to watch both the Max Error as well as the Average Error. While increasing Position P-gain, it should be apparent that both the Max Error as well as the Average Error decrease.



Position P-Gain set too LOW
Large Position Error occurring and large error
in final positioning achieved

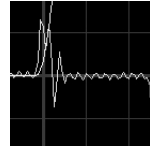


Increased Position P-Gain
Shows improvement to the maximum error
and the final positioning accuracy

At some point while increasing the P-Gain, additional oscillations (Average Error) will start to appear on the position error waveform.



Further Increased Position P-Gain
Shows very good reduction to the maximum
error but with additional oscillations starting
to occur.

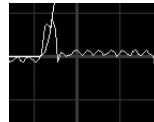


Step 2: Setting the Position D-Gain

Slowly increase the D-Gain while watching the position error waveform on oscilloscope Channel 1. As the D-Gain is increased, the position error oscillation caused by the P-Gain, should start to decrease. Continue to increase the D-Gain until oscillation is gone or until D-Gain is no longer having any apparent effect.



Adjustment of Position D-Gain
in conjunction with the P-Gain dampens
out additional oscillations while improving
position error.

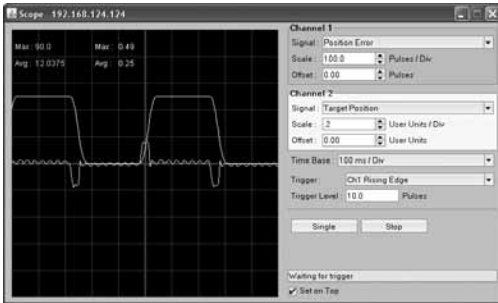


For optimum tuning, it is sometimes required to repeat the process of increasing the P-Gain until a slight oscillation occurs and then increase the D-Gain to suppress that oscillation. This procedure can be repeated until the increasing of D-Gain has negligible effect on the position error waveform.

Operation

Step 3: Setting the Position I-Gain and Position I-Gain Limit

The objective here is to minimize the position error during steady state operation and improve positioning accuracy. Start to increase the Position I-gain. Increasing the I-gain will increase the drive's reaction time while the I-Limit will set the maximum influence that the I-Gain can have on the Integral loop. When adjusting the I-gain start with a very small value for the I-gain (e.g. 1) then increase the I-gain parameter value until stand-still error is compensated and positioning accuracy is satisfactory. Remember that large values of Position I-limit can cause a large instability in the control loop and unsettled oscillation of the system mechanics.



Position Error trace following the tuning of Position P-, I- and D-Gains

Step 4: Check Motor Currents

Set the oscilloscope channel 2 to 'Phase Current RMS'

Channel 2:

Signal = "Phase Current RMS"

Scale = as appropriate to peak current limit set in drive parameters (MotionView)

Timebase: = as appropriate to the "Period" of the moves being generated

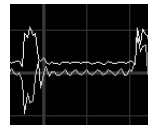
Trigger: = Ch1 Rising Edge

Level: = 10 Pulses

Observe the Current waveform to make sure that there are no significant oscillations during the steady state sections of the position profile (times when target position is not changing). If so then decrease the gains values until the oscillations are either removed or reduced to an acceptable level.



Minimal oscillation when motor positioned to target position.





8) Setting the Position Error Limits

Look at the position error waveform on the oscilloscope. Note the maximum time that position errors exist (from the time axis of the scope) and the maximum peak errors being seen (from the value at the top of the screen). Use these values to set the position error limits to provide suitable position error protection for the application.

Open the 'Limits' folder and 'Position Limits' sub-folder within the MotionView node tree and set suitable values for the 'Position Error' and 'Max Error Time' parameters.



Maximum error and time period for error existing.

Max : 95.0
Avg : 11.85



Time Base : 100 ms / Div

In this particular example maximum error in pulses is 95.0. The time this peak error occurs can be read from the oscilloscope at approximately $\frac{1}{2}$ of a division with each division equal to 100ms, hence the error pulse lasts approximately 50ms. Suitable settings for position error within this application might be as follows, although looser or tighter limits could be applied depending on the requirements of the application.

Description	Value
Position Error	100
Max Error Time	50

9) End Tuning

Remove the Enable Input from input A3 (disable the drive).

Click on the [Indexer Program] folder in MotionView. Click the [Reset] button at the top of the indexer programming screen.

Tuning is now complete.



7 Quick Start Reference

This section provides instructions for External Control, Minimum Connections and Parameter Settings to quickly setup a PositionServo drive for External Torque, Velocity or Positioning Modes. The sections are NOT a substitute for reading the entire PositionServo User Manual. Observe all safety notices in this manual.

7.1 Quick Start - External Torque Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector - Pin	Input Name	Description
P3-22	ACOM	Analog Common Reference from Controller
P3-24	AIN1+	Analog Torque Reference from Controller – Positive
P3-25	AIN1-	Analog Torque Reference from Controller – Negative
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (If required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z-	Buffered Encoder Output
P3-23	AO	Analog Output
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter

Mandatory Parameter Settings:

These Parameters are required to be set prior to running the drive

Folder / Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Torque]
	Reference	Set to [External]
IO / Analog IO	Analog Input (Current Scale)	Set to required current per 1V input from controller
	Analog Input Dead band	Set zero torque Dead band in mV
	Analog Input Offset	Set Analog Offset for Torque Reference
IO / Digital IO	Enable Switch Function	Set to [Run]



Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of a Zero Speed Output
	At Speed	Set Target Speed for activation of a At Speed Output
	Speed Window	Set bandwidth for activation of a At Speed Output

7.2 Quick Start - External Velocity Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector - Pin	Input Name	Description
P3-22	ACOM	Analog Common Reference from Controller
P3-24	AIN1+	Analog Velocity Reference from Controller – Positive
P3-25	AIN1-	Analog Velocity Reference from Controller – Negative
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (If required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z-	Buffered Encoder Output
P3-23	A0	Analog Output
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter



Reference

Mandatory Parameter Settings:

These parameters are required to be set prior to running the drive.

Folder/Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Velocity]
	Reference	Set to [External]
	Enable Velocity Accel / Decel Limits	Enable Ramp rates for Velocity Mode
	Velocity Accel Limit	Set required Acceleration Limit for Velocity command
	Velocity Decel Limit	Set required Deceleration Limit for Velocity command
IO / Analog IO	Analog Input (Velocity Scale)	Set to required velocity per 1 volt input from controller
	Analog Input Dead band	Set zero velocity Dead band in mV
	Analog Input Offset	Set Analog Offset for velocity Reference
IO / Digital IO	Enable Switch Function	Set to [Run]
Compensation (see tuning section)	Velocity P-Gain	Set P-Gain for Velocity loop
	Velocity I_Gain	Set I-Gain for Velocity loop
	Gain Scaling	Set Gain Scaling Parameter

Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of Zero Speed Output
	At Speed	Set Target Speed for activation of At Speed Output
	Speed Window	Set bandwidth for activation of At Speed Output



7.3 Quick Start - External Positioning Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector-Pin	Input Name	Description
P3-1	MA+	Position Reference Input for Master Encoder / Step-Direction Input
P3-2	MA-	Position Reference Input for Master Encoder / Step-Direction Input
P3-3	MB+	Position Reference Input for Master Encoder / Step-Direction Input
P3-4	MB-	Position Reference Input for Master Encoder / Step-Direction Input
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (If required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z-	Buffered Encoder Output
P3-22	ACOM	Analog Common Reference from Controller
P3-23	AO	Analog Output
P3-27	IN_A1	Positive Limit Switch: Required if Limit Switch Function is used
P3-28	IN_A2	Negative Limit Switch: Required if Limit Switch Function is used
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter



Reference

Mandatory Parameter Settings:

These parameters are required to be set prior to running the drive.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Position]
	Reference	Set to [External]
	Step Input Type	Set to [S/D] or [Master Encoder]. (S/D = Step + Direction)
	System to Master Ratio	Set 'Master' and 'Slave' values to gear position input pulses to pulse revolution of the motor shaft
IO / Digital IO	Enable Switch Function	Set to [Run]
Limits / Position Limits	Position Error	Set Position Error Limit specific to application
	Max Error Time	Set Position Error Time specific to application
Compensation (see tuning section)	Velocity P-Gain	Set P-Gain for Velocity loop
	Velocity I_Gain	Set I-Gain for Velocity loop
	Position P-Gain	Set P-Gain for Position Loop
	Position I-Gain	Set I-Gain for Position Loop
	Position D-Gain	Set D-Gain for Position Loop
	Position I-Limit	Set I-Limit for Position Loop
	Gain Scaling	Set Gain Scaling Parameter

Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
	Hard Limit Switch Actions	Set if Hard Limit Switches used in Application
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of a Zero Speed Output
	At Speed	Set Target Speed for activation of a At Speed Output
	Speed Window	Set bandwidth for activation of a At Speed Output



8 Diagnostics

8.1 Display

The PositionServo drives are equipped with a diagnostic LED display and 3 push buttons to select displayed information and to edit a limited set of parameter values.

Parameters can be scrolled by using the “UP” and “DOWN” (▲▼) buttons. To view a value, press “Enter” (↵). To return back to scroll mode press “Enter” again. After pressing the “Enter” button on editable parameters, the yellow LED “C” (see figure in the next section) will blink indicating that parameter value can be changed. Use “UP” and “DOWN” buttons to change the value. Press “Enter” to store new setting and return back to scroll mode.

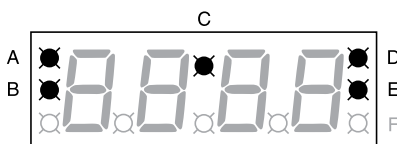
Display	Description
StRt	current drive status - ↵ to view: rUn - drive running d.S - drive disabled F.XX - drive fault. Where XX is the fault code (section 8.3.2)
Hx.xx	Hardware revision (e.g. H2.00)
Fx.xx	Firmware revision (e.g. F2.06)
bRtUd	RS232/RS485(normal mode) baud rate - ↵ to set ▲▼ selects from 2400 to 115200 baudrates
Rdr	Drive's address - ↵ to set ▲▼ sets 0 - 31 drive's address
FtL5	Stored fault's history - ↵ to view ▲▼ scroll through stored faults F0XX - F7XX, “XX” is the fault code (section 8.3.2)
Ht	Heatsink temperature - ↵ to view Shows heatsink temperature in °C if greater than 40°C. Otherwise shows “LO” (low).
EnC	Encoder activity - ↵ to view Shows primary encoder counts for encoder diagnostics activity
HRLl	Displays motor's hall sensor states - ↵ to view Shows motor hall states in form XXX, where X is 1 or 0 - sensor logic states.
boot	0 = Autoboot disabled 1 = Autoboot enabled (Feature available in FW 3.50 or higher)
bU5	Displays drive DC bus voltage - ↵ to view Shows DC bus voltage value
CUrr	Displays motor's phase current (RMS) Shows current value if drive is enabled, otherwise shows “d.S”
CRnb	CAN Baudrate
CRnR	CAN Address
CRno	CAN Operational Mode
CRnd	CAN Delay
CRnE	CAN Enable/disable
dHCP	Ehternet DHCP Configuration: 0="dHCP" is disabled; 1="dHCP" is enabled.
IP_4	IP Adress Octet 4
IP_3	IP Adress Octet 3
IP_2	IP Adress Octet 2
IP_1	IP Adress Octet 1



Diagnostics

8.2 LEDs

The PositionServo has five diagnostic LEDs located around the periphery of the front panel display as shown in the drawing below. These LEDs are designed to help monitor system status and activity as well as troubleshoot any faults.



S913

LED	Function	Description
A	Enable	Orange LED indicates that the drive is ENABLED (running).
B	Regen	Yellow LED indicates the drive is in regeneration mode.
C	Data Entry	Yellow LED will flash when changing.
D	Comm Fault	Red LED illuminates upon a communication fault. (in CANbus only)
E	Comm Activity	Green LED flashes to indicate communication activity.

8.3 Faults

8.3.1 Fault Codes

Listen herein are fault codes caused mostly by hardware operations. Refer to the PositionServo Programming Manual for additional fault codes related to programming.

Fault Code (Display)	Fault	Description
F_0U	Over voltage	Drive bus voltage reached the maximum level, typically due to motor regeneration
F_Fb	Feedback error	Invalid Hall sensors code; Resolver signal lost or at least one motor hall sensor is inoperable or not connected.
F_0C	Over current	Drive exceeded peak current limit. Software incapable of regulating current within 15% for more than 20mS. Usually results in wrong motor data or poor tuning.
F_0t	Over temperature	Drive heatsink temperature has reached maximum rating. Trip Point = 100°C for all drives except 480V 6A & 9A drives Trip Point = 108°C for 480V 6A & 9A drives
F_0S	Over speed	Motor has reached velocity above its specified limit
F_PE	Position Error Excess	Position error has exceeded maximum value.
F_bd	Bad motor data	Motor profile data is invalid or no motor is selected.
F_EP	EPM failure	EPM failure on power up
-EP-	EPM missing	EPM not recognized (connected) on power up
F_09	Motor over temperature	Motor over temperature switch activated; Optional motor temperature sensor (PTC) indicates that the motor windings have reached maximum temperature
F_10	Subprocessor failure	Error in data exchange between processors. Usually occurs when EMI level is high due to poor shielding and grounding.
F_14	Under voltage	Occurs when the bus voltage level drops below 50% of nominal bus voltage while drive is operating. An attempt to enable the drive with low bus voltage will also result in this fault



Fault Code (Display)	Fault	Description
F_15	Hardware overload protection	Occurs when the phase current becomes higher than 400% of total drive's current capability for more than 5µs.
F_18	Arithmetic Error Division by zero	Statement executed within the Indexer Program results in a division by 0 being performed. Drive programming error (error in drive source code).
F_19	Arithmetic Error Register overflow	Statement executed within the Indexer Program results in a value being generated that is too big to be stored in the requested register. Drive programming error (error in drive source code).
F_20	Subroutine stack overflow	Exceeded 32 levels subroutines stack depth. Caused by executing excessive subroutine calls without a RETURN statement. Drive programming error (error in drive source code).
F_21	Subroutine stack underflow	Executing RETURN statement without preceding call to subroutine. Drive programming error (error in drive source code).
F_22	Arithmetic stack overflow	Variable evaluation stack overflow. Expression too complicated for compiler to process. Drive programming error (error in drive source code).
F_23	Motion Queue overflow	32 levels depth exceeded. Drive programming error (error in drive source code).
F_24	Motion Queue underflow	Relates to the MDV statements in the Indexer Program. Drive programming error (error in drive source code).
F_25	Unknown opcode	Byte code interpreter error; May occur when program is missing the closing END statement; when subroutine has no RETURN statement; or if data in EPM is corrupted at run-time
F_26	Unknown byte code	Byte code interpreter error; May occur when program is missing the closing END statement; when subroutine has no RETURN statement; or if data in EPM is corrupted at run-time
F_27	Drive disabled	Attempt to execute motion while drive is disabled. Drive programming error (error in drive source code).
F_28	Accel too high	Motion statement parameters calculate an Accel value above the system capability. Drive programming error (error in drive source code).
F_29	Accel too low	Motion statement parameters calculate an Accel value below the system capability. Drive programming error (error in drive source code).
F_30	Velocity too high	Motion statement parameters calculate a velocity above the system capability. Drive programming error (error in drive source code).
F_31	Velocity too low	Motion statement parameters calculate a velocity below the system capability. Drive programming error (error in drive source code).
F_32	Positive Limit Switch	Positive limit switch is activated. (Only available while drive is in position mode)
F_33	Negative Limit Switch	Negative limit switch is activated. (Only available while drive is in position mode)
F_34	Positive motion w/ Pos Lim Sw ON	Attempt at positive motion with engaged positive limit switch
F_35	Negative motion w/ Neg Lim Sw ON	Attempt at negative motion with engaged negative limit switch
F_36	Drive Disabled by User at Enable Input	The drive is disabled while operating or an attempt is made to enable the drive without deactivating "Inhibit input". "Inhibit" input has reverse polarity
F_39	Positive soft limit reached	Programmed (Soft) absolute limits reached during motion
F_40	Negative soft limit reached	Programmed (Soft) absolute limits reached during motion
F_41	Unknown Variable ID	Attempt to use variable with unknown ID from user program. Drive programming error (error in drive source code).
F_45	2nd Encoder Position Error	Secondary encoder position error has exceeded maximum value



8.3.2 Fault Event

When drive encounters any fault, the following events occur:

- Drive is disabled
- Internal status is set to “Fault”
- Fault number is logged in the drive’s internal memory for later interrogation
- Digital output(s), if configured for “Run Time Fault”, are asserted
- Digital output(s), if configured for READY, are de asserted
- If the display is in the default status mode, the LEDs display F0XX where XX is current fault code.
- “Enable” LED turns OFF

8.3.3 Fault Reset

Fault reset is accomplished by disabling or re-enabling the drive depending on the setting of the “Reset option” parameter (section 5.3.10).

8.4 Troubleshooting



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait at least 60 seconds before servicing drive. Capacitors retain charge after power is removed.

Before troubleshooting

Perform the following steps before starting any procedure in this section:

- Disconnect AC or DC voltage input from the PositionServo. Wait at least 60 seconds for the power to discharge.
- Check the PositionServo closely for damaged components.
- Check that no foreign material has fallen or become lodged in the PositionServo.
- Verify that every connection is correct and in good condition.
- Verify that there are no short circuits or grounded connections.
- Check that the drive’s rated phase current and RMS voltage are consistent with the motor ratings.

For additional assistance, contact your local PositionServo® authorized distributor.

Problem	External line fuse blows
Possible Cause	Line fuses are the wrong size Motor leads or incoming power leads are shorted to ground. <i>Nuisance tripping caused by EMI noise spikes caused by poor grounding and/or shielding.</i>
Suggested Solution	<ul style="list-style-type: none"> • Check that line fuses are properly sized for the motor being used. • Check motor cable and incoming power for shorts. • Check that you follow recommendation for shielding and grounding listed in section “shielding and grounding” early in this manual.



Problem	Ready LED is on but motor does not run.
Suggested Solution	<p>If in Torque or Velocity mode: Reference voltage input signal is not applied. Reference signal is not connected to the PositionServo input properly; connections are open. In MotionView program check <Parameters> <Reference> set to <External></p> <p>For Velocity mode only: In MotionView check <Parameters> <Compensation><Velocity loop filter> P-gain must be set to value more then 0 in order to run. Without load motor will run with P-gain set as low as 20 but under load might not. If P-gain is set to 0 motor will not run at all.</p> <p>In Position mode with master encoder motion source (no program) Reference voltage input signal source is not properly selected. In MotionView program check <Parameters> <Reference> set to <External></p> <p>In Position mode using indexing program Variables ACCEL, DECEL,MAXV, UNITS are not set or set to 0. Before attempting the move set values of motion parameters ACCEL, DECEL,MAXV, UNITS</p>
Problem	In velocity mode, the motor runs away.
Possible Cause	<ul style="list-style-type: none"> • Hall sensors or encoder mis-wired. • PositionServo not programmed for motor connected.
Suggested Solution	<ul style="list-style-type: none"> • Check Hall sensor and encoder connections. • Check that the proper motor is selected.

Notes

Notes

Lenze AC Tech Corporation
630 Douglas Street • Uxbridge, MA 01569 • USA
Sales: (800) 217-9100 • Service (508) 278 9100
www.lenze-actech.com

Document
S94P01G-e1



MotionView®
OnBoard

PositionServo with MVOB
Users Manual
Valid for Hardware Version 2

Copyright ©2013 - 2010 by Lenze AC Tech Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from Lenze AC Tech Corporation. The information and technical data in this manual are subject to change without notice. Lenze AC Tech makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. Lenze AC Tech assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView®, PositionServo®, and all related indicia are either registered trademarks or trademarks of Lenze AG in the United States and other countries.

Windows™ and all related indicia are registered trademarks of Microsoft Corporation.

Java™ and all related indicia are registered trademarks of Sun Microsystems, Incorporated.

CANopen® is a registered trademark of 'CAN in Automation (CiA)'.

DeviceNet™, EtherNet/IP™, and all related indicia are trademarks of ODVA (Open Device Vendors Association).

PROFIBUS DP™ is a registered trademark of PROFIBUS International.



1	Introduction	5
1.1	Safety Information	5
1.2	Legal Regulations	5
1.3	General Drive Information	6
1.3.1	Mains Configuration	6
1.3.2	Operating Modes	6
1.3.3	Feedback	7
1.3.4	Software	7
1.4	Part Number Designation	8
1.4.1	Drive Part Number	8
1.4.2	Filter Part Number	8
1.4.3	Option Part Number	9
2	Technical Data	10
2.1	Electrical Characteristics	10
2.2	Power Ratings	11
2.3	Fuse Recommendations	12
2.4	Digital and Analog I/O Ratings	12
2.5	Environment	12
2.6	Operating Modes	12
2.7	Connections and I/O	13
2.8	PositionServo Dimensions	14
2.9	Clearance for Cooling Air Circulation	15
3	Installation	16
3.1	Wiring	17
3.2	Shielding and Grounding	17
3.2.1	General Guidelines	17
3.2.2	EMI Protection	18
3.2.3	Enclosure	18
3.3	Line Filtering	18
3.4	Heat Sinking	19
3.5	Line (Mains) Fusing	19
4	Interface	20
4.1	External Connectors	20
4.1.1	P1 & P7 - Input Power and Output Power Connections	20
4.1.2	P2 - Ethernet Communications Port	21
4.1.3	P3 - Controller I/O	22
4.1.4	P4 - Motor Feedback	23
4.1.5	P5 - 24 VDC Back-up Power Input	24
4.1.6	P6 - Braking Resistor and DC Bus	24
4.1.7	Connector and Wiring Notes	25
4.1.8	P8 - ISO 13849-1 Safety Circuit (option)	26
4.2	Digital I/O Details	31
4.2.1	Step & Direction/Master Encoder Inputs (P3, pins 1-4)	31
4.2.2	Buffered Encoder Output (P3, pins 7-12)	32
4.2.3	Digital Outputs	32
4.2.4	Digital Inputs	33
4.3	Analog I/O Details	34
4.3.1	Analog Reference Input	34
4.3.2	Analog Output	35
4.4	Communication Interfaces	35
4.4.1	Ethernet Interface (standard)	35
4.4.2	RS485 Interface (option)	35
4.4.3	Modbus RTU Support	36
4.4.4	CANopen Interface	36
4.4.5	DeviceNet Interface	36
4.4.6	PROFIBUS DP Interface	37
4.5	Motor Selection	37
4.5.1	Motor Connection	37
4.5.2	Motor Over-Temperature Protection	37



Contents

5	Parameters	38
5.1	Drive Identification	39
5.1.1	Drive Name	39
5.1.2	Group ID	39
5.2	Motor	40
5.2.1	Motor Setup	40
5.2.2	Using a Custom Motor	41
5.2.3	Creating Custom Motor Parameters	41
5.2.4	Autophasing	42
5.2.5	Custom Motor Data Entry	43
5.3	Parameters	47
5.3.1	Drive Mode	48
5.3.2	Reference	48
5.3.3	Drive PWM Frequency	49
5.3.4	Current Limit	49
5.3.5	To Change Current Limits	49
5.3.6	Peak Current Limit (8 kHz and 16 kHz)	49
5.3.7	Accel/Decel Limits (velocity mode only)	49
5.3.8	Fault Reset	49
5.3.9	Motor Temperature Sensor	50
5.3.10	Motor PTC Cutoff Resistance	50
5.3.11	Regen Duty Cycle	50
5.3.12	Master Encoder Input Type (position mode only)	51
5.3.13	Master Encoder - System to Master Ratio	51
5.3.14	Autoboot	51
5.3.15	User Units	51
5.3.16	Rotation Direction	51
5.3.17	Resolver Tracks	51
5.4	Communication	52
5.4.1	Ethernet	52
5.4.2	RS-485	52
5.4.3	CAN	52
5.4.4	PROFIBUS	52
5.5	Analog I/O	53
5.5.1	Analog Output	53
5.5.2	Analog Output Current Scale (Volt/Amps)	53
5.5.3	Analog Output Velocity Scale (mV/RPM)	53
5.5.4	Analog Input Current Scale (Amps/Volt)	53
5.5.5	Analog Input Velocity Scale (RPM/Volt)	53
5.5.6	Analog Input Dead Band	54
5.5.7	Analog Input Offset	54
5.6	Digital I/O	54
5.6.1	Digital Output	54
5.6.2	Digital Input De-bounce Time	54
5.6.3	Hard Limit Switch Action	54
5.6.4	Enable Switch Function	54
5.6.5	Brake Release Delay	55
5.7	Velocity Limits	55
5.7.1	Zero Speed	55
5.7.2	Speed Window	55
5.7.3	At Speed	55
5.8	Position Limits	56
5.8.1	Position Error	56
5.8.2	Max Error Time	56
5.8.3	Soft Limits	56



5.9	Compensation	56
5.9.1	Velocity P-gain (proportional)	56
5.9.2	Velocity I-gain (integral)	56
5.9.3	Position P-gain (proportional)	57
5.9.4	Position I-gain (integral)	57
5.9.5	Position D-gain (differential)	57
5.9.6	Position I-limit	57
5.9.7	Gain Scaling Window	57
5.9.8	Disable High Performance Mode	57
5.9.9	Auto Tuning	57
5.9.10	Set Default Gains	58
5.9.11	Feedback and Loop Filters	58
5.10	Tools	59
5.10.1	Oscilloscope	59
5.10.2	Parameter & I/O View	60
5.11	Faults	61
5.12	Monitor	62
6	Operation	63
6.1	Minimum Connections	63
6.2	Ethernet Connection	63
6.2.1	PositionServo Ethernet Port Configuration	64
6.2.2	Configuring the PC IP Address (Windows XP)	66
6.2.3	Initial Connection to the Drive	69
6.2.4	Launching MotionView & Communicating to the PS Drive	70
6.3	Parameter Storage and EPM Operation	73
6.3.1	Parameter Storage	73
6.3.2	EPM Operation	73
6.3.3	EPM Fault	73
6.4	Configuration of the PositionServo	74
6.5	Position Mode Operation (gearing)	75
6.6	Enabling the PositionServo	75
6.7	Drive Tuning	76
6.7.1	Auto Tuning the Drive	76
6.7.2	Manually Tuning the Drive in Velocity Mode	77
6.7.3	Manually Tuning the Drive in Position Mode	82
6.8	Upgrading Firmware	87
7	Quick Start Reference	88
7.1	Quick Start - External Torque Mode	88
7.2	Quick Start - External Velocity Mode	89
7.3	Quick Start - External Positioning Mode	91
8	Diagnostics	93
8.1	Diagnostic Display	93
8.2	Diagnostic LEDs	94
8.3	Stop/Reset	94
8.4	Faults	95
8.4.1	Fault Codes	95
8.4.2	Fault Event	97
8.4.3	Fault Reset	97
8.4	Troubleshooting	97

About These Instructions

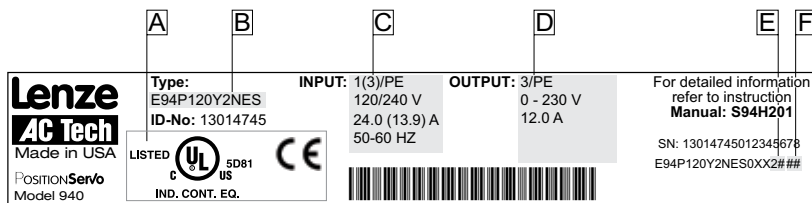
This documentation pertains to the PositionServo drive with Hardware Version 2. This documentation contains important technical data regarding the installation, operation and commissioning of the drive. Observe all safety instructions. Read this document in its entirety before operating or servicing a PositionServo drive.

Drive Hardware Version

For hardware version 2, the drive dataplate (identification label) displays “2” in the fourth to last digit of the drive identification number. Refer to “E” designation in the drive identification label. Upon power-up the drive LED display will read “9402” to indicate 940 PositionServo, hardware version 2.

If upon power-up the drive LED reads “940”, the drive has hardware version 1. Refer to User Manual S94PM01 for hardware version 1 drives.

Drive Identification Label



A	B	C	D	E	F
Certifications	Model Number	Input Ratings	Output Ratings	Hardware Version	Software Version

Package Contents

Scope of Supply	Important
1 Model PositionServo: Type E94P or E94R 1 Mounting Instructions (English) 1 MotionView CD ROM including: - configuration software - documentation	After reception of the delivery, immediately check whether the scope of supply matches the accompanying papers. Lenz- AC Tech does not accept any liability for deficiencies claimed subsequently. Claim: - visible transport damage immediately to the forwarder - visible deficiencies / incompleteness immediately to your Lenz representative.

Related Documents

The documentation listed herein contains information relevant to the operation of the PositionServo and MotionView OnBoard. To obtain the latest documentation, visit the Technical Documentation section of <http://www.lenze.com>.

Table 1: Reference Documentation

Document #	Description
P94MI01	PositionServo (with MVOB) Mounting Instructions
PM94H201	PositionServo (with MVOB) Programming Manual
P94MOD01	Position Servo ModBus RTU over RS485 ; Modbus TCP/IP
P94CAN01	PositionServo CANopen Communications Reference Guide
P94DVN01	PositionServo DeviceNet Communications Reference Guide
P94ETH01	PositionServo EtherNet/IP Communications Reference Guide
P94PFB01	PositionServo PROFIBUS Communications Reference Guide



1 Introduction

1.1 Safety Information

The safety information provided in this documentation has the layout shown herein.



Signal Word! (Characteristics the severity of the danger)

Note (describes the danger and informs on how to proceed)

Table 2: Pictographs used in these Instructions

Icon		Signal Words	
	Warning of hazardous electrical voltage	DANGER!	Warns of impending danger. Consequences if disregarded: Death or severe injuries.
	Warning of a general danger	WARNING!	Warns of potential, very hazardous situations . Consequences if disregarded: Death or severe injuries.
	Warning of damage to equipment	STOP!	Warns of potential damage to material and equipment . Consequences if disregarded: Damage to the controller/drive or its environment.
	Information	NOTE	Designates a general, useful note. If you observe it, handling the controller/drive system is made easier.

1.2 Legal Regulations

Table 2 lists the identification, application, liability, warranty and disposal information for the PositionServo drive.

Table 3: Legal Disclaimers

Claim	Description		
Identification	Nameplate	CE Identification	Manufacturer
	Lenze controllers are unambiguously designated by the contents of the nameplate	In compliance with the EC Low-Voltage Directive	Lenze AC Tech Corporation 630 Douglas Street Uxbridge, MA 01569 USA
Application as directed	E94P or E94R servo controller <ul style="list-style-type: none"> • must only be operated under the conditions prescribed in these Instructions. • are components for: <ul style="list-style-type: none"> - Closed loop control of Velocity, Torque, or Positioning applications with AC synchronous motors. - installation in a machine. - assembly with other components to form a machine. • are electric units for installation in control cabinets or similarly enclosed housing. • comply with the requirements of the Low-Voltage Directive. • are not machines for the purpose of the Machinery Directive. • are not to be used as domestic appliances, but only for industrial purposes. 		
Application as directed	Drive systems with E94P or E94R servo inverters <ul style="list-style-type: none"> • comply with the EMC Directive if they are installed according to the guidelines of CE-typical drive systems. • can be used for: <ul style="list-style-type: none"> - for operation on public and non-public mains - for operation in industrial premises and residential areas. • The user is responsible for the compliance of his application with the EC directives. <p>Any other use shall be deemed as inappropriate!</p>		

Note: Table 3 continued on next page.



Introduction

Claim	Description		
Liability	<ul style="list-style-type: none"> • The information, data, and notes in these instructions met the latest design and implementation of the drive at the time of publication. Claims on modifications referring to controllers that have already been supplied cannot be derived from the information, illustrations, and descriptions. • The specifications, processes and circuitry described in these instructions are for guidance only and must be adapted to your own specific application. Lenze does not take responsibility for the suitability of the process and circuit proposals. • The specifications in these Instructions describe the product features without guaranteeing them. • Lenze does not accept any liability for damage and operating interference caused by: <ul style="list-style-type: none"> - Disregarding the operating instructions - Unauthorized modifications to the controller - Operating errors - Improper working on and with the controller 		
Warranty	• Warranty conditions: refer to Lenze AC Tech Terms and Conditions of Sale, document TD03.		
Disposal	Material	Recycle	Dispose
	Metal	•	-
	Plastic	•	-
	Assembled PCB's	-	•

1.3 General Drive Information

1.3.1 Mains Configuration

The PositionServo is available in four mains (input power) configurations:

1. 120/240V Single Phase (Voltage Doubler) Units

When wired for **Doubler mode** (L1-N), the input is for 120V nominal only and can range from 70 VAC to 132 VAC and the maximum output voltage is double the input voltage. When wired to terminals L1-L2/N, the input can range from 80 VAC to 264 VAC and the maximum output voltage is equal to the input voltage.

2. 120/240V Single Phase (Filtered) Units

120/240V (nominal) single phase input with integrated input mains (line) filter. Actual input voltage range: 80VAC to 264VAC. The maximum output voltage is approximately equal to the input voltage.

3. 120/240V Single or Three Phase Units

120V or 240V (nominal) single or three phase input. Actual input voltage range: 80VAC to 264VAC. The maximum output voltage is approximately equal to the input voltage. An external input mains (line) filter is available.

4. 400/480V Three phase Units

400/480V (nominal) three phase input. An external input mains (line) filter is available. Actual voltage range: 320 - 528 VAC.

1.3.2 Operating Modes

The PositionServo drive can operate in one of three mode settings, torque (current), velocity, or positioning. The drive's command or reference signal can come from one of three sources. The first is an external reference. An external reference can be an analog input signal, a step and direction input or an input from a master encoder. The second reference is an internal reference. An internal reference is when the commanded reference is derived from the drive's user program. The third reference is when the commanded reference is given by a host device over a communications network. This Host device can be an external motion controller, PLC, HMI or PC. The communication network can be over, RS485 (Point-to-Point or Modbus RTU), Modbus over TCP/IP, CANopen (DS301), EtherNet/IP, DeviceNet or PROFIBUS DP.



1.3.3 Feedback

Depending on the primary feedback, there are two types of drives: the Model 940 PositionServo encoder-based drive which accepts an incremental encoder with Hall channel inputs and the Model 941 PositionServo resolver-based drive which accepts resolver inputs. The feedback signal is brought back to the P4 connector on the drive. This connector will be a 15 pin D-sub for the encoder version and a 9 pin D-sub for the resolver version.

1.3.4 Software

MotionView software is the setup and management tool for the PositionServo drive. All parameters can be set and monitored via this software tool. It has a real-time oscilloscope tool for analysis and optimum tuning. The users program, written with SimpleMotion Programming Language (SML), can be utilized to command motion and handle the drive's analog and digital I/O (inputs and outputs). The programming language is a Basic-like language designed to be very intuitive and easy to implement. For programming details, refer to the PositionServo Programming Manual. All PositionServo related manuals can be downloaded from the Technical Documentation section on the Lenze website (<http://www.lenze.com>).

On each PositionServo drive, there is an Electronic Programming Module (EPM), which stores all drive setup and tuning gain settings. This module can be removed from the drive and reinstalled into another drive, making the field replacement of the drive extremely easy. This also makes it easy to duplicate the settings for several drives.

The PositionServo drive supports a variety of communication protocols, including Point-to-Point (PPP), Modbus RTU over RS485, Ethernet TCP/IP, Modbus over TCP/IP, CANopen (DS301), EtherNet/IP, DeviceNet and PROFIBUS DP.



Introduction

1.4 Part Number Designation

The table herein describes the part number designation for the PositionServo drive. The available filter and communication options are detailed in separate tables.

1.4.1 Drive Part Number

E94	P	020	S	1	N	E	M
Electrical Products in the 94x Series							
P = PositionServo Model 940 with Encoder Feedback R = PositionServo Model 941 with Resolver Feedback							
Drive Rating in Amps:							
020 = 2 Amps		090 = 9 Amps					
040 = 4 Amps		100 = 10 Amps					
060 = 6 Amps		120 = 12 Amps					
080 = 8 Amps		180 = 18 Amps					
Input Phase:							
S = Single Phase Input only							
Y = Single or Three Phase Input							
T = Three Phase Input only							
Input Voltage:							
1 = 120 VAC Doubler (120V, 1~ in/ 240V, 3~ out)							
2 = 200/240 VAC							
4 = 400/480 VAC							
Line Filter:							
N = No Line Filter*							
F = Integrated Line Filter							
Secondary Feedback:							
E = Incremental Encoder							
R = Standard Resolver							
Safety Option:							
M = MotionView OnBoard, no ISO 13849-1 safety compliance							
S = MotionView OnBoard, with ISO 13849-1 safety compliance							

* For 3-phase EMC installation, model 940 EMC footprint/side mount filters are required.

1.4.2 Filter Part Number

E94Z	F	4	T	4	A1
Electrical Option in the 94x Series					
F = EMC Filter					
Filter Current Rating in Amps:					
04 = 4.4 Amps		12 = 12 Amps			
07 = 6.9 Amps		15 = 15 Amps			
10 = 10 Amps		24 = 24 Amps			
Input Phase:					
S = Single Phase					
T = Three Phase					
Max Voltage:					
2 = 240 VAC					
4 = 400/480 VAC					
Degree of Filtering/Variation					
A1 = Industrial/1st Variation					
A2 = Industrial/2nd Variation					



1.4.3 Option Part Number

	E94Z	A	CAN	1
Electrical Option in the 94x Series				
A = Communication or Breakout Module				
Module Type:				
Communication:		Breakout:		
CAN = CANopen COMM Module		HBK = Motor Brake Terminal Module		
RS4 = RS485 COMM Module		TBO = Terminal Block I/O Module		
DVN = DeviceNet COMM Module		SCA = Panel Saver I/O Module		
PFB = PROFIBUS COMM Module				
Variations				
1 = 1st Variation				
2 = 2nd Variation				
3 = 3rd Variation				



2 Technical Data

2.1 Electrical Characteristics

Single-Phase Models					
Type ⁽¹⁾	Mains Voltage ⁽²⁾	1~ Mains Current (doubler)	1~ Mains Current (Std.)	Rated Output Current ⁽⁵⁾	Peak Output Current ⁽⁶⁾
E94_020S1N_~	120V ⁽³⁾ or 240V ⁽⁴⁾	9.7	5.0	2.0	6
E94_040S1N_~		15	8.6	4.0	12
E94_020S2F_~	120 / 240V ⁽⁴⁾ (80 V -0%...264 V +0%)	--	5.0	2.0	6
E94_040S2F_~		--	8.6	4.0	12
E94_080S2F_~		--	15.0	8.0	24
E94_100S2F_~		--	18.8	10.0	30
Single/Three-Phase Models					
Type ⁽¹⁾	Mains Voltage ⁽²⁾	1~ Mains Current	3~ Mains Current	Rated Output Current ⁽⁵⁾	Peak Output Current ⁽⁶⁾
E94_020Y2N_~	120 / 240V ⁽⁴⁾ 1~ or 3~ (80 V -0%...264 V +0%)	5.0	3.0	2.0	6
E94_040Y2N_~		8.6	5.0	4.0	12
E94_080Y2N_~		15.0	8.7	8.0	24
E94_100Y2N_~		18.8	10.9	10.0	30
E94_120Y2N_~		24.0	13.9	12.0	36
E94_180T2N_~	240V 3~ (180 V -0%...264 V +0%)	--	19.6	18.0	54
E94_020T4N_~	400 / 480V 3~ (320 V -0%...528 V +0%)	--	2.7	2.0	6
E94_040T4N_~		--	5.5	4.0	12
E94_060T4N_~		--	7.9	6.0	18
E94_090T4N_~		--	12.0	9.0	27

- (1) The first "_" equals "P" for the 940 encoder based drive or "R" for the 941 resolver based drive.
The second "_" equals "E" for incremental encoder (must have E94P drive) or "R" for the standard resolver (must have E94R drive).
The last digit "~" equals "M" for MV OnBoard and no ISO 13849-1 circuit or "S" for MV OnBoard plus the ISO 13849-1 circuit.
- (2) Mains voltage for operation on 50/60 Hz AC supplies (48 Hz -0% ... 62Hz +0%).
- (3) Connection of 120VAC (70 V ... 132 V) to input power terminals L1 and N on these models doubles the voltage on motor output terminals U-V-W for use with 230VAC motors.
- (4) Connection of 240VAC or 120VAC to input power terminals L1 and L2 on these models delivers an equal voltage as maximum to motor output terminals U-V-W allowing operation with either 120VAC or 230VAC motors.
- (5) Drive rated at 8kHz Carrier Frequency. Derate Continuous current by 17% at 16kHz.
- (6) Peak RMS current allowed for up to 2 seconds. Peak current rated at 8kHz. Derate by 17% at 16kHz.
- (7) Derate rated output current and peak output current by 2.5% for every °C above 40°C up to 55°C maximum.



Electrical Specifications applicable to all models:

Acceleration Time Range (Zero to Max Speed)	0.1 ... 5x10 ⁶ RPM/sec
Deceleration Time Range (Max Speed to Zero)	0.1 ... 5x10 ⁶ RPM/sec
Speed Regulation (typical)	± 1 RPM
Input Impedance (AIN+ to COM and AIN+ to AIN-)	47 kΩ
Power Device Carrier Frequency (sinusoidal commutation)	8, 16 kHz
Power Supply (max)	+5 VDC @ 300 mA
Maximum Encoder Feedback Frequency	2.1 MHz (per channel)
Maximum Output Frequency (to motor)	400 Hz
Resolver Carrier Frequency	4.5 - 5.5kHz (5kHz nom)
Resolver Turns Ratio: Reference to SIN/COS signal	2:1
Resolver Voltage	10V peak to peak
Maximum Resolver Feedback Speed	6500 rpm

2.2 Power Ratings

Type ⁽¹⁾	Output Power at Rated Output Current (8kHz) ⁽²⁾	Leakage Current	Power Loss at Rated Output Current (8kHz)	Power Loss at Rated Output Current (16 kHz) ⁽³⁾
Units	kVA	mA	Watts	Watts
E94_020S1N_~	0.8	Typically >3.5 mA. Consult factory for applications requiring <3.5 mA.	19	21
E94_040S1N_~	1.7		29	30
E94_020S2F_~	0.8		19	21
E94_040S2F_~	1.7		29	30
E94_080S2F_~	3.3		61	63
E94_100S2F_~	4.2		80	85
E94_020Y2N_~	0.8		19	21
E94_040Y2N_~	1.7		29	30
E94_080Y2N_~	3.3		61	63
E94_120Y2N_~	5.0		114	129
E94_180T2N_~	7.5		171	195
E94_020T4N_~	1.7		31	41
E94_040T4N_~	3.3		50	73
E94_060T4N_~	5.0		93	122
E94_090T4N_~	7.5		138	182

- (1) The first "_" equals "P" for the Model 940 encoder based drive or "R" for the Model 941 resolver based drive.
The second "_" equals "E" for incremental encoder (must have E94P drive) or "R" for the standard resolver (must have E94R drive).
The last digit "~" equals "M" for MV OnBoard and no ISO 13849-1 circuit or "S" for MV OnBoard plus the ISO 13849-1 circuit.
- (2) At 240 VAC line input for drives with suffixes "S1N", "S2F", "Y2N". At 480 VAC line input for drives with suffixes "T4N".
- The output power is calculated from the formula: output kVA = $[(\sqrt{3}) \times U_{L-L} \times I_{max}] / 1000$
 - The actual output power (kW) depends on the motor in use due to variations in motor rated voltage, rated speed and power factor, as well as actual max operating speed and desired overload capacity.
 - Typical max continuous power (kW) for PM servo motors run 50-70% of the kVA ratings listed.
- (3) At 16 kHz, de-rate continuous current by 17%



2.3 Fuse Recommendations

Type ⁽¹⁾	AC Line Input Fuse (1 ϕ /3 ϕ)	Miniature Circuit Breaker ⁽⁴⁾ (1 ϕ /3 ϕ)	AC Line Input Fuse or Breaker ^{(5) (6)} (N. America)	DC Bus Input Fuse ⁽⁷⁾
Amp Ratings				
E94_020S1N_~	M20/M10	C20/C10	20/10	10
E94_040S1N_~	M32/M20	C32/C20	30/20	20
E94_020S2F_~	M20	C20	20	15
E94_040S2F_~	M20	C20	20	20
E94_080S2F_~	M32	C32	32	40
E94_100S2F_~	M40	C40	40	45
E94_020Y2N_~	M20/M16	C20/C16	20/15	15
E94_040Y2N_~	M20/M16	C20/C16	20/15	20
E94_080Y2N_~	M32/M20	C32/C20	30/20	40
E94_120Y2N_~	M50/M32	C50/C32	50/30	55
E94_180T2N_~	M40	C40	40	80
E94_020T4N_~	M10	C10	10	10
E94_040T4N_~	M10	C10	10	20
E94_060T4N_~	M20	C20	20	30
E94_090T4N_~	M25	C25	25	40

- (1) The first "_" equals "P" for the Model 940 encoder based drive or "R" for the Model 941 resolver based drive.
The second "_" equals "E" for incremental encoder (must have E94P drive) or "R" for the standard resolver (must have E94R drive).
The last digit "--" equals "M" for MV OnBoard and no ISO 13849-1 circuit or "S" for MV OnBoard plus the ISO 13849-1 circuit.
- (4) Installations with high fault current due to large supply mains may require a type D circuit breaker.
- (5) UL Class CC or T fast-acting current-limiting type fuses, 200,000 AIC, preferred. Bussman KTK-R, JIN, JJS or equivalent.
- (6) Thermal-magnetic type breakers preferred.
- (7) DC-rated fuses, rated for the applied voltage. Examples Bussman KTM or JIN as appropriate.

2.4 Digital and Analog I/O Ratings

I/O	Scan Times	Linearity	Temperature Drift	Offset	Current	Input Impedance	Voltage Range
Units	μ S	%	%	%	mA	Ohm	VDC
Digital Inputs⁽¹⁾	512				Depend on load	2.4 k ⁽²⁾	5-24
Digital Outputs	512				100 max	N/A	30 max
Analog Inputs	512	± 0.013	0.1% per °C rise	± 0 adjustable	Depend on load	47 k	± 10
Analog Outputs	512		0.1% per °C rise	± 0 adjustable	10 max	N/A	± 10

- (1) Inputs do not have scan time. Their values are read directly by indexer program statement.
De-bounce time is programmable and can be set as low as 0. Propagation delay is typical 20 μ s
- (2) Input Impedance is 1.2k Ω for drive with Hardware Revision 2A.

2.5 Environment

Vibration	2 g (10 - 2000 Hz)
Ambient Operating Temperature Range	0 to 40°C (Derate rated output current and peak output current by 2.5% for every °C above 40°C up to 55°C)
Ambient Storage Temperature Range	-10 to 70°C
Temperature Drift	0.1% per °C rise
Humidity	5 - 90% non-condensing
Altitude	1500m/5000ft [derate by 1% per 300m (1000 ft) above 1500m (5000 ft)]

2.6 Operating Modes

Torque Reference	± 10 VDC 12-bit; scalable
Torque Range	100:1
Current-Loop Bandwidth	Up to 1.5 kHz*

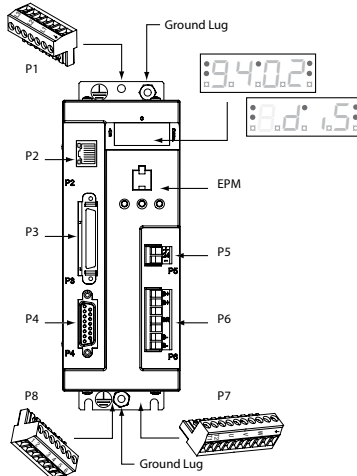


Velocity	
Reference	± 10 VDC or 0...10 VDC; 12-bit; scalable
Regulation	± 1 RPM
Velocity-Loop Bandwidth	Up to 200 Hz*
Speed Range	5000:1 with 5000 ppr encoder
Position	
Reference	0...2 MHz Step & Direction or 2 channels quadrature input; scalable
Minimum Pulse Width	500 nanoseconds
Loop Bandwidth	Up to 150 Hz*
Accuracy	±1 encoder count for encoder feedback ±1.32 arc-minutes for resolver feedback (14-bit resolution)

* = motor and application dependent

2.7 Connections and I/O

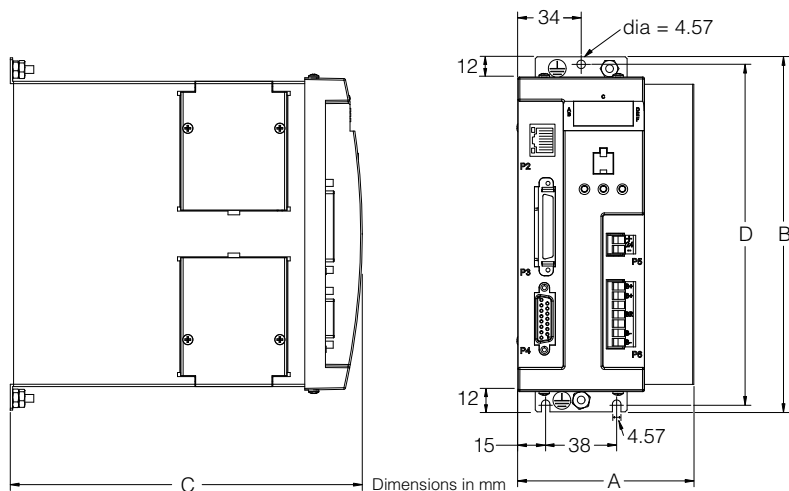
Mains Power	3-pin or 4-pin removable terminal block	(P1)
Ethernet Port	Standard RJ45 Connector	(P2)
I/O Connector	Standard 50-pin SCSI	(P3)
- Buffered Encoder Output	A, B, & Z channels with compliments (5V @ 20mA)	(P3)
- Digital Inputs	11 programmable plus 1 dedicated (5-24V)	(P3)
- Digital Outputs	4 programmable plus 1 dedicated (5-24V @ 100mA)	(P3)
- Analog Input	2 differential; ±10 VDC (12-bits each)	(P3)
- Analog Output	1 single ended; ±10 VDC (10-bit)	(P3)
- Position Reference Input	Step & Direction or Master Encoder (TTL)	(P3)
Encoder Feedback (E94P drive)	Feedback connector, 15-pin D-shell	(P4)
Resolver Feedback (E94R drive)	Feedback connector, 9-pin D-shell	(P4)
24VDC Power "Keep Alive"	2-pin removable terminal block	(P5)
Regen and Bus Power	5-pin removable terminal block	(P6)
Motor Power	6-pin pin removable terminal block	(P7)
ISO 13849-1 Safety Circuit (option)	6-pin quick connect terminal block	(P8)
RS485 Option Module	3-pin terminal block (installed in Option Bay 1)	(P21)
CAN Option Module	3-pin terminal block (installed in Option Bay 1)	(P21)
DeviceNet Option Module	5-pin terminal block (installed in Option Bay 1)	(P23)
PROFIBUS Option Module	9-pin D-shell connector (installed in Option Bay 1)	(P24)
MotionView OnBoard	Embedded Software (Java-based)	
Maximum Servo Cable Length	20 meters (10m if EN55011 compliance required, see 3.2.1)	





Technical Data

2.8 PositionServo Dimensions



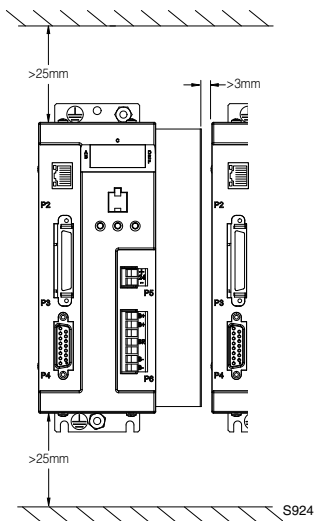
S923

Type ⁽¹⁾	A (mm)	B (mm)	C (mm)	D (mm)	Weight (kg)
E94_020S1N_~	68	190	190	182	1.1
E94_040S1N_~	69	190	190	182	1.2
E94_020S2F_~	68	190	235	182	1.3
E94_040S2F_~	69	190	235	182	1.5
E94_080S2F_~	87	190	235	182	1.9
E94_100S2F_~	102	190	235	182	2.2
E94_020Y2N_~	68	190	190	182	1.3
E94_040Y2N_~	69	190	190	182	1.5
E94_080Y2N_~	95	190	190	182	1.9
E94_100Y2N_~	114	190	190	182	2.2
E94_120Y2N_~	68	190	235	182	1.5
E94_180T2N_~	68	242	235	233	2.0
E94_020T4N_~	68	190	190	182	1.5
E94_040T4N_~	95	190	190	182	1.9
E94_060T4N_~	68	190	235	182	1.4
E94_090T4N_~	68	242	235	233	2.0

- (1) The first "_" equals "P" for the Model 940 encoder based drive or "R" for the Model 941 resolver based drive.
 The second "_" equals "E" for incremental encoder (must have E94P drive) or "R" for the standard resolver (must have E94R drive).
 The last digit "~" equals M" for MV OnBoard and no ISO 13849-1 circuit or "S" for MV OnBoard plus the ISO 13849-1 circuit.



2.9 Clearance for Cooling Air Circulation





3 Installation

Perform the minimum system connection. Refer to section 6.1 for minimum connection requirements. Observe the rules and warnings below carefully:



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.



STOP!

- The PositionServo must be mounted vertically for safe operation and to ensure enough cooling air circulation.
- Printed circuit board components are sensitive to electrostatic fields. Avoid contact with the printed circuit board directly. Hold the PositionServo by its case only.
- Protect the drive from dirt, filings, airborne particles, moisture, and accidental contact. Provide sufficient room for access to the terminal block.
- Mount the drive away from any and all heat sources. Operate within the specified ambient operating temperature range. Additional cooling with an external fan may be required in certain applications.
- Avoid excessive vibration to prevent intermittent connections
- DO NOT connect incoming (mains) power to the output motor terminals (U, V, W)! Severe damage to the drive will result.
- Do not disconnect any of the motor leads from the PositionServo drive unless (mains) power is removed. Opening any one motor lead may cause failure.
- Control Terminals provide basic isolation (insulation per EN 61800-5-1). Protection against contact can only be ensured by additional measures, e.g., supplemental insulation.
- Do not cycle mains power more than once every 2 minutes. Otherwise damage to the drive may result.



WARNING!

For compliance with EN 61800-5-1, the following warning applies.

This product can cause a d.c. current in the protective earthing conductor. Where a residual current-operated protective (RCD) or monitoring (RCM) device is used for protection in case of direct or indirect contact, only an RCD or RCM of Type B is allowed on the supply side of this product.



UL INSTALLATION INFORMATION

- Suitable for use on a circuit capable of delivering not more than 200,000 rms symmetrical amperes, at the maximum voltage rating marked on the drive.
- Use Class 1 wiring with minimum of 75°C copper wire only.
- Shall be installed in a pollution degree 2 macro-environment.
- The PositionServo does not provide motor over-temperature protection. The user may connect a KTY motor thermal sensor to the drive as detailed in section 4.1.1 and 4.5.2 if necessary to satisfy NEC requirements.



3.1 Wiring

**DANGER!**

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing the drive. Capacitors retain charge after power is removed.

**WARNING!**

Leakage current may exceed 3.5mA AC. Minimum size of the protective earth conductor shall comply with local safety regulations for high leakage current equipment.

**STOP!**

Under no circumstances should power and control wiring be bundled together. Induced voltage can cause unpredictable behavior in any electronic device, including motor controls.

**WARNING!**

The PositionServo drive runs on phase-to-phase voltage. For the standard drive, either a delta or wye transformer may be used for 3-phase input. However, for reinforced insulation of user accessible I/O circuits, each phase voltage to ground must be less than or equal to 300VAC rms. This means that the power system must use center grounded wye secondary configuration for 400/480VAC mains.

Refer to section 4.1.1 for Power wiring specifications.

3.2 Shielding and Grounding

3.2.1 General Guidelines

Lenze recommends the use of single-point grounding (SPG) for panel-mounted controls. Serial grounding (a “daisy chain”) is not recommended. The SPG for all enclosures must be tied to earth ground at the same point. The system ground and equipment grounds for all panel-mounted enclosures must be individually connected to the SPG for that panel using 14 AWG (2.5 mm²) or larger wire.

In order to minimize EMI, the chassis must be grounded to the mounting. Use 14 AWG (2.5 mm²) or larger wire to join the enclosure to earth ground. A lock washer must be installed between the enclosure and ground terminal. To ensure maximum contact between the terminal and enclosure, remove paint in a minimum radius of 0.25 in (6 mm) around the screw hole of the enclosure.

Lenze recommends the use of the special PositionServo drive cables provided by Lenze. If you specify cables other than those provided by Lenze, please make certain all cables are shielded and properly grounded.

It may be necessary to earth ground the shielded cable. Ground the shield at both the drive end and at the motor end.

If the PositionServo drive continues to pick up noise after grounding the shield, it may be necessary to add an AC line filtering device and/or an output filter (between the drive and servo motor).



Installation

EMC

Compliance with EN 61800-3:2004

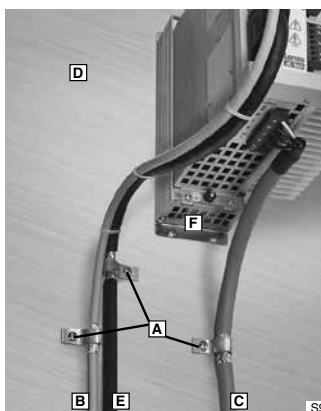
In a domestic environment this product may cause radio interference. The user may be required to take adequate measures

Noise emission

Drive Models ending in the suffix "2F" are in compliance with class A limits according to EN 55011 if installed in a control cabinet and the motor cable length does not exceed 10m. Models ending in "N" will require an appropriate line filter.

- [A] Screen clamps
- [B] Control cable
- [C] Low-capacitance motor cable
(core/core < 75 pF/m, core/screen < 150 pF/m)
- [D] Earth grounded conductive mounting plate
- [E] Encoder/Resolver Feedback Cable
- [F] Footprint or Sidemount Filter (optional)

Installation according to EMC Requirements



3.2.2 EMI Protection

Electromagnetic interference (EMI) is an important concern for users of digital servo control systems. EMI will cause control systems to behave in unexpected and sometimes dangerous ways. Therefore, reducing EMI is of primary concern not only for servo control manufacturers such as Lenze, but the user as well. Proper shielding, grounding and installation practices are critical to EMI reduction.

3.2.3 Enclosure

The panel in which the PositionServo is mounted must be made of metal, and must be grounded using the SPG method outlined in section 3.2.1.

Proper wire routing inside the panel is critical; power and logic leads must be routed in different avenues inside the panel.

You must ensure that the panel contains sufficient clearance around the drive. Refer to section 2.9 suggested cooling air clearance.

3.3 Line Filtering

In addition to EMI/RFI safeguards inherent in the PositionServo design, external filtering may be required. High frequency energy can be coupled between the circuits via radiation or conduction. The AC power wiring is one of the most important paths for both types of coupling mechanisms. In order to comply with IEC 61800-3:2004, an appropriate filter must be installed within 20cm of the drive power inputs.

Line filters should be placed inside the shielded panel. Connect the filter to the incoming power lines immediately after the safety mains and before any critical control components. Wire the AC line filter as close as possible to the PositionServo drive.

**NOTE**

The ground connection from the filter must be wired to solid earth ground, not machine ground.

If the end-user is using a CE-approved motor, the AC filter combined with the recommended motor and encoder feedback cables (maximum cable length of 10m), is all that is necessary to meet the EMC directives listed herein. The end user must use the compatible filter to comply with CE specifications. The OEM may choose to provide alternative filtering that encompasses the PositionServo drive and other electronics within the same panel. The OEM has this liberty because CE requirements are for the total system.

3.4 Heat Sinking

The PositionServo drive contains sufficient heat sinking within the specified ambient operating temperature in its basic configuration. There is no need for additional heat sinking. However, the user must ensure that there is sufficient clearance for proper air circulation. As a minimum, an air gap of 25 mm above and below the drive is necessary.

3.5 Line (Mains) Fusing

External line fuses must be installed on all PositionServo drives. Connect the external line fuse in series with the AC line voltage input. Use fast-acting fuses rated for 250 VAC or 600 VAC (depending on model), and approximately 200% of the maximum RMS phase current. Refer to section 2.3 for fuse recommendations.



Interface

4 Interface

The standard PositionServo drive is equipped with seven connectors including: four quick-connect terminal blocks, one SCSI connector, one subminiature type "D" connector and one ethernet RJ45 connector. These connectors provide communications from a PLC or host controller, power to the drive, and feedback from the motor. Prefabricated cable assemblies may be purchased from Lenze to facilitate wiring the drive, motor and host computer. Contact your Lenze Sales Representative for assistance.

As this manual makes reference to specific pins on specific connectors, the convention PX.Y is used, where X is the connector number and Y is the pin number.

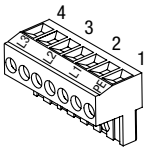
4.1 External Connectors

4.1.1 P1 & P7 - Input Power and Output Power Connections

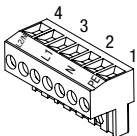
Located on the top of the drive, P1 is a 3 or 4-pin quick-connect terminal block used for input (mains) power. Located on the bottom of the drive, P7 is a 6-pin quick-connect terminal block used for output power to the motor. P7 also has a thermistor (PTC) input for motor over-temperature protection (refer to paragraph 4.5.2). The P1 and P7 connector pin assignments are listed in the tables herein.

P1 Pin Assignments (Input Power)

Standard Models		
Pin	Name	Function
1	PE	Protective Earth (Ground)
2	L1	AC Power in
3	L2	AC Power in
4	L3	AC Power in (3~ models only)

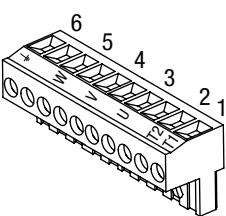


Doubler Models		
Pin	Name	Function
1	PE	Protective Earth (Ground)
2	N	AC Power Neutral (120V Doubler only)
3	L1	AC Power in
4	L2/N	AC Power in (non-doubler operation)



P7 Pin Assignments (Output Power)

Pin	Terminal	Function
1	T1	Thermistor (PTC) Input
2	T2	Thermistor (PTC) Input
3	U	Motor Power Out
4	V	Motor Power Out
5	W	Motor Power Out
6	PE	Protective Earth (Chassis Ground)




DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.



STOP!

DO NOT connect incoming power to the output motor terminals (U, V, W) Severe damage to the PositionServo will result.

Check phase wiring (U, V, W) and thermal input (T1, T2) before powering up drive. If miswired, severe damage to the PositionServo will result.



All conductors must be enclosed in one shield with a jacket around them. The shield on the drive end of the motor power cable should be terminated to the conductive machine panel using screen clamps as shown in section 3.2. The other end should be properly terminated at the motor shield. Feedback cable shields should be terminated in a like manner. Lenze recommends Lenze cables for both the motor power and feedback. These are available with appropriate connectors and in various lengths. Contact your Lenze representative for assistance.

Wire Size

Current A (rms)	Terminal Torque (lb-in)	Wire Size
$I \leq 8$	4.5	16 AWG (1.5mm ²) or 14 AWG (2.5mm ²)
$8 < I \leq 12$	4.5	14 AWG (2.5mm ²) or 12 AWG (4.0mm ²)
$12 < I \leq 15$	4.5	12 AWG (4.0mm ²)
$15 < I \leq 20$	5.0 - 7.0	10 AWG (6.0mm ²)
$20 < I \leq 24$	11.0 - 15.0	10 AWG (6.0mm ²)

4.1.2 P2 - Ethernet Communications Port

P2 is a RJ45 Standard Ethernet connector that is used to communicate with a host computer via Ethernet TCP/IP.

P2 Pin Assignments (Communications)

Pin	Name	Function	
1	+ TX	Transmit Port (+) Data Terminal	
2	- TX	Transmit Port (-) Data Terminal	
3	+ RX	Receive Port (+) Data Terminal	
4	N.C.		
5	N.C.		
6	- RX	Receive Port (-) Data Terminal	
7	N.C.		
8	N.C.		



NOTE

To communicate from the PC directly to the drive a crossover cable is recommended. If using a hub or switch, use a regular patch cable.



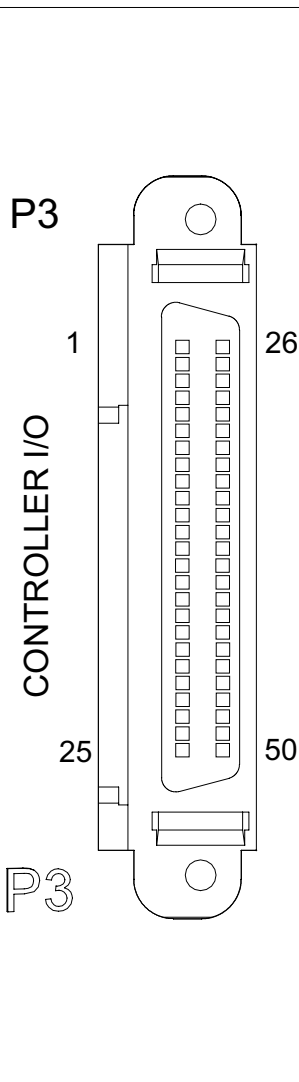
Interface

4.1.3 P3 - Controller I/O

P3 is a 50-pin SCSI connector to interface with the front-end of the controller. It is strongly recommended that OEM cables be used to aid in satisfying CE requirements. Contact your Lenze representative for assistance.

P3 Pin Assignments (Controller Interface)

Pin	Name	Function
1	MA+	Master Encoder A+ / Step+ input ⁽²⁾
2	MA-	Master Encoder A- / Step- input ⁽²⁾
3	MB+	Master Encoder B+ / Direction+ input ⁽²⁾
4	MB-	Master Encoder B- / Direction- input ⁽²⁾
5	GND	Drive Logic Common
6	5+	+5V output (max 100mA)
7	BA+	Buffered Encoder Output: Channel A+ ⁽¹⁾
8	BA-	Buffered Encoder Output: Channel A- ⁽¹⁾
9	BB+	Buffered Encoder Output: Channel B+ ⁽¹⁾
10	BB-	Buffered Encoder Output: Channel B- ⁽¹⁾
11	BZ+	Buffered Encoder Output: Channel Z+ ⁽¹⁾
12	BZ-	Buffered Encoder Output: Channel Z- ⁽¹⁾
13-19		Empty
20	AIN2+	Positive (+) of Analog signal input
21	AIN2-	Negative (-) of Analog signal input
22	ACOM	Analog common
23	AO	Analog output (max 10 mA)
24	AIN1+	Positive (+) of Analog signal input
25	AIN1 -	Negative (-) of Analog signal input
26	IN_A_COM	Digital input group ACOM terminal ⁽³⁾
27	IN_A1	Digital input A1
28	IN_A2	Digital input A2
29	IN_A3	Digital input A3 ⁽³⁾
30	IN_A4	Digital input A4
31	IN_B_COM	Digital input group BCOM terminal
32	IN_B1	Digital input B1
33	IN_B2	Digital input B2
34	IN_B3	Digital input B3
35	IN_B4	Digital input B4
36	IN_C_COM	Digital input group CCOM terminal
37	IN_C1	Digital input C1
38	IN_C2	Digital input C2
39	IN_C3	Digital input C3
40	IN_C4	Digital input C4
41	RDY+	Ready output Collector
42	RDY-	Ready output Emitter
43	OUT1-C	Programmable output #1 Collector
44	OUT1-E	Programmable output #1 Emitter
45	OUT2-C	Programmable output #2 Collector
46	OUT2-E	Programmable output #2 Emitter
47	OUT3-C	Programmable output #3 Collector
48	OUT3-E	Programmable output #3 Emitter
49	OUT4-C	Programmable output #4 Collector
50	OUT4-E	Programmable output #4 Emitter



(1) Refer to Note 1, Section 4.1.7 - Connector and Wiring Notes

(2) Refer to Note 2, Section 4.1.7 - Connector and Wiring Notes

(3) Refer to Note 3, Section 4.1.7 - Connector and Wiring Notes



4.1.4 P4 - Motor Feedback

For encoder-based 940 drives, P4 is a 15-pin DB connector that contains connections for an incremental encoder with Hall emulation tracks or Hall sensors. For synchronous servo motors, Hall sensors or Hall emulation tracks are necessary for commutation. For pin assignments, refer to the Table P4A. Encoder inputs on P4 have 26LS32 or compatible differential receivers for increased noise immunity. Inputs have all necessary filtering and line balancing components so no external noise suppression networks are needed.

For resolver-based 941 drives, P4 is a 9-pin DB connector for connecting resolver feedback and thermal sensor. For pin assignments, refer to the Table P4B. The resolver feedback is translated to 65,536 counts per revolution.

All conductors must be enclosed in one shield with a jacket around them. Lenze recommends that each and every pair (for example, EA+ and EA-) be twisted. In order to satisfy CE requirements, use of an OEM cable is recommended. Contact your Lenze representative for assistance.

The PositionServo buffers encoder/resolver feedback from P4 to P3. For example, when encoder feedback is used, channel A on P4, is Buffered Encoder Output channel A on P3. For more information on this refer to section 4.2.2 "Buffered Encoder Outputs".



STOP!

Use only +5 VDC encoders. Do not connect any other type of encoder to the PositionServo reference voltage terminals. When using a front-end controller, it is critical that the +5 VDC supply on the front-end controller NOT be connected to the PositionServo's +5 VDC supply, as this will result in damage to the PositionServo.

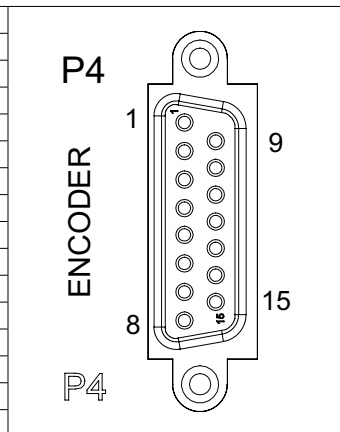


NOTE

- The PositionServo encoder inputs are designed to accept differentially driven hall signals. Single-ended or open-collector type hall signals are also acceptable by connecting "HA+", "HB+", "HC+" and leaving "HA-, HB-, HC-" inputs unconnected. The user does not need to supply pull-up resistors for open-collector hall sensors. The necessary pull-up circuits are already provided.
- Encoder connections (A, B and Z) must be full differential. The PositionServo does not support single-ended or open-collector type outputs from the encoder.
- An encoder resolution of 2000 PPR (pre-quadrature) or higher is recommended.

P4A Pin Assignments (Encoder Feedback - E94P Drives)

Pin	Name	Function
1	EA+	Encoder Channel A+ Input ⁽¹⁾
2	EA-	Encoder Channel A- Input ⁽¹⁾
3	EB+	Encoder Channel B+ Input ⁽¹⁾
4	EB-	Encoder Channel B- Input ⁽¹⁾
5	EZ+	Encoder Channel Z+ Input ⁽¹⁾
6	EZ-	Encoder Channel Z- Input ⁽¹⁾
7	GND	Drive Logic Common/Encoder Ground
8	SHLD	Shield
9	PWR	Encoder supply (+5VDC)
10	HA-	Hall Sensor A- Input ⁽²⁾
11	HA+	Hall Sensor A+ Input ⁽²⁾
12	HB+	Hall Sensor B+ Input ⁽²⁾
13	HC+	Hall Sensor C+ Input ⁽²⁾
14	HB-	Hall Sensor B- Input ⁽²⁾
15	HC-	Hall Sensor C- Input ⁽²⁾



(1) Refer to Note 1, Section 4.1.7 - Connector and Wiring Notes

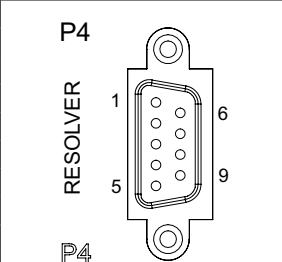
(2) For asynchronous servo motor, an incremental encoder without Hall effect sensors (commutation tracks) can be used.



Interface

P4B Pin Assignments (Resolver Feedback - E94R Drives)

Pin	Name	Function
1	Ref +	Resolver reference connection
2	Ref -	
3	N/C	No Connection
4	Cos+	Resolver Cosine connections
5	Cos-	
6	Sin+	Resolver Sine connections
7	Sin-	
8	PTC+	Motor PTC Temperature Sensor
9	PTC-	




STOP!

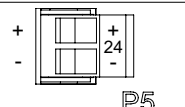
Use only 10 V (peak to peak) or less resolvers. Use of higher voltage resolvers may result in feedback failure and damage to the drive.

4.1.5 P5 - 24 VDC Back-up Power Input

P5 is a 2-pin quick-connect terminal block that can be used with an external 24 VDC (500mA) power supply to provide “Keep Alive” capability: during a power loss, the logic and communications will remain active. Applied voltage must be greater than 20VDC.

P5 Pin Assignments (Back-up Power)

Pin	Name	Function
1	+24 VDC	Positive 24 VDC Input
2	Return	24V power supply return




WARNING!

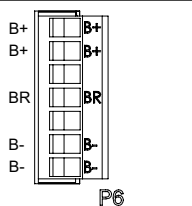
Hazard of unintended operation! When the enable input remains asserted, the “Keep Alive” circuit will restart the motor upon restoration of mains power. If this action is not desired, then remove the enable input prior to re-application of input power.

4.1.6 P6 - Braking Resistor and DC Bus

P6 is a 5-pin quick-connect terminal block that can be used with an external braking resistor (the PositionServo has the regen circuitry built-in). The Brake Resistor connects between the Positive DC Bus (either P6.1 or 2) and P6.3.

P6 Terminal Assignments (Brake Resistor and DC Bus)

Pin	Terminal	Function
1	B+	Positive DC Bus / Brake Resistor
2	B+	
3	BR	Brake Resistor
4	B-	Negative DC Bus
5	B-	




DANGER!

Hazard of electrical shock! Voltage up to 480 VAC above earth ground is possible. Avoid direct contact with live terminals and circuit elements. Disconnect incoming power and wait 60 seconds before opening or servicing the drive. Capacitors retain charge after power is removed.



4.1.7 Connector and Wiring Notes

Note 1 - Buffered Encoder Outputs

Each of the encoder output pins on P3 is a buffered pass-through of the corresponding input signal on P4. Refer to section 4.2.2 "Buffered Encoder Outputs". This can be either from a motor mounted encoder or an encoder emulation of the resolver. The parameter "Resolver Tracks" configures the resolution of the encoder emulation (refer to 5.3.17).

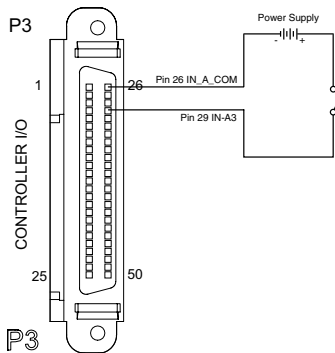
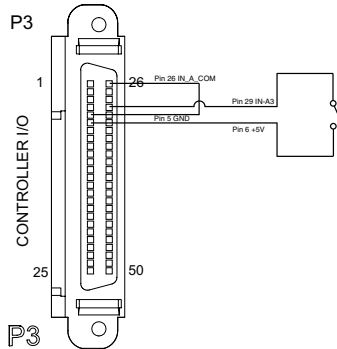
Note 2 - Master Encoder Inputs or Step/Direction Inputs

An external pulse train signal ("step") supplied by an external device, such as a PLC or stepper indexer, can control the speed and position of the servomotor. The speed of the motor is controlled by the frequency of the "step" signal, while the number of pulses that are supplied to the PositionServo determines the position of the servomotor. Direction input controls direction of the motion.

Note 3 - Digital Input A3

For the drive to function, an ENABLE input must be wired to the drive, and should be connected to IN_A3, (P3.29), which is, by the default the ENABLE input on the drive. This triggering mechanism can either be a switch or an input from an external PLC or motion controller. The input can be wired either sinking or sourcing (section 4.2.3). The Enable circuit will accept 5-24V control voltage.

Wiring the ENABLE Switch:





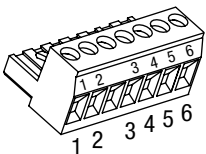
Interface

4.1.8 P8 - ISO 13849-1 Safety Circuit (option)

If installed, the ISO 13849-1 Safety Circuit connector, P8, is located on the bottom of the PositionServo. P8, a 6-pin quick-connect terminal block.

P8 Pin Assignments (ISO 13849-1 Safety Function)

Pin	Name	Function
1	Bypass Voltage	ISO 13849-1 Bypass Voltage (+24VDC)
2	Bypass COM	ISO 13849-1 Bypass Common
3	Safety Status	ISO 13849-1 Safety Status
4	Safety Input1	ISO 13849-1 Safety Input 1 (+24VDC to Enable)
5	Safety COM	ISO 13849-1 Safety Common
6	Safety Input2	ISO 13849-1 Safety Input 2 (+24VDC to Enable)

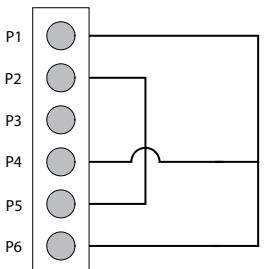



WARNING!

The drive is supplied from the factory with the ISO 13849-1 safety circuit enabled. The drive is not operational until +24V is present at terminals 4 and 6. For the proper safety connections, refer to the "Connection of Two Safety Circuits with External +24V Supply" diagram. Under certain applications when safety connections are not required the drive may be operated with the safety circuit disabled. The diagram below illustrates how to bypass the safety circuit.

Wiring Diagram to Bypass ISO 13849-1 Safety Circuit

Pin	Name	Function
1	Bypass Voltage	ISO 13849-1 Bypass Voltage (+24VDC) *1
2	Bypass COM	ISO 13849-1 Bypass Common *1
3	Safety Status	ISO 13849-1 Safety Status
4	Safety Input1	ISO 13849-1 Safety Input 1 (+24VDC to Enable) *2, *3
5	Safety COM	ISO 13849-1 Safety Common *2, *3
6	Safety Input2	ISO 13849-1 Safety Input 2 (+24VDC to Enable) *2, *3



*1 – This voltage must under no circumstances be used to supply the ISO 13849-1 Safety circuits (terminals 3 to 6). This voltage is intended only for use in bypassing (disabling) the ISO 13849-1 circuits should they not be required.

*2 – A Separate +24VDC supply providing reinforced isolation (SELV or PELV), must be supplied to operate these inputs. This supply should not be floating but should be referenced within 20V peak of PE at the drive.

*3 – Unsnubbed inductive loads must NOT be used on the 24VDC safety circuit wiring.

PositionServo drives with the following "S" designation in the model number have been fitted with the optional ISO 13849-1 Safe Torque Off function.

Drive Model Number:	E94	P	020	S	1	N	E	S
---------------------	-----	---	-----	---	---	---	---	---

The last "S" denotes ISO 13849-1 option fitted to drive at manufacturer.

This option can only be fitted at the factory at the time of unit manufacturer.

This option provides additional methods (Inputs) to disable the drive output so that the drive cannot cause torque to be generated in the motor. This safety function is often referred to as the "Safe Torque Off" function and meets the requirements of the following standard: ISO 13849-1 Safety of Machinery, Safety-related Parts of Control Systems, Category (Cat.) 3, Performance Level (PL) d and Safety Integrity Level (SIL) 2, per EN 61800-5-2 2007.



WARNING!

It is required that all information contained within this ISO 13849-1 standard be observed when implementing any part of this safety circuit functionality with the PositionServo drive.



Operation of the ISO 13849-1 Safety Circuit

ISO 13849-1 Cat 3, PL d designates that the enable function of the drive be designed in such a way that a single fault in any of the parts of this enable circuit cannot lead to a loss of this safety function. The ISO 13849-1 safe torque off function has been designed and certified as meeting the requirements of this standard.

PositionServo drives equipped with the ISO 13849-1 safety circuit option can be used in application requiring conformance to this standard, and also in safety-related applications or in other applications where the integrity of the enable / disable function is paramount to the safety of personnel and machinery.

The ISO 13849-1 safety circuit can interrupt the power supply to the motor without the AC line input to the drive being removed. However, for the purposes of maintenance and mechanical work on the drive system it is recommended that the AC (work swap) Line input be removed and the drives internal bus voltages allowed to discharge before any such work is attempted. The ISO 13849-1 category 3 standard does not provide for electrical safety of all components within the drive system.

For normal operation (enable) of the PositionServo drive, both the Safety Input 1 and Safety Input 2 are required to be active. These inputs act as a Inhibit function, preventing the drive from being enabled until both are active, and causing the drive to disable once either one or both of the inputs are removed. The activation of both inputs will not automatically cause the drive to enable but will allow enable through the standard methods provided for enable of the drive.

If an attempt is made to enable the drive by executing the program statement "ENABLE" or from activating the input IN_A3 with the ISO 13849-1 safety inputs not being present then the drive will generate an ISO 13849-1 Safety Fault (F_EF).

When the drive is disabled through the ISO 13849-1 safety inputs (by removing the +24VDC assertion level to either Safety Input 1 or Safety Input 2 or both while the drive is enabled) the drive output is turned off and further torque cannot be produced by the drive in the motor. The drive will go to the "F_EF" fault condition to indicate disable of the drive was by means of the safety circuits. With the drive output disabled the motor will perform an uncontrolled stop or free-wheel deceleration to stand-still (unless driven by the load). Rotation of the motor will not stop immediately and the time to reach standstill will depend on the inertia contained within the system.



WARNING!

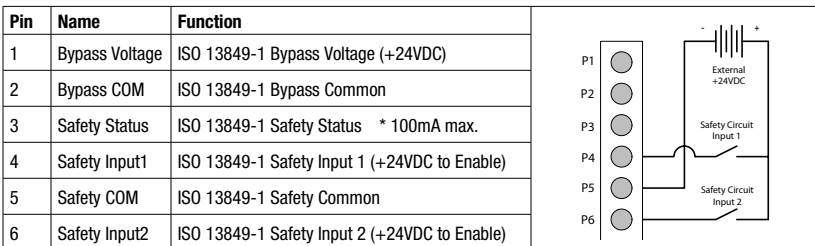
Ensure motion has stopped and the machine is in a safe condition before approaching the application.

If the system is required to be brought to zero speed on loss of the safety circuit function then a motor with a fail-safe mechanical brake should be used and the necessary mechanism implemented.

Due to ISO 13849-1 regulations, a separate +24VDC external dedicated safety power supply must be provided to the drive Safety circuits. The bypass +24V supply is intended for bypass purposes only and must not be used as the control voltage to these circuits.

Installation and Connection

Connection of Two Safety Circuits with External +24V Supply





Interface

Evaluation and Testing of the ISO 13849-1 Safety Circuit

As part of the regulations for ISO 13849-1 safety circuit provision must be made for the user to periodically test the safety circuits and that testing should be capable of identifying a single fault. The PositionServo drive uses the safety status output (Pin 3) in conjunction with the display of the drive to allow the testing of the safety circuits.

The safety status output becomes active to indicate partial or full enable of the safety input circuits 1 and 2. If safety input 1 or safety input 2 or both inputs are on then the safety status output will become active. The safety status output must be connected to some visible indication for the operator to reference during test of the circuit.

As well as being used to test the correct operation of the safety circuits the safety status output can be used as an indicator that the drive has been placed in the fully shut down condition (all safety circuits off). For example, if both Safety Inputs have been Deactivated, the Safety Status is also Deactivated. If one of the Safety Inputs signals failed to call for a shutdown, or if one of the Safety Circuits failed to shut down, the Safety Status signal remains Asserted to alert the operator to the problem.

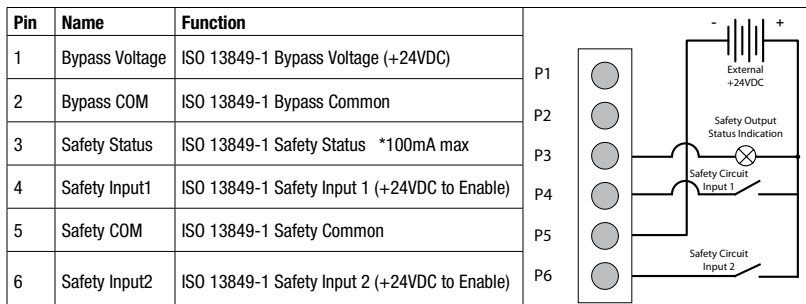
The procedure for testing the ISO 13849-1 safety circuit and the identification of a single fault on the system is given below. The safety status output should be connected to a visible indicator (such as a lamp or LED) so the operator can interpret its condition.



NOTE

Customer must size load so as not to pull more than 100mA.

Safety Status Output Indication



Setting up the Drive in a Maintenance Mode:



WARNING!

During test of the ISO 13849-1 circuit, as laid out in this documentation the drive will go to run (enabled) condition and motion from the motor may be generated. It is the responsibility of the system designer to ensure the system remains in a safe condition during the enclosed maintenance procedure.



Guidance of setting up the drive to allow testing on the ISO 13849-1 circuit:

External Reference:

If the drive is getting its command signal from an external reference then Parameters should be set accordingly.

From the Parameter Folder:

Drive Operating Mode	
Drive Mode	Velocity
Reference	External
Drive PWM Frequency	16 KHZ
Fault Reset	On Disable

From the Digital IO Folder:

Enable Switch Function	Run
------------------------	-----

In this mode your external analog input will command movement. For safety purposes, measures should be made to sure that velocity is at a minimum. From here you can proceed to the ISO 13849-1 Test Procedure.

Internal Reference:

If an Indexer program is used to operate the drive then it must contain a means of placing the drive into a maintenance mode so that the ISO 13849-1 safety circuit can be safely tested. Responsibility lies with the programmer on the safe implementation of a maintenance mode within the indexer program.



WARNING!

If no maintenance mode has been incorporated into the Indexer program then the Indexer program must be erased prior to testing the ISO 13849-1 circuit. Save any code that is required but has not previously been saved and then delete all code from the indexer folder. Press the [Load W Source] button on the program toolbar to remove any residual code from the drive memory.

The following truth table shows logical conditions for ISO 13849-1 circuits.

Safety Input 1	Safety Input 2	Safety Status Output	Drive Display ^{*1}
1	1	1	Run
1	0	1	F_EF
0	1	1	F_EF
0	0	0	F_EF

*1 – Drive display will change to condition shown on enable of the drive (Input A3 Enable)

Place Input A3, hardware enable in the deactivated state.

Test Procedure for ISO 13849-1 Safety Circuit:

Test Step	Action	Drive Display Indication	Safety Status Output Indication	Failed Test Indication
1	Activate both safety circuit inputs 1 & 2. Set Input A3 to Enable	'Run'	'Activated'	Trip on display (F_EF) = one of the safety inputs failed to activate. Status Output Deactivated = Both Safety Inputs Failed to activate
2	Set Input A3 to Disable	'Dis'	'Activated'	Status Output Deactivated = Both Safety Inputs Failed to activate
3	Deactivate Safety Input 1. Set Input A3 to Enable	'F EF'	'Activated'	No Trip on display (F_EF) = Safety Input 1 failed to deactivate. Status Output Deactivated = Safety Input 2 Failed to activate
4	Activate Safety Input 1. Set Input A3 to disable	'Dis'	'Activated'	Status Output Deactivated = Both Safety Inputs Failed to activate



Interface

Test Step	Action	Drive Display Indication	Safety Status Output Indication	Failed Test Indication
5	Deactivate Safety Input 2. Set Input A3 to Enable	'F EF'	'Activated'	No Trip on display (F_EF) = Safety Input 2 failed to deactivate. Status Output Deactivated = Safety Input 1 Failed to activate
6	Set Input A3 to disable	'Dis'	'Activated'	Status Output Deactivated = Both Safety Inputs Failed to activate
7	Deactivate Safety Input 1. Set Input A3 to Enable	'F EF'	'Deactivated'	No Trip on display (F_EF) = Safety Inputs 1 & 2 failed to deactivate. Status Output Activated = Safety Input 1 or Safety Input 2 Failed to deactivate

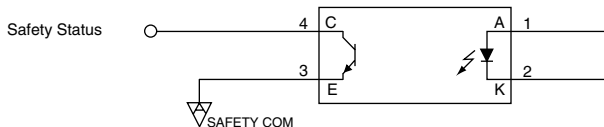
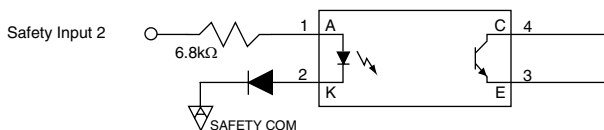
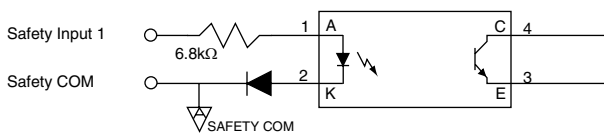
This procedure will evaluate the following conditions:

- All Circuits (safety inputs 1 & 2) working Correctly
- Safety Input 1 failing to activate
- Safety Input 1 failing to deactivate
- Safety Input 2 failing to activate
- Safety Input 2 failing to deactivate
- Both Safety input 1 and 2 failing to activate
- Both Safety input 1 and 2 failing to deactivate

Electrical Characteristics

Safety Input1, Safety Input2 and Safety Status are fully isolated from the rest of the drive circuits as shown in the following diagram.

Safety Inputs	Insulated, compatible with single-ended output (+24VDC) Enable voltage range: 18 to 30VDC Disable voltage range: 0 to 1.0 VDC
Input Impedance	6.8 k Ω
Safety Status	Isolated Open Collector (Grounded Emitter)
Output Load Capability	100mA
Output Max Voltage	30VDC (Collector-Emitter)

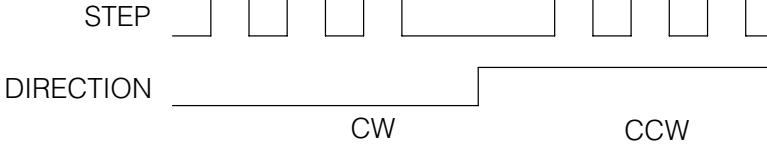




4.2 Digital I/O Details

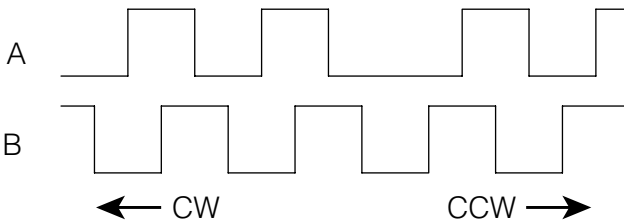
4.2.1 Step & Direction/Master Encoder Inputs (P3, pins 1-4)

A master encoder with quadrature outputs or a step and direction pair of signals can be connected to the PositionServo to control position in the external positioning operating mode. These inputs are optically isolated from the rest of the drive circuits and from each other. Both inputs can operate from any voltage source in the range of 5 to 24 VDC and do not require additional series resistors for normal operation.



Timing Diagram for Step & Direction Signals

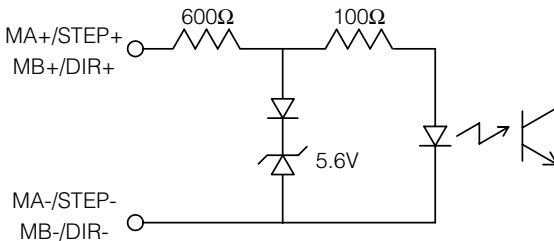
S904



Timing Diagram for Master Encoder Signals

S905

Input type/output compatibility	Insulated, compatible with single-ended or differential outputs (5-24 VDC)
Max frequency (per input)	2 MHz
Min pulse width (negative or positive)	500nS
Input impedance	700 Ω (approx)



S906

Master Encoder Step & Direction Input Circuit

Differential signal inputs are preferred when using Step and Direction. Single ended inputs can be used but are not recommended. Sinking or sourcing outputs may also be connected to these inputs. The function of these inputs "Master Encoder" or "Step and Direction" is software selectable. Use the MotionView set up program to choose the desirable function.



Interface

4.2.2 Buffered Encoder Output (P3, pins 7-12)

There are many applications where it is desired to close the feedback loop to an external device. This feature is built into the PositionServo drive and is referred to as the "Buffer Encoder Output". If a motor with encoder feedback is being used, the A+, A-, B+, B-, Z+ and Z- signals are directly passed through the drive through pins 7-12 with no delays, up to a speed of 2MHz. If a motor with resolver feedback is being used a minimal encoder feedback is transmitted. The default resolution of the simulated encoder is 1024 pulses per revolution, pre-quad. If a different resolution is desired refer to section 5.3.19 "Resolver Tracks". There is a small additional delay when using a resolver. With Encoder pass through the delay is approximately 100nS; with Resolver pass through, the delay is approximately 62uS. Refer to Note 1 in section 4.1.7.

4.2.3 Digital Outputs

There are a total of five digital outputs ("OUT1" - "OUT4" and "RDY") available on the PositionServo drive. These outputs are accessible from the P3 connector. Outputs are open collector/emitter and are fully isolated from the rest of the drive circuits as shown in the figures below. These outputs can be used by the drive's internal User Program or they can be configured as Special Purpose outputs. When used as Special Purpose, each output (OUT1-OUT4) can be assigned to one of the following functions:

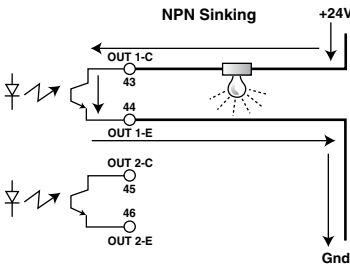
- Not assigned
- Zero speed
- In-speed window
- Current limit
- Run-time fault
- Ready
- Brake (motor brake release)

Note that if an output is assigned as a Special Purpose Output then that output can **not** be utilized by the User Program. The "RDY" Output has a fixed function, "ENABLE", that will become active when the drive is enabled and the output power transistors become energized.

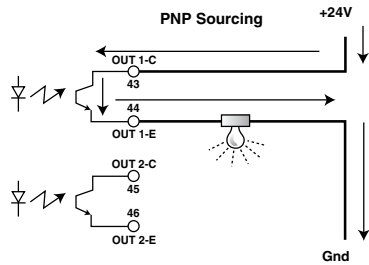
Digital outputs electrical characteristics

Circuit type	Isolated open collector/emitter
Digital outputs load capability	100mA
Digital outputs Collector-Emitter max voltage	30V

The digital outputs have a typical 1 volt leakage. Apply the appropriate relays based on the application. The outputs on the drive can be wired as either sinking (NPN) or sourcing (PNP), as illustrated herein.



mb101



mb102

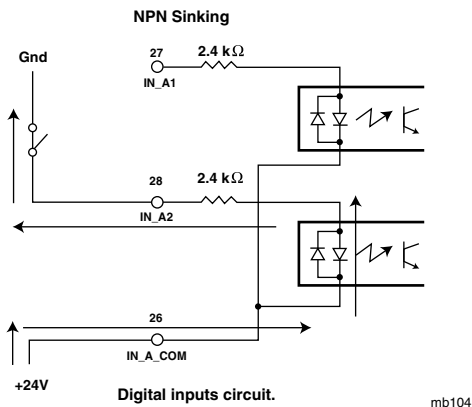
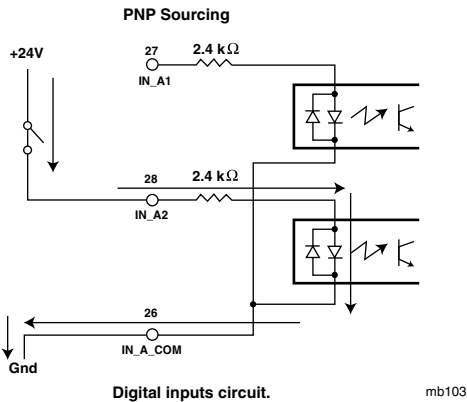


4.2.4 Digital Inputs

IN_Ax, IN_Bx, IN_Cx (P3.26-30, P3.31-35, P3.36-40)

The PositionServo drive has 12 optically isolated inputs. These inputs are compatible with a 5 - 24V voltage source. No additional series resistors are needed for circuit operation. The 12 inputs are segmented into three groups of 4, Inputs A1 - A4, Inputs B1 - B4, and Inputs C1 - C4. Each group, (A, B and C) have their own corresponding shared COM terminal, (ACOM, BCOM and CCOM). Each group or bank can be wired as sinking or sourcing. Refer to the PNP Sourcing and NPN Sinking wiring examples herein. All inputs have a separate software adjustable de-bounce time. Some of the inputs can be set up as Special Purpose Inputs. For example, inputs A1 and A2 can be configured as hardware limit switch inputs, input A3 is always set up as an Enable input and input C3 can be used as a registration input. Refer to the PositionServo Programming Manual for more detail.

For the registration input (C3), the registration time is 3 μ s for an encoder and 7 μ s for a resolver.





Interface

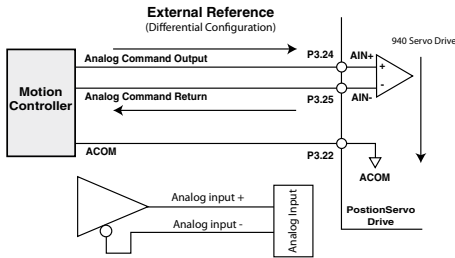
4.3 Analog I/O Details

4.3.1 Analog Reference Input

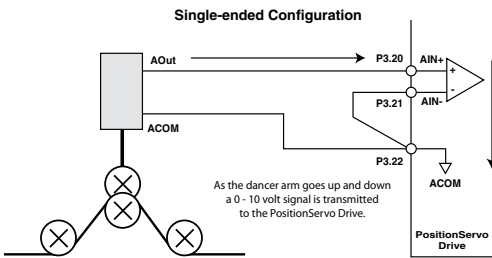
AIN1+, AIN1- (P3.24 and P3.25)

The analog reference input can accept up to a $\pm 10V$ analog signal across AIN1+ and AIN1-. The maximum limit with respect to analog common (ACOM) on each input is $\pm 18VDC$. The analog signal will be converted to a digital value with 12 bit resolution (11-bit plus sign). This input is used to control speed or torque of the motor in velocity or torque mode. The total reference voltage as seen by the drive is the voltage difference between AIN1+ and AIN1-. If used in single-ended mode, one of the inputs must be connected to a voltage source while the other one must be connected to Analog Common (ACOM). If used in differential mode, the voltage source is connected across AIN1+ and AIN1- and the driving circuit common (if any) needs to be connected to the drive Analog Common (ACOM) terminal. Refer to the External Reference and Single-Ended Configuration wiring examples below.

Reference as seen by drive: $V_{ref} = (AIN1+) - (AIN1-)$ and $-10V \leq V_{ref} \leq +10V$



mb105



mb106

AIN2+, AIN2- (P3.20 and P3.21)

The analog reference input can accept up to a $\pm 10V$ analog signal across AIN2+ and AIN2-. The maximum limit with respect to analog common (ACOM) on each input is $\pm 18VDC$. The analog signal will be converted to a digital value with 12 bit resolution (11-bit plus sign). This input is available to the User's program. This input does not have a predefined function.



4.3.2 Analog Output

AO (P3.23)

The analog output is a single-ended signal (with reference to Analog Common (ACOM) which can represent the following motor data:

- Not Assigned
- Phase R Current
- Iq Current
- RMS Phase Current
- Phase S Current
- Id Current
- Peak Phase Current
- Phase T Current
- Motor Velocity

Motor phase U, V and W correspond to R, S and T respectively.

MotionView Setup program can be used to select the signal source for the analog output as well as its scaling.

If the output function is set to "Not Assigned" then the output can be controlled directly from user's program. Refer to the PositionServo Programming Manual for details.



STOP!

Upon application of power to the PositionServo, the Analog Output supplies -10VDC until bootup is complete. Once bootup is complete, the Analog Output will supply the commanded voltage.

4.4 Communication Interfaces

4.4.1 Ethernet Interface (standard)

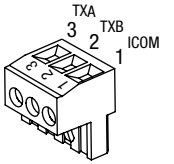
Programming and diagnostics of the drive are performed over the standard Ethernet communication port. The drive's IP address is addressable from the drive's front panel display. The interface supports both 100 BASE-TX as well as 10 BASE-T. This configuration allows the user to monitor and program multiple drives from MotionView. Refer to section 5.4.1 for PC configuration information.

4.4.2 RS485 Interface (option)

PositionServo drives can be equipped with an RS485 communication interface option module (E94ZARS41) that is optically isolated from the rest of the drive's circuitry. The option module can be used for communications to the drive as a Modbus RTU slave or over UPPP protocol. The PositionServo drive supports 7 different baud rates from 2400 to 115200. As a Modbus RTU slave, drives are addressable at up to 247 addresses (repeaters are required above 31 devices on the network). The factory setting for the baud rate is 38,400 with a node address of "1". The drive's address and baud rate can be set from the front panel of the drive or in MotionView.

RS485 Interface Pin Assignments

Pin	Name	Function
1	ICOM	Isolated Common
2	TXB	Transmit B(+)
3	TXA	Transmit A(-)





Interface

4.4.3 Modbus RTU Support

The RS485 interface is configured through the MotionView program. When configured for Modbus operation, the baud rate for RS485 is set using the parameter “RS485 baud rate”. Modbus RTU requires 8 data bits. The Modbus RTU slave interface protocol definitions can be found on the MotionView CD in “Product Manuals”, P94MOD01.



NOTE

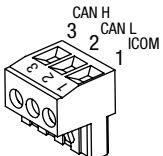
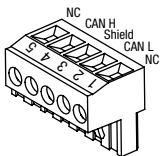
Only one communication option module (RS485, CANopen, DeviceNet or PROFIBUS DP) can be installed in the Option Bay 1 at a time. The COMM modules can be exchanged out and replaced with another of a different type. The Ethernet interface supports Modbus TCP/IP and EtherNet/IP.

4.4.4 CANopen Interface

An optional CANopen communication module (E94ZACAN1) is available for the PositionServo drive. Installed in Option Bay 1 as P21, the CANopen module is optically isolated from the rest of the drive’s circuitry. The 3-pin CANopen module is for HW/SW 1A10 and the 5-pin CANopen module is for HW/SW 1B10 or higher. Refer to the PS CANopen Reference Guide (P94CAN01) for more information.

CANopen Interface Pin Assignments

Pin	Name	Function	Pin	Name	Function
1	ICOM	Isolated Common	1	NC	No connection
2	CAN L	CAN Bus Low	2	CAN L	CAN Bus Low
3	CAN H	CAN Bus High	3	Shield	
			4	CAN H	CAN Bus High
			5	NC	No connection

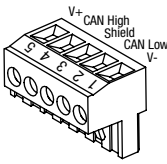



4.4.5 DeviceNet Interface

An optional DeviceNet communication module (E94ZADVN1) is available for the PositionServo drive. Installed in Option Bay 1 as P23, the DeviceNet module is optically isolated from the rest of the drive’s circuitry. The DeviceNet module is a 5-pin quick connect terminal block. Refer to the PS DeviceNet Communications Reference Guide (P94DVN01) for detailed information.

DeviceNet Interface Pin Assignments

Pin	Name	Function
1	V-	0V
2	CAN L	CAN Bus Low (Negative data line)
3	Shield	
4	CAN H	CAN Bus High (Positive data line)
5	V+	11-25VDC power supply

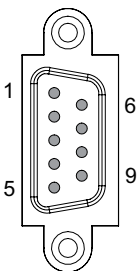




4.4.6 PROFIBUS DP Interface

An optional PROFIBUS DP communication module (E94ZAPFB1) is available for the PositionServo drive. Installed in Option Bay 1 as P24, the PROFIBUS DP module is optically isolated from the rest of the drive's circuitry. The PROFIBUS module is a female DB-9 connector. Refer to the PS PROFIBUS Communications Reference Guide (P94PFB01) for detailed information.

Pin	Name	Function
1	Shield	Cable Shield Connection
2	N/C	No Connection
3	RxD/TxD-P	Data Line B (Red)
4	N/C	No Connection
5	DGND	Data Ground
6	+5V	5V Output Supply
7	N/C	No Connection
8	RxD/TxD-N	Data Line A (Green)
9	N/C	No Connection



4.5 Motor Selection

The PositionServo drive is compatible with many 3-phase AC synchronous servo motors. MotionView OnBoard is equipped with a motor database that contains hundreds of motors for use with the PositionServo drive. If the desired motor is in the database, no data is needed to set it up. Just select the motor and click "OK". However, if the motor is not in the database, it can still be used, but some electrical and mechanical data must be provided to create a custom motor profile. The auto-phasing feature of the PositionServo drive allows the user to correctly determine the relationship between phase voltage and hall sensor signals or resolver offset, eliminating the need to determine feedback orientation by other means.

4.5.1 Motor Connection

Motor phase U, V, W (or R, S, T) are connected to terminal P7. It is very important that motor cable shield is connected to Earth ground terminal (PE) or the drive's case. The motor's encoder/resolver feedback cable must be connected to terminal P4.

4.5.2 Motor Over-Temperature Protection



NOTE

The PositionServo does not provide motor over-temperature protection. The user may connect a KTY motor thermal sensor to the drive as detailed in section 4.1.1 and this paragraph, 4.5.2, if necessary to satisfy NEC requirements.

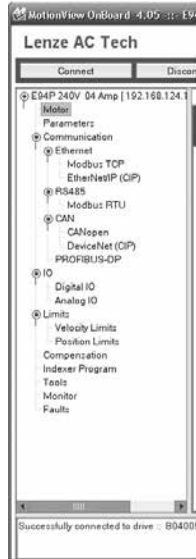
If using a motor equipped with an encoder and PTC thermal sensor, the encoder feedback cable will have flying leads exiting the P4 connector to be wired to the P7.1 (T1) and P7.2 (T2) terminals. If using a motor equipped with a Resolver and a PTC sensor, the thermal feedback is passed directly to the drive via the resolver 9-pin D shell connector.

Use parameter "Motor PTC cut-off resistance" (section 5.3.10) to set the resistance that corresponds to maximum motor allowed temperature. The parameter "Motor temperature sensor" must also be set to ENABLE. If the motor doesn't have a PTC sensor, set this parameter to DISABLE. This input will also work with N.C. thermal switches which have only two states; Open or Closed. In this case "Motor PTC cut-off resistance" parameter can be set to the default value.



5 Parameters

The PositionServo drive has many programmable features accessible via the universal software MotionView. This chapter covers the drive's programmable features and parameters in the order they appear in the Parameter Tree of MotionView. Programmable parameters are divided into folders. Each folder contains one or more user adjustable parameters.



Parameter (Node) Tree

All drives can execute a User Program in parallel with motion. Motion can be specified by variety of sources and in three different modes:

Torque

Velocity

Position

In Torque and Velocity mode the reference can be taken from Analog Input AIN1 or from the User Program by setting a particular variable (digital reference). In Position mode, the reference can be taken from MA/MB master encoder/step and directions inputs (available in terminal P3) or from trajectory generator. Access to the trajectory generator is provided through the User Program's motion statements, MOVEx and MDV. Refer to the PositionServo Programming Manual for details on programming. Whether the reference comes from an external device, (AIN1 or MA/MB) or from the drives internal variables (digital reference and trajectory generator) will depend on the parameter settings.



5.1 Drive Identification

At the top of the Node Tree, click the Drive name [E94P 240V 04Amp ...]. The drive ID string, device family, firmware revision, vector processor revision, hardware revision, MotionView OnBoard revision, motor database revision, indexer compiler revision, serial number, drive name and group ID are displayed as illustrated herein. With the exception of the Drive Name and Group ID, the drive identification parameters are fixed and provided for information only.

The screenshot shows the 'Lenze AC Tech' software window. The title bar reads 'MotionView OnBoard 4.05 - E94P 240V 04 Amp [192.168.124.120] : STOPPED'. The interface includes a menu bar with 'Connect', 'Disconnect', 'Save Connection', 'Load Connection', 'Print', and 'Save'. A tree view on the left shows the drive's configuration structure, with 'E94P 240V 04 Amp [192.168.124.120]' selected. The main area displays a table of parameters:

Description	Value	Us
Drive ID String	B04009014100000	
Device Family	804	
Firmware Revision	0.09	
Vector Processor Revision	0.14	
Hardware Revision	1.00	
Deviation Revision	000	
MotionView OnBoard Revision	4.04	
Motor Database Revision	4.11	
Indexer Compiler Revision	3.02	
Serial Number	1	
Drive Name	<input type="text"/>	
Group ID	<input type="text" value="0"/>	

At the bottom of the window, a status bar indicates: 'Successfully connected to drive : B04009014100000_192.168.124.120'.

The drive identifier (E94P 240V 04Amp [192.168.124.120] : STOPPED) in the node tree consists of three segments: the drive's name, the drive's IP address and the status of the Indexer Program.

Drive name: E94P 240V 04Amp
 Drive IP address: 192.168.124.120
 Indexer program status: STOPPED (indexer program is stopped)
 RUNNING (indexer program is running)

The drive identifier also indicates the status of the drive. When the drive identifier in the node tree is highlighted in green, the drive is enabled. When the drive identifier is gray, the drive is disabled.

5.1.1 Drive Name

To assign a name to the drive click in the box adjacent to Drive Name. A alpha-numeric name may be entered to identify the drive.

5.1.2 Group ID

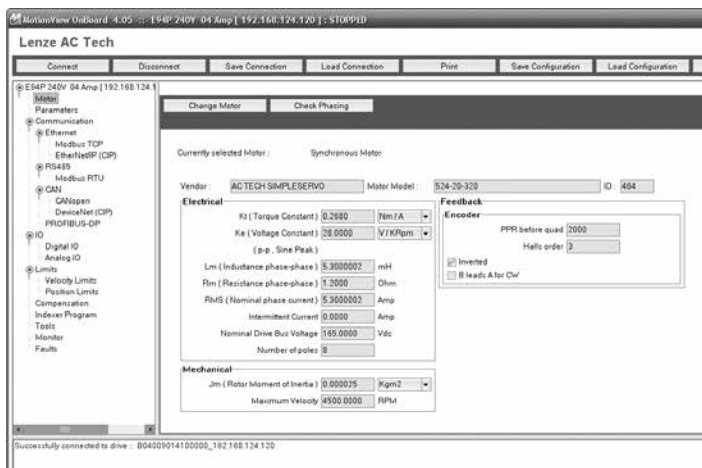
The Group ID feature allows the user to group PositionServo drives together via an Ethernet network. When used with the SEND and SENDTO command, drives in the same group can share and update variables. Group ID Numbers can be set between 0 and 32767. See statements SEND and SENDTO for further explanations.



Parameters

5.2 Motor

The motor folder displays the data for the currently selected motor. A motor may be selected from the database or a custom motor may be configured.



5.2.1 Motor Setup

Select the [Motor] folder in the right-hand "Parameter View Window". To select a new motor click the [Change Motor] button. When [Change Motor] is selected, the Motor Database dialog box will open. Select the Motor Type from the node tree in the left-hand window.



NOTE: The drive must be DISABLED (display: "d ,S") to setup a new motor.



To make a new motor selection:

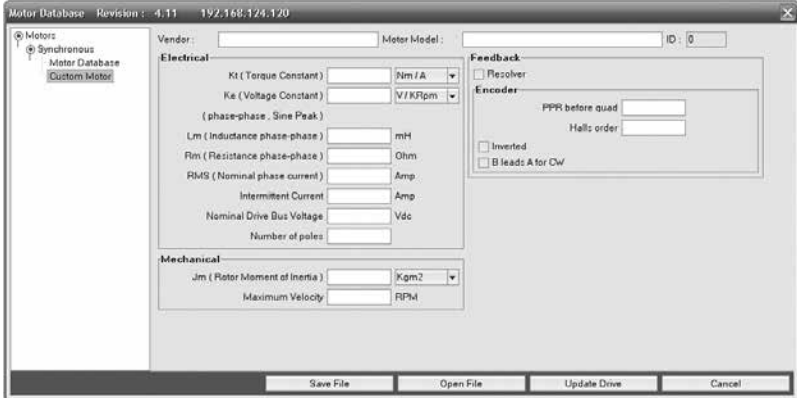
- Click [Change Motor] in the Parameter View Window.
- Select motor Vendor and Motor Model from the pull down menus.
- Click [Update Drive] to complete the motor selection, dismiss the dialog box and return to MotionView OnBoard's main program.
- If using a motor not listed in the current motor database, select [Create Custom] and refer to section 5.2.2 "Using a Custom Motor".

**NOTE**

To help prevent the motor from drawing too much current and possibly overheating it is recommended that the drive's "Current Limit" be checked against the motors "Nominal Phase Current" and set accordingly.

5.2.2 Using a Custom Motor

Follow these instructions to load a custom motor from a file or create a new custom motor. From the Parameter tree select the [Motor] folder. From the Parameter view window select [Change Motor].



- With the Motor Database dialog box open, Click [Custom Motor] under the Motor Type from the left-hand window: Synchronous or Induction/Asynchronous.
- Input the Motor data manually or from a previously saved motor file. To load motor data from a file click [Open file], select file path and click [OK] to open.
- To add this new custom file to your computer's hard drive, click [Save File], select file path and click [OK] to save.
- To load this file to the drive, click [Update Drive].
- When selecting [OK] for a custom motor, a dialog box will appear prompting for a decision to perform/not perform "Autophasing" (refer to section 5.2.4).

5.2.3 Creating Custom Motor Parameters

**STOP!**

Use extreme caution when entering custom parameters! Incorrect settings may damage the drive or motor! If unsure of the settings, refer to the materials distributed with the motor, or contact the motor manufacturer for assistance.

1. Enter custom motor data in the Motor Parameters dialog fields. Complete all sections of dialog: Electrical, Mechanical, Feedback and Motor Gain Scaling.

**NOTE**

If unsure of the motor halls order and encoder channels A and B relationship, leave "B leads A for CW", "Halls order" and "inverted" fields as they are. Use Autophasing (section 5.2.4) to set them correctly.

2. Enter motor model and vendor in the top edit boxes. Motor ID cannot be entered, this is set to 0 for custom motors. Likewise, if unsure of resolver offset and direction of rotation leave at default and correct using the Autophasing.
3. Click [Save File] button and enter filename without extension. The default extension .cmt will be given when you click OK on file dialog box.



NOTE

Save the file even if the autophasing feature will be used and some of the final parameters are not known. After autophasing is completed, the corrected motor file can be updated before loading it to memory.

- Click [Close] to exit from the Motor Parameters dialog.
- MotionView will prompt to autophase/not autophase the custom motor. Answer [No] to cancel without applying the changes made in the Motor database window. Answer [Yes] and the motor dialog will be dismissed and the drive will start the autophasing sequence. Refer to section 5.2.4, Autophasing.
- If [Yes] is selected, the same motor selection dialog box will be displayed after autophasing is complete. For motors with incremental encoders, the fields “B leads A for CW”, “Halls order” and “inverted” will be assigned correct values. For motors with resolvers, the fields “Offset in degree” and “CW for positive” will be assigned correct values.
- Click [Save File] to save the custom motor file and then click [Update Drive] to exit the dialog box and load the data to the drive.

5.2.4 Autophasing

The Autophasing feature determines important motor parameters when using a motor that is not in MotionView’s database. For motors equipped with incremental encoders, Autophasing will determine the Hall order sequence, Hall sensor polarity and encoder channel relationship (B leads A or A leads B for CW rotation). For motors equipped with resolvers, Autophasing will determine resolver angle offset and angle increment direction (“CW for positive”).

To perform autophasing:

- Complete the steps in “Creating custom motor parameters”. If the motor file to be autophased already exists, simply load it as described under “Using a custom motor”.
- Make sure that the motor’s shaft is not connected to any mechanical load and can freely rotate.



STOP!

Autophasing will energize the motor and will rotate the shaft. Make sure that the motor’s shaft is not connected to any mechanical load and can freely and safely rotate.

3. Make sure that the drive is not enabled.

- For Encoder it is not necessary to edit the field “Hall order” and check boxes “inverted” and “B leads A for CW” as these values are ignored for autophasing. For Resolver it is not necessary to set “Offset in degree” and “CW for positive”.
- Click [Update Drive] to dismiss motor selection dialog. MotionView responds with the question “Do you want to perform autophasing?”
- Click [OK]. A safety reminder dialog appears. Verify that it is safe to run the motor then click [Yes] and wait until autophasing is completed.



NOTE

If a problem occurs with the motor, hall sensor or resolver connections, MotionView will send an error message. The source of the error is commonly the power, shield and ground terminations or the use of an improper cable. Correct the wiring problem(s) and repeat steps 1 - 6.

If the error message repeats, exchange motor phases U and V (R and S) and repeat. If problems persist, contact the factory.

- If autophasing is completed with no error then MotionView will return to the motor dialog box. For motors with incremental encoders, the parameter field “Hall order” and the check boxes “inverted”, “B leads A for CW” will be filled in with correct values. For resolver equipped motors, fields “Offset” and “CW for positive” will be correctly set.



- Click [Save File] to save the completed motor file (use same filename as the initial data in step 1).
Click [Update Drive] to load the motor data to the drive.

5.2.5 Custom Motor Data Entry

A Custom Motor file is created by entering motor data into the “Motor Parameters” dialog box. This box is divided up into four sections: Electrical constants, Mechanical constants, Feedback and Gain Scaling.

Parameter Type	Synchronous Motor	Asynchronous (Induction) Motor
Identification	Vendor, Motor Model, ID	Vendor, Motor Model, ID
Electrical	K_t , K_e , L_m , R_m , I_{RMS}^* , Nominal V_{BUS}^* , # of poles	$\cos \phi$, f_{BASE}^* , L_m , R_m , I_{RMS}^* , Nominal V_{BUS}^* , # of poles
Feedback	Primary feedback, Resolver Offset	Resolver FB, Encoder PPR before quad, B leads A CW
Motor Gain Scaling	Velocity P-gain, Velocity I-gain, Gain Scaling	Velocity P-gain, Velocity I-gain, Gain Scaling
Mechanical	J_m , Vel_{MAX}	J_m , $Vel_{NOMINAL}^*$, Vel_{MAX}

When creating a custom motor, input the value of all parameters listed for the specific motor type. All entries are mandatory except motor inertia (J_m). Enter a value of 0 for the motor inertia if the actual value is unknown.

5.2.5.1 Electrical & Mechanical Constants

Motor Torque Constant (Kt)

Enter the value and select proper units from the drop-down list.



NOTE

Round the calculated result to 3 significant places.

Motor Voltage Constant (Ke)

The program expects K_e to be entered as a phase-to-phase Peak voltage. If you have K_e as an RMS value, multiply this value by 1.414 for the correct K_e Peak value.

Phase-to-phase winding Inductance (Lm)

This must be set in millihenries (mH). The phase-to-phase winding Inductance (L) will typically be between 0.1 and 200.0 mH.



NOTE

If the units for the phase-to-phase winding Inductance (L) are given in micro-henries (μH), then divide by 1000 to get mH.

Phase-to-phase winding Resistance (Rm) in Ohms

This is also listed as the terminal resistance (R_t). The phase-to-phase winding Resistance (R) will typically be between 0.05 and 200 Ohms.

Nominal phase current (RMS Amps)

Nominal continuous phase current rating (I_n) in Amps RMS. Do not use the peak current rating.

**NOTE**

If the phase current rating is not given, use this equation to obtain the nominal continuous phase-to-phase winding current:

$$I_n = \text{Continuous Stall Torque} / \text{Motor Torque Constant (Kt)}$$

The same force x distance units must be used in the numerator and denominator in the equation above. If torque (T) is expressed in units of pound-inches (lb-in), then Kt must be expressed in pound-inches per Amp (lb-in/A). Likewise, if T is expressed in units of Newton-meters (N-m), then units for Kt must be expressed in Newton-meters per Amp (N-m/A).

Example:

Suppose that the nominal continuous phase to phase winding current (I_n) is not given. Instead, we look up and obtain the following:

Continuous stall torque $T = 3.0$ lb-in

Motor torque constant $K_t = 0.69$ lb-in/A

Dividing, we obtain:

$$I_n = 3.0 \text{ lb-in} / 0.69 \text{ lb-in/A} = 4.35 \text{ (A)}$$

Our entry for (I_n) would be 4.35.

Note that the torque (lb-in) units are cancelled in the equation above leaving just Amps (A). We would have to use another conversion factor if the numerator and denominator had different force x distance units.

Nominal Bus Voltage (Vbus)

The Nominal Bus Voltage can be calculated by multiplying the Nominal AC mains voltage supplied by 1.41. When using a model with the suffix "S1N" where the mains are wired to the "Doubler" connection, the Nominal Bus Voltage will be doubled.

Example:

If the mains voltage is 230VAC, $V_{bus} = 230 \times 1.41 = 325V$

This value is the initial voltage for the drive and the correct voltage will be calculated dynamically depending on the drive's incoming voltage value.

Number of Poles

This is a positive integer number that represents the number of motor poles, normally 2, 4, 6 or 8.

Rotor Moment of Inertia (Jm)

From motor manufacturer or nameplate.

**NOTE**

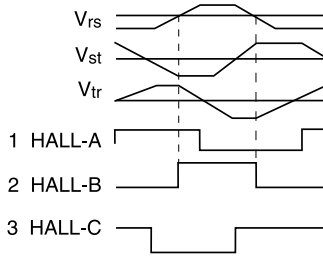
Round the calculated result to 3 significant places.

5.2.5.2 For Incremental Encoder - Equipped Motors Only**Encoder Line Count**

The encoders for servomotors normally have line counts of 1000, 1024, 2000, 2048, 4000, or 4096. The Encoder Line Count is pre-quadrature and a positive integer.

Halls Order

Each hall signal is in phase with one of the three phase-phase voltages from the motor windings. Hall order number defines which hall sensor matches which phase-phase voltage. Motor phases are usually called R-S-T or U-V-W or A-B-C. Phase-Phase voltages are called V_{rs} , V_{st} , V_{tr} . Halls are usually called HALL-A, HALL-B, HALL-C or just Halls 1, 2, 3. A motor's phase diagram is supplied by motor vendor and usually can be found in the motor data sheet or by making a request to the motor manufacturer. A sample phase diagram is illustrated in Figure S912.



S912

The Halls Order is obtained as follows:

1. Look at the “Vrs” Output Voltage and determine the Hall Voltage that is lined up with (or in phase with) this voltage. To determine which Hall Voltage is in phase with the Vrs Output Voltage draw vertical lines at those points where it crosses the horizontal line (zero). The dashed lines at the zero crossings (above) indicate that Hall B output is lined up with (and in phase with) the Vrs Output Voltage.
2. Look at the “Vst” Output Voltage. Determine which Hall Voltage is in phase with this Voltage. Per Figure S912, the Hall C output is in phase with the Vst Output Voltage.
3. Look at the “Vtr” Output Voltage. Determine which Hall Voltage is in phase with this Voltage. Per Figure S912, the Hall A output is in phase with the Vtr Output Voltage.



NOTE

If hall sensors are in phase with the corresponding phase voltage but are inverted 180 degrees (hall sensor waveform edge aligns with the phase-phase voltage waveform but the positive hall sensor cycle matches the negative phase-phase waveform or visa-versa), you must check the “Inverted” check box.

4. The phases that correspond to the Vrs, Vst and Vtr voltages are Hall B then Hall C then Hall A or Halls number 2 then 3 then 1. Referring to the following table, we find that 2-3-1 sequence is Halls Order number 3. We would then enter 3 for the Halls Order field in the motor dialog box.

Hall Order Numbers for Different Hall Sequences

Halls Order	Hall Sequence
0	1-2-3
1	1-3-2
2	2-1-3
3	2-3-1
4	3-1-2
5	3-2-1



NOTE

Each Hall Voltage is in phase with one and only one Output Voltage.



B leads A for CW

This is the encoder phase relationship for CW/CCW shaft rotation. When you obtain the diagram for your motor phasing similar to shown above, it's assumed by the software that the motor shaft rotates CW when looking at the rear of the motor. For that rotation Encoder phase A must lead phase B. If it does, leave the check box unchecked. Otherwise (if B leads A), check B leads A in the CW box.



NOTE

The reference for direction of rotation is from the rear of the motor.



NOTE

This parameter does not reverse the direction of motor rotation. It is used to setup the motor commutation. See "Rotation Direction" in the Parameters menu to reverse the direction of forward rotation.

5.2.5.3 For Resolver Equipped Motors Only

If parameter "Resolver" is checked, following parameters appear on the form:

Offset in degree (electrical)

This parameter represents offset between resolver's "0 degree" and motor's windings "0 degree".

CW for positive

This parameter sets the direction for positive angle increment.

"Offset in degree" and "CW for positive" will be set during Auto-Phasing of the motor.



5.3 Parameters

Lenze AC Tech

Connected	Disconnected	Save Connection	Load Connection	Print	Save
-----------	--------------	-----------------	-----------------	-------	------

Motor

Parameters

- Communication
 - Ethernet
 - Modbus TCP
 - EtherNet/IP (OP)
 - RS485
 - Modbus RTU
 - CAN
 - CANopen
 - DeviceNet (CP)
 - PROFIBUS-DP
- IO
 - Digital IO
 - Analog IO
- Limits
 - Velocity Limits
 - Position Limits
- Compensation
- Inverter Program
- Tests
- Monitor
- Faults

Description	Value
Drive Operating Mode	
Drive Mode	Position
Reference	External
Drive PkM Frequency	18 KHZ
Current Limits	
To Change Current Limits	<input type="checkbox"/>
Warning: Setting the Current Limit or Peak Current Limit at level bigger than the Motor RMS - H	
Current Limit	4.0000
8 KHZ Peak Current Limit	10.0000
18 KHZ Peak Current Limit	15.0000
Velocity Mode Acceleration	
Enable Accel / Decel Limits	Disable
Accel Limit	1000.0000
Decel Limit	1000.0000
Fault Reset	On Enable
Motor Temperature Sensor	Disable
Motor PTC out-of-Resistance	2500
Regen Duty Cycle	10
Master Encoder Input Type	Master Encoder

Successfully connected to drive: B0450014100000_192.168.124.130

Parameters List - Top

Lenze AC Tech

Connected	Disconnected	Save Connection	Load Connection	Print	Save
-----------	--------------	-----------------	-----------------	-------	------

Motor

Parameters

- Communication
 - Ethernet
 - Modbus TCP
 - EtherNet/IP (OP)
 - RS485
 - Modbus RTU
 - CAN
 - CANopen
 - DeviceNet (CP)
 - PROFIBUS-DP
- IO
 - Digital IO
 - Analog IO
- Limits
 - Velocity Limits
 - Position Limits
- Compensation
- Inverter Program
- Tests
- Monitor
- Faults

Description	Value
Current Limit	
Current Limit	3.1000
8 KHZ Peak Current Limit	5.2500
18 KHZ Peak Current Limit	5.2500
Velocity Mode Acceleration	
Enable Accel / Decel Limits	Disable
Accel Limit	1000.0000
Decel Limit	1000.0000
Fault Reset	On Disable
Motor Temperature Sensor	Disable
Motor PTC out-of-Resistance	2500
Regen Duty Cycle	10
Master Encoder Input Type	Master Encoder
MASTER ENCODER - Master To System Ratio	
Master	1
System	1
Autoboot	Disable
User Limits	1.0000
Rotational Direction	Normal
(Change applied after power-cycle)	
Resolver Tracks	Track 0 - 1024 PPR

Successfully connected to drive: B04012015100000_192.168.124.130

Parameters List - Bottom



5.3.1 Drive Mode

The PositionServo has 3 operating mode selections: Torque, Velocity and Position.

For Torque and Velocity modes the drive will accept an analog input voltage on the AIN1+ and AIN1- pins of P3 (refer to section 4.3.1). This voltage is used to provide a torque or speed reference.

For Position mode the drive will accept step and direction logic signals or a quadrature pulse train on pins P3.1- P3.4.

5.3.1.1 Torque Mode

In torque mode, the servo control provides a current output proportional to the analog input signal at input AIN1, if parameter “Reference” is set to “External”. Otherwise the reference is taken from the drive’s internal variable. (Refer to the PositionServo Programming Manual for details)

For analog reference “Set Current”, (current the drive will try to provide), is calculated using the following formula:

$$\text{Set Current(A)} = \text{Vinput(Volt)} \times \text{Iscale (A/Volt)}$$

where:

- Vinput is the voltage at analog input
- Iscale is the current scale factor (input sensitivity) set by the Analog input (Current Scale) parameter (section 5.5.4).

5.3.1.2 Velocity Mode

In velocity mode, the servo controller regulates motor shaft speed (velocity) proportional to the analog input voltage at input AIN1, if parameter “Reference” is set to “External”. Otherwise the reference is taken from the drive’s internal variable. Refer to the PositionServo Programming Manual for details.

For analog reference, Target speed (set speed) is calculated using the following formula:

$$\text{Set Velocity (RPM)} = \text{Vinput (Volt)} \times \text{Vscale (RPM/Volt)}$$

where:

- Vinput is the voltage at analog input (AIN1+ and AIN1-)
- Vscale is the velocity scale factor (input sensitivity) set by the Analog input (Velocity scale) parameter (section 5.5.5).

5.3.1.3 Position Mode

In this mode the drive reference is a pulse-train applied to P3.1-4 terminals, if the parameter “Reference” is set to “External”. Otherwise the reference is taken from the drive’s internal motion commands. (Refer to the PositionServo Programming Manual for details).

P3.1-4 inputs can be configured for two types of signals: step and direction and Master encoder quadrature signal. Refer to section 4.2.1 for details on these inputs connections. Refer to section 6.4 for details about positioning and gearing.

When the Reference is set to Internal, the drives reference position, (theoretical or Target position), is generated by trajectory generator. Access to the trajectory generator is provided by motion statements, MOVEx and MDV, from the User Program. Refer to the PositionServo Programming Manual for details.

5.3.2 Reference

The REFERENCE setting selects the reference signal being used by the drive. This reference signal can be either External or Internal. An External Reference can be one of three types, an Analog Input signal, a Step and Direction Input or an Input from a external Master Encoder. The Analog Input reference is used when the drive is either in torque or velocity mode. The Master Encoder and Step and Direction reference is used when the drive is in position mode. An Internal Reference is used when the motion being generated is derived from drive’s internal variable(s), i.e., User Program. Refer to the PositionServo Programming Manual.



5.3.3 Drive PWM Frequency

This parameter sets the PWM carrier frequency. Frequency can be changed only when the drive is disabled. Maximum overload current is 300% of the drive rated current when the carrier is set to 8kHz. It is limited to 250% at 16kHz.

5.3.4 Current Limit

The Current Limit setting determines the nominal currents, in amps RMS per phase, that output to the motor phases. To prevent the motor from overloading, this parameter is usually set equal to the motor nominal (or rated) phase current. The Current Limit is set equal to the nominal motor phase current by default when a motor model is selected.

5.3.5 To Change Current Limits

To modify/overwrite the Current Limit place a checkmark in the box. If this box is checked, the parameters "Current limit", "8 kHz peak current limit" and "16 kHz peak current limit" can be overwritten. To prevent the motor from overloading, the "current Limit", "8 kHz peak current limit" and "16 kHz peak current limit" shall be set to values no higher than the corresponding current limits of the motor in use.

5.3.6 Peak Current Limit (8 kHz and 16 kHz)

Peak Current Limit sets the motor RMS phase current that is allowed for up to 2 seconds. After this two second limit, the drive output current to motor will be reduced to the value set by the Current Limit parameter. When the motor current drops below nominal current for two seconds, the drive will automatically re-enable the peak current level. This technique allows for high peak torque on demanding fast moves and fast start/stop operations with high regulation bandwidth. If 8 kHz is used for Drive PWM frequency, use the parameter 8 kHz Peak Current Limit, otherwise, use 16 kHz Peak Current Limit.

The Peak Current Limit is set equal to 2.5 times the nominal motor phase current by default when a motor model is selected. The maximum of 3 times nominal motor phase current can be obtained at 8kHz. To prevent motor from overloading, the Peak Current Limit shall be set no higher than the maximum motor current. Otherwise, the motor may be damaged due to overheating. To modify this limit, refer to section 5.3.5.

5.3.7 Accel/Decel Limits (velocity mode only)

The Accel setting determines the time the motor takes to ramp to a higher speed. The Decel setting determines the time the motor takes to ramp to a lower speed. If the Enable Accel/Decel Limits is set to [Disable], the drive will automatically accelerate and decelerate at maximum acceleration limited only by the current limit established by the Peak Current Limit and Current Limit settings. This parameter is only utilized when the drive is set to Velocity mode (refer to 5.3.1).

5.3.8 Fault Reset

Fault Reset selects the type of action required to reset the drive after a FAULT condition has been generated by the drive. On Disable clears the fault when the drive is disabled. This is useful if you have a single drive and motor connected in a single drive system. The On Enable option clears the fault when the drive is re-enabled. Choose On Enable if you have a complex servo system with multiple drives connected to an external controller. This makes troubleshooting easier since the fault will not be reset until the drive is re-enabled. Thus, a technician can more easily determine which component of a complex servo system has caused the fault.



5.3.9 Motor Temperature Sensor

This parameter enables / disables motor over-temperature detection. It must be disabled if the motor PTC sensor is not wired to either P7.1-2 or to the resolver feedback input (P4 or P11).

5.3.10 Motor PTC Cutoff Resistance

This parameter sets the cut-off resistance of the PTC that defines when the motor reaches the maximum allowable temperature. Refer to section 4.5.2 for details on how to connect the motor's PTC.

5.3.11 Regen Duty Cycle

This parameter sets the maximum duty cycle for the brake (regeneration) resistor. This parameter can be used to prevent brake resistor overload. Use the following formula to calculate the maximum value for this parameter. If this parameter is set equal to the calculated value, the regeneration resistor is most effective without overload. One may set this parameter with a value smaller than the calculated one if the drive will not experience over voltage fault during regeneration.

$$D = P * R / (U_{max})^2 * (1/D_{application}) * 100\%$$

Where:

D (%) regeneration duty cycle

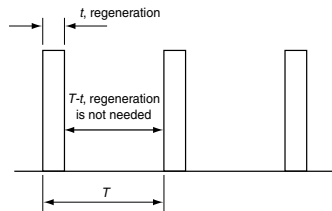
U_{max} (VDC) bus voltage at regeneration conditions

$$U_{max} = 390 \text{ VDC for } 120/240 \text{ VAC drives and } 770 \text{ VDC for } 400/480 \text{ VAC drives.}$$

R (Ohm) regeneration resistor value

P (W) regeneration resistor rated power

D_{application} (%) application duty cycle. For the continuous regeneration applications, use D_{application} = 1. For the intermittent regeneration applications, use D_{application} = t/T, where t is the duration when regeneration is needed and T is the time interval between two regenerations. Both t and T must use the same time unit, e.g., seconds



If calculation of D is greater than 100% set it to 100% value. If calculation of D is less than 10% then resistor power rating is too low. For more information refer to the PositionServo Dynamic Braking Manual (G94BR01).

Minimum Required Dynamic Braking Resistance

Drive Model	DB Minimum Resistance (Ω)
E94_180T2N~~	15
E94_080S2F~~, E94_080Y2N~~, E94_100S2F~~, E94_100Y2N~~	20
E94_120Y2N	30
E94_020S1N~~, E94_020S2F~~, E94_020Y2N~~, E94_040S1N~~, E94_040S2F~~, E94_040Y2N~~	40
E94_090T4N~~	45
E94_040T4N~~, E94_050T4N~~, E94_060T4N~~	75
E94_020T4N~~	150



5.3.12 Master Encoder Input Type (position mode only)

This parameter sets the type of input for position reference the drive expects to see. Signal type can be step and direction [Step & Direction] type or quadrature pulse-train [Master Encoder]. Refer to section 4.2.1 for details on these inputs.

5.3.13 Master Encoder - System to Master Ratio

This parameter is used to set the scale between the reference pulse train (when operating in position mode) and the system feedback device. The system feedback device is the motor encoder or resolver.

5.3.14 Autoboot

When set to "Enabled" the drive will start to execute the user's program immediately after cold boot (reset). Otherwise the user program has to be started from MotionView or from the Host interface.

5.3.15 User Units

This parameter sets up the relationship between User Units and motor revolutions. From here you can determine how many User Units there is in one motor revolution. This parameter allows the user to scale motion moves to represent a desired unit of measure, (inches, meters, in/sec, meters/sec, etc).

User Units Example: A linear actuator allows a displacement of 2.5" with every revolution of the motor's shaft.

$$\text{Units} = \text{Units} / \text{Revolutions}$$

$$\text{Units} = 2.5 \text{ Inches} / \text{Revolution}$$

$$\text{Units} = 2.5$$

5.3.16 Rotation Direction

This parameter sets up the direction of forward (positive) rotation. To reverse the direction of positive rotation for a specific installation, change Rotation Direction from "Normal" to "Reversed".

5.3.17 Resolver Tracks

The Resolver Tracks parameter is used in conjunction with the resolver motors and Buffered Encoder Outputs (Section 4.2.2). If a motor with resolver feedback is being used a simulated encoder feedback is transmitted out the Buffered Encoder Outputs, P3.7 to P3.12. The default resolution of this feedback is 1024 pulses per revolution, pre quad. If a different resolution is required then the Resolver Tracks parameter is utilized. The number entered into this field, 0-15, correlates to a specific encoder resolution.

Resolver Tracks Configuration

Resolver Track	Resolution Before Quad	Resolver Track	Resolution Before Quad
0	1024	8	1000
1	256	9	1024
2	360	10	2000
3	400	11	2048
4	500	12	2500
5	512	13	2880
6	720	14	250
7	800	15	4096



5.4 Communication

The Communication folder contains four sub-folders: Ethernet, RS-485, CAN and PROFIBUS plus sub-sub folders to program the parameters specific to the communication type. Select the Fieldbus used from the pull-down menu (None, CANOpen Simple 301, DeviceNet or PROFIBUS).



NOTE

Ethernet is always enabled regardless of the fieldbus selected. Gatewaying is not supported between fieldbus and Ethernet.

5.4.1 Ethernet

Refer to section 6.2 on setting an IP address. The Ethernet folder displays the IP Address, Subnet Mask and Default Gateway for the drive selected in the Node Tree. The TCP Reply Delay can be set in 1 millisecond increments from 0 to 15ms. To obtain the IP address via DHCP, check the box adjacent to [Obtain IP address using DHCP].

The Ethernet folder contains the sub-folders: Modbus TCP and EtherNet/IP. Defined by the Ethernet hardware settings, no further settings are necessary to communicate via Modbus TCP. Also defined by the Ethernet hardware settings, the EtherNet/IP folder contains the configuration parameters for the EtherNet/IP (Industrial Protocol). In general, there is no need to change parameters for multicast operations. Consult your IT administrator for these settings as their configuration is very network-specific.

5.4.2 RS-485

To configure the RS485 interface, option module E94ZARS41, set the following parameters: RS485 Configuration, RS485 Baud Rate, RS485 Parity, RS485 Stop Bits and RS485 Address. The RS485 interface can be configured for UPPP operation or as a Modbus RTU slave.

The RS-485 folder contains one sub-folder: Modbus RTU. The Modbus RTU folder contains the Modbus Reply Delay parameter which sets the time delay between the drive's reply to the Modbus RTU master. This delay is needed for some types of Modbus masters to function correctly.

5.4.3 CAN

The CAN baud rate and CAN address are set in the main CAN folder. The main CAN folder contains two sub-folders: CANOpen and DeviceNet. In the CANOpen sub-folder, the CAN Bootup Mode, CAN Bootup Delay and CAN Heart Beat Time parameters are set. Mapping of the CAN process data objects (PDO) is also carried out from this folder. In the DeviceNet folder, the DeviceNet Poll I/O Scaling parameter is set.

5.4.4 PROFIBUS

These parameters are set in the PROFIBUS folder: PROFIBUS Address, Acyclic Mode, Data Exchange Timeout plus the IN/OUT Data Size, Parameter ID Number and Mapping Type.



5.5 Analog I/O

5.5.1 Analog Output

The PositionServo has one analog output with 10-bit resolution on P3 pin 23. The signal is scaled to $\pm 10V$. The analog output can be assigned to the following functions:

- Not Assigned
- Phase current RMS
- Phase current Peak
- Motor Velocity
- Phase R current
- Phase S current
- Phase T current
- Iq current (Torque component)
- Id current (Direct component)

5.5.2 Analog Output Current Scale (Volt/Amps)

Applies scaling to all functions representing CURRENT values.

5.5.3 Analog Output Velocity Scale (mV/RPM)

Applies scaling to all functions representing VELOCITY values. (Note: that mV/RPM scaling units are numerically equivalent to volts/kRPM)

5.5.4 Analog Input Current Scale (Amps/Volt)

This parameter sets the analog input sensitivity for current reference used when the drive operates in torque mode. Units for this parameter are A/Volt. To calculate this value use the following formula:

$$I_{scale} = I_{max} / V_{in\ max}$$

I_{max} maximum desired output current (motor phase current RMS)

$V_{in\ max}$ max voltage fed to analog input at I_{max}

Example: I_{max} = 5A (phase RMS)
 $V_{in\ max}$ = 10V
 I_{scale} = $I_{max} / V_{in\ max}$
 = $5A / 10V = 0.5\ A / Volt$ (value to enter)

5.5.5 Analog Input Velocity Scale (RPM/Volt)

This parameter sets the analog input sensitivity for the velocity reference used when the drive operates in velocity mode. Units for this parameter are RPM/Volt. To calculate this value use the following formula:

$$V_{scale} = VELOCITY_{max} / V_{in\ max}$$

$VELOCITY_{max}$ maximum desired velocity in RPM

$V_{in\ max}$ max voltage fed to analog input at $Velocity_{max}$

Example: $VELOCITY_{max}$ = 2000 RPM
 $V_{in\ max}$ = 10V
 V_{scale} = $VELOCITY_{max} / V_{in\ max}$
 = $2000 / 10V$
 = 200 RPM / Volt (value to enter)



5.5.6 Analog Input Dead Band

Allows the setting of a voltage window (in mV) at the reference input AIN1+ and AIN1- (P3 pins 24 and 25) such that any voltage within that window will be treated as zero volts. This is useful if the analog input voltage drifts resulting in motor rotation when commanded to zero.

5.5.7 Analog Input Offset

This function allows the drive to automatically adjust the analog input voltage offset. To use it, command the external reference source input at AIN1+ and AIN1- (P3 pins 24 and 25) to zero volts and then click the [<<] button adjacent to the [Analog Input Offset] box. Any offset voltage at the analog input will be adjusted out and the adjustment value will be stored in the [Analog input offset] parameter.

5.6 Digital I/O

5.6.1 Digital Output

The PositionServo has four programmable digital outputs. These outputs can be assigned to one of the following functions, or used by the drive's internal User Program.

Not Assigned	No function assigned. Output can be used by the User program.
Zero Speed	Output activated when drive is at zero speed, refer to "Velocity Limits Group" (section 5.7) for settings.
In Speed Window	Output activated when drive is in set speed window, refer to "Velocity Limits Group" (section 5.7) for settings.
Current Limit	Output activated when drive detects current limit.
Run Time Fault	A fault has occurred. Refer to section 7.3 for details on faults.
Ready	Drive is enabled.
Brake	Output is active for the time programmed by the Brake Release Delay parameter after the drive is enabled and deactivates after the drive is disabled for control of a motor mechanical brake.
In position	Position mode only. Refer to the PS Programming Manual.

5.6.2 Digital Input De-bounce Time

Sets de-bounce time for the digital inputs to compensate for bouncing of the switch or relay contacts. This is the time following an input transition when any further transitions will be ignored (not recognized by the drive).

5.6.3 Hard Limit Switch Action

Digital inputs IN_A1 and IN_A2 can be used as limit switches if their function is set to "Fault" or "Stop and Fault". Activation of these inputs while the drive is enabled will cause the drive to Disable and go to a Fault state. The "Stop and Fault" action is available only in Position mode when the "Reference" parameter is set to "Internal", i.e., when the source for the motion is the Trajectory generator. Refer to the PositionServo Programming Manual for details on "Stop and Fault" behavior.

5.6.4 Enable Switch Function

The Enable input (IN_A3) on PositionServo can be set to function as either a 'Run' Input or an 'Inhibit' Input. The run function allows input A3 control of switching the drive between enable and disable states (Enabling or disabling output to the motor). The Run function is typically used in centralized systems where a PLC or Motion Control output is required to control the enable/disable of the drive.

When input A3 becomes active the drive will go immediately to an enable state, and when it becomes inactive the drive will go immediately to a disabled state.



The inhibit function allows input A3 to inhibit (prevent) power being applied to the motor but does not provide the enable or disable command for the drive. This function is typically used in a centralized system where the drive's internal programming determines when the drive should enable or disable (these statements are executed within the drive programming). In the inhibit mode Input A3 acts as a hardware level inhibit, only allowing the drive to go to an enable state (when instructed from the internal programming) providing the input A3 is active. Attempting to enable from the internal user program while input A3 is inactive will cause the drive to trip (Fault F_36) as will removal of input A3 while the drive is in an enabled state.

Input A3 cannot be bypassed, it must be present to obtain any power to the motor or motion.

5.6.5 Brake Release Delay

The Brake Release Delay controls the amount of time an output configured as “brake” waits after the drive enables to activate the brake output. The range for Brake Release Delay is 0-2000 milliseconds and the default value is 0ms.

5.7 Velocity Limits

In the Velocity Limits folder are 3 programmable parameters: Zero Speed, Speed Window and At Speed. These parameters are active in Velocity Mode Only.

5.7.1 Zero Speed

Specifies the upper threshold for motor zero speed in RPM. When the motor shaft speed is at or below the specified value, the zero speed condition is set to true in the internal controller logic. The zero speed condition can also trigger a programmable digital output, if selected. The Zero Speed range is 0 to 100 RPM and the default value is 10 RPM.

5.7.2 Speed Window

Speed Window specifies the width used with the “In speed window” output. The Speed Window range is 10 to 10,000 RPM and the default value is 100 RPM.

5.7.3 At Speed

At Speed specifies the speed window center used with the “In speed window” output. The At Speed range is -10000 to 10000 RPM and the default value is 1000 RPM.

Speed Window and At Speed specify speed limits. If motor shaft speed is within these limits then the condition AT SPEED is set to TRUE in the internal controller logic. The AT SPEED condition can also trigger a programmable digital output, if selected.

For example if “AT SPEED” is set for 1000 RPM, and the “SPEED WINDOW” is set for 100, then “AT SPEED” will be true when the motor velocity is between 950 -1050 RPM.



5.8 Position Limits

5.8.1 Position Error

Specifies the maximum allowable position error in the primary (motor mounted) feedback device before enabling the “Max error time” clock. When using an encoder, the position error is in post-quadrature encoder counts. When using a resolver, position error is measured at a fixed resolution of 65,536 counts per motor revolution.

**STOP!**

If Position Error is set to 0, position error checking is disabled. Carefully evaluate the application for safety aspects before disabling position error checking.

5.8.2 Max Error Time

Specifies maximum allowable time (in mS) during which a position error can exceed the value set for the “Position error” parameter before a Position Error Excess fault is generated. If the Position Error is set to the max setting then the drive will trip and not use the Error time when the error exceeds the above setting.

5.8.3 Soft Limits

Enables/disables the usage of a software defined limit. Do not enable this feature until after the drive is homed for the specific application. Like all parameters, this setting can be set/reset logically within the Indexer program.

Positive Limit Soft limit switch location in User Units

Negative Limit Soft limit switch location in User Units

5.9 Compensation

5.9.1 Velocity P-gain (proportional)

Proportional gain adjusts the system’s overall response to a velocity error. The velocity error is the difference between the commanded velocity of a motor shaft and the actual shaft velocity as measured by the primary feedback device. By adjusting the proportional gain, the bandwidth of the drive is more closely matched to the bandwidth of the control signal, ensuring more precise response of the servo loop to the input signal.

5.9.2 Velocity I-gain (integral)

The output of the velocity integral gain compensator is proportional to the accumulative error over cycle time, with I-gain controlling how fast the error accumulates. Integral gain also increases the overall loop gain at the lower frequencies, minimizing total error. Thus, its greatest effect is on a system running at low speed, or in a steady state without rapid or frequent changes in velocity.

**NOTE**

The following 4 position gain settings are only active if the drive is operating in Position mode. They have no effect in Velocity or Torque modes.



5.9.3 Position P-gain (proportional)

Position P-gain adjusts the system's overall response to position error. Position error is the difference between the commanded position of the motor shaft and the actual shaft position. By adjusting the proportional gain, the bandwidth of the drive is more closely matched to the bandwidth of the control signal, ensuring more precise response of the servo loop to the input signal.

5.9.4 Position I-gain (integral)

The output of the Position I-gain compensator is proportional to accumulative error over cycle time, with I-gain controlling how fast the error accumulates. Integral gain also increases overall loop gain at the lower frequencies, minimizing total error. Thus, its greatest effect is on a system running at low speed, or in a steady state without rapid or frequent changes in position.

5.9.5 Position D-gain (differential)

The output of the Position D-gain compensator is proportional to the difference between the current position error and the position error measured in the previous servo cycle. D-gain decreases the bandwidth and increases the overall system stability. It is responsible for removing oscillations caused by load inertia.

5.9.6 Position I-limit

The Position I-limit will clamp the Position I-gain compensator to prevent excessive torque overshooting caused by an over accumulation of the I-gain. It is defined in terms of RPM. This is especially helpful when position error is integrated over a long period of time.

5.9.7 Gain Scaling Window

Sets the total velocity loop gain multiplier ($2n$) where n is the velocity regulation window. If, during motor tuning, the velocity gains become too small or too large, this parameter is used to adjust loop sensitivity. If the velocity gains are too small, decrease the total loop gain value, by decreasing this parameter. If gains are at their maximum setting and you need to increase them even more, use a larger value for this parameter.

5.9.8 Disable High Performance Mode

If the box is checked, the drive uses the gain modeling algorithm from hardware revision 1 of the PositionServo. This setting is enabled by default to facilitate the replacement of legacy platform 940/941 (hardware revision 1) installations without re-tuning. This setting should be de-selected for best results with Auto Tuning.

5.9.9 Auto Tuning

Click the [Autotuning] button to access the Auto Tuning parameter. this parameter auto tunes the compensation gains for the motor/load applied.



NOTE

For best results, de-select [Disable High Performance Mode] prior to auto tuning.



5.9.10 Set Default Gains

Click the [Set Default Gains] button to access the Default Gains parameter. Selecting [Set Default Gains] will reset the gains to the default values in the motor file.

5.9.11 Feedback and Loop Filters

Hardware Version 2 provides for the use of 1 feedback filter and 2 cascaded loop filters. Loop filters are identical in structure and operation.

The feedback filter is a low-pass first order filter used primarily to filter noise from the feedback device. The time constant of the filter is settable and from 2mS and up. Values of 2 – 8mS are generally adequate for most applications.

The Loop filter can be configured as a Low-Pass, Notch or Resonant type filter.

The Low pass filter is used to lower noise in the system produced by control and feedback signal disturbances and quantization noise. The Low-pass cut off frequency is usually set at 5-10 times the desired velocity loop bandwidth.

If enabled, the loop filter is installed between the velocity and current loop.

The Loop filter can be configured as a Notch or Resonant type filter. Both configurations implement band-stop filtering for solving certain mechanical compliance problems. A common problem is torsional resonance due to mechanical compliance between load inertia and motor inertia. Consider a motor coupled with a long load shaft with an inert load at the opposite end. Such a system will have a resonant frequency of

$$f_r = \frac{1}{2\pi} \sqrt{\frac{K_s}{J_p}} \quad \text{[Hz]}$$

where

JL =	load inertia	[kgm ²]
JM =	motor inertia	[kgm ²]
Ks =	total stiffness of coupling and shaft	[Nm/rad]
Jp =	(JL * JM) / (JL + JM)	
π =	3.1416	

Applying the loop filter at this frequency in the configuration “Resonant Filter” will cancel the resonant pole effectively allowing higher overall loop gain without losing stability.

The Resonant filter setting allows the user to set the resonant frequency, the bandwidth of the filter as well as maximum attenuation gain in dB.

The Notch filter serves a similar purpose as the resonant filter. It has programmable bandwidth and center frequency. The Gain in the center frequency point is not programmable and depends on the bandwidth of the filter which is programmable. The resonant filter is a second order bi-quad filter with -20dB/dec roll-off.

This resonant filter is good for applications where resonances have a wide bandwidth rather than in those that have a big amplitude and narrow bandwidth.



5.10 Tools

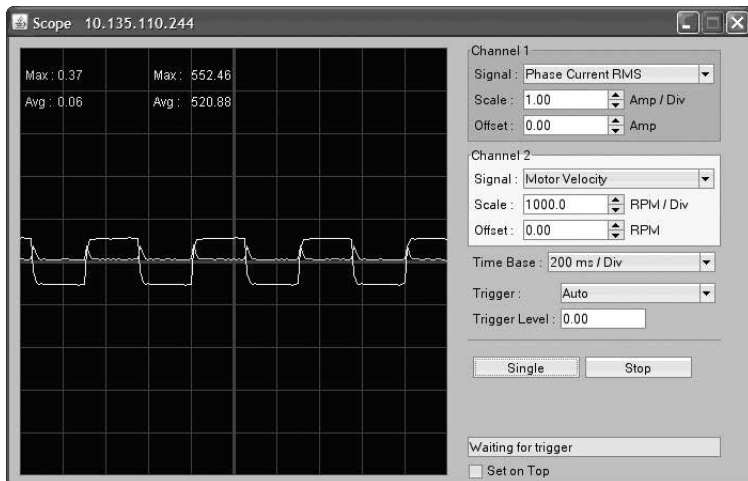
The [Tools] folder contains two action buttons: Oscilloscope and Parameter I/O View. These tools allow the user to perform real-time diagnostics.

5.10.1 Oscilloscope

The Oscilloscope tool provides a real-time display of the different electrical signals inside the PositionServo drive. The signals in the following table can be observed on the two channels of the Oscilloscope tool. Click on the [Oscilloscope] tool to open the Oscilloscope in a separate window.

Oscilloscope Parameters

Signal	Description
Phase Current RMS	Motor phase (RMS) current
Phase Current Peak	Motor phase peak current
Iq Current	Motor Iq (torque producing) current
Motor Velocity	Actual motor speed in RPM
Command Velocity	Desired motor speed in RPM (Velocity mode only)
Velocity Error	Difference in RPM between actual and commanded motor speed
Position Error	Difference between actual and commanded position (Step & Direction mode only)
Bus Voltage	DC bus voltage
Analog Input	Voltage at the drive's analog input AIN1
Target Position	Requested position
Target Position Pulses	Requested position expressed in pulses of the primary feedback device
Absolute Position	Absolute position (actual position)
Absolute Position Pulses	Absolute position expressed in pulses of the primary feedback device
Position Increment	Commanded position increment



Oscilloscope Display



Parameters

Signal Name

The user can customize the information presented on the Scope tool by choosing the drop-down box in each channel. The set of available signals depends on the drive mode. Refer to the Oscilloscope Parameters table for the list of the signals.

Scale

Scale sets the sensitivity of the display. Each division is considered one unit of the selected scale. A scale of 100 RPM/div, for example, means that the signal will rise (or descend) by one vertical division for every change of 100 RPM in the signal level. Thus, a 500-RPM signal would deflect the signal by five vertical divisions from the central reference line.

Offset

Offset sets the vertical distance from the central base line to the signal trace. This is useful if you want to compare two signals. For example, if you wish to compare the actual vs. commanded motor velocity, you would enter an offset that would move the two signals to alternate sides of the central reference line.

Time Base

Time base sets the number of milliseconds displayed per horizontal division. Higher frequencies have a shorter time base than lower frequencies. If you wanted to display one cycle of a particular signal, your time base setting would therefore be lower for high-frequency signals than for low-frequency signals.

Trigger/Trigger Level

Trigger level specifies the signal level after which the scope starts acquiring data. You can also specify which channel will be a source for the trigger. The oscilloscope display will continue to run while the signal level crosses the specified level (above if the trigger is set for rising or leading edge, or below if the trigger is set for falling or trailing edge).

Single

Also called one-shot trigger. If Single Sweep is selected, data acquisition will be stopped after the scope buffer is filled and data displayed on the screen (frozen data). To repeat data acquisition, you will need to click the **Single** button again.

Run / Stop

Select [Run] for a continuous trigger. Select [Stop] to disable the trigger.

Set on Top

Select this button to display the oscilloscope window on top of all other windows.

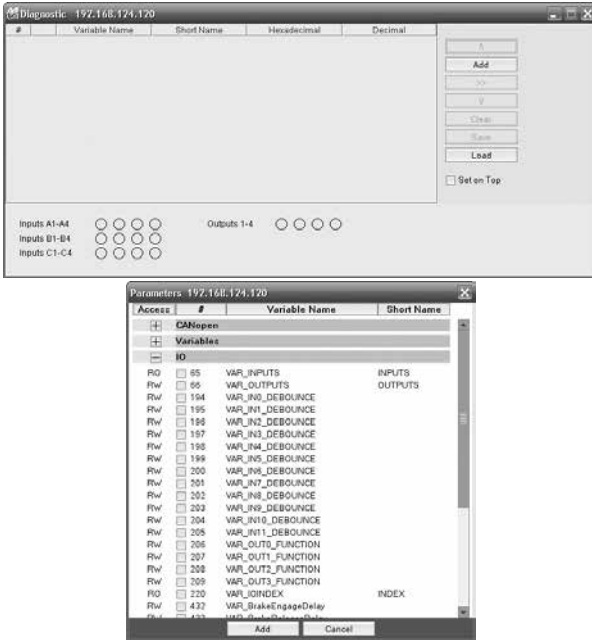
5.10.2 Parameter & I/O View

The [Parameter & IO View] tool permits the user to access the list of variables. Click on the [Parameter & IO View] button to open the diagnostic tool in a separate window. Click on the box adjacent to [Set on Top] to keep this window on top. Also known as the Debug Tool, the Parameter and I/O View permits the user to view the values of the drive's variables plus the I/O status.

To add a variable to the View List, click on [Add] then browse to [Variable Name] in the pop-up window, then click on the left arrow button. To remove a variable from this View List, click on the variable name in the View list and then click on the right arrow button. To save the variable list, click [Save]. To load the variable list, click [Load].



To edit a parameter's value, double click the [Decimal] field of the parameter. When the text is double-clicked, the background color will change. The parameter value will stop updating allowing you to change the value. However, if the interface device or user's program manipulates the value of the parameter, then your change will be overwritten in a concurrent manner.



Parameter I/O View with Variables



NOTE

By clicking [#], the variables can be automatically sorted in descending or ascending order. They can also be sorted alphabetically by clicking [Variable Name] and/or [Short Name].



NOTE

Write-only variables may not contain valid data in the "Parameter and I/O" view screen as all write-only variables in the drive use a common display buffer.

5.11 Faults

The [Faults] folder contains three action buttons upon opening and displays the most recent fault. [Load Faults] permits the user to load the entire stored fault history of the drive onto the computer. The sixteen most recent faults are displayed with the newer faults replacing the older faults in a first-in, first-out manner. In all cases, the fault on top of the list is the most recent fault. [Clear Faults] clears the fault history of the drive from within the MotionView program. The device time of the fault is the time from last power up (Power-up time = 00:00). Each fault has its code and explanation of the fault. Refer to section 8.4 for details on faults. [Clear Fault/Reset] clears the active fault and resets the drive.

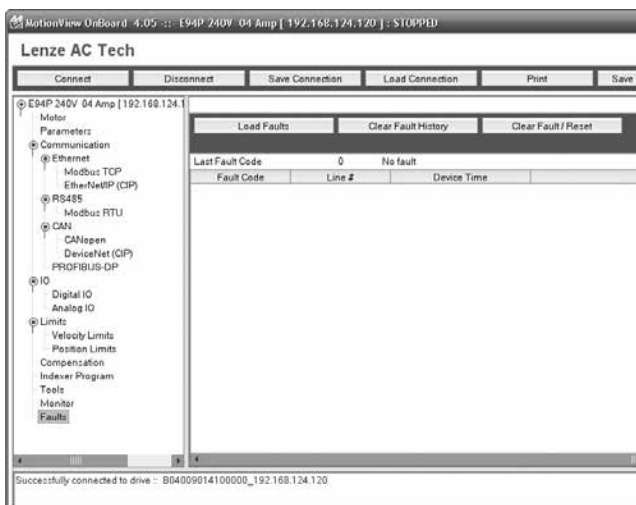


Parameters



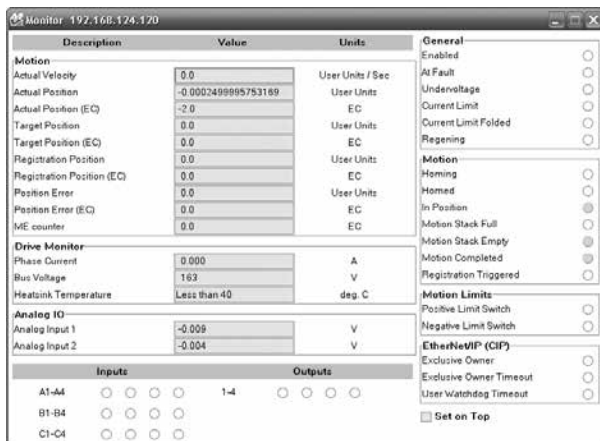
NOTE

The [Clear Faults] operation will disrupt motion and the program being executed. It is recommended not to clear faults while running an application.



5.12 Monitor

The Monitor window displays common diagnostic information for the drive's status. Click the [Set on Top] box to keep the Monitor displayed while manipulating other screens in MotionView.





6 Operation

This section offers guidance on configuring the PositionServo drive for operations in torque, velocity or position modes without requiring a user program. To use advanced programming features of PositionServo please perform all steps below and then refer to the PositionServo Programming Manual for details on how to write motion programs.

6.1 Minimum Connections

For the most basic operation, connect the PositionServo to mains (line) power at terminal P1, the servomotor power at P7 and the motor feedback as appropriate.

**DANGER!**

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait at least 60 seconds before servicing drive. Capacitors retain charge after power is removed.

As a minimum these connections must be made:

- Connect an Ethernet crossover cable between PositionServo's P2 and your PC's Ethernet port. A straight patch cable can be used if using a hub or switch.
- Connect mains power to terminal P1. Mains power must be as defined on the drive's data label (Refer to label in 'About These Instructions' section).
- When connecting to an encoder-based drive, take the encoder feedback cable and connect it to the 15 pin D-sub connector located at P4. When connecting to a resolver-based drive, take the resolver feedback cable and connect it to the 9 pin D-sub connector located at P4.
- Connect motor windings U, V, W (a.k.a. R, S, T) to terminal P7 as shown in section 4.1.1. Make sure the motor cable shield is connected as in section 3.2.
- Provide an Enable switch according to Section 6.6.
- Perform drive configuration as described in the next section.

**NOTE**

To run MotionView OnBoard on a Mac OS, run the PC emulation tool first.

**NOTE**

The recommended screen setting size for the PC is 1680 x 1050.

6.2 Ethernet Connection

Configuration, Programming and diagnostics of the PositionServo drive are typically performed over the standard 10/100 Mbps Ethernet communication port using the 'MotionView OnBoard' software contained within the drive itself.

To access the MotionView OnBoard software and configure the drive the PositionServo drive and PC must be configured to operate on the same Ethernet network. The IP addresses of the PositionServo drive, the PC, or both drive and PC may be required to be configured to enable Ethernet communications between the two devices.

**NOTE**

Any changes made to the Ethernet communication settings on the PositionServo do not take effect until the drive is powered off and powered on again. Until this time the drive will continue to use its previous settings.



NOTE

For any PC that will need regular configuration to communicate with a PositionServo Drive and if the default PC Ethernet port on your computer is already being used for another purpose (such as email, web browsing, etc.) then it may be more convenient for the operator to add an additional Ethernet port to the PC.

The most common and cost effective way to do this is by using a USB / Ethernet dongle or a PCMCIA Ethernet card. The additional port can be configured for communication to the PositionServo drive without effecting the operation of other PC functions.

6.2.1 PositionServo Ethernet Port Configuration

The IP address of the PositionServo drive is composed of four sub-octets that are separated by three dots to conform to the Class C Subnet structure. Each sub-octet can be configured with number between 1 and 254. As shipped from the factory the default IP address of a drive is:

192.168.124.120.

There are two methods of changing the current IP address. An address can be assigned to the drive automatically (dynamic IP address) when the drive is connected to a DHCP (Dynamic Host Configuration Protocol) enabled server, or the drive can have an IP address assigned to it manually by the user (static IP address). Both methods of configuring the drive's IP address are detailed herein.

6.2.1.1 Obtaining the PositionServo's Current Ethernet Settings

The current Ethernet setting and IP address of the PositionServo drive can be obtained from the drive display and keypad. Press the recessed 'mode' button (←) on the display and use the "UP" and "DOWN" buttons (▲ ▼) to access parameters IP_1, IP_2, IP_3 and IP_4. Each of these parameters contain one sub-octet of the full IP address, for example in the case of the drive default (factory set) address parameters:

IP_1 = 192
IP_2 = 168
IP_3 = 124
IP_4 = 120

By accessing these four parameters the full IP address on the drive can be obtained.

If parameters IP_1, IP_2, IP_3 and IP_4 all contain '----' rather than a numerical values it means that the drive has DHCP enabled and the DHCP server is yet to assign the drive its dynamic IP address. As soon as an IP address is assigned by the server the address assigned will be display by the drive in the above parameters. See section on obtaining IP addresses through DHCP.

6.2.1.2 Configuring the IP Address Manually (Static Address)

When connecting directly from PositionServo drive to the PC without a server or when connecting to a private network (where all devices have static IP addresses) the IP address of the PositionServo drive will need to be assigned manually.

To assign the address manually, the drive must have its DHCP mode disabled. This can be done using the drive keypad and display. Press the recessed 'mode' button (←) on the display and use the "UP" and "DOWN" buttons (▲ ▼) to access parameter 'DHCP'. Check this parameter is set to a value of '0'. If the DHCP parameter is set to '1' then use the 'mode' (←) and down (▼) arrows to set to '0' and then cycle power to the drive in order for this change to take effect. When DHCP is disabled and power cycled to the drive, it will revert back to its previous static IP address.



It is most common for the PositionServo drive IP address to be left at its default value (192.168.124.120) and to configure the PC Ethernet port to communicate on this subnet. If more than one drive needs to be connected to the PC at any one time then the IP_4 parameter can be accessed via the keypad and changed to provide a unique IP address on the network for each drive. Note that IP_4 is the only octet that can be changed (IP_1, IP2, and IP_3 are read-only) and that power must be cycled to the drive for any changes to take effect.

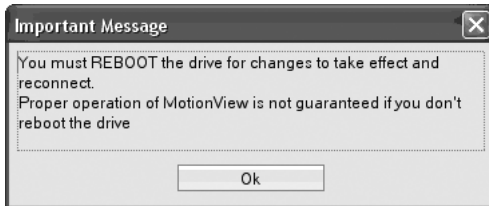
If the PositionServo drive(s) needs to be configured for a specific subnet with different values to default (for IP_1, IP_2, and IP_3, and IP_4) then this needs to be performed with the MotionView configuration tool. First establish communications using the default drive address or with an address that was established by changing IP_4 parameters via the drive keypad. Follow the rest of these instructions in order to establish communications and launch MotionView using this address. Once within the MotionView software a full IP address can be assigned.

From the Node tree within MotionView select the [Communications] folder and then the [Ethernet] sub-folder as shown herein. The settings reflect those that will appear in the software parameter view window.



The IP address, subnet mask, and default gateway address can all be edited in this screen. If the text in any of these boxes turns red once it has been entered then this means that the values or format used is invalid and the values will not be applied.

To enable DHCP, click the box adjacent to [Obtain IP Address using DHCP] to place a check mark in this box . To disable DHCP, click the box again. Power must be cycled for any changes to [Configure IP Address] to take effect. On changing any ethernet parameter value, the following dialog box will appear. Click [Ok] and cycle power for changes to take effect.





Operation

6.2.1.3 Configuring the IP Address Automatically (Dynamic Address)

When connecting a PositionServo drive onto a network domain with a DHCP enabled server (where all devices have dynamic IP addresses assigned by the server) the IP address of the PositionServo drive can also be assigned automatically by the server.

To have the address assigned automatically the drive must have its DHCP mode enabled. This can be done by using the drive keypad and display. Press the 'mode' button on the display and use the "UP" and "DOWN" buttons to access parameter 'DHCP'. Check this parameter is set to a value of '1'. If the DHCP parameter is set to '0' then use the 'mode' and up arrow to set to '1' and then cycle power to the drive in order for this change to take effect.

When the PositionServo drive is waiting for an IP address to be assigned to it by the server it will display '----' in each of the four octet parameters (IP_1, IP_2, IP_3, and IP_4) on its display. Once the address is assigned by the server it will appear in these parameters. If this parameters continue to display '----' then it is likely that a connection between the drive and server has not been established, or the server is not DHCP enabled.

DHCP can be enabled through the MotionView software for convenience should the operator wish to configure the drive using a manual (static) IP address and switch over to an automatic (dynamic) address once configuration is complete. See section 6.2.1.1 for information on enabling DHCP from within the MotionView software.



NOTE

A useful feature of the MotionView software and communications interface to the PositionServo drive is the ability to assign the drive a name (text string). This name can then be used to discover the drive's IP address and is useful when the drive has its IP address assigned automatically by the server for easy connection. Refer to section on MotionView connection window, below.

6.2.2 Configuring the PC IP Address (Windows XP)



NOTE

This section of the manual gives some guidance on how to configure the Ethernet communications setting on a PC to communicate with a PositionServo drive. Additional material for other operating systems/platforms may be available from the website or as an appendix to existing drive documentation. If the drive and PC are both assigned automatic IP addresses from a DHCP enabled server then configuration of the PC port should not be necessary.

The following is a step by step guide to configure the PC IP address in Windows XP using either the classic or category viewing mode.

To access the network settings on a Windows XP based PC:

Category (Default) View:

[Start]
[Control Panel]
[Network & Internet Connections]
[Network Connections]

Classic View:

[Start]
[Settings]
[Control Panel]
[Network Connections]



Start Menus - Windows XP	
Category (Default View)	Classic View

One of the following screens will be displayed, depending on the user's configuration of Windows XP software.

Control Panel Displays - Windows XP	
Category (Default View)	Classic View

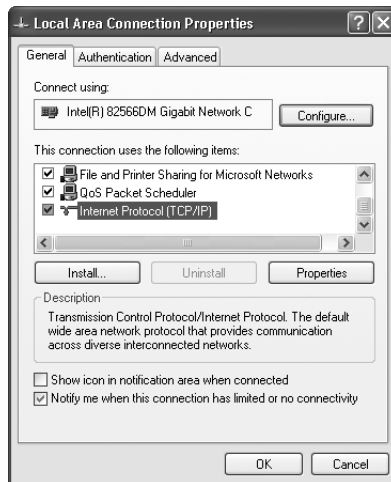


Operation

Regardless of the Windows XP viewing mode the following [Network Connections] screen will appear. Hereafter all configuration screens are the same regardless of selected Windows XP viewing mode.



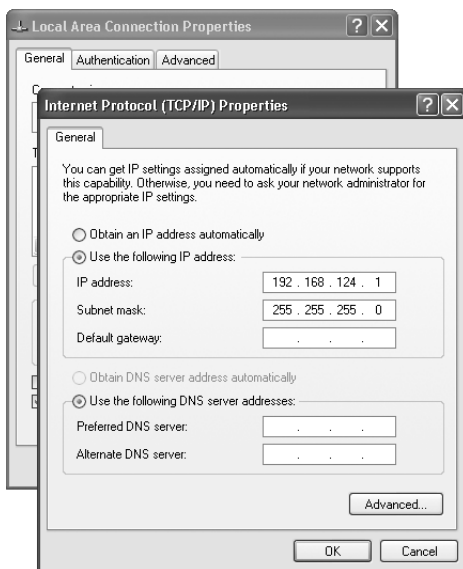
Select the connection you wish configure. [Local Area Connection] is typically the standard or local Ethernet port on the PC (the port supplied with the PC), with any additional hardwire ports displayed as [Local Area Connection x] (with x being a numerical value). Double-click the icon for the port you wish to configure. The [Local Area Connection Properties] screen will appear.



Use the vertical scroll bar on the right hand side of the screen to scroll down to the [Internet Protocol (TCP/IP)] option in the selection window. Select this option and click the [Properties] button. The [Internet Protocol (TCP/IP) Properties] screen will appear.



Select [Use the following IP address]. The IP address and Subnet mask text boxes can now be edited.



Enter an IP address for the PC. This IP address will need to be unique to the PC (different to any other device on the network) but still allow communication on the same subnet that the drive is set to. To set up the PC IP address in this way enter the first three values of the IP address box to be identical to those set in IP_1, IP_2, and IP_3 parameters on the PositionServo drive. For the last value (IP_4) pick a unique value different to any other device on that network.

If the drive IP address has been left at its factory (default) value then a logical IP address to assign to the PC might be 192.168.124.1

When exiting the IP address box the value in the subnet mask text box should default to 255.255.255.0. This value tells the PC that all other devices on the network share the same values for the first 3 Octets of their IP addresses with the last octet beginning unique to those devices. Typically the default value can be left unchanged unless a larger network needs to be specified.



NOTE

If the PC and drive need to obtain an IP address from a DHCP enabled server then the [Obtain an IP address automatically] option should remain ticked and no values should be present for either the IP address or subnet mask.

6.2.3 Initial Connection to the Drive

Before connecting to the PositionServo drive and attempting to run the MotionView software check the PC has the following features installed:

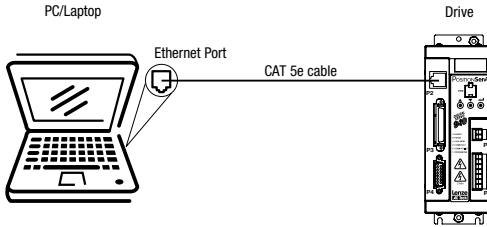
- Java Run Time Environment 1.4 or higher
(download latest version at <http://www.java.com>)
- Web Browser (Internet Explorer, Mozilla Firefox, Netscape, etc)



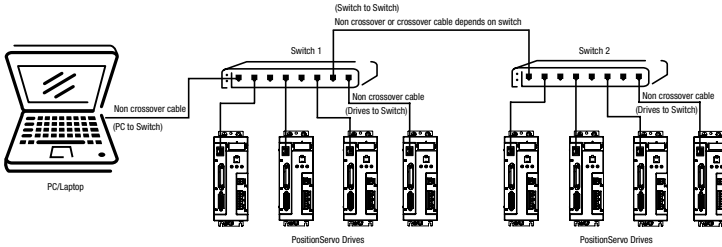
Operation

Physically connect the Drive to the PC:

To connect directly between a PC and a PositionServo drive it is recommended that a CAT 5e crossover cable be connected between the P2 port on the PositionServo drive and the Ethernet port on the PC.



To Connect from a PC to a PositionServo drive via an Ethernet switch or hub it is recommended that a CAT 5e straight through cable be connected from both the drive and PC directly to the Hub or switch.



6.2.4 Launching MotionView & Communicating to the PS Drive

Open your PC's web browser.

Enter the drive's default IP address [192.168.124.120] in the browser's Address window.



The authentication screen may be displayed if the PC does not have Java RTE version 1.4 or higher. To remedy this situation, download the latest Java RTE from <http://www.java.com>.



Java Authentication



Java Splash Screen

When MotionView has finished installing, a Java icon entitled [MotionView OnBoard] will appear on your desktop and the MVOB splash screen is displayed. Click [Run] to enter the MotionView program.

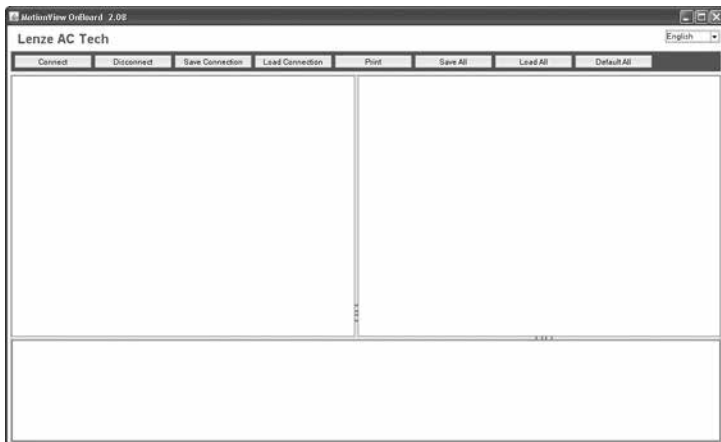


MotionView OnBoard Splash Screen



WARNING Statement on Initial MotionView Display

Once MotionView has launched, verify motor is safe to operate, click [YES, I have] then select [Connect] from the Main toolbar (top left).

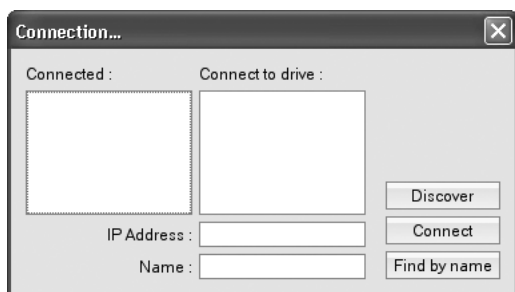


Initial MotionView Display



Operation

The Connection dialog box will appear.



Connection Dialog Box

Select [Discover] to find the drive(s) on the network available for connection.



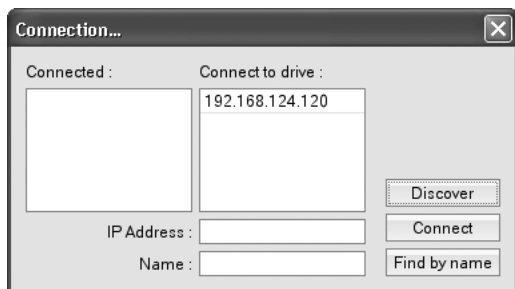
NOTE

[Discover] may fail to find the drive's IP address on a computer with both a wireless network card and a wired network card. If this happens, try one of these remedies:

- Disable the wireless network card and then use [Discover].
- Type in the drive's IP address manually at the box [IP Address].

Then click [Connect].

Highlight the drive (or drives) to be connected and click [Connect] in the dialog box.



Connection Box with Discovered Drive

In the lower left of the MotionView display, the Message Window will contain the connection status message. The message "Successfully connected to drive B04402200450_192.168.124.120" indicates that the drive B04402200450 with IP address 192.168.124.120 is connected.

Sequentially connecting 2 Ethernet-based Drives

If when trying to sequentially commission several Ethernet-based drives with the same PC, MotionView discovers the IP address, but then reports that the drive cannot be connected; open the Command window on your PC and run the command "arp -d" just before connecting MotionView to another drive.

ARP is the Address Resolution Protocol. Each PositionServo drive has two addresses, one MAC address and one IP address. ARP links these two addresses together. Each PositionServo drive has the same factory default IP address, but a different MAC address. After connecting the first drive, the Ethernet hub will cache its IP and MAC address for about 2 minutes. When another drive with the same IP address and different MAC address is connected to the network, ARP will observe the mismatching between the IP address and MAC address.



6.3 Parameter Storage and EPM Operation

6.3.1 Parameter Storage

All settable parameters are stored in the drive's internal non-volatile memory. Parameters are saved automatically when they are changed. In addition, parameters are copied to the EPM memory module located on the drive's front panel. In the unlikely event of drive failure, the EPM can be removed and inserted into the replacement drive, thus making an exact copy of the drive being replaced. This shortens down time by eliminating the configuration procedure. The EPM can also be used for replication of the drive's settings.

6.3.2 EPM Operation

When the drive is powered up, a comparison is made between the drive's internal memory and the EPM. If a correctly formatted EPM is inserted, the EPM will over-ride the internal memory with the settings of the new EPM. This allows the user to replace/clone existing drives. If the drive being cloned is of a lower power rating than the original drive and the current settings exceed the max settings of the new drive, then the current settings will default to the max settings.

**STOP!**

Never install or remove the EPM module while the drive is powered.

Most Lenze-AC Tech products use the EPM for memory storage on the drive. The memory size of the EPM is denoted by its color and drive format structure. The PositionServo drive uses a white EPM module. When the drive is powered up it checks the format style of the EPM in the EPM port. If the EPM Port is empty, or a different color EPM is inserted, the drive will display “-EP-” and no further operation is possible until a white EPM is inserted. If a white EPM with an older format or a new/blank EPM is inserted, the drive will display “FEP?” (format EPM). The drive is asking the user if he wants to reformat the EPM. To reformat the EPM, press the recessed carriage return button [↵] on the front of the drive. If you do not wish to reformat, power down the drive and remove the EPM from the drive.

**STOP!**

If the EPM contains any data from an existing drive, that data will be overwritten during this procedure. During the reformatting process, some of the data from the internal memory will be written to the EPM and some of the settings will be set to default. Check all parameters.

6.3.3 EPM Fault

If the EPM fails during operation or the EPM is removed from the EPM Port, the drive will generate a fault and display “-EP-”. The fault is logged to the drive's fault history. The fault log will list fault code 38, EPM Failure. Further operation is not possible until the EPM is replaced (inserted) and the drive's power is cycled.



6.4 Configuration of the PositionServo

Regardless of the mode in which the user wishes to operate, he must first configure the PositionServo for his particular motor, mode of operation, and additional features if used. Drive configuration consists of following steps:

- Motor Selection
- Mode of operation selection
- **Reference source selection (Very Important)**
- Drive parameters (i.e. current limit, acceleration / deceleration) setup
- Operational limits (velocity or position limits) setup
- Input / Output (I/O) setup
- Velocity / position compensator (gains) setup (Auto Tuning)
- Optionally store drive settings in a PC file and exit the MotionView program.

To configure drive:

1. Ensure that the control is properly installed and mounted. Refer to section 3 for installation instructions.
2. Perform wiring to the motor and external equipment suitable for desired operating mode and your system requirements.
3. Connect the Ethernet port P2 on the drive to your PC Ethernet port. If connecting directly to the drive from the PC, a crossover cable is required.
4. Make sure that the drive is disabled.
5. Apply power to the drive and wait until “d iS” shows on the display. For anything other than this, refer to the chart below before proceeding.

Drive Display	Fault	Remedy
-EP-	EPM missing	Insert EPM
FEEP	Format EPM	Reformatting EPM
- - -	No valid firmware	Update firmware

6. Confirm that the PC and the drive have the correct IP setting. (Section 6.2.2)
7. Launch MotionView software on your computer.
8. From the main toolbar select [Connect].
9. In the Connect dialog box, click [Discover] to ping the network for any drives. If a drive is located the address will appear in the dialog box. If no address appears then you can type the IP address in. The default address for the drive is 192.168.124.120. Click [Connect] to connect to the drive.
10. Once connected, the drive name and identifier are displayed in the upper left-hand corner of the Parameter Tree Window.
11. Select the [Motor] to be used (section 4.5).
12. Click on [Parameters] and set the following:
 - [Drive Mode]: Torque, Velocity or Position (Refer to section 6.3.1)
 - [Current limit]: enter current limit (in A RMS per phase) i.a.w. the motor.
 - [Peak current limit]: peak current limit (in A RMS per phase i.a.w. the motor)
 - [Drive PWM frequency]: 8kHz or 16kHzSet up additional parameters suitable for the drive mode selected above.
13. After drive is configured, tune the drive if operating in “Velocity”, or “Position” mode. “Torque” mode doesn’t require additional tuning or calibration. Refer to section 6.8 for details on tuning.



6.5 Position Mode Operation (gearing)

In position mode the drive will follow the master reference signals at the 1-4 inputs of P3. The distance the motor shaft rotates per each master pulse is established by the ratio of the master signal pulses to motor encoder pulses (in single loop configuration). The ratio is set by "System to Master ratio" parameter (see section 5.3.16).

Example 1

- Problem:** Setup the drive to follow a master encoder output where 1 revolution of the master encoder results in 1 revolution of the motor
- Given:** Master encoder: 4000 pulses/revolution (post quadrature)
Motor encoder: 8000 pulses/revolution (post quadrature)
- Solution:** Ratio of System (motor encoder) to Master Encoder is $8000/4000 = 2/1$
Set parameter "System to master ratio" to 2:1

Example 2

- Problem:** Setup drive so motor can follow a master encoder wheel where 1 revolution of the master encoder results in 3 revolutions of the motor
- Given:** Motor encoder: 4000 pulses/revolution (post quadrature)
Master encoder: 1000 pulses/revolution (post quadrature).
Desired "gear ratio" is 3:1
- Solution:** Ratio = (motor encoder PPR / master encoder PPR) x the "gear ratio":
(Motor PPR/Master PPR)*(3/1) => (4000/1000)*(3/1) => 12/1
Set parameter "System to master ratio" to 12:1

6.6 Enabling the PositionServo

Regardless of the selected operating mode, the PositionServo must be enabled before it can operate. A voltage in the range of 5-24 VDC connected between P3 pins 26 and 29 (input IN_A3) is used to enable the drive (section 4.1.7, note 3). There is a difference in the behavior of input IN_A3 depending on how the "Enable switch function" is set. **TIP!** If using the onboard +5VDC power supply for this purpose, wire your switch between pins P3.6 and P3.29. Jumper P3.5 to P3.26. If doing this, all inputs in group must be powered by P3.6.

When the "Enable switch function" is set to "RUN":

IN_A3 acts as positive logic ENABLE or negative logic INHIBIT input depending on:

- If user program is not running: Activating IN_A3 enables the drive
- If user program is running: Activating IN_A3 acts as negative logic
"Inhibit" and operates exactly as if parameter
"Enable switch function" set to "Inhibit"

When the "Enable switch function" set to "Inhibit":

IN_A3 acts as negative logic INHIBIT input regardless of mode or program status.

Activating input IN_A3 doesn't enable the drive. The drive can be enabled from the user's program or interface only when IN_A3 is active. Attempt to enable drive by executing the program statement "ENABLE" or from interface will cause the drive to generate a fault, F_36. Regardless of the mode of operation, if the input is deactivated while the drive is enabled, the drive will be disabled and will generate a fault, F_36.



WARNING!

Enabling the drive allows the motor to operate depending on the reference command. Before enabling the drive, make sure that the motor and machine are safe to operate and that moving elements are appropriately guarded.
Failure to comply could result in damage to equipment and/or injury to personnel!



6.7 Drive Tuning

The PositionServo Drive will likely require some tuning of its gains parameters in order to achieve best performance in the application in which it is being applied. Only when the drive is placed in Torque Mode are the gain values not required to be tuned. The table herein lists the gains parameters that should be adjusted for each of the drive operating modes. These parameters are found within the 'Compensation' folder.

MotionView Parameter	Torque Mode	Velocity Mode	Positioning Mode
Velocity P Gain	No	Yes	Yes
Velocity I Gain	No	Yes	Yes
Position P Gain	No	No	Yes
Position I Gain	No	No	Yes
Position D Gain	No	No	Yes
Position I-Limit	No	No	Yes
Gain Scaling	No	Yes	Yes

Before using the tuning procedures detailed in the next sections, ensure that the system is in a safe condition for tuning to be carried out. It is often beneficial to first tune the motor off-load to obtain approximate gains setting before fine tuning in the application.

Check that the drive output to the motor is disabled (via Input A3) and that the drive is powered up. Save any user program code previously entered into the [Indexer Program] folder in MotionView prior to tuning so it can be recalled after tuning is complete.

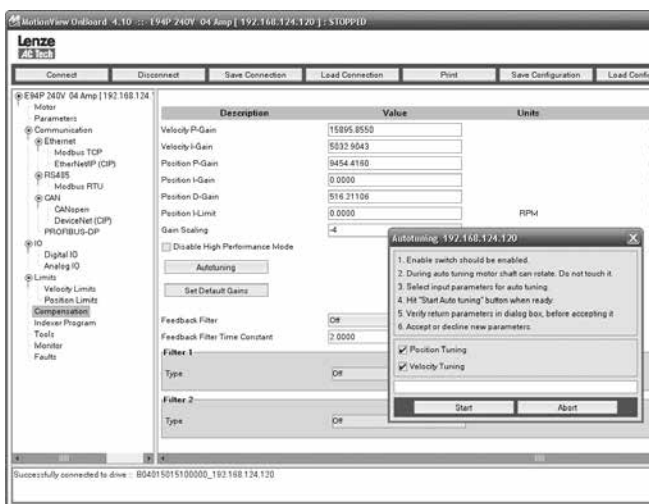


WARNING!

During both the Velocity and Position tuning procedures the PositionServo drive will perform rotation (motion) of the motor shaft in the forward and reverse directions at velocities based on the user settings. Ensure that the motor and associated mechanics of the system are safe to operate in the way specified during these procedures.

6.7.1 Auto Tuning the Drive

PositionServo drives with hardware revision 2 and higher feature Auto Tuning. To Auto Tune the drive, disable the drive. Ensure that the Indexer program is not running. Select the {Compensation} folder in the navigation tree. De-select [Disable High Performance Mode] and select [Auto Tuning]. The velocity and position loops can be tuned either individually or together.





6.7.2 Manually Tuning the Drive in Velocity Mode

The PositionServo drive may also be tuned manually. Follow the procedure in this paragraph to tune the drive in Velocity mode.

1) Parameter Setup

Set up the motor as per the instructions given in the relevant section of this manual. The motor must be configured correctly prior to tuning taking place.

The parameters Drive Mode, Reference and Enable Switch Function are configured automatically by the velocity tuning program. They are not required to be set at this stage.

2) Importing the Velocity Tuning Program

Before importing the Velocity Tuning Program, the example programs must be installed from the Documentation CD that shipped with the drive. If this has not been done then please do so now.

To load the TuneV program file to the drive, select [Indexer Program] in the MotionView Parameter Tree. Select [Import] on main toolbar. Navigate to [C:\Lenze\ACTech\MVOB\Programming_Examples]. If during the installation of the Documentation CD files a different default directory was selected, then navigate to that directory. Click on the [TuneV.txt] file and select [Open].



3) Editing the Velocity Tuning Program

The Tune Velocity Program creates a step velocity demand in the forward and reverse directions that the drive will attempt to follow (based on its velocity gain settings). The drive will run for a set time in the forward direction and then reverse the reference and run for the same set time in the reverse direction, showing the acceleration, deceleration and steady state performance.

The speed and period (time for one complete cycle - forward and reverse) is set in the Indexer program with the following statements:

```
; Motion Parameters
Define SpeedReference 5           ; speed reference in Rps
Define Period 500                 ; time in millisec
```

Adjust these parameters to values suitable to the application in which the drive is used before going to the next step.

4) Compile and Download Indexer Program to Drive

In the [Indexer program] folder in MotionView, select the [Load W Source] button on the program toolbar. The TuneV program will be compiled and sent to the drive. Click [Run] on the program toolbar to run the TuneV program. Do NOT enable the drive (via input A3) at this stage.



Operation

5) Oscilloscope Settings

Open the [Tools] folder in MotionView and select the [Oscilloscope] tool. Click the [Set on Top] box to place a checkmark in it and keep the scope on top.

In the Scope Tool Window make the following settings:

- Channel 1: Signal = "Commanded Velocity"
Scale = appropriate to "SpeedReference" value set in Indexer Program
- Channel 2: Signal = "Motor Velocity"
Scale = appropriate to "SpeedReference" value set in Indexer Program
- Timebase: = as appropriate to "Period" value of Indexer Program
- Trigger: = Channel 1, Rising Edge
- Level: = 10 RPM

For better resolution, adjust these scaling factors during the tuning procedure.

6) Compensation Folder

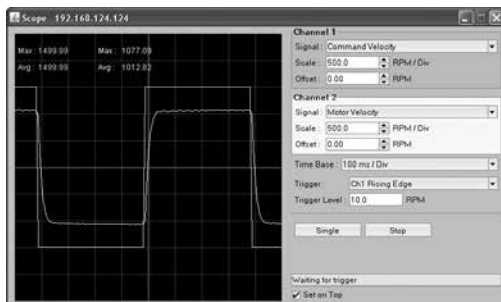
In MotionView, open the [Compensation] folder for the drive. Set [Gain Scaling] to a relatively low value, e.g. -6 for Encoder motor and -8 for a Resolver Motor. Set the [Velocity P-gain] to a mid-value (16000) and set the [Velocity I-Gain] to 0.

7) Gain Tuning

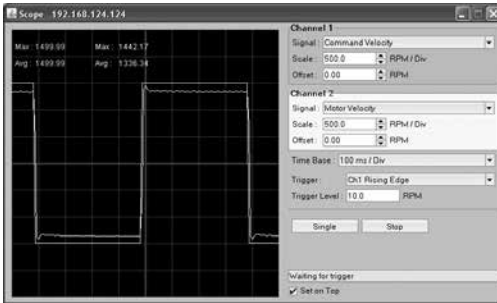
The system should now be ready to start tuning the velocity gains. Start the Oscilloscope by clicking [Run]. Apply the Enable input to Input A3 to enable the drive. At this point of the procedure it is desirable to have little to no motion until we start to increase the gain settings. If the motor vibrates uncontrollably disable the drive, lower the Gain Scaling parameter value and repeat the input enable.

Step 1: Setting the Gain Scaling Parameter

The gain scaling parameter is a 'course adjustment' of the other gain's parameter values. Steadily increase the value of the gain scaling parameter until a reasonable response is obtained from the motor (motor velocity starts to resemble the commanded velocity).



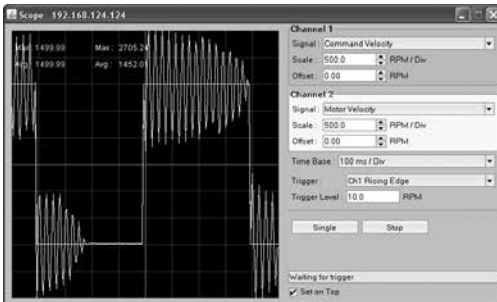
Gain Scaling set too LOW
Motor Velocity significantly different than
Commanded Velocity.



Gain Scaling set OK
Motor Velocity resembles Commanded Velocity. Motor Velocity is reasonably close with a slight overshoot.



Gain Scaling set too HIGH
Motor Velocity shows significant overshoot following the acceleration periods.



Gain Scaling set significantly too HIGH
Motor Velocity exhibits instability throughout the steady state Commanded Velocity.

Depending on the system being tuned, the motor may go from stable operation (little to no overshoot with stable steady state velocity) to instability (continuous and pronounced oscillations during steady state command) very quickly as gain scaling is increased. The bandwidth for allowing some overshoot with a quick settle time may be very small and may only be achieved through adjustment of the Velocity P-Gain, as described in Step 2. Set the gain scaling parameter to the value preceding that where significant overshoot or continuous instability occurs. With the Gain scaling parameter set move onto tuning the velocity P and I gains.



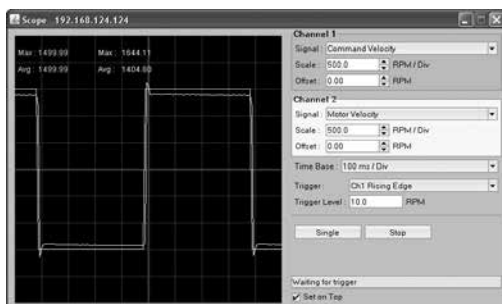
Operation

Step 2: Fine Tuning the Velocity P-Gain

Slowly alter the Velocity P-Gain (increase and decrease) and observe the motor velocity waveform on the oscilloscope. As the P-Gain increases the gradient of the velocity during acceleration and deceleration will also increase as will the final steady state velocity that is achieved. The application of too much P-Gain will eventually result in an overshoot in the motor velocity, and further increases will result in larger overshooting to the point that instability (continuous oscillation) occurs.

Increase the velocity P-gain until some overshoot occurs. Some overshoot is generally ok, and the objective is typically to achieve the shortest possible settle time (steady state velocity). When the system appears to have reached the shortest possible settle time, with acceptable overshoot, cease from increasing the P-Gain.

Scope traces will be similar to those shown in Step 1, however the P-gain will now be given a more precise adjustment in order to obtain the best possible tuning.

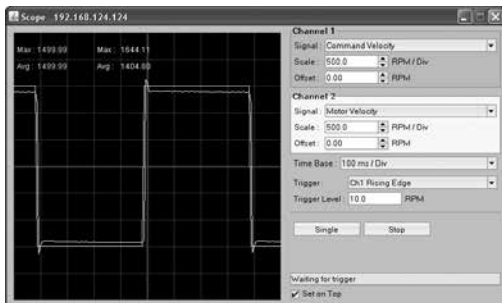


Good Fine Tuning of the P-Gain
Small overshoot with excellent settle time and steady state velocity regulation.

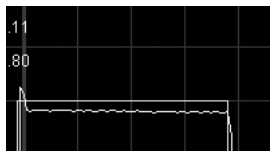
Step 3: Setting the Velocity I-Gain

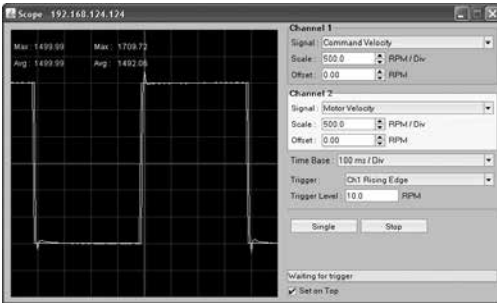
The purpose of the velocity I-gain is to correct any error that is present between the commanded velocity and the steady state velocity that could not be rectified by adjustment of the velocity P-Gain. Adjustment of the velocity I-gain can also reduce the steady state ripple that may occur in the velocity waveform. Lastly, velocity I-gain has a positive effect on the holding torque produced by the motor.

Slowly increase the "Velocity I-Gain" and check for correction of the steady state error in the velocity waveform. Continuing to increase the velocity I-gain will eventually result in increased overshoot and instability in the motor velocity waveform. Stop increasing the I-Gain when additional overshoot or instability starts to occur.



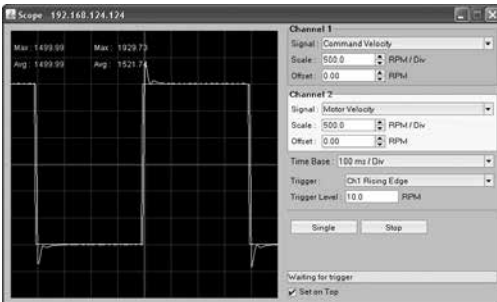
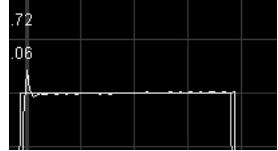
I-Gain set too LOW
Error exists between Commanded steady state velocity and Actual steady state velocity





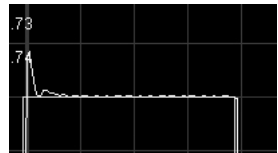
I-Gain set OK

No error between Commanded steady state velocity and Actual steady state velocity with excellent stability.



I-Gain set too HIGH

Additional overshoot and oscillations are starting to occur. Steady state velocity regulation



Step 4: Check Motor Currents

Finally check the motor currents on the Oscilloscope. Make the following settings to the oscilloscope.

Channel 1:

Signal = "Phase Current RMS"

Scale = as appropriate to peak current limit set in drive parameters (MotionView)

Timebase: = as appropriate to "Period" value of Indexer Program

Trigger: = Channel 2, Rising Edge

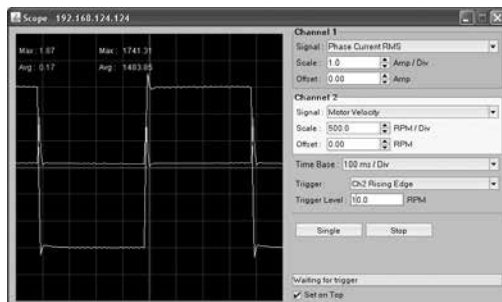
Level: = 10 RPM

Observe the waveforms to insure there are no significant oscillations. Reduce the gains values if necessary.

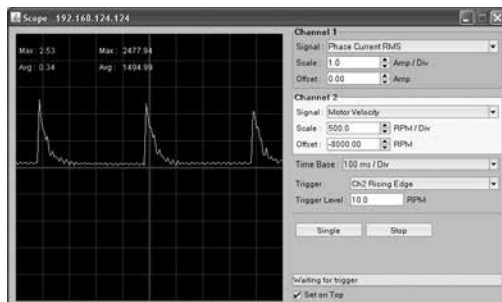
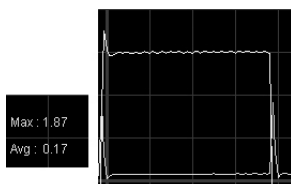
The current waveform should be showing spikes of current during acceleration / deceleration and steady state current during any steady state velocity. The maximum value (peak value) of the current waveform is shown at the top of the oscilloscope screen. This maximum value can be compared to the drive nominal current and peak current settings to check how much of the motors potential performance is being used and if optimum performance is being achieved.



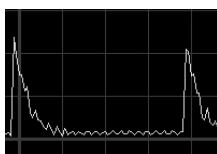
Operation



Good Current Trace
Uniform current pulses during accel/
deceleration and stable current during steady
state velocity.



Instability in Drive Output Current
(Note: Channel 2 trace has been removed for
clarity).



8) End Velocity Tuning

Remove the Enable Input from input A3 (disable the drive). In MotionView, click on the [Indexer] folder for the drive. Click [Reset] on the program toolbar. If the drive is to be run in just velocity mode then tuning is now complete. If the drive is to be used in Positioning mode continue with 'Tuning the Drive in Position Mode', section 6.8.3.

6.7.3 Manually Tuning the Drive in Position Mode

The Position Loop can also be manually tuned. Manual Velocity Tuning should be carried out prior to the manual tuning of the position loop. Refer to the Velocity Tuning section, 6.7.2.

1) Parameter Set up

In MotionView, open the [Limits] folder and then the [Position Limits] sub-folder. Set the [Position Error] and [Max Error Time] parameters to their maximum values to effectively disable the position error trip while tuning takes place. Ensure the system is safe to operate in this manner.

Position Error = 32767

Max Error Time = 8000

The Drive Mode, Reference and Enable Switch Function parameters are automatically configured by the velocity tuning program. They do not require setting at this stage.

2) Importing the Position Tuning Program

Before importing the Position Tuning Program, the example programs must be installed from the Documentation CD that shipped with the drive. If this has not been done then please do so now.

To load the TuneP program file to the drive, select [Indexer Program] in MotionView. Select [Import] on main toolbar. Navigate to [C:\Lenze-ACTech\MVOB\Programming_Examples]. If during the installation of the Documentation CD files a different default directory was selected, then navigate to that directory. Click on the [TuneP.txt] file and select [Open].



Operation

6) Compensation Folder

Open the [Compensation] folder in MotionView.

Leave the Velocity P-Gain and Velocity I Gain unchanged, as they should already have been setup during velocity tuning. Do not adjust the Gain Scaling Parameter during this procedure.

Set the [Position P-gain] to a low value (e.g. 100) and set the [Position I-Gain] and [Position D-Gain] to 0.

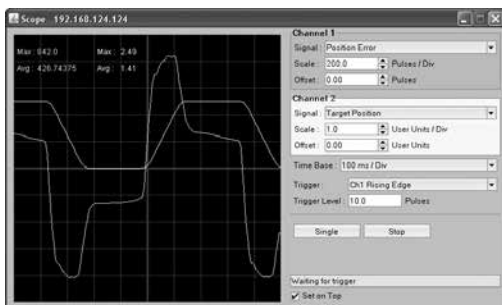
7) Gain Tuning

The system should now be ready to start tuning the position loop. Start the Oscilloscope by clicking [Run]. Apply the Enable input A3 to enable the drive.

The general goal in tuning the position loop is to achieve the minimum position error while maintaining system stability. Some experimentation with gain values will be required to achieve the best performance for the application.

Step 1: Setting the Position P-Gain

Slowly increase the Position P-Gain while watching the position error waveform on oscilloscope Channel 1. It is important to watch both the Max Error as well as the Average Error. While increasing Position P-gain, it becomes apparent that both the Max Error as well as the Average Error decrease.



Position P-Gain set too LOW
Large Position Error occurring and large error
in final positioning achieved

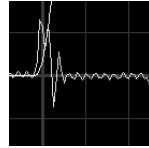


Increased Position P-Gain
Shows improvement to the maximum error
and the final positioning accuracy

At some point while increasing the P-Gain, additional oscillations (Average Error) will start to appear on the position error waveform.



Further Increased Position P-Gain Shows very good reduction to the maximum error but with additional oscillations starting to occur.

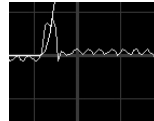


Step 2: Setting the Position D-Gain

Slowly increase the D-Gain while watching the position error waveform on oscilloscope Channel 1. As the D-Gain is increased, the position error oscillation caused by the P-Gain, should start to decrease. Continue to increase the D-Gain until oscillation is gone or until D-Gain is no longer having any apparent effect.



Adjustment of Position D-Gain in conjunction with the P-Gain dampens out additional oscillations while improving position error.



For optimum tuning, it is sometimes required to repeat the process of increasing the P-Gain until a slight oscillation occurs and then increase the D-Gain to suppress that oscillation. This procedure can be repeated until the increasing of D-Gain has negligible effect on the position error waveform.

Step 3: Setting the Position I-Gain and Position I-Gain Limit

The objective here is to minimize the position error during steady state operation and improve positioning accuracy. Start to increase the Position I-gain. Increasing the I-gain will increase the drive's reaction time while the I-Limit will set the maximum influence that the I-Gain can have on the Integral loop. When adjusting the I-gain start with a very small value for the I-gain (e.g. 1) then increase the I-gain parameter value until stand-still error is compensated and positioning accuracy is satisfactory. Remember that large values of Position I-limit can cause a large instability in the control loop and unsettled oscillation of the system mechanics.



Position Error trace following the tuning of Position P-, I- and D-Gains



Operation

Step 4: Check Motor Currents

Set the oscilloscope channel 2 to 'Phase Current RMS'

Channel 2:

Signal = "Phase Current RMS"

Scale = as appropriate to peak current limit set in drive parameters (MotionView)

Timebase: = as appropriate to the "Period" of the moves being generated

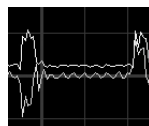
Trigger: = Ch1 Rising Edge

Level: = 10 Pulses

Observe the Current waveform to make sure that there are no significant oscillations during the steady state sections of the position profile (times when target position is not changing). If so then decrease the gains values until the oscillations are either removed or reduced to an acceptable level.



Minimal oscillation when motor positioned to target position.



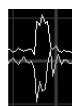
8) Setting the Position Error Limits

Look at the position error waveform on the oscilloscope. Note the maximum time that position errors exist (from the time axis of the scope) and the maximum peak errors being seen (from the value at the top of the screen). Use this values to set the position error limits to provide suitable position error protection for the application.

Open the 'Limits' folder and 'Position Limits' sub-folder within the MotionView node tree and set suitable values for the 'Position Error' and 'Max Error Time' parameters.



Maximum error and time period for error existing.



Time Base : 100 ms / Div



In this particular example maximum error in pulses is 95.0. The time this peak error occurs can be read from the oscilloscope at approximately ½ of a division with each division equal to 100ms, hence the error pulse lasts approximately 50mS. Suitable settings for position error within this application might be as follows, although looser or tighter limits could be applied depending on the requirements of the application.

Description	Value
Position Error	100
Max Error Time	50

9) End Tuning

Remove the Enable Input from input A3 (disable the drive).

Click on the [Indexer Program] folder in MotionView. Click the [Reset] button at the top of the indexer programming screen.

Tuning is now complete.

6.8 Upgrading Firmware

Starting with hardware revision 2 and higher, MotionView OnBoard (MVOB) features an [Upgrade] action button located in the top right-hand corner. The [Upgrade] selection launches a firmware loading utility to easily upgrade the drive's firmware revision. Browse to the firmware ".jar" file on your local PC and follow the prompts.

After upgrading the firmware re-download MVOB from the drive as the firmware may contain a newer version. If the drive displays "FEP?", the new firmware contains additional parameter data from the previously installed firmware. Press and hold the drive's [->] button until the drive display reads "bUSY". Release the button and the drive will format the EPM to the new firmware revision.



Upgrade Pop-up Window



7 Quick Start Reference

This section provides instructions for External Control, Minimum Connections and Parameter Settings to quickly setup a PositionServo drive for External Torque, Velocity or Positioning Modes. The sections are NOT a substitute for reading the entire PositionServo User Manual. Observe all safety notices in this manual.

7.1 Quick Start - External Torque Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector - Pin	Input Name	Description
P3-22	ACOM	Analog Common Reference from Controller
P3-24	AIN1+	Analog Torque Reference from Controller – Positive
P3-25	AIN1-	Analog Torque Reference from Controller – Negative
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (If required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z=	Buffered Encoder Output
P3-23	AO	Analog Output
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter

Mandatory Parameter Settings:

These Parameters are required to be set prior to running the drive

Folder / Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Torque]
	Reference	Set to [External]
IO / Analog IO	Analog Input (Current Scale)	Set to required current per 1V input from controller
	Analog Input Dead band	Set zero torque Dead band in mV
	Analog Input Offset	Set Analog Offset for Torque Reference
IO / Digital IO	Enable Switch Function	Set to [Run]



Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of a Zero Speed Output
	At Speed	Set Target Speed for activation of a At Speed Output
	Speed Window	Set bandwidth for activation of a At Speed Output

7.2 Quick Start - External Velocity Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector - Pin	Input Name	Description
P3-22	ACOM	Analog Common Reference from Controller
P3-24	AIN1+	Analog Velocity Reference from Controller – Positive
P3-25	AIN1-	Analog Velocity Reference from Controller – Negative
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (If required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z=	Buffered Encoder Output
P3-23	A0	Analog Output
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter



Quick Start Reference

Mandatory Parameter Settings:

These parameters are required to be set prior to running the drive.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Velocity]
	Reference	Set to [External]
	Enable Velocity Accel / Decel Limits	Enable Ramp rates for Velocity Mode
	Velocity Accel Limit	Set required Acceleration Limit for Velocity command
	Velocity Decel Limit	Set required Deceleration Limit for Velocity command
IO / Analog IO	Analog Input (Velocity Scale)	Set to required velocity per 1 volt input from controller
	Analog Input Dead band	Set zero velocity Dead band in mV
	Analog Input Offset	Set Analog Offset for velocity Reference
IO / Digital IO	Enable Switch Function	Set to [Run]
Compensation	Velocity P-Gain	Set P-Gain for Velocity loop
(see tuning section)	Velocity I_Gain	Set I-Gain for Velocity loop
	Gain Scaling	Set Gain Scaling Parameter

Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of Zero Speed Output
	At Speed	Set Target Speed for activation of At Speed Output
	Speed Window	Set bandwidth for activation of At Speed Output



7.3 Quick Start - External Positioning Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector-Pin	Input Name	Description
P3-1	MA+	Position Reference Input for Master Encoder / Step-Direction Input
P3-2	MA-	Position Reference Input for Master Encoder / Step-Direction Input
P3-3	MB+	Position Reference Input for Master Encoder / Step-Direction Input
P3-4	MB-	Position Reference Input for Master Encoder / Step-Direction Input
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (if required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z=	Buffered Encoder Output
P3-22	ACOM	Analog Common Reference from Controller
P3-23	AO	Analog Output
P3-27	IN_A1	Positive Limit Switch: Required if Limit Switch Function is used
P3-28	IN_A2	Negative Limit Switch: Required if Limit Switch Function is used
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter



Quick Start Reference

Mandatory Parameter Settings:

These parameters are required to be set prior to running the drive

Folder / Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Position]
	Reference	Set to [External]
	Step Input Type	Set to [S/D] or [Master Encoder]. (S/D = Step + Direction)
	System to Master Ratio	Set 'Master' and 'Slave' values to gear position input pulses to pulse revolution of the motor shaft
IO / Digital IO	Enable Switch Function	Set to [Run]
Limits / Position Limits	Position Error	Set Position Error Limit specific to application
	Max Error Time	Set Position Error Time specific to application
Compensation (see tuning section)	Velocity P-Gain	Set P-Gain for Velocity loop
	Velocity I_Gain	Set I-Gain for Velocity loop
	Position P-Gain	Set P-Gain for Position Loop
	Position I-Gain	Set I-Gain for Position Loop
	Position D-Gain	Set D-Gain for Position Loop
	Position I-Limit	Set I-Limit for Position Loop
	Gain Scaling	Set Gain Scaling Parameter

Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
	Hard Limit Switch Actions	Set if Hard Limit Switches used in Application
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of a Zero Speed Output
	At Speed	Set Target Speed for activation of a At Speed Output
	Speed Window	Set bandwidth for activation of a At Speed Output



8 Diagnostics

8.1 Diagnostic Display

Apply power to the drive and wait until “d ,5” shows on the display. For anything other than “d ,5”, refer to the chart below before proceeding.

Drive Display	Fault	Remedy
-EP-	EPM missing	Insert EPM
FEPP	Format EPM	Reformatting EPM
---	No valid firmware	Update firmware

PositionServo drives are equipped with a diagnostic LED display and three push buttons to select displayed information and to edit a limited set of parameter values.

Parameters can be scrolled by using the “UP” and “DOWN” (▲▼) buttons. To view a value, press “Enter” (⏏). To return back to scroll mode press “Enter” again. After pressing the “Enter” button on editable parameters, the yellow LED “C” will blink indicating that parameter value can be changed. Use “UP” and “DOWN” buttons to change the value. Press “Enter” to store new setting and return back to scroll mode.

Display	Description
StAt	current drive status - ⏏ to view: rUn - drive running d ,5 - drive disabled F_XX - drive fault. Where XX is the fault code (section 8.4.1)
Hx.xx	Hardware revision (e.g. H2.00)
Fx.xx	Firmware revision (e.g. F2.06)
bAUd	RS232/RS485(normal mode) baud rate - ⏏ to set ▲▼ selects from 2400 to 115200 baudrates
Adr	Drive's address - ⏏ to set ▲▼ sets 0 - 31 drive's address
FLtS	Stored fault's history - ⏏ to view ▲▼ scroll through stored faults F0XX - F7XX, “XX” is the fault code (section 8.4.1)
Ht	Heatsink temperature - ⏏ to view Shows heatsink temperature in °C if greater than 40°C. Otherwise shows “LO” (low).
EnC	Encoder activity - ⏏ to view Shows primary encoder counts for encoder diagnostics activity
HALL	Displays motor's hall sensor states - ⏏ to view Shows motor hall states in form XXX, where X is 1 or 0 - sensor logic states.
boot	0 = Autoboot disabled 1 = Autoboot enabled (Feature available in FW 3.50 or higher)
bUS	Displays drive DC bus voltage - ⏏ to view Shows DC bus voltage value
Urr	Displays motor's phase current (RMS) Shows current value if drive is enabled, otherwise shows “d ,5”
CRnb	CAN Baudrate
CRnA	CAN Address
CRno	CAN Operational Mode
CRnd	CAN Delay
CRnE	CAN Enable/disable

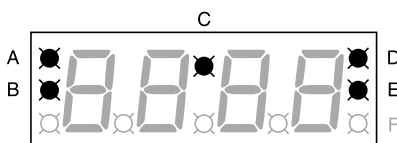


Diagnostics

Display	Description
dHCP	Ethernet DHCP Configuration: 0="dHCP" is disabled; 1="dHCP" is enabled.
I P_4	IP Address Octet 4
I P_3	IP Address Octet 3
I P_2	IP Address Octet 2
I P_1	IP Address Octet 1
Ptc	Displays the motor ptc resistance in ohms
A in 1	Displays the voltage on Drive Analog Input 1 (Ain1)
A in 2	Displays the voltage on Drive Analog Input 2 (Ain2)

8.2 Diagnostic LEDs

The PositionServo has five diagnostic LEDs located around the periphery of the front panel display as shown in the drawing below. These LEDs are designed to help monitor system status and activity as well as troubleshoot any faults.



S913

LED	Function	Description
A	Enable	Orange LED indicates that the drive is ENABLED (running).
B	Regen	Yellow LED indicates the drive is in regeneration mode.
C	Data Entry	Yellow LED will flash when changing.
D	Comm Fault	Red LED illuminates upon a communication fault. (in CANbus only)
E	Comm Activity	Green LED flashes to indicate communication activity.

8.3 Stop/Reset

With hardware version 2 and higher, MotionView OnBoard (MVOB) features a [Stop/Reset] action button in the top right-hand corner. Pressing the red [Stop/Reset] button causes all motion to stop and resets the drive.



Stop/Reset Button



8.4 Faults

8.4.1 Fault Codes

Faults in the drive are immediately shown on the drive display. The fault condition is also recorded to the drive trip log and the DFaults register inside the drive. The various trip conditions, as they appear on the display of the drive are listed in the table below.

Fault Codes as Displayed on the Drive

Fault Code (Display)	Fault	Description
F_OU	Over voltage	Drive bus voltage reached the maximum level, typically due to motor regeneration
F_Fb	Feedback error	Invalid Hall sensors code (DFAULT = 2); or Resolver signal lost (DFAULT = 11).
F_OC	Over current	Drive exceeded peak current limit. Software incapable of regulating current within 15% for more than 20mS. Usually results in wrong motor data or poor tuning.
F_Ot	Over temperature	Drive heatsink temperature has reached maximum rating. Trip Point = 100°C for all drives except 480V 6A & 9A drives Trip Point = 108°C for 480V 6A & 9A drives
F_EF	ISO13849-1 fault	The drive is disabled by the ISO13849-1 Safety Function
F_OS	Over speed	Motor has reached velocity above its specified limit
F_PE	Position Error Excess	Position error has exceeded maximum value.
F_bd	Bad motor data	Motor profile data is invalid or no motor is selected.
F_EP	EPM failure	EPM failure on power up
-EP-	EPM missing	EPM not recognized (connected) on power up
F_OT	Motor over temperature	Motor over temperature switch activated; Optional motor temperature sensor (PTC) indicates that the motor windings have reached maximum temperature
F_ID	Subprocessor failure	Error in data exchange between processors. Usually occurs when EMI level is high due to poor shielding and grounding.
F_I3	Current feedback error	Current sensor offset is too big (usually noise related).
F_I4	Under voltage	(Applies to drive's with hardware version 1). Occurs when the bus voltage level drops below 50% of nominal bus voltage while drive is operating. An attempt to enable the drive with low bus voltage will also result in this fault.
F_I5	Hardware overload protection	Occurs when the phase current becomes higher than 400% of total drive's current capability for more than 5µs.
F_I6	Internal Error	Associated with noise. Troubleshoot grounding. If error persists contact factory for technical support.
F_I7	Internal Error	Associated with noise. Troubleshoot grounding. If error persists contact factory for technical support.
F_I8	Arithmetic Error Division by zero	Statement executed within the Indexer Program results in a division by 0 being performed. Drive programming error (error in drive source code).
F_I9	Arithmetic Error Register overflow	Statement executed within the Indexer Program results in a value being generated that is too big to be stored in the requested register. Drive programming error (error in drive source code).
F_I20	Subroutine stack overflow	Exceeded 32 levels subroutines stack depth. Caused by executing excessive subroutine calls without a RETURN statement. Drive programming error (error in drive source code).
F_I21	Subroutine stack underflow	Executing RETURN statement without preceding call to subroutine. Drive programming error (error in drive source code).
F_I22	Arithmetic stack overflow	Variable evaluation stack overflow. Expression too complicated for compiler to process. Drive programming error (error in drive source code).
F_I23	Motion Queue overflow	32 levels depth exceeded. Drive programming error (in drive source code).
F_I24	Motion Queue underflow	Relates to the MDV statements in the Indexer Program. Drive programming error (error in drive source code).
F_I25	Unknown opcode	Byte code interpreter error; May occur when program is missing the closing END statement; when subroutine has no RETURN statement; or if data in EPM is corrupted at run-time



Diagnostics

Fault Code (Display)	Fault	Description
F_26	Unknown byte code	Byte code interpreter error; May occur when program is missing the closing END statement; when subroutine has no RETURN statement; or if data in EPM is corrupted at run-time
F_27	Drive disabled	Attempt to execute motion while drive is disabled. Drive programming error (error in drive source code).
F_28	Accel too high	Motion statement parameters calculate an Accel value above the system capability. Drive programming error (error in drive source code).
F_29	Accel too low	Motion statement parameters calculate an Accel value below the system capability. Drive programming error (error in drive source code).
F_30	Velocity too high	Motion statement parameters calculate a velocity above the system capability. Drive programming error (error in drive source code).
F_31	Velocity too low	Motion statement parameters calculate a velocity below the system capability. Drive programming error (error in drive source code).
F_32	Positive Limit Switch	Positive limit switch is activated. (Only available while drive is in position mode)
F_33	Negative Limit Switch	Negative limit switch is activated. (Only available while drive is in position mode)
F_34	Positive motion w/ Pos Lim Sw ON	Attempt at positive motion with engaged positive limit switch
F_35	Negative motion w/ Neg Lim Sw ON	Attempt at negative motion with engaged negative limit switch
F_36	Drive Disabled by User at Enable Input	The drive is disabled while operating or an attempt is made to enable the drive without deactivating "Inhibit input". "Inhibit" input has reverse polarity
F_37	Under voltage	(Applies to drive's with hardware version 2 and higher). Occurs when the bus voltage level drops below 50% of nominal bus voltage while drive is operating. An attempt to enable the drive with low bus voltage will also result in this fault.
F_38	EPM Loss	EPM Failure
F_39	Positive soft limit reached	Programmed (Soft) absolute limits reached during motion
F_40	Negative soft limit reached	Programmed (Soft) absolute limits reached during motion
F_41	Unknown Variable ID	Attempt to use variable with unknown ID from user program. Drive programming error (error in drive source code).
F_42	Missing Hardware	Ethernet port failure.
F_43	DeviceNet Module Error	DeviceNet module configured but not detected.
F_44	Bad Memory Index	Memory index out of limits when writing user variables to RAM.
F_45	2nd Encoder Position Error	Secondary encoder position error limit has exceeded maximum value (applies to hardware version 1)
F_46	PFB module error	PROFIBUS module configured but not detected.
F_47	PFB monitor timeout	PROFIBUS network monitor timeout error.
F_48	PFB exchange timeout	PROFIBUS data exchange timeout error.
F_49	Illegal manipulation of APOS	The APOS variable cannot be manipulated while a MOVE is being executed
F_51	Unspecified DSP fault	General internal fault.
F_52	Drive disabled-motion	Drive disabled while in motion.



8.4.2 Fault Event

When the drive encounters any fault, the following events occur:

- Drive is disabled
- Internal status is set to “Fault”
- Fault number is logged in the drive’s internal memory for later interrogation
- Digital output(s), if configured for “Run Time Fault”, are asserted
- Digital output(s), if configured for READY, are de-asserted
- If the display is in the default status mode, the LEDs display FOXX where XX is current fault code.
- “Enable” LED turns OFF

8.4.3 Fault Reset

Fault reset is accomplished by disabling or re-enabling the drive depending on the setting of the [Fault Reset] parameter (section 5.3.8).

If [On Disable] is selected, the fault is cleared when the drive is disabled.

If [On Enable] is selected, the fault is cleared when the drive is re-enabled.

8.4 Troubleshooting



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait at least 60 seconds before servicing drive. Capacitors retain charge after power is removed.

Before troubleshooting

Perform the following steps before starting any procedure in this section:

- Disconnect AC or DC voltage input from the PositionServo. Wait at least 60 seconds for the power to discharge.
- Check the PositionServo closely for damaged components.
- Check that no foreign material has fallen or become lodged in the PositionServo.
- Verify that every connection is correct and in good condition.
- Verify that there are no short circuits or grounded connections.
- Check that the drive’s rated phase current and RMS voltage are consistent with the motor ratings.

For additional assistance, contact your local PositionServo authorized distributor.

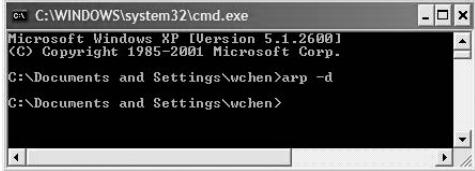
Problem	External line fuse blows
Possible Cause	Line fuses are the wrong size Motor leads or incoming power leads are shorted to ground. <i>Nuisance tripping caused by EMI noise spikes caused by poor grounding and/or shielding.</i>
Suggested Solution	<ul style="list-style-type: none"> • Check that line fuses are properly sized for the motor being used. • Check motor cable and incoming power for shorts. • Check that you follow recommendation for shielding and grounding listed in section “shielding and grounding” early in this manual.



Diagnostics

Problem	Ready LED is on but motor does not run.
Suggested Solution	<p>If in Torque or Velocity mode: Reference voltage input signal is not applied. Reference signal is not connected to the PositionServo input properly; connections are open. In MotionView program check <Parameters> <Reference> set to <External></p> <p>For Velocity mode only: In MotionView check <Parameters> <Compensation><Velocity loop filter> P-gain must be set to value more then 0 in order to run. Without load motor will run with P-gain set as low as 20 but under load might not. If P-gain is set to 0 motor will not run at all.</p> <p>In Position mode with master encoder motion source (no program) Reference voltage input signal source is not properly selected. In MotionView program check <Parameters> <Reference> set to <External></p> <p>In Position mode using indexing program Variables ACCEL, DECEL,MAXV, UNITS are not set or set to 0. Before attempting the move set values of motion parameters ACCEL, DECEL,MAXV, UNITS</p>

Problem	In velocity mode, the motor runs away.
Possible Cause	<ul style="list-style-type: none">• Hall sensors or encoder mis-wired.• PositionServo not programmed for motor connected.
Suggested Solution	<ul style="list-style-type: none">• Check Hall sensor and encoder connections.• Check that the proper motor is selected.

Problem	Cannot connect second drive when sequentially connecting 2 Ethernet-based Drives
Possible Cause	If when trying to sequentially commission several Ethernet-based drives with the same PC, MotionView discovers the IP address, but then reports that the drive cannot be connected. Due to the fact that the PS drives are shipped with the same factory default IP address.
Suggested Solution	<p>After the first drive is connected, open the Command window on your PC and run the command "arp -d" just before connecting MotionView to another drive.</p>  <pre>C:\WINDOWS\system32\cmd.exe Microsoft Windows XP [Version 5.1.2600] (G) Copyright 1985-2001 Microsoft Corp. C:\Documents and Settings\uchen>arp -d C:\Documents and Settings\uchen></pre> <p>ARP is the Address Resolution Protocol. Each PositionServo drive has two addresses, one MAC address and one IP address. ARP links these two addresses together. Each PositionServo drive has the same factory default IP address, but a different MAC address. After connecting the first drive, the Ethernet hub will cache its IP and MAC address for about 2 minutes. When another drive with the same IP address and different MAC address is connected to the network, ARP will observe the mismatching between the IP address and MAC address.</p>

Lenze Americas Corporation • Lenze AC Tech Corporation
630 Douglas Street • Uxbridge, MA 01569 • USA
Sales: (800) 217 9100 • Service (508) 278 9100
www.lenze.com

S94H201E-e1



S929

PositionServo with RS-232
Users Manual

Copyright ©2005 by AC Technology Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from AC Technology Corporation. The information and technical data in this manual are subject to change without notice. AC Tech makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. AC Tech assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView[®], PositionServo[®], and all related indicia are either registered trademarks or trademarks of Lenze AG in the United States and other countries.

This document printed in the United States of America



1	Introduction	5
1.1	About These Instructions	5
1.2	Scope of Supply	6
1.3	Legal Regulations	6
2	Technical Data	7
2.1	Electrical Characteristics	7
2.2	Environment	7
2.3	Operating Modes	8
2.4	Connections and I/O	8
2.5	Digital I/O Ratings	8
2.6	Power Ratings	9
2.7	Dimensions	9
2.8	Clearance for Cooling Air Circulation	10
3	Installation	11
3.1	Wiring	12
3.2	Shielding and Grounding	12
3.2.1	General Guidelines	12
3.2.2	EMI Protection	13
3.2.3	Enclosure	13
3.3	Line Filtering	13
3.4	Heat Sinking	14
3.5	Line (Mains) Fusing	14
3.6	Fuse Recommendations	14
4	Interface	15
4.1	External Connectors	15
4.1.1	P1 & P7 - Input Power and Output Power Connections	15
4.1.2	P2 - Serial Communications Port	16
4.1.3	P3 - Controller Interface	16
4.1.4	P4 - Motor Feedback / Second Loop Encoder Input	17
4.1.5	P5 - 24 VDC Back-up Power Input	19
4.1.6	P6 - Braking Resistor and DC Bus	19
4.1.7	Connector and Wiring Notes	19
4.1.8	P11 - Resolver Interface Module (option)	20
4.1.9	P12 - Second Encoder Interface Module (option)	21
4.2	Digital I/O Details	22
4.2.1	Step & Direction / Master Encoder Inputs (P3, pins 1-4)	22
4.2.2	Digital Outputs	23
4.2.3	Digital Inputs	24
4.3	Analog I/O Details	25
4.3.1	Analog Reference Input	25
4.3.2	Analog Output	25
4.4	Communication Interfaces	26
4.4.1	RS232 Interface (standard)	26
4.4.2	RS485 Interface (option)	26
4.4.3	Using RS232 and RS485 Interfaces Simultaneously	26
4.4.4	MODBUS RTU Support	27
4.5	Motor Selection	27
4.5.1	Motor Connection	27
4.5.2	Motor Over-Temperature Protection	27
4.5.3	Motor Set-up	28
4.6	Using a Custom Motor	29
4.6.1	Creating Custom Motor Parameters	29
4.6.2	Autophasing	30
4.6.3	Custom Motor Data Entry	30



Contents

5	Parameters	35
5.1	Parameter Storage and EPM Operation	35
5.1.1	Parameter Storage	35
5.1.2	EPM Operation	35
5.1.3	EPM Fault	36
5.2	Motor Group	36
5.3	Parameters	36
5.3.1	Drive Operating Modes	36
5.3.2	Drive PWM Frequency	37
5.3.3	Current Limit	37
5.3.4	Peak Current Limit (8kHz and 16 kHz)	37
5.3.5	Analog Input Scale (Current)	37
5.3.6	Analog Input Scale (Velocity)	37
5.3.7	ACCEL/DECEL Limits (Velocity Mode Only)	38
5.3.8	Reference	38
5.3.9	Step Input Type (Position Mode Only)	38
5.3.10	Fault Reset Option	38
5.3.11	Motor Temperature Sensor	38
5.3.12	Motor PTC Cut-off Resistance	38
5.3.13	Second Encoder	38
5.3.14	Regen Duty Cycle	39
5.3.15	Encoder Repeat Source	39
5.3.16	System to Master Ratio	39
5.3.17	Second to Prime Encoder Ratio	39
5.3.18	Autoboot	39
5.3.19	Group ID	40
5.3.20	Enable Switch Function	40
5.3.21	User Units	40
5.4	Communication	40
5.4.1	IP Setup	40
5.4.2	RS-485 Configuration	40
5.4.3	Modbus Baud Rate	40
5.4.4	Modbus Reply Delay	40
5.5	Analog I/O	40
5.5.1	Analog Output	40
5.5.2	Analog Output Current Scale (Volt/amps)	41
5.5.3	Analog Output Velocity Scale (mV/RPM)	41
5.5.4	Analog Input Dead Band	41
5.5.5	Analog Input Offset Parameter	41
5.5.6	Adjust Analog Voltage Offset	41
5.6	Digital I/O	41
5.6.1	Digital Input De-bounce Time	41
5.6.2	Hard Limit Switch Action	42
5.7	Velocity Limits	42
5.7.1	Zero Speed	42
5.7.2	Speed Window	42
5.7.3	At Speed	42
5.8	Position Limits	42
5.8.1	Position Error	42
5.8.2	Max Error Time	42
5.8.3	Second Encoder Position Error	42
5.8.4	Second Encoder Max Error Time	42



5.9	Compensation	43
5.9.1	Velocity P-gain (Proportional)	43
5.9.2	Velocity I-gain (Integral)	43
5.9.3	Position P-gain (Proportional)	43
5.9.4	Position I-gain (Integral)	43
5.9.5	Position D-gain (Differential)	43
5.9.6	Position I-limit	43
5.9.7	Gain Scaling Window	44
5.10	Tools	44
5.10.1	Oscilloscope Tool	44
5.10.2	Run Panels	44
5.11	Faults Group	44
6	Operation	45
6.1	Minimum Connections	45
6.2	Configuration of the PositionServo	45
6.3	Position Mode Operation (gearing)	47
6.4	Dual-loop Feedback	47
6.5	Enabling the PositionServo	48
6.6	Drive Tuning	48
6.6.1	Tuning the Drive in Velocity Mode	49
6.6.2	Tuning the Drive in Position Mode	54
7	Quick Start Reference	60
7.1	Quick Start - External Torque Mode	60
7.2	Quick Start - External Velocity Mode	61
7.3	Quick Start - External Positioning Mode	63
8	Diagnostics	65
8.1	Display	65
8.2	LEDs	66
8.3	Faults	66
8.3.1	Fault Codes	66
8.3.2	Fault Event	68
8.3.3	Fault Reset	68
8.4	Troubleshooting	68

Safety Information





All safety information given in these Operating Instruction has a similar layout:



Signal Word! (Characteristics the severity of the danger)

Note (describes the danger and informs on how to proceed)

Pictographs used in these instructions:

Icon		Signal Words	
	Warning of hazardous electrical voltage	DANGER!	Warns of impending danger . Consequences if disregarded: Death or severe injuries.
	Warning of a general danger	WARNING!	Warns of potential, very hazardous situations . Consequences if disregarded: Death or severe injuries.
	Warning of damage to equipment	STOP!	Warns of potential damage to material and equipment . Consequences if disregarded: Damage to the controller/drive or its environment.
	Information	NOTE	Designates a general, useful note. If you observe it, handling the controller/drive system is made easier.



1 Introduction

The PositionServo line of advanced general purpose servo drives utilizes the latest technology in power semiconductors and packaging. The PositionServo uses Field Oriented control to enable high quality motion.

The PositionServo Model 940 is available in four mains (input power) configurations:

1. **400/480V** (nominal) three phase input. An external input mains (line) filter is available. These drives have the suffix "T4N". Actual voltage can range from 320 - 528 VAC.
2. **120/240V** (nominal) Single Phase input with integrated input mains (line) filter, Actual input voltage can range from 80VAC to 264VAC. The maximum output voltage is approximately equal to the input voltage. These drives have the suffix "S2F".
3. **120V or 240V** (nominal) Single or Three Phase input. Actual input voltage can range from 80VAC to 264VAC. The maximum output voltage is approximately equal to the input voltage. An external input mains (line) filter is available. These drives have the suffix "Y2N".
4. **120V or 240V** (nominal) single phase input. When wired for **Doubler mode** (L1-N), the input is for 120V nominal only and can range from 45VAC to 132 VAC and the maximum output voltage is double the input voltage. When wired to terminals L1-L2/N, the input can range from 80 VAC to 264 VAC and the maximum output voltage is equal to the input voltage. These drives have the suffix "S1N".

The PositionServo 940 will accept feedback from an incremental encoder (that includes Hall channel information) or from a resolver. It accepts commands from a variety of sources, including analog voltage, RS485 interface (PPP and Modbus RTU), Ethernet interface, CANopen interface, digital pulse train, and master encoder reference. The control will operate in current (torque), velocity, or position (step and direction / master encoder) modes.

The 940 utilizes a software package called MotionView. MotionView provides a window into the drive allowing the user to check and set parameter. It has a real-time oscilloscope tool, for analyses and optimum tuning, as well as a User Program. This User Program can be utilized to command motion and handle the drives I/O. The MotionView programming language is designed to be very user friendly and easy to implement.

The EPM (Electronic Programming Module) stores all drive setup and tuning information. This module can be removed from the drive and reinstalled into another PositionServo 940, making field replacement of the PositionServo 940 extremely easy.

The PositionServo 940 controls supports Point-to-Point (PPP) and Modbus RTU over RS485, Ethernet TCP/IP and CANopen (DS301, DS402) communication protocols.

The PositionServo 940 supports incremental quadrature encoder or resolver feedback devices. A second encoder can also be supported during position and velocity modes.

1.1 About These Instructions

- These Operating Instructions are provided to assist the user in connecting and commissioning the PositionServo drive equipped with an RS232 interface in P2. Observe all safety instructions contained in this document.
- All persons working on and with the controller must have the Operating Instructions available and must observe the information and notes relevant for their work.
- Read these Operating Instructions in their entirety before operating the drive.



Introduction

1.2 Scope of Supply

Scope of Supply	Important
<ul style="list-style-type: none"> • 1 Model 940 Servo type E94P... • 1 Users Manual (English) • 1 MotionView CD ROM including <ul style="list-style-type: none"> - configuration software - documentation (Adobe Acrobat) 	<p>After reception of the delivery, check immediately whether the scope of supply matches the accompanying papers. Lenze does not accept any liability for deficiencies claimed subsequently.</p> <p>Claim</p> <ul style="list-style-type: none"> • visible transport damage immediately to the forwarder • visible deficiencies / incompleteness immediately to your Lenze representative.

1.3 Legal Regulations

Identification	Nameplate	CE Identification	Manufacturer
	Lenze controllers are unambiguously designated by the contents of the nameplate	In compliance with the EC Low-Voltage Directive	AC Technology Corp. member of the Lenze Group 630 Douglas Street Uxbridge, MA 01569 USA
Application as directed	<p>E94P... servo controller</p> <ul style="list-style-type: none"> • must only be operated under the conditions prescribed in these Instructions. • are components <ul style="list-style-type: none"> - for closed loop control of variable speed and torque applications with PM synchronous motors - for installation in a machine. - for assembly with other components to form a machine. • are electric units for the installation into control cabinets or similar enclosed operating housing. • comply with the requirements of the Low-Voltage Directive. • are not machines for the purpose of the Machinery Directive. • are not to be used as domestic appliances, but only for industrial purposes. <p>Drive systems with E94P... servo inverters</p> <ul style="list-style-type: none"> • comply with the EMC Directive if they are installed according to the guidelines of CE-typical drive systems. • can be used <ul style="list-style-type: none"> - for operation on public and non-public mains - for operation in industrial premises and residential areas. • The user is responsible for the compliance of his application with the EC directives. <p>Any other use shall be deemed as inappropriate!</p>		
Liability	<ul style="list-style-type: none"> • The information, data, and notes in these instructions met the state of the art at the time of publication. Claims on modifications referring to controllers which have already been supplied cannot be derived from the information, illustrations, and descriptions. • The specifications, processes and circuitry described in these instructions are for guidance only and must be adapted to your own specific application. Lenze does not take responsibility for the suitability of the process and circuit proposals. • The specifications in these Instructions describe the product features without guaranteeing them. • Lenze does not accept any liability for damage and operating interference caused by: <ul style="list-style-type: none"> - Disregarding the operating instructions - Unauthorized modifications to the controller - Operating errors - Improper working on and with the controller 		
Warranty	<ul style="list-style-type: none"> • Warranty conditions: see Sales and Delivery Conditions of Lenze Drive Systems GmbH. • Warranty claims must be made to Lenze immediately after detecting the deficiency or fault. • The warranty is void in all cases where liability claims cannot be made. 		
Disposal	Material	Recycle	Dispose
	Metal	•	-
	Plastic	•	-
	Assembled PCB's	-	•



2 Technical Data

2.1 Electrical Characteristics

Single-Phase Models					
Type	Mains Voltage ⁽¹⁾	1~ Mains Current (doubler)	1~ Mains Current (Std.)	Rated Output Current ⁽⁴⁾	Peak Output Current ⁽⁵⁾
E94P020S1N	120V ⁽²⁾ or 240V ⁽³⁾	9.7	5.0	2.0	6
E94P040S1N		15	8.6	4.0	12
E94P020S2F	120 / 240V ⁽³⁾ (80 V -0%...264 V +0%)	--	5.0	2.0	6
E94P040S2F		--	8.6	4.0	12
E94P080S2F		--	15.0	8.0	24
E94P100S2F		--	18.8	10.0	30
Single/Three-Phase Models					
Type	Mains Voltage ⁽¹⁾	1~ Mains Current	3~ Mains Current	Rated Output Current ⁽⁴⁾	Peak Output Current ⁽⁵⁾
E94P020Y2N	120 / 240V ⁽³⁾ 1~ or 3~ (80 V -0%...264 V +0%)	5.0	3.0	2.0	6
E94P040Y2N		8.6	5.0	4.0	12
E94P080Y2N		15.0	8.7	8.0	24
E94P100Y2N		18.8	10.9	10.0	30
E94P120Y2N		24.0	13.9	12.0	36
E94P020T4N	400 / 480V 3~ (320 V -0%...528 V +0%)	--	2.7	2.0	6
E94P040T4N		--	5.5	4.0	12
E94P050T4N		--	6.9	5.0	15
E94P060T4N		--	7.9	6.0	18

(1) Mains voltage for operation on 50/60 Hz AC supplies (48 Hz -0% ... 62Hz +0%).

(2) Connection of 120VAC (45 V ... 132 V) to input power terminals L1 and N on these models doubles the voltage on motor output terminals U-V-W for use with 230VAC motors.

(3) Connection of 240VAC or 120VAC to input power terminals L1 and L2 on these models delivers an equal voltage as maximum to motor output terminals U-V-W allowing operation with either 120VAC or 230VAC motors.

(4) Drive rated at 8kHz Carrier Frequency. Derate Continuous current by 17% at 16kHz.

(5) Peak RMS current allowed for up to 2 seconds. Peak current rated at 8kHz. Derate by 17% at 16kHz.

Applies to all models:

Acceleration Time Range (Zero to Max Speed)	0.1 ... 5x10 ⁶ RPM/sec
Deceleration Time Range (Max Speed to Zero)	0.1 ... 5x10 ⁶ RPM/sec
Speed Regulation (typical)	± 1 RPM
Input Impedance (AIN+ to COM and AIN+ to AIN-)	47k Ω
Power Device Carrier Frequency (sinusoidal commutation)	8,16 kHz
Encoder power supply (max)	+5 VDC @ 300 mA
Maximum encoder feedback frequency	2.1 MHz (per channel)

2.2 Environment

Vibration	2 g (10 - 2000 Hz)
Ambient Operating Temperature Range	0 to 40°C
Ambient Storage Temperature Range	-10 to 70°C
Temperature Drift	0.1% per °C rise
Humidity	5 - 90% non-condensing
Altitude	1500 m/5000 ft [derate by 1% per 300m (1000 ft) above 1500m (5000 ft)]



Technical Data

2.3 Operating Modes

Torque	Reference	± 10 VDC 16-bit; scalable
	Torque Range	100:1
	Current-Loop Bandwidth	Up to 1.5 kHz*
Velocity	Reference	± 10 VDC or 0...10 VDC; scalable
	Regulation	± 1 RPM
	Velocity-Loop Bandwidth	Up to 200 Hz*
	Speed Range	5000:1 with 5000 ppr encoder
Position	Reference	0...2 MHz Step and Direction or 2 channels quadrature input; scalable
	Minimum Pulse Width	500 nanoseconds
	Loop Bandwidth	Up to 200 Hz*
	Accuracy	±1 encoder count

* = motor and application dependent

2.4 Connections and I/O

Mains Power	4-pin removable terminal block	(P1)
RS232 Port	Standard 9-pin D-shell (DCE) Connector	(P2)
I/O Connector	Standard 50-pin SCSI.	(P3)
- Buffered Encoder Output	A, B & Z channels with compliments (5V @ 20mA)	(P3)
- Digital Inputs	12 programmable, 1 dedicated (5-24V)	(P3)
- Digital Outputs	4 programmable, 1 dedicated(5-24V @ 15mA)	(P3)
- Analog Input	1 differential; ±10 VDC (16 bit)	(P3)
- Analog Output	1 single ended; ±10 VDC (10-bit)	(P3)
Encoder Feedback (primary)	Standard 15-pin D-shell	(P4)
24VDC Power "Keep Alive"	2-pin removable terminal block	(P5)
Regen and Bus Power	5-pin removable terminal block	(P6)
Motor Power	6-pin pin removable terminal block	(P7)
Resolver feedback (option bay)	Option module with standard 9-pin D-shell	(P11)
Encoder Feedback (option bay)	Option module with standard 9-pin D-shell	(P12)
Comm Option Bay	Optional Comm Modules (CAN, Ethernet, RS485)	(P21)
Windows® Software:	MotionView (Windows 98, NT, 2000, XP)	

2.5 Digital I/O Ratings

	Scan Times	Linearity	Temperature Drift	Offset	Current	Input Impedance	Voltage Range
Units	µs	%	%	%	mA	Ohm	VDC
Digital Inputs⁽¹⁾	512				Depend on load	2.2 k	5-24
Digital Outputs	512				15 max	N/A	30 max
Analog Inputs	512	± 0.013	0.1% per °C rise	± 0 adjustable	Depend on load	47 k	± 18
Analog Outputs	512		0.1% per °C rise	± 0 adjustable	10 max	N/A	± 10

(1) Inputs do not have scan time. Their values are read directly by indexer program statement.

Notes for Power Ratings Table in section 2.6:

- At 240 VAC line input for drives with suffixes "S1N", "S2F", "Y2N". At 480 VAC line input for drives with suffixes "T4N".
 - The output power is calculated from the formula: output kVA = $\{(\sqrt{3}) \times U_{L-L} \times I_{rated}\} / 1000$
 - The actual output power (kW) depends on the motor in use due to variations in motor rated voltage, rated speed and power factor, as well as actual max operating speed and desired overload capacity.
 - Typical max continuous power (kW) for PM servo motors runs 50-70% of the kVA ratings listed.
- At 16 kHz, de-rate continuous current by 17%
- Leakage Current is typically >3.5mA. Contact factory for applications requiring <3.5mA

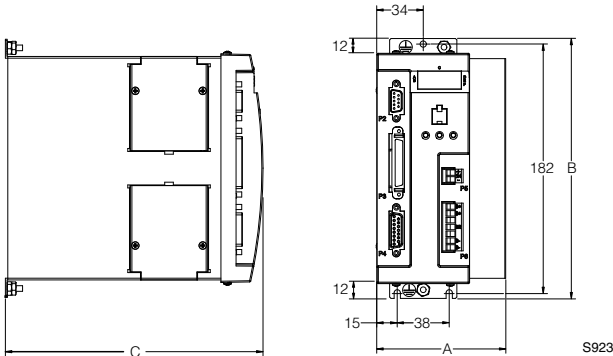


2.6 Power Ratings

Type ⁽¹⁾	Output kVA at Rated Output Current (8kHz) ⁽¹⁾	Leakage Current ⁽⁹⁾	Power Loss at Rated Output Current (8kHz)	Power Loss at Rated Output Current (16 kHz) ⁽²⁾
Units	kVA		Watts	Watts
E94P020S1N	0.8	Typical: >3.5mA*	19	21
E94P040S1N	1.7		29	30
E94P020S2F	0.8		19	21
E94P040S2F	1.7	Typical: >3.5mA*	29	30
E94P080S2F	3.3		61	63
E94P100S2F	4.2		80	85
E94P020Y2N	0.8	Typical: >3.5mA*	19	21
E94P040Y2N	1.7		29	30
E94P080Y2N	3.3		61	63
E94P100Y2N	4.2		80	85
E94P120Y2N	5.0		114	129
E94P020T4N	1.7	Typical: >3.5mA*	31	41
E94P040T4N	3.3		50	73
E94P050T4N	4.2		70	90
E94P060T4N	5.0		93	122

Refer to Notes that precede this table.

2.7 Dimensions

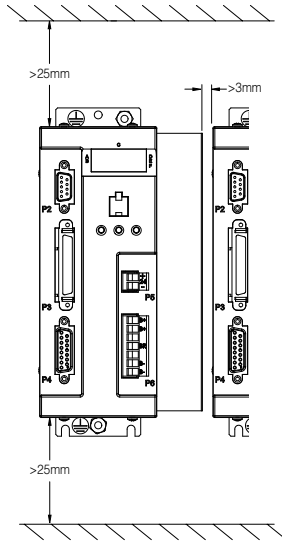


Type	A (mm)	B (mm)	C (mm)	Weight (kg)
E94P020S1N	68	190	190	1.1
E94P040S1N	69	190	190	1.2
E94P020S2F	68	190	235	1.3
E94P040S2F	69	190	235	1.5
E94P080S2F	87	190	235	1.9
E94P100S2F	102	190	235	2.2
E94P020Y2N	68	190	190	1.3
E94P040Y2N	69	190	190	1.5
E94P080Y2N	95	190	190	1.9
E94P100Y2N	114	190	190	2.2
E94P120Y2N	68	190	235	1.5
E94P020T4N	68	190	190	1.5
E94P040T4N	95	190	190	1.9
E94P050T4N	114	190	190	2.2
E94P060T4N	68	190	235	1.4



Technical Data

2.8 Clearance for Cooling Air Circulation





3 Installation

Perform the minimum system connection. Please refer to section 6.1 for minimum connection requirements. Observe the rules and warnings below carefully:



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.



STOP!

- The PositionServo 940 must be mounted vertically for safe operation and enough cooling air circulation.
- Printed circuit board components are sensitive to electrostatic fields. Avoid contact with the printed circuit board directly. Hold the PositionServo 940 by its case only.
- Protect the drive from dirt, filings, airborne particles, moisture, and accidental contact. Provide sufficient room for access to the terminal block.
- Mount the drive away from any and all heat sources. Operate within the specified ambient operating temperature range. Additional cooling with an external fan may be recommended in certain applications.
- Avoid excessive vibration to prevent intermittent connections
- DO NOT connect incoming (mains) power to the output motor terminals (U, V, W)! Severe damage to the drive will result.
- Do not disconnect any of the motor leads from the PositionServo 940 drive unless (mains) power is removed. Opening any one motor lead may cause failure.
- Control Terminals provide basic isolation (insulation per EN61800-5-1). Protection against contact can only be ensured by additional measures, e.g., supplemental insulation.



WARNING!

For compliance with EN61800-5-1, the following warning applies.

This product can cause a d.c. current in the protective earthing conductor. Where a residual current-operated protective (RCD) or monitoring (RCM) device is used for protection in case of direct or indirect contact, only an RCD or RCM of Type B is allowed on the supply side of this product.



Installation



UL INSTALLATION INFORMATION

- Suitable for use on a circuit capable of delivering not more than 200,000 rms symmetrical amperes, at the maximum voltage rating marked on the drive.
- Use Class 1 wiring with minimum of 75°C copper wire only.
- Shall be installed in a pollution degree 2 macro-environment.

3.1 Wiring



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing the drive. Capacitors retain charge after power is removed.



WARNING!

Leakage current may exceed 3.5mA AC. Minimum size of the protective earth conductor shall comply with local safety regulations for high leakage current equipment.



STOP!

Under no circumstances should power and control wiring be bundled together. Induced voltage can cause unpredictable behavior in any electronic device, including motor controls.

Refer to section 4.1.1 for power wiring specifications.

3.2 Shielding and Grounding

3.2.1 General Guidelines

Lenze recommends the use of single-point grounding (SPG) for panel-mounted controls. Serial grounding (a “daisy chain”) is not recommended. The SPG for all enclosures must be tied to earth ground at the same point. The system ground and equipment grounds for all panel-mounted enclosures must be individually connected to the SPG for that panel using 14 AWG (2.5 mm²) or larger wire.

In order to minimize EMI, the chassis must be grounded to the mounting. Use 14 AWG (2.5 mm²) or larger wire to join the enclosure to earth ground. A lock washer must be installed between the enclosure and ground terminal. To ensure maximum contact between the terminal and enclosure, remove paint in a minimum radius of 0.25 in (6 mm) around the screw hole of the enclosure.

Lenze recommends the use of the special PositionServo 940 drive cables provided by Lenze. If you specify cables other than those provided by Lenze, please make certain all cables are shielded and properly grounded.

It may be necessary to earth ground the shielded cable. Ground the shield at both the drive end and at the motor end.

If the PositionServo 940 drive continues to pick up noise after grounding the shield, it may be necessary to add an AC line filtering device and/or an output filter (between drive and servo motor).



EMC

Compliance with EN 61800-3/A11

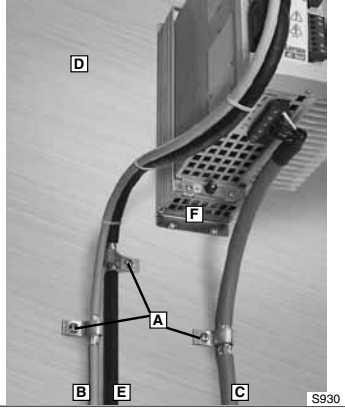
This is a product of the restricted sales distribution class according to IEC 61800-3. In a domestic environment this product may cause radio interference in which the user may be required to take adequate measures

Noise emission

Drive Models ending in the suffix "2F" are in compliance with class A limits according to EN 55011 if installed in a control cabinet and the motor cable length does not exceed 10m. Models ending in "N" will require an appropriate line filter.

- A** Screen clamps
- B** Control cable
- C** Low-capacitance motor cable
(core/core < 75 pF/m, core/screen < 150 pF/m)
- D** Earth grounded conductive mounting plate
- E** Encoder Feedback Cable
- F** Footprint or Sidemount Filter (optional)

Installation according to EMC Requirements



3.2.2 EMI Protection

Electromagnetic interference (EMI) is an important concern for users of digital servo control systems. EMI will cause control systems to behave in unexpected and sometimes dangerous ways. Therefore, reducing EMI is of primary concern not only for servo control manufacturers such as Lenze, but the user as well. Proper shielding, grounding and installation practices are critical to EMI reduction.

3.2.3 Enclosure

The panel in which the PositionServo 940 is mounted must be made of metal, and must be grounded using the SPG method outlined in section 3.2.1.

Proper wire routing inside the panel is critical; power and logic leads must be routed in different avenues inside the panel.

You must ensure that the panel contains sufficient clearance around the drive. Refer to Section 2.6 suggested cooling air clearance.

3.3 Line Filtering

In addition to EMI/RFI safeguards inherent in the PositionServo 940 design, external filtering may be required. High frequency energy can be coupled between the circuits via radiation or conduction. The AC power wiring is one of the most important paths for both types of coupling mechanisms. In order to comply with IEC61800-3, an appropriate filter must be installed within 20cm of the drive power inputs.

Line filters should be placed inside the shielded panel. Connect the filter to the incoming power lines immediately after the safety mains and before any critical control components. Wire the AC line filter as close as possible to the PositionServo 940 drive.



Installation



NOTE

The ground connection from the filter must be wired to solid earth ground, not machine ground.

If the end-user is using a CE-approved motor, the AC filter combined with the recommended motor and encoder cables, is all that is necessary to meet the EMC directives listed herein. The end user must use the compatible filter to comply with CE specifications. The OEM may choose to provide alternative filtering that encompasses the PositionServo 940 drive and other electronics within the same panel. The OEM has this liberty because CE requirements are for the total system.

3.4 Heat Sinking

The PositionServo 940 drive contains sufficient heat sinking within the specified ambient operating temperature in their basic configuration. There is no need for additional heat sinking. However, you must ensure that there is sufficient clearance for proper air circulation. As a minimum, you must allow an air gap of 25 mm above and below the drive.

3.5 Line (Mains) Fusing

External line fuses must be installed on all PositionServo drives. Connect the external line fuse in series with the AC line voltage input. Use fast-acting fuses rated for 250 VAC or 600 VAC (depending on model), and approximately 200% of the maximum RMS phase current.

3.6 Fuse Recommendations

Type ⁽¹⁾	AC Line Input Fuse (1ø/3ø)	Miniature Circuit Breaker ⁽¹⁾ (1ø/3ø)	AC Line Input Fuse or Breaker ^{(2) (3)} (N. America)	DC Bus Input Fuse ⁽⁴⁾
Amp Ratings				
E94P020S1N	M20/M10	C20/C10	20/10	10
E94P040S1N	M32/M20	C32/C20	30/20	20
E94P020S2F	M20	C20	20	15
E94P040S2F	M20	C20	20	20
E94P080S2F	M32	C32	32	40
E94P100S2F	M40	C40	40	45
E94P020Y2N	M20/M16	C20/C16	20/15	15
E94P040Y2N	M20/M16	C20/C16	20/15	20
E94P080Y2N	M32/M20	C32/C20	30/20	40
E94P100Y2N	M40/M25	C40/C25	40/25	45
E94P120Y2N	M50/M32	C50/C32	50/30	55
E94P020T4N	M10	C10	10	10
E94P040T4N	M10	C10	10	20
E94P050T4N	M16	C16	15	25
E94P060T4N	M20	C20	20	30

(1) Installations with high fault current due to large supply mains may require a type D circuit breaker.

(2) UL Class CC or T fast-acting current-limiting type fuses, 200,000 AIC, preferred. Bussman KTK-R, JJJ, JJS or equivalent.

(3) Thermal-magnetic type breakers preferred.

(4) DC-rated fuses, rated for the applied voltage. Examples Bussman KTM or JJJ as appropriate.



4 Interface

The standard PositionServo 940 drive contains seven connectors: four quick-connect terminal blocks, one SCSI connector and two subminiature type “D” connectors. These connectors provide communications from a PLC or host controller, power to the drive, and feedback from the motor. Prefabricated cable assemblies may be purchased from Lenze to facilitate wiring the drive, motor and host computer. Contact your Lenze Sales Representative for assistance.

As this manual makes reference to specific pins on specific connectors, we will use the convention PX.Y where X is the connector number and Y is the pin number.

4.1 External Connectors

4.1.1 P1 & P7 - Input Power and Output Power Connections

P1 is a 3 or 4-pin quick-connect terminal block used for input (mains) power. P7 is a 6-pin quick-connect terminal block used for output power to the motor. P7 also has a thermistor (PTC) input for motor over-temperature protection. The tables below identify connector pin assignments.



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.



STOP!

DO NOT connect incoming power to the output motor terminals (U, V, W)! Severe damage to the PositionServo will result.

All conductors must be enclosed in one shield and jacket around them. The shield on the drive end of the motor power cable should be terminated to the conductive machine panel using screen clamps as shown in section 3.2. The other end should be properly terminated at the motor shield. Feedback cable shields should be terminated in a like manner. Lenze recommends Lenze cables for both the motor power and feedback. These are available with appropriate connectors and in various lengths. Contact your Lenze representative for assistance.

Wire Size

Current (Arms)	Terminal Torque (lb-in)	Wire Size
$I \leq 8$	4.5	16 AWG (1.5mm ²) or 14 AWG (2.5mm ²)
$8 < I \leq 12$	4.5	14 AWG (2.5mm ²) or 12 AWG (4.0mm ²)
$12 < I \leq 15$	4.5	12 AWG (4.0mm ²)
$15 < I \leq 20$	5.0 - 7.0	10 AWG (6.0mm ²)
$20 < I \leq 24$	11.0 - 15.0	10 AWG (6.0mm ²)

P1 PIN ASSIGNMENTS (INPUT POWER)

Pin	Standard Models		Doubler Models	
	Name	Function	Name	Function
1	PE	Protective Earth (Ground)	PE	Protective Earth (Ground)
2	L1	AC Power in	N	AC Power Neutral (120V Doubler only)
3	L2	AC Power in	L1	AC Power in
4	L3	AC Power in (3- models only)	L2/N	AC Power in (non-doubler operation)



Interface

P7 PIN ASSIGNMENTS (OUTPUT POWER)

Pin	Terminal	Function
1	T1	Thermistor (PTC) Input
2	T2	Thermistor (PTC) Input
3	U	Motor Power Out
4	V	Motor Power Out
5	W	Motor Power Out
6	PE	Protective Earth (Chassis Ground)

4.1.2 P2 - Serial Communications Port

P2 is a 9-pin D-sub connector that is used to communicate with a host computer via standard RS-232 interface using a proprietary Point-to-Point Protocol (PPP). This port is present on all Model 94 and 940 RS-232-based drives. All levels must be RS-232C compliant.

P2 PIN ASSIGNMENTS (COMMUNICATIONS)

Pin	Name	Function	RS-232 Connector
1	Reserved		
2	TX	RS-232 (transmit)	
3	RX	RS-232 (receive)	
4	Reserved		
5	GND	Common	
6	Reserved		
7	Reserved		
8	Reserved		
9	Reserved		



STOP!

Do not make any connection to Reserved pins!



NOTE

If you purchase serial cables from a third party, you must use a pass-through cable, not Null-Modem (not crossover)

4.1.3 P3 - Controller Interface

P3 is a 50-pin SCSI connector for interfacing to the front-end of the controllers. It is strongly recommended that you use OEM cables to aid in satisfying CE requirements. Contact your Lenze representative for assistance.

P3 PIN ASSIGNMENTS (CONTROLLER INTERFACE)

Pin	Name	Function
1	MA+	Master Encoder A+ / Step+ input ⁽²⁾
2	MA-	Master Encoder A- / Step- input ⁽²⁾
3	MB+	Master Encoder B+ / Direction+ input ⁽²⁾
4	MB-	Master Encoder B- / Direction- input ⁽²⁾
5	GND	Drive Logic Common
6	5+	+5V output (max 100mA)
7	BA+	Buffered Encoder Output: Channel A+ ⁽¹⁾
8	BA-	Buffered Encoder Output: Channel A- ⁽¹⁾
9	BB+	Buffered Encoder Output: Channel B+ ⁽¹⁾



Pin	Name	Function
10	BB-	Buffered Encoder Output: Channel B- ⁽¹⁾
11	BZ+	Buffered Encoder Output: Channel Z+ ⁽¹⁾
12	BZ-	Buffered Encoder Output: Channel Z- ⁽¹⁾
13-19		Empty
20	AIN2+	Positive (+) of Analog signal input
21	AIN2-	Negative (-) of Analog signal input
22	ACOM	Analog common
23	AO	Analog output
24	AIN1+	Positive (+) of Analog signal input
25	AIN1 -	Negative (-) of Analog signal input
26	IN_A_COM	Digital input group ACOM terminal ⁽³⁾
27	IN_A1	Digital input A1
28	IN_A2	Digital input A2
29	IN_A3	Digital input A3 ⁽³⁾
30	IN_A4	Digital input A4
31	IN_B_COM	Digital input group BCOM terminal
32	IN_B1	Digital input B1
33	IN_B2	Digital input B2
34	IN_B3	Digital input B3
35	IN_B4	Digital input B4
36	IN_C_COM	Digital input group CCOM terminal
37	IN_C1	Digital input C1
38	IN_C2	Digital input C2
39	IN_C3	Digital input C3
40	IN_C4	Digital input C4
41	RDY+	Ready output Collector
42	RDY-	Ready output Emitter
43	OUT1-C	Programmable output #1 Collector
44	OUT1-E	Programmable output #1 Emitter
45	OUT2-C	Programmable output #2 Collector
46	OUT2-E	Programmable output #2 Emitter
47	OUT3-C	Programmable output #3 Collector
48	OUT3-E	Programmable output #3 Emitter
49	OUT4-C	Programmable output #4 Collector
50	OUT4-E	Programmable output #4 Emitter

⁽¹⁾ See Note 1, Section 4.1.7 - Connector and Wiring Notes

⁽²⁾ See Note 2, Section 4.1.7 - Connector and Wiring Notes

⁽³⁾ See Note 3, Section 4.1.7 - Connector and Wiring Notes

4.1.4 P4 - Motor Feedback / Second Loop Encoder Input

P4 is a 15-pin DB connector that contains connections for Hall Effect sensors and incremental encoder feedback. Refer to the P4 pin assignments table for the connector pin assignments. Encoder inputs on P4 have 26LS32 or compatible differential receivers for increased noise immunity. Inputs have all necessary filtering and line balancing components so no external noise suppression networks are needed.

All conductors must be enclosed in one shield and jacket around them. Lenze recommends that each and every pair (for example, EA+ and EA-) be twisted. In order to satisfy CE requirements, use of an OEM cable is recommended. Contact your Lenze representative for assistance.



Interface

The PositionServo 940 buffers encoder feedback from P4 to P3. Encoder Feedback channel A on P4, for example, is Buffered Encoder Output channel A on P3. The Hall sensors from the motor must be wired to the 15-pin connector (P4).



STOP!

Use only +5 VDC encoders. Do not connect any other type of encoder to the PositionServo 940 reference voltage terminals. When using a front-end controller, it is critical that the +5 VDC supply on the front-end controller NOT be connected to the PositionServo 940's +5 VDC supply, as this will result in damage to the PositionServo 940.



NOTE

- The PositionServo 940 encoder inputs are designed to accept differentially driven hall signals. Single-ended or open-collector type hall signals are also acceptable by connecting "HA+", "HB+", "HC+" and leaving "HA-, HB-, HC-" inputs unconnected. You do not need to supply pull-up resistors for open-collector hall sensors. The necessary pull-up circuits are already provided.
- Encoder connections (A, B, and Z) must be full differential. PositionServo doesn't support single-ended or open-collector type outputs from the encoder.
- An encoder resolution of 2000 PPR (pre-quadrature) or higher is recommended.

Using P4 as second encoder input for dual-loop operation.

P4 can be used as a second loop encoder input in situations where the motor is equipped with a resolver as the primary feedback. If such a motor is used, the drive must have a resolver feedback option module installed. A second encoder can then be connected to the A and B lines of the P4 connector for dual loop operation. Refer to "Dual-loop Feedback Operation" for details (Section 6.4).

P4 PIN ASSIGNMENTS (ENCODER)

Pin	Name	Function
1	EA+	Encoder Channel A+ Input ⁽¹⁾
2	EA-	Encoder Channel A- Input ⁽¹⁾
3	EB+	Encoder Channel B+ Input ⁽¹⁾
4	EB-	Encoder Channel B- Input ⁽¹⁾
5	EZ+	Encoder Channel Z+ Input ⁽¹⁾
6	EZ-	Encoder Channel Z- Input ⁽¹⁾
7	GND	Drive Logic Common/Encoder Ground
8	SHLD	Shield
9	PWR	Encoder supply (+5VDC)
10	HA-	Hall Sensor A- Input
11	HA+	Hall Sensor A+ Input
12	HB+	Hall Sensor B+ Input
13	HC+	Hall Sensor C+ Input
14	HB-	Hall Sensor B- Input
15	HC-	Hall Sensor C- Input

⁽¹⁾ See Note 1, Section 4.1.7 - Connector and Wiring Notes



4.1.5 P5 - 24 VDC Back-up Power Input

P5 is a 2-pin quick-connect terminal block that can be used with an external 24 VDC (500mA) power supply to provide “Keep Alive” capability: during a power loss, the logic and communications will remain active. Applied voltage must be greater than 20VDC.

P5 PIN ASSIGNMENTS (BACK-UP POWER)

Pin	Name	Function
1	+24 VDC	Positive 24 VDC Input
2	Return	24V power supply return



WARNING!

Hazard of unintended operation! The “Keep Alive” circuit will restart the motor upon restoration of mains power when the enable input remains asserted. If this action is not desired, then the enable input must be removed prior to re-application of input power.

4.1.6 P6 - Braking Resistor and DC Bus

P6 is a 5-pin quick-connect terminal block that can be used with an external braking resistor (the PositionServo 940 has the regen circuitry built-in). The Brake Resistor connects between the Positive DC Bus (either P6.1 or 2) and P6.3.

P6 TERMINAL ASSIGNMENTS (BRAKE RESISTOR AND DC BUS)

Pin	Terminal	Function
1	B+	Positive DC Bus / Brake Resistor
2	B+	
3	BR	Brake Resistor
4	B-	Negative DC Bus
5	B-	

4.1.7 Connector and Wiring Notes

Note 1 - Encoder Inputs

Each of the encoder output pins on P3 is a buffered pass-through of the corresponding input signal on P4. This can be either from a motor mounted (primary feedback) encoder or from an auxiliary encoder when a resolver is the primary feedback device on the motor.

Via software, these pins can be re-programmed to be a buffered pass through of the signals from a feedback option card. This can be either the second encoder option module (E94ZAENC1) or an encoder emulation of the resolver connected to the resolver option module (E94ZARSV1, E94ZARSV2 or E94ZARSV3).

Note 2 - Encoder Outputs

An external pulse train signal (“step”) supplied by an external device, such as a PLC or stepper indexer, can control the speed and position. of the servomotor. The speed of the motor is controlled by the frequency of the “step” signal, while the number of pulses that are supplied to the PositionServo 940 determines the position of the servomotor. “DIR” input controls direction of the motion.

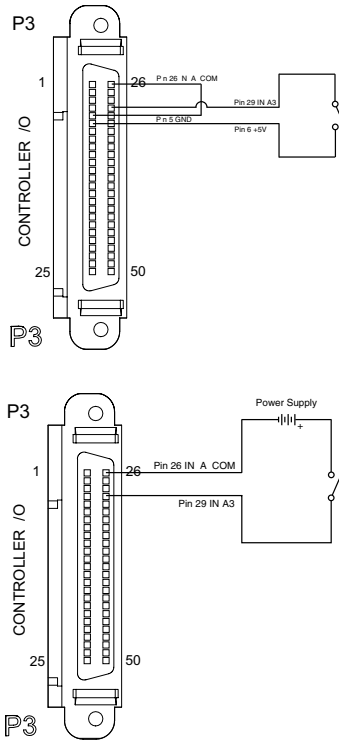


Interface

Note 3 - Digital Input A3

The ENABLE pin (IN_A3, P3.29) must be wired through a switch or an output on a front-end controller to digital input common (IN_ACOM, P3.26). If a controller is present, it should supervise the PositionServo 940's enable function. The ENABLE circuit will accept 5-24V control voltage.

Wiring the ENABLE Switch:



4.1.8 P11 - Resolver Interface Module (option)

PositionServo drives can operate motors equipped with resolvers. Resolver connections are made to a 9 pin D-shell female connector (P11) on the resolver option module E94ZARSV2 (scalable) or E94ZARSV3 (standard). When the motor profile is loaded from the motor database or from a custom motor file, the drive will select the primary feedback source based on the motor data entry.

When using a Lenze motor with resolver feedback and a Lenze resolver cable, the pins are already configured for operation. If a non-Lenze motor is used, the resolver connections are made as follows:



P11 PIN ASSIGNMENTS (Resolver Feedback)

Pin	Name	Function
1	Ref +	Resolver reference connection
2	Ref -	
3	N/C	No Connection
4	Cos+	Resolver Cosine connections
5	Cos-	
6	Sin+	Resolver Sine connections
7	Sin-	
8	PTC+	Thermal sensor
9	PTC-	



STOP!

Use only 10 V (peak to peak) or less resolvers. Use of higher voltage resolvers may result in feedback failure and damage to the resolver option module.

4.1.9 P12 - Second Encoder Interface Module (option)

PositionServo drives can support a second incremental encoder interface for dual-loop systems. Depending on the motor's primary feedback type (encoder or resolver) a second encoder can be connected as follows:

- If the primary motor feedback is an encoder (connected to P4), the second encoder interfaces through the encoder option module (E94ZAENC1) at P12 on Option Bay 2.
- If the motor primary feedback is a resolver connected to the resolver option module (E94ZARSV1) at P11 on Option Bay 2, the second encoder connects to the P4 connector on the drive. In this case, the hall inputs on P4 are not used.

The 2nd Encoder Option Module includes a 9 pin D-shell male connector. When using a Lenze motor with encoder feedback and a Lenze encoder cable, the pins are already configured for operation. If a non-Lenze motor is used, the encoder connections are made as follows:

P12 PIN ASSIGNMENTS (Second Encoder Feedback)

Pin	Name	Function
1	E2B+	Second Encoder Channel B+ Input
2	E2A-	Second Encoder Channel A- Input
3	E2A+	Second Encoder Channel A+ Input
4	+5v	Supply voltage for Second Encoder
5	COM	Supply common
6	E2Z-	Second Encoder Channel Z- Input
7	E2Z+	Second Encoder Channel Z+ Input
8	N/C	No Connection
9	E2B-	Second Encoder Channel B- Input

The second encoder needs to be enabled using MotionView software. Refer to "Dual-loop Feedback" (Section 6.4) for details.



STOP!

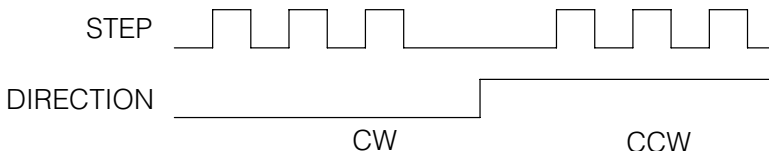
Use only +5 VDC encoders. Do not connect any other type of encoder to the option module otherwise damage to drive's circuitry may result.



4.2 Digital I/O Details

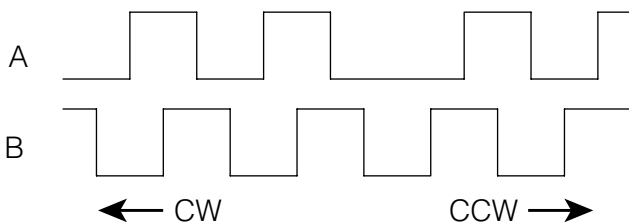
4.2.1 Step & Direction / Master Encoder Inputs (P3, pins 1-4)

You can connect a master encoder with quadrature outputs or a step and direction pair of signals to control position in step / direction operating mode (stepper motor emulation). These inputs are optically isolated from the rest of the drive circuits and from each other. Both inputs can operate from any voltage source in the range of 5 to 24 VDC and do not require additional series resistors for normal operation.



Timing characteristics for Step And Direction signals

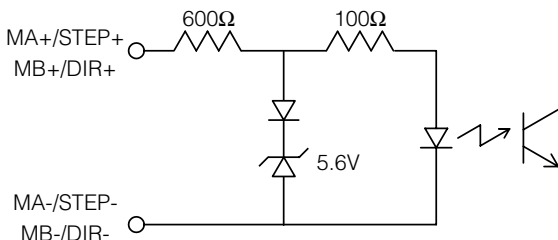
S904



Timing characteristics for Master Encoder signals

S905

Input type/ output compatibility	Insulated, compatible with Single-ended or differential outputs (5-24 VDC)
Max frequency (per input)	2 MHz
Min pulse width (negative or positive)	500nS
Input impedance	700 Ω (approx)



S906

Master encoder/step and direction input circuit

Differential signal inputs are preferred when using Step and Direction. Single ended inputs can be used but are not recommended. Sinking or sourcing outputs may also be connected to these inputs. The function of these inputs “Master Encoder” or “Step and Direction” is software selectable. Use MotionView setup program to choose desirable function.



4.2.2 Digital Outputs

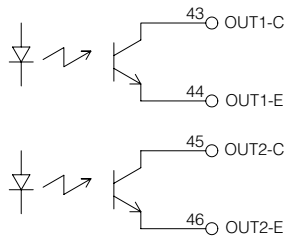
There are a total of five digital outputs (“OUT1” - “OUT4” and “RDY”) available on the PositionServo 940 drive. These outputs are accessible from the P3 connector. Outputs are open collector type that are fully isolated from the rest of the drive circuits. See the following figure for the electrical diagram. These outputs can be either used via the drives internal User Program or they can be configured as Special Purpose outputs. When used as Special Purpose, each output (OUT1-OUT4) can be assigned to one of the following functions:

- Not assigned
- Zero speed
- In-speed window
- Current limit
- Run-time fault
- Ready
- Brake (motor brake release)

Please note that if you assign an output as a Special Purpose Output then that output can not be utilized by the User Program. The “RDY” Output has a fixed function, “ENABLE”, which will become active when the drive is enabled and the output power transistors becomes energized.

Digital outputs electrical characteristics

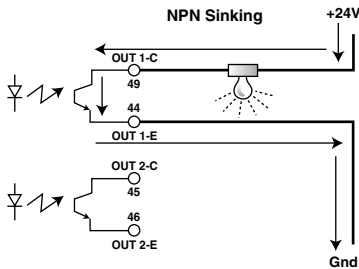
Circuit type	Isolated Open Collector
Digital outputs load capability	15mA
Digital outputs Collector-Emitter max voltage	30V



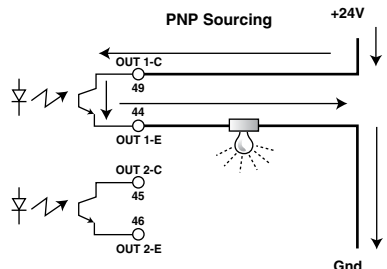
S907

Digital outputs circuit

The outputs on the drive can be wired as either sinking (NPN) or sourcing (PNP) as illustrated in wiring examples mb101 and mb102.



mb101



mb102



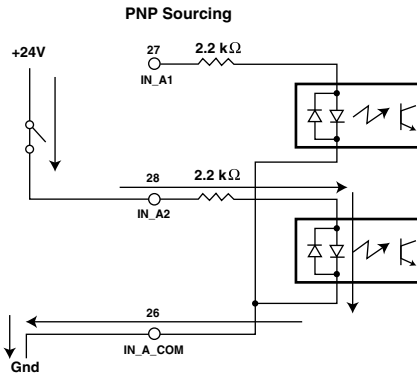
Interface

4.2.3 Digital Inputs

IN_Ax, IN_Bx, IN_Cx (P3.26-30, P3.31-35, P3.36-40)

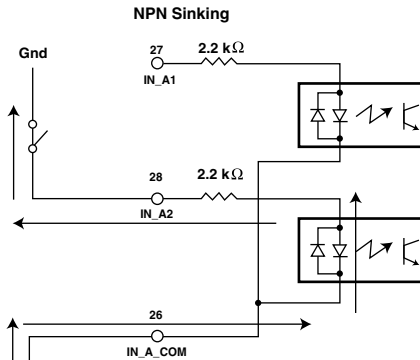
The PositionServo 940 Drive has 12 optically isolated inputs. These inputs are compatible with a 5 -24V voltage source. No additional series resistors are needed for circuit operation. The 12 inputs are segmented into three groups of 4, Inputs A1 - A4, Inputs B1 - B4, and Inputs C1 - C4. Each group, (A, B and C) have their own corresponding shared COM terminal, (ACOM, BCOM and CCOM). All inputs have separate software adjustable de-bounce time. Some of the inputs can be set up as Special Purpose Inputs. For example inputs A1 and A2 can be configured as limit inputs, input A3 can be set up as an Enable input and input C3 can be used as a registration input. Reference the 940 Programming Manual for more detail.

For the registration input (C3), the registration time is $3\mu\text{s}$ for an encoder and $7\mu\text{s}$ for a resolver.



Digital inputs circuit.

mb103



Digital inputs circuit.

mb104

Digital inputs circuit



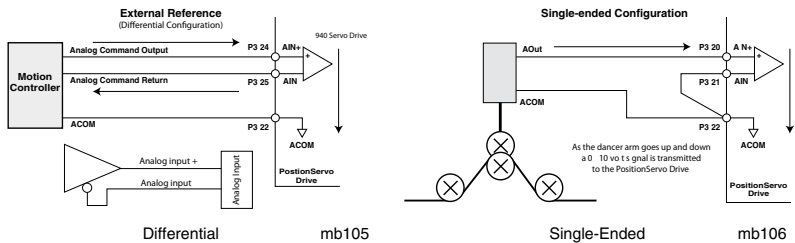
4.3 Analog I/O Details

4.3.1 Analog Reference Input

AIN+, AIN- (P3.24 and P3.25)

The analog reference input can accept up to a $\pm 10\text{V}$ analog signal across AIN1+ and AIN2-. The maximum limit with respect to analog common (ACOM) on each input is $\pm 18\text{VDC}$. The analog signal will be converted to a digital value with 16 bit resolution (15 bit plus sign). This input is used to control speed or torque of the motor in velocity or torque mode. The total reference voltage as seen by the drive is the voltage difference between AIN1+ and AIN1-. If used in single-ended mode, one of the inputs must be connected to a voltage source while the other one must be connected to Analog Common (ACOM). If used in differential mode, the voltage source is connected across AIN+ and AIN- and the driving circuit common (if any) needs to be connected to the drive Analog Common (ACOM) terminal.

Reference as seen by drive: $V_{ref} = (AIN1+) - (AIN1-)$ and $-10\text{V} \leq V_{ref} \leq +10\text{V}$



AIN2+, AIN2- (P3.20 and P3.21)

The analog reference input can accept up to a $\pm 10\text{V}$ analog signal across AIN2+ and AIN2-. The maximum limit with respect to analog common (ACOM) on each input is $\pm 18\text{VDC}$. The analog signal will be converted to a digital value with 10 bit resolution (9 bit plus sign). This input is available to the User's program. This input does not have a predefined function. Scaling of this input is identical to AIN1.

4.3.2 Analog Output

AO (P3.23)

The analog output is a single-ended signal (with reference to Analog Common (ACOM)) which can represent the following Motor data:

- Not Assigned
- Phase R Current
- Phase S Current
- Phase T Current
- Iq current
- Id current
- Motor Velocity

Motor phase U, V and W correspond to R, S and T respectively.

MotionView Setup program can be used to select the signal source for the analog output as well as its scaling.

If the output function is set to "Not Assigned" then the output can be controlled directly from user's program. Refer to the 940 Programming Manual.



STOP!

Upon application of power to the PositionServo, the Analog Output supplies -10VDC until bootup is complete. Once bootup is complete, the Analog Output will supply the commanded voltage.



Interface

4.4 Communication Interfaces

4.4.1 RS232 Interface (standard)

Programming and diagnostics of the 940 drive is done over the standard RS232 communication port. The baud rate for this port can be configured to one of 7 different settings, ranging from 2400 to 115200. Drives are addressable with up to 32 addresses from 0-31. Communication speed and address are set from the drive's front panel display.

4.4.2 RS485 Interface (option)

PositionServo 940 drives can be equipped with an RS485 communication interface option module (E94ZARS41) which is optically isolated from the rest of the drive's circuitry. This option module can be used for two functions: drive programming and diagnostics using MotionView from a PC (with RS485 port) or as a Modbus RTU slave. The 940 family of drives support 7 different baud rates, ranging from 2400 to 115200. Drives are addressable with up to 32 addresses from 0-31. The factory setting for the baud rate is 38,400 with a node address of "1". The drives address must be set from the front panel display of the drive. When used with MotionView software, the communication speed is also set from the front panel display. Please note that baud rate and address are applied to both RS232 and RS485 interfaces in this case. If used for Modbus RTU communications, the Modbus baud rate is set as a parameter within MotionView.

PIN ASSIGNMENTS (RS485 interface)

Pin	Name	Function
1	ICOM	Isolated Common
2	TXB	Transmit B
3	TXA	Transmit A

4.4.3 Using RS232 and RS485 Interfaces Simultaneously

When establishing communication between MotionView and a 940 drive, a communication method must be selected. The connection choice can be either "UPP over RS485/RS232" or "Ethernet". The "UPP over RS485/RS232" selection establishes a RS232 connection between MotionView and the first drive on the network. Multiple drives can then be added to the network via RS485. Each drive on the network must have a different Node Address. When setting up communications the node address of the target drive must be set. MotionView will then send out a communications packet to the first drive on the network, via the RS232 connection. If the node address set in this packet doesn't match the node address of the drive, the drive will resend the packet, via RS485, to the next drive on the network. This process will continue until the target drive is reached. The following message, "Device with address # not present in the network" will appear if the target node could not be found.



4.4.4 MODBUS RTU Support

As a default, the RS232 and RS485 interfaces are configured to support MotionView program operations. In addition, the RS485 interface can be configured to support the MODBUS RTU slave protocol. The interface can be configured through the MotionView program. When configured for MODBUS operation, the baud rate for RS485 is set by the parameter “Modbus baud rate” in MotionView, while the RS232 baud rate is set on the drive’s front panel. Thus RS485 and RS232 can have different speeds at the same time if RS485 is configured for MODBUS operation. Please note that if RS485 is configured for MODBUS operation, the command repeat function (see 4.4.3) is unavailable even if baud rates are set the same for both interfaces.

The Modbus RTU slave interface protocol definitions can be found in the MotionView help menu under “Product Manuals”.

4.5 Motor Selection

The PosiionServo 940 drive is compatible with many 3-phase AC synchronous servo motors. MotionView is equipped with a motor database which contains over 600 motors for use with the 940 drive. If the desired motor is in the database, no data to set it up is needed. Just select the motor and click “OK”. However, if your motor is not in the database, it can still be used, but some electrical and mechanical data will need to be provided to create a custom motor profile. The auto-phasing feature of the 940 allows the user to correctly determine the relationship between phase voltage and hall sensor signals, eliminating the need to use a multi-channel oscilloscope.

4.5.1 Motor Connection

Motor phase U, V, W (or R, S, T) are connected to terminal P7. It is very important that motor cable shield is connected to Earth ground terminal (PE) or the drive’s case.

The motor feedback cable must be connected to encoder terminal P4 if the motor is equipped with an incremental encoder. If the motor is equipped with a resolver it needs to be connected to terminal P11 on the resolver option module.

4.5.2 Motor Over-Temperature Protection

If using a motor equipped with an encoder and PTC thermal sensor, the encoder feedback cable will have flying leads exiting the P4 connector to be wired to the P7.1 (T1) and P7.2 (T2) terminals. If using a motor equipped with a Resolver and a PTC sensor, the connector on the Resolver Option Module (P11) provides this connection.

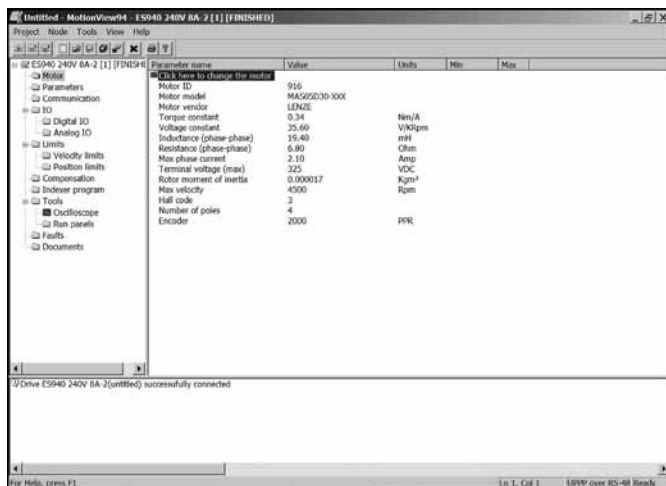
Use parameter “Motor PTC Cut-off Resistance” (section 5.3.12) to set the resistance which corresponds to maximum motor allowed temperature. The parameter “Motor temperature sensor” must also be set to ENABLE. If the motor doesn’t have a PTC sensor, set this parameter to DISABLE. This input will also work with N.C. thermal switches which have only two states; Open or Closed. In this case “Motor PTC Cut-off Resistance” parameter can be set to the default value.



Interface

4.5.3 Motor Set-up

Once you are connected to the PositionServo 940 via MotionView a “Parameter Tree” will appear in the “Parameter Tree Window”. The various parameters of the drive are shown here as folders and files. If the “Motor” folder is selected, all motor parameters can be viewed in the “Parameter View Window”. To view selected motor parameters or to select a new motor click the section marked “CLICK HERE TO CHANGE”.



MotionView's “Motor Group” folder and its contents

S911



NOTE

If the drive is ENABLED, a new motor cannot be set. You can only set a new motor when the drive is DISABLED.

To View selected motor parameters or to make a new motor selection:

- Click “Click here to change the motor” from the Parameter View Window (see figure above). If you are just viewing motor parameters click Cancel on Motor Parameters dialog when done to dismiss the dialog box.
- Select motor Vendor from the right list box and desired motor from the left list box.
- If you will be using a “custom” motor (not listed in our motor database) go to “Using a custom motor” topic in the next section.
- Finally, click the OK button to dismiss the dialog and return to MotionView's main program.



4.6 Using a Custom Motor

You can load a custom motor from a file or you can create a new custom motor.

- To create a custom motor click “CREATE CUSTOM” and follow the instructions in the next section “Creating custom motor parameters”.
- To load a custom motor click “OPEN CUSTOM” button then select the motor file and click the “OPEN” button to select or click the “CANCEL” button to return to the previous dialog box.
- Click OK to load the motor data and return to the main MotionView menu or Cancel to abandon changes. When clicking OK for a custom motor, a dialog box will appear asking if you want to execute “Autophasing” (section 4.6.2).

4.6.1 Creating Custom Motor Parameters



STOP!

Use extreme caution when entering custom parameters! Incorrect settings may cause damage to the drive or motor! If you are unsure of the settings, refer to the materials that were distributed with your motor, or contact the motor manufacturer for assistance.

1. Enter custom motor data in the Motor Parameters dialog fields. Complete all sections of dialog: Electrical, Mechanical, Feedback. Refer to Section 4.6.3 for explanation of motor parameters and how to enter them.



NOTE

If unsure of the motor halls order and encoder channels A and B relationship, leave “B leads A for CW”, “Halls order” and “inverted” fields as they are. You can execute autophasing (section 4.6.2) to set them correctly.

2. Enter motor model and vendor in the top edit boxes. Motor ID cannot be entered, this is set to 0 for custom motors.
3. Click “Save File” button and enter filename without extension. Default extension .cmt will be given when you click OK on file dialog box.



NOTE

Saving the file is necessary even if the autophasing feature will be used and some of the final parameters are not known. After autophasing is completed the corrected motor file can be updated before loading it to memory.

4. Click OK to exit from the Motor Parameters dialog.
5. MotionView will ask if you want to autophase your custom motor. If you answer “No”, the motor data will be loaded immediately to the drive’s memory. If you answer “Yes”, the motor dialog will be dismissed and the drive will start the autophasing sequence. Refer to section 4.6.2 for autophasing information.
6. If you answered “Yes” for autophasing, you will be returned to the same motor selection dialog box after autophasing is complete. For motors with incremental encoders, the fields “B leads A for CW”, “Halls order” and “inverted” will be assigned correct values. For motors with resolvers, the fields “Offset in degree” and “CW for positive” will be assigned correct values.
7. Click “Save File” to save the custom motor file and then click “OK” to exit the dialog box and load the data to the drive.



4.6.2 Autophasing

The Autophasing feature determines important motor parameters when using a motor that is not in MotionView's database. For motors equipped with incremental encoders, Autophasing will determine the Hall order sequence, Hall sensor polarity and encoder channel relationship (B leads A or A leads B for CW rotation). For motors equipped with resolvers, Autophasing will determine resolver angle offset and angle increment direction ("CW for positive").

To perform autophasing:

1. Complete the steps in the previous section "Setting custom motor parameters". If the motor file you are trying to autophase already exists, simply load it as described under "Using a custom motor" at the beginning of this section.
2. Make sure that the motor's shaft is not connected to any mechanical load and can freely rotate.



STOP!

Autophasing will energize the motor and will rotate the shaft. Make sure that the motor's shaft is not connected to any mechanical load and can freely and safely rotate.

3. **Make sure that the drive is not enabled.**
4. It is not necessary to edit the field "Hall order" and check boxes "inverted" and "B leads A for CW" as these values are ignored for autophasing.
5. Click OK to dismiss motor selection dialog. MotionView responds with the question "Do you want to perform autophasing?"
6. Click OK. A safety reminder dialog appears. Verify that it is safe to run the motor then click "Proceed" and wait until autophasing is completed.



NOTE

If there was a problem with the motor connection, hall sensor connection or resolver connection, MotionView will respond with an error message. Common problems include power, shield and ground terminations or use of an improper cable. Correct the wiring problem(s) and repeat steps 1 - 6.

If the error message repeats, exchange motor phases U and V (R and S) and repeat. If problems persist, contact the factory.

7. If autophasing is completed with no error then MotionView will return to the motor dialog box. For motors with incremental encoders, the parameter field "Hall order" and the check boxes "inverted", "B leads A for CW" will be filled in with correct values. For resolver equipped motors, fields "Offset" and "CW for positive" will be correctly set.
8. Click "Save File" to save the completed motor file (you can use the same filename as you use to save initial data in step 1) and click OK to load the motor data to the drive.

4.6.3 Custom Motor Data Entry

A Custom Motor file is created by entering motor data into the "Motor Parameters" dialog box. This box is divided up into the following three sections, or frames:

- Electrical constants
- Mechanical constants
- Feedback

When creating a custom motor you must supply all parameters listed in these sections. All entries are mandatory except the motor inertia (J_m) parameter. A value of 0 may be entered for the motor inertia if the actual value is unknown.



4.6.3.1 Electrical Constants

Motor Torque Constant (Kt)

Enter the value and select proper units from the drop-down list.



NOTE

Round the calculated result to 3 significant places.

Motor Voltage Constant (Ke)

The program expects Ke to be entered as a phase-to-phase Peak voltage. If you have Ke as an RMS value, multiply this value by 1.414 for the correct Ke Peak value.

Phase-to-phase winding Resistance (R) in Ohms (Ω)

This is also listed as the terminal resistance (Rt). The phase-to-phase winding Resistance (R) will typically be between 0.05 and 200 Ohms.

Phase-to-phase winding Inductance (L)

This must be set in millihenries (mH). The phase-to-phase winding Inductance (L) will typically be between 0.1 and 200.0 mH.



NOTE

If the units for the phase-to-phase winding Inductance (L) are given in micro-henries (μ H), then divide by 1000 to get mH.

Nominal phase current (RMS Amps)

Nominal continuous phase current rating (In) in Amps RMS. Do not use the peak current rating.



NOTE

Sometimes the phase current rating will not be given. The equation below may be used to obtain the nominal continuous phase-to-phase winding current from other variables.

$$I_n = \text{Continuous Stall Torque} / \text{Motor Torque Constant (Kt)}$$

The same force x distance units must be used in the numerator and denominator in the equation above. If torque (T) is expressed in units of pound-inches (lb-in) then, Kt must be expressed in pound-inches per Amp (lb-in/A). Likewise, if T is expressed in units of Newton-meters (N-m), then units for Kt must be expressed in Newton-meters per Amp (N-m/A).

Example:

Suppose that the nominal continuous phase to phase winding current (In) is not given. Instead, we look up and obtain the following:

Continuous stall torque T = 3.0 lb-in

Motor torque constant Kt = 0.69 lb-in/A

Dividing, we obtain:

$$I_n = 3.0 \text{ lb-in} / 0.69 \text{ lb-in/A} = 4.35 \text{ (A)}$$

Our entry for (In) would be 4.35.

Note that the torque (lb-in) units cancelled in the equation above leaving only Amps (A). We would have to use another conversion factor if the numerator and denominator had different force x distance units.



Interface

Nominal Bus Voltage (Vbus)

The Nominal Bus Voltage can be calculated by multiplying the Nominal AC mains voltage supplied by 1.41. When using a model with the suffix "S1N" where the mains are wired to the "Doubler" connection, the Nominal Bus Voltage will be doubled.

Example:

If the mains voltage is 230VAC, $V_{bus} = 230 \times 1.41 = 325V$

This value is the initial voltage for the drive and the correct voltage will be calculated dynamically depending on the drive's incoming voltage value.

Rotor Moment of Inertia (Jm)

From motor manufacturer or nameplate.



NOTE

Round the calculated result to 3 significant places.

Maximum Motor Speed in RPM

This is also listed as "Speed @ Vt" (motor speed at the terminal voltage rating). The maximum motor speed will typically be a round even value between 1000 and 6000 RPM.

Number of Poles

This is a positive integer number that represents the number of motor poles, normally 2, 4, 6 or 8.

4.6.3.2 For Motors Equipped with Incremental Encoders Only

Encoder Line Count

The Encoders for servomotors normally have Line Counts of 1000, 1024, 2000, 2048, 4000, or 4096. The Encoder Line Count must be a positive integer and must be pre-quadrature.

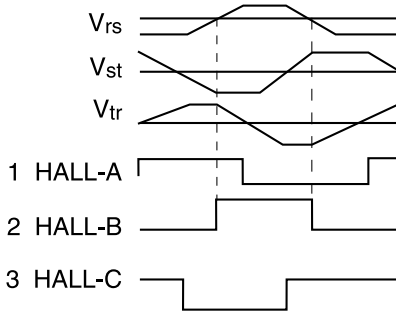
Index pulse offset. Enter 0 (zero)

Index marker pulse position. This field is reserved for backward compatibility. All PositionServo drives determine actual marker pulse position automatically.



Halls Order

Each hall signal is in phase with one of the three phase-phase voltages from the motor windings. Hall order number defines which hall sensor matches which phase-phase voltage. Motor phases are usually called R-S-T or U-V-W or A-B-C. Phase-Phase voltages are called V_{rs} , V_{st} , V_{tr} . Halls are usually called HALL-A, HALL-B, HALL-C or just Halls 1, 2, 3. A motor's phase diagram is supplied by motor vendor and usually can be found in the motor data sheet or by making a request to the motor manufacturer. A sample phase diagram is shown below.



S912

The Halls Order is obtained as follows:

1. By looking at the " V_{rs} " Output Voltage, determine which Hall Voltage is lined up with (or in phase with) this voltage. We can determine which Hall Voltage is in phase with the V_{rs} Output Voltage by drawing vertical lines at those points where it crosses the horizontal line (zero). The dashed lines at the zero crossings (above) indicate that Hall B output is lined up with (and in phase with) the V_{rs} Output Voltage.
2. Look at the " V_{st} " Output Voltage. Determine which Hall Voltage is in phase with this Voltage. As can be seen, Hall C output is in phase with the V_{st} Output Voltage.
3. Look at the " V_{tr} " Output Voltage. Determine which Hall Voltage is in phase with this Voltage. As can be seen, Hall A output is in phase with the V_{tr} Output Voltage.



NOTE

If hall sensors are in phase with the corresponding phase voltage but are inverted 180 degrees (hall sensor waveform edge aligns with the phase-phase voltage waveform but the positive hall sensor cycle matches the negative phase-phase waveform or visa-versa), you must check the "Inverted" check box.



Interface

- The phases that correspond to the Vrs Vst Vtr voltages are Hall B then Hall C then Hall A or Halls number 2 then 3 then 1. Referring to the following table, we find that 2-3-1 sequence is Halls Order number 3. We would enter 3 for the Halls Order field in motor dialog.

HALL ORDER NUMBERS FOR DIFFERENT HALL SEQUENCES

Halls Order	Hall Sequence
0	1-2-3
1	1-3-2
2	2-1-3
3	2-3-1
4	3-1-2
5	3-2-1



NOTE

Each Hall Voltage will be in phase with one and only one Output Voltage.

B leads A for CW

This is the encoder phase relationship for CW/CCW shaft rotation. When you obtain the diagram for your motor phasing similar to shown above, it's assumed by the software that the motor shaft rotates CW when looking at the mounting face of the motor. For that rotation Encoder phase A must lead phase B. If it does leave check box unchecked. Otherwise (if B leads A) check B leads A for CW box.



NOTE

Lenze convention references the shaft direction of rotation from the front (shaft end) of the motor. Some manufacturers' timing diagrams are CW when viewed from the "rear" of the motor.

4.6.3.3 For Resolver Equipped Motors Only

If parameter "Resolver" is checked, following parameters appear on the form:

Offset in degree (electrical)

This parameter represents offset between resolver's "0 degree" and motor's windings "0 degree".

CW for positive

This parameter sets the direction for positive angle increment.

"Offset in degree" and "CW for positive" will be set during Auto-Phasing of the motor.



5 Parameters

PositionServo 940 series drives are configured through one of the interfaces: RS232, RS485 or Ethernet. The drives have many programmable and configurable features and parameters. These features and parameters are accessible via a universal software called MotionView. Please refer to the MotionView Manual for details on how to make a connection to the drive and change parameter values.

This chapter covers programmable features and parameters specific to the PositionServo Model 940 drive in the order they appear in the Parameter Tree of MotionView. Programmable parameters are divided into groups. Each group holds one or more user's adjustable parameters.

All 940 series drives can execute a User Program in parallel with motion. Motion can be specified by variety of sources and in three different modes:

Torque Velocity Position

In Torque and Velocity mode Reference can be taken from Analog Input AIN1 or from the User Program by setting a particular variable (digital reference). See Programmer's Manual for details on programming. In Position mode, the reference could be taken from MA/MB master encoder/step and directions inputs (available in terminal P3) or from trajectory generator. Access to the trajectory generator is provided through the User Program's motion statements, MOVEx and MDV. Refer to the PositionServo Programmer's Manual for details on programming.

Whether the reference comes from an external device, (AIN1 or MA/MB) or from the drives internal variables (digital reference and trajectory generator) will depend on the parameter settings. Refer to "Parameters" group in MotionView.

5.1 Parameter Storage and EPM Operation

5.1.1 Parameter Storage

All settable parameters are stored in the drive's internal non-volatile memory. Parameters are saved automatically when they are changed and are copied to the EPM memory module located on the drive's front panel. In the unlikely event of drive failure, the EPM can be removed and inserted into the replacement drive, thus making an exact copy of the drive being replaced. This shortens down time by eliminating the configuration procedure. The EPM can also be used for replication of the drive's settings.

5.1.2 EPM Operation

When the drive is powered up it first checks for a white EPM in the EPM Port. If the EPM Port is empty, no further operation is possible until a white EPM is installed into the EPM Port. The drive will display "EP-" until an EPM is inserted. Do not insert or remove the EPM module while the drive is powered.

If a different color EPM is inserted the drive may appear to function however, some operations will not be correct and the drive may hang. The white EPM is the only acceptable EPM for the PositionServo 940 drive. If a white EPM is detected, the drive compares data in the EPM to that in its internal memory. In order for the drive to operate, the contents of the drive's memory and EPM must be the same. Press the enter button to load the EPM, this will take a moment. The drive will display "BUSY" during this time and will return to normal display when update is complete.

**STOP!**

If the EPM contains any data from an inverter drive, that data will be overwritten during this procedure.



5.1.3 EPM Fault

If the EPM fails during operation or the EPM is removed from the EPM Port the drive will generate a fault and will be disabled (if enabled). The fault is logged to the drives memory. Further operation is not possible until the EPM is replaced (inserted) and the drive's power is cycled. The fault log on the display shows "F_EP" fault.

5.2 Motor Group

The motor group shows the data for the currently selected motor. Refer to Section 4.5 for details on how to select another motor from the motor database or to configure a custom motor.

5.3 Parameters

5.3.1 Drive Operating Modes

The PositionServo has 3 operating mode selections: Torque, Velocity and Position.

For Torque and Velocity modes the drive will accept an analog input voltage on the AIN+ and AIN- pins of P3 (refer to section 4.3.1). This voltage is used to provide a torque or speed reference.

For Position mode the drive will accept step and direction logic signals or a quadrature pulse train on pins P3.11-14.

5.3.1.1 Velocity Mode

In velocity mode, the servo controller regulates motor shaft speed (velocity) proportional to the analog input voltage at input AIN1, if parameter "Reference" is set to "External". Otherwise the reference is taken from the drive's internal variable, IREF. (Refer to Programmer's manual for details).

For analog reference, Target speed (set speed) is calculated using the following formula:

$$\text{Set Velocity (RPM)} = \text{Vinput (Volt)} \times \text{Vscale (RPM/Volt)}$$

where:

- Vinput is the voltage at analog input (AIN+ and AIN-)
- Vscale is the velocity scale factor (input sensitivity) set by the Analog input (Velocity scale) parameter (section 5.3.6).

$$\text{Set Velocity (RPM)} = \text{Vinput (Volt)} \times \text{Vscale (RPM/Volt)}$$

5.3.1.2 Torque Mode

In torque mode, the servo control provides a current output proportional to the analog input signal at input AIN1, if parameter "Reference" is set to "External". Otherwise the reference is taken from the drive's internal variable, IREF. (Refer to Programmer's manual).

For analog reference "Set Current", (current the drive will try to provide), is calculated using the following formula:

$$\text{Set Current(A)} = \text{Vinput(Volt)} \times \text{Iscale (A/Volt)}$$

where:

- Vinput is the voltage at analog input
- Vscale is the current scale factor (input sensitivity) set by the Analog input (Current Scale) parameter (section 5.5.2).



5.3.1.3 Position Mode

In this mode the drive reference is a pulse-train applied to P3.1-4 terminals, if the parameter "Reference" is set to "External". Otherwise the reference is taken from the drive's internal variables. (Refer to Programmer's manual for details).

P3.1-4 inputs can be configured for two types of signals: step and direction and Master encoder quadrature signal. Refer to section 4.2.1 for details on these inputs connections. Refer to section 8.3 for details about positioning and gearing.

When the Reference is set to Internal, the drives reference position, (theoretical or Target position), is generated by trajectory generator. Access to the trajectory generator is provided by motion statements, MOVEx and MDV, in the User Program.

5.3.2 Drive PWM Frequency

This parameter sets the PWM carrier frequency. Frequency can be changed only when the drive is disabled. Maximum overload current is 300% of the drive rated current when the carrier is set to 8kHz, it is limited to 250% at 16kHz.

5.3.3 Current Limit

The CURRENT LIMIT setting determines the nominal current, in amps RMS per phase.

5.3.4 Peak Current Limit (8kHz and 16 kHz)

Peak current sets the motor RMS phase current that is allowed for up to 2 seconds. After this two second limit, the current limit will be reduced to the value set in the Current Limit parameter. When the motor current drops below nominal current for two seconds, the drive will automatically re-enable the peak current level. This technique allows for high peak torque on demanding fast moves and fast start/stop operations with high regulation bandwidth. The control will use only the Peak current limit parameter for the carrier frequency selected.

5.3.5 Analog Input Scale (Current)

This parameter sets the analog input sensitivity for current reference used when the drive operates in Torque mode. Units for this parameter are A/Volt. To calculate this value use the following formula:

$$I_{scale} = I_{max} / V_{in\ max}$$

I_{max} maximum desired output current (motor phase current RMS)
 $V_{in\ max}$ max voltage fed to analog input at I_{max}

Example: $I_{max} = 5A$ (phase RMS)
 $V_{in\ max} = 10V$
 $I_{scale} = I_{max} / V_{in\ max} = 5A / 10V = 0.5 A / Volt$ (value to enter)

5.3.6 Analog Input Scale (Velocity)

This parameter sets the analog input sensitivity for the velocity reference used when the drive operates in Velocity mode. Units for this parameter are RPM/Volt. To calculate this value use the following formula:

$$V_{scale} = VELOCITY_{max} / V_{in\ max}$$

$VELOCITY_{max}$ maximum desired velocity in RPM
 $V_{in\ max}$ max voltage fed to analog input at $Velocity_{max}$

Example: $VELOCITY_{max} = 2000$ RPM
 $V_{in\ max} = 10V$
 $V_{scale} = VELOCITY_{max} / V_{in\ max}$
 $= 2000 / 10V$
 $= 200 RPM / Volt$ (value to enter)



Parameters

5.3.7 ACCEL/DECEL Limits (Velocity Mode Only)

The ACCEL setting determines the time the motor takes to ramp to a higher speed. The DECEL setting determines the time the motor takes to ramp to a lower speed. If the ENABLE ACCEL/DECEL LIMITS is set to DISABLE, the drive will automatically accelerate and decelerate at maximum acceleration limited only by the current limit established by the PEAK CURRENT LIMIT and CURRENT LIMIT settings.

5.3.8 Reference

The REFERENCE setting selects the reference signal being used by the drive. This reference signal can be either External or Internal. An External Reference can be one of three types, a Analog Input signal, a Step and Direction Input or a Input from a external Master Encoder. The Analog Input reference is used when the drive is either in Torque or Velocity mode. The Master Encoder and Step and Direction reference is used when the drive is in Position Mode. An Internal Reference is used when the motion being generated is derived from drive's internal variable(s), or User Program, (See programmer's manual).

5.3.9 Step Input Type (Position Mode Only)

This parameter sets the type of input for position reference the drive expects to see. Signal type can be step and direction (S/D) type or quadrature pulse-train (Master Encoder / Electronic Gearing). Refer to section 4.2.1 for details on these inputs.

5.3.10 Fault Reset Option

The FAULT RESET OPTION selects the type of action required to reset the drive after a FAULT signal has been generated by the drive. ON DISABLE clears the fault when the drive is disabled. This is useful if you have a single drive and motor connected in a single drive system. The ON ENABLE option clears the fault when the drive is re-enabled. Choose ON ENABLE if you have a complex servo system with multiple drives connected to an external controller. This makes troubleshooting easier since the fault will not be reset until the drive is re-enabled. Thus, a technician can more easily determine which component of a complex servo system has caused the fault.

5.3.11 Motor Temperature Sensor

This parameter enables / disables motor over-temperature detection. It must be disabled if the motor PTC sensor is not wired to either P7.1-2 or to the resolver option module (P11).

5.3.12 Motor PTC Cut-off Resistance

This parameter sets the cut-off resistance of the PTC which defines when the motor reaches the maximum allowable temperature. Refer to section 4.5.2 for details how to connect motor's PTC.

5.3.13 Second Encoder

Disables or enables second encoder. Effectively selects single-loop or double-loop configuration in position mode. The second encoder connects to the Encoder Option Module (E94ZAENC1) connector P12, Refer to section 6.4 for details on dual loop operation.



5.3.14 Regen Duty Cycle

This parameter sets the maximum duty cycle for the brake (regen) resistor. This parameter can be used to prevent brake resistor overload. Use the following formula to set the correct value for this parameter.

$$D = P * R / (U_{max})^2 * 100\%$$

where:

D (%) regen duty cycle

U_{max} (V) bus voltage at regen conditions.
 U_{max}=390V for 230VAC drives and 770V for 400/480VAC drives

R (ohm) regen resistor value

P (W) regen resistor rated power

If calculation of D is greater than 100% set it to 100% value. If calculation of D is less than 10% then resistor power rating is too low. For more information refer to the PositionServo Dynamic Braking Manual (G94BR01).

Minimum Required Dynamic Braking Resistance

Drive Model	DB Resistor Minimum Resistance (Ω)
E94_080S2F, E94_080Y2N, E94_100S2F, E94_100Y2N	20
E94_120Y2N	30
E94_020S1N, E94_020S2F, E94_020Y2N, E94_040S1N, E94_040S2F	40
E94_040T4N, E94_050T4N, E94_060T4N	75
E94_020T4N	150

5.3.15 Encoder Repeat Source

This parameter sets the feedback source signal for the buffered encoder repeat outputs (P3.1-6). The source can be the drive's encoder input (P4) or an optional feedback module (resolver, second encoder etc.)

5.3.16 System to Master Ratio

This parameter is used to set the scale between the reference pulse train (when operating in position mode) and the system feedback device. In a single loop configuration, the system feedback device is the motor encoder or resolver. In a dual-loop system the system encoder is the second encoder. Refer to sections 6.3 and 6.4 for details.

5.3.17 Second to Prime Encoder Ratio

This parameter sets the ratio between the secondary encoder and the primary feedback device when the drive is configured to operate in dual-loop mode. When the primary feedback device is a resolver, the pulse count is fixed at 65,536. The resolutions of encoders are "post quadrature" (PPR x 4). Refer to section 6.4.



NOTE

Post quadrature pulse count is 4X the pulses-per-revolution (PPR) of the encoder.

5.3.18 Autobot

When set to "Enabled" the drive will start to execute the user's program immediately after cold boot (reset). Otherwise the user program has to be started from MotionView or from the Host interface.



Parameters

5.3.19 Group ID

Refer to the Programmer's manual for details. This parameter is only needed for operations over an Ethernet network.

5.3.20 Enable Switch Function

If set to "Run", input IN_A3 (P3.29) acts as an "Enable" input when the user program is not executing. If the user program is executing, the function will always be "Inhibit" regardless of the setting. This parameter is needed so the drive can be Enabled/Disabled without running a user's program.

5.3.21 User Units

This parameter sets up the relationship between User Units and motor revolutions. From here you can determine how many User Units there is in one motor revolution. This parameter allows the user to scale motion moves to represent a desired unit of measure, (inches, meters, in/sec, meters/sec, etc).

For example:

A linear actuator allows a displacement of 2.5" with every revolution of the motor's shaft.

Units = Units / Revolutions

Units = 2.5 Inches / Revolution

Units = 2.5

5.4 Communication

5.4.1 IP Setup

This action button opens dialog for TCP/IP related parameters setup.

5.4.2 RS-485 Configuration

This parameter sets how the optional RS485 interface will function. The RS485 interface can be configured for normal operation (programming and diagnostics using MotionView software) or as a Modbus RTU slave. See section 4.4 for comm interfaces.

5.4.3 Modbus Baud Rate

This parameter sets the baud rate for RS485 interface in Modbus RTU mode. When the drive is operating in normal mode the baud rate is set to the same setting as the RS232 interface.

5.4.4 Modbus Reply Delay

This parameter sets the time delay between the drives reply to the Modbus RTU master. This delay is needed for some types of Modbus masters to function correctly.

5.5 Analog I/O

5.5.1 Analog Output

The PositionServo 940 has one analog output with 10-bit resolution on P3.23. The signal is scaled to $\pm 10V$. The analog output can be assigned to following functions:

- Not Assigned
- Phase R current
- Iq current (Torque component)
- Phase current RMS
- Phase S current
- Id current (Direct component)
- Phase current Peak
- Phase T current
- Motor Velocity



5.5.2 Analog Output Current Scale (Volt/amps)

Applies scaling to all functions representing CURRENT values.

5.5.3 Analog Output Velocity Scale (mV/RPM)

Applies scaling to all functions representing VELOCITY values. (Note: that mV/RPM scaling units are numerically equivalent to volts/kRPM).

5.5.4 Analog Input Dead Band

Allows the setting of a voltage window (in mV) at the reference input AIN1+ and AIN1- (P3.24 and 25) such that any voltage within that window will be treated as zero volts. This is useful if the analog input voltage drifts resulting in motor rotation when commanded to zero.

5.5.5 Analog Input Offset Parameter

Allows you to adjust the offset voltage at AIN1+ and AIN1- (P3.24 and P3.25). This function is equivalent to the balance trim potentiometer found in analog drives. Lenze recommends that this adjustment be made automatically using the "Adjust analog voltage offset" button while the external analog reference signal commands zero speed.

5.5.6 Adjust Analog Voltage Offset

This control button is useful to allow the drive to automatically adjust the analog input voltage offset. To use it, command the external reference source input at AIN1+ and AIN1- (P3.24 and 25) to zero volts and then click this button. Any offset voltage at the analog input will be adjusted out and the adjustment value will be stored in the "Analog input offset" parameter.

5.6 Digital I/O

The 940 has four digital outputs. These outputs can be either assigned to one of the following functions, or be used by the drives internal User Program

- **Not Assigned** No special function assigned. Output can be used by the User Program.
- **Zero Speed** Output activated when drive is at zero speed, refer to "Velocity Limits Group" (Section 5.7) for settings.
- **In Speed Window** Output activated when drive is in set speed window, refer to "Velocity Limits Group" (Section 5.7) for settings.
- **Current Limit** Output activated when drive detects current limit.
- **Run Time Fault** A fault has occurred. Refer to Section 8.3 for fault details.
- **Ready** Drive is enabled.
- **Brake** Command for the holding brake option (E94ZAHBK2) for control of a motor mounted brake. This output is active 10ms after the drive is enabled and deactivates 10ms before the drive is disabled.
- **In position** Position mode only. Refer to Programming Manual for details

5.6.1 Digital Input De-bounce Time

Sets de-bounce time for the digital inputs to compensate for bouncing of the switch or relay contacts. This is the time during an input transition that the signal must be stable before it is recognized by the drive.



5.6.2 Hard Limit Switch Action

Digital inputs IN_A1-IN_A2 can be used as limit switches if their function is set to “Fault” or “Stop and Fault”. Activation of this input while the drive is enabled will cause the drive to Disable and go to a Fault state. The “Stop and Fault” action is available only in Position mode when the “Reference” parameter is set to “Internal” i.e. when the source for the motion is the Trajectory generator. IN_A1 is used as the negative limit switch. IN_A2 is used as the positive limit switch. Both are treated as normally open.

5.7 Velocity Limits

These parameters are active in Velocity Mode Only.

5.7.1 Zero Speed

Specifies the upper threshold for motor zero speed in RPM. When the motor shaft speed is at or below the specified value the zero speed condition is set to true in the internal controller logic. The zero speed condition can also trigger a programmable digital output, if selected.

5.7.2 Speed Window

Specifies the speed window width used with the “In speed window” output.

5.7.3 At Speed

Specifies the speed window center used with the “In speed window” output.

These last two parameters specify speed limits. If motor shaft speed is within these limits then the condition AT SPEED is set to TRUE in the internal controller logic. The AT SPEED condition can also trigger a programmable digital output, if selected. For example if “AT SPEED” is set for 1000 RPM, and the “SPEED WINDOW” is set for 100, then “AT SPEED” will be true when the motor velocity is between 950 -1050 RPM.

5.8 Position Limits

5.8.1 Position Error

Specifies the maximum allowable position error in the primary (motor mounted) feedback device before enabling the “Max error time” clock (described next). When using an encoder, the position error is in post-quadrature encoder counts. When using a resolver, position error is measured at a fixed resolution of 65,536 counts per motor revolution.

5.8.2 Max Error Time

Specifies maximum allowable time (in mS) during which a position error can exceed the value set for the “Position error” parameter before a Position Error Excess fault is generated.

5.8.3 Second Encoder Position Error

Specifies the maximum allowable error of the second encoder in post quadrature encoder counts before enabling the “Second encoder max error time” clock.

5.8.4 Second Encoder Max Error Time

Specifies maximum allowable time (in mS) during which the second encoder’s position error can exceed the value set for the “Second encoder position error” parameter before a Position Error Excess fault is generated.



5.9 Compensation

5.9.1 Velocity P-gain (Proportional)

Proportional gain adjusts the system's overall response to a velocity error. The velocity error is the difference between the commanded velocity of a motor shaft and the actual shaft velocity as measured by the primary feedback device. By adjusting the proportional gain, the bandwidth of the drive is more closely matched to the bandwidth of the control signal, ensuring more precise response of the servo loop to the input signal.

5.9.2 Velocity I-gain (Integral)

The output of the velocity integral gain compensator is proportional to the accumulative error over cycle time, with I-gain controlling how fast the error accumulates. Integral gain also increases the overall loop gain at the lower frequencies, minimizing total error. Thus, its greatest effect is on a system running at low speed, or in a steady state without rapid or frequent changes in velocity.



NOTE

The following four position gain settings are only active if the drive is operating in Position mode. They have no effect in Velocity or Torque modes.

5.9.3 Position P-gain (Proportional)

Position P-gain adjusts the system's overall response to position error. Position error is the difference between the commanded position of the motor shaft and the actual shaft position. By adjusting the proportional gain, the bandwidth of the drive is more closely matched to the bandwidth of the control signal, ensuring more precise response of the servo loop to the input signal.

5.9.4 Position I-gain (Integral)

The output of the Position I-gain compensator is proportional to accumulative error over cycle time, with I-gain controlling how fast the error accumulates. Integral gain also increases overall loop gain at the lower frequencies, minimizing total error. Thus, its greatest effect is on a system running at low speed, or in a steady state without rapid or frequent changes in position.

5.9.5 Position D-gain (Differential)

The output of the Position D-gain compensator is proportional to the difference between the current position error and the position error measured in the previous servo cycle. D-gain decreases the bandwidth and increases the overall system stability. It is responsible for removing oscillations caused by load inertia and acts similar to a shock-absorber in a car.

5.9.6 Position I-limit

The Position I-limit will clamp the Position I-gain compensator to prevent excessive torque overshooting caused by an over accumulation of the I-gain. It is defined in terms of percent of maximum drive velocity. This is especially helpful when position error is integrated over a long period of time.



Parameters

5.9.7 Gain Scaling Window

Sets the total velocity loop gain multiplier (2^n) where n is the velocity regulation window. If, during motor tuning, the velocity gains become too small or too large, this parameter is used to adjust loop sensitivity. If the velocity gains are too small, decrease the total loop gain value, by decreasing this parameter. If gains are at their maximum setting and you need to increase them even more, use a larger value for this parameter.

5.10 Tools

5.10.1 Oscilloscope Tool

The oscilloscope tool gives real time representation of different signals inside the PositionServo 940 drive and is helpful when debugging and tuning drives. Operation of the oscilloscope tool is described in more detail in the MotionView Software User's Manual. The following are the signals that can be observed with the oscilloscope tool:

Phase Current (RMS):	Motor phase current
Phase Current (Peak):	Motor peak current
Iq Current:	Measures the motor Iq (torque producing) current
Motor Velocity:	Actual motor speed in RPM
Commanded Velocity:	Desired motor speed in RPM (velocity mode only)
Velocity Error:	Difference in RPM between actual and commanded motor speed
Position Error:	Difference between actual and commanded position (Step & Direction mode only)
Bus Voltage:	DC bus voltage
Analog Input:	Voltage at drive's analog input
Absolute Position:	Absolute (actual) position
Absolute Position Pulses:	Absolute position expressed in pulses of the primary feedback device
Secondary Abs Position:	Absolute (actual) position of secondary feedback device
Secondary Position Error:	Difference between actual and commanded position of secondary feedback device
Target Position:	Requested position
Target Position Pulses:	Requested position expressed in pulses of the primary feedback device
Position Increment:	Commanded position increment

5.10.2 Run Panels

Check Phasing

This button activates the Autophasing feature as described in section 4.6.2. However, in this panel only the motor phasing is checked, the motor data is not modified.

5.11 Faults Group

Faults Group loads the fault history from the drive. The 8 most recent faults are displayed with the newer faults replacing the older faults in a first-in, first-out manner. In all cases fault # 0 is the most recent fault. To clear the faults history from the drive's memory click on the "Reset Fault history" button. Each fault has its code and explanation of the fault. Refer to section 8.3 for details on faults.



6 Operation

This section offers guidance on configuring the PositionServo drive for operations in torque, velocity or position modes without requiring a user program. To use advanced programming features of PositionServo please perform all steps below and then refer to the Programmer's Manual for details on how to write motion programs.

6.1 Minimum Connections

For the most basic operation, connect the PositionServo to mains (line) power at terminal P1, the servomotor power at P7 and the motor feedback as appropriate.



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait at least 60 seconds before servicing drive. Capacitors retain charge after power is removed.

Below is a list of the minimum necessary connections:

- Connect a serial cable between PositionServo's P2 and your PC serial port using a straight-through 9 pin RS232 cable (available as EWLC003BA1NA).
- Connect mains power to terminal P1. Mains power must be as defined on the drive's data label (section 2.1).
- If the motor is equipped with an encoder, connect the encoder cable to the PositionServo feedback connector P4.
If the motor is equipped with a resolver, install the "Resolver option module" (E94ZARSV1) in the lower option bay and connect the resolver cable to P11.
- Connect motor windings U, V, W (sometimes called R, S, T) to terminal P7 according to Section 4.1.1. Make sure that motor cable shield is connected as described in section 3.2.
- Provide an Enable switch (IN_A3) according to Section 6.5.
- Perform drive configuration as described in the next section.



NOTE

You must configure the drive before it can be operated.
Proceed to Section 6.2.

6.2 Configuration of the PositionServo

Regardless of the mode in which you wish to operate, you must first configure the PositionServo 940 for your particular motor, mode of operation, and additional features if used.

Drive configuration consists of following steps:

- Motor Selection
- Mode of operation selection
- **Reference source selection (Very Important)**
- Drive parameters (i.e. current limit, acceleration / deceleration) setup
- Operational limits (velocity or position limits) setup
- Input / Output (I/O) setup
- Velocity / position compensator (gains) setup
- Optionally store drive settings in a PC file and exit the MotionView program.



Operation

To configure drive:

1. Ensure that the control is properly installed and mounted. Refer to Section 3 for installation instructions.
2. Perform wiring to the motor and external equipment suitable for desired operating mode and your system requirements.
3. Connect the serial port P2 on the drive to your PC serial port.
4. Make sure that the drive is disabled.
5. Apply power to the drive and wait until “d ,5” shows on the display. For anything other than this, refer to the chart below before proceeding.

Drive display:	Meaning
EP	EPM missing; Refer to 6.1.2
EPP	EPM data; Refer to 6.1.2
----	No valid firmware
----	Monitor mode

6. Using the drive’s keypad and display, check that the baud rate is set to 38.4 (kbps).
7. Using the drive’s keypad and display, check that the address is set to 1. Set if necessary.
8. Launch MotionView software on your computer.
9. From the MotionView menu, select <Project> <Connection setup>.
10. Select “UPPP over RS-485/RS-232”, then select <Properties> and select the computer’s serial port that the drive is connected to.
 - Select the Comm port that matches the serial port of the computer used for this connection
 - Set baud rate at 38400 and rest of the parameters at default.
11. Click <OK> twice to dismiss both dialog boxes.
12. From <Node> menu choose <Connect Drive>.
13. Click “Connect one” button, type “1” in the address box and press “OK” to dismiss dialog.
14. Drive connects and its icon appears in the left node tree of the MotionView’s screen.



NOTE

MotionView’s “Connection setup” properties need only be configured the first time MotionView is operated or if the port connection is changed. Refer to MotionView User’s Manual for details on how to make a connection to the drive.

15. Double-click on the drive’s icon to expand parameter group’s folders.
16. Select the motor to be used according to the Section 4.5.
17. Expand the folder “Parameters” and choose the operating mode for the drive. Refer to Section 5.3.1 for details on operating modes.
18. Click on the “Current limit” parameter (5.3.3) and enter current limit (in Amp RMS per phase) appropriate for the motor.
19. Click on the appropriate “Peak current limit” parameter (5.3.4) based on the “Drive PWM frequency” parameter (5.3.2) used and enter the peak current limit (in Amp RMS per phase) appropriate for your motor.
20. Set up additional parameters suitable for the operating mode selected in step 17.
21. After you configure the drive, proceed to the tuning procedure if operating in “Velocity”, or “Position” mode. “Torque” mode doesn’t require additional tuning or calibration. Refer to Section 6.6 for details on tuning.



6.3 Position Mode Operation (gearing)

In position mode the drive will follow the master reference signals at the P3. 1-4 inputs. The distance the motor shaft rotates per each master pulse is established by the ratio of the master signal pulses to motor encoder pulses (in single loop configuration). The ratio is set by “System to Master ratio” parameter (section 5.3.16).

Example 1

Problem: Setup the drive to follow a master encoder output where 1 revolution of the master encoder results in 1 revolution of the motor

Given: Master encoder: 4000 pulses/revolution (post quadrature)
 Motor encoder: 8000 pulses/revolution (post quadrature)

Solution: Ratio of System (motor encoder) to Master Encoder is $8000/4000 = 2/1$
 Set parameter “System to master ratio” to 2:1

Example 2

Problem: Setup drive so motor can follow a master encoder wheel where 1 revolution of the master encoder results in 3 revolutions of the motor

Given: Motor encoder: 4000 pulses/revolution (post quadrature)
 Master encoder: 1000 pulses/revolution (post quadrature).
 Desired “gear ratio” is 3:1

Solution: Ratio is motor encoder PPR divided by master encoder PPR times the “gear ratio”:
 $(\text{Motor PPR} / \text{Master PPR}) * (3/1) \Rightarrow (4000/1000) * (3/1) \Rightarrow 12/1$
 Set parameter “System to master ratio” to 12:1

6.4 Dual-loop Feedback

In dual-loop operation (position mode only) the relationship between the Master input and mechanical system movement requires that two parameters be set:

(1) “System to master ratio” sets the ratio between the second encoder pulses (system encoder) and the master input pulses.

(2) “Prime to second encoder ratio” sets the ratio between the second and primary (motor) encoder. If the motor is equipped with a resolver connected to the resolver option module, the primary encoder resolution of 65536 (post quadrature) must be used.

When operating in this mode the second encoder input is applied to integral portion of the position compensator. Therefore it is important that the Position I-gain and Position I-limit parameters are set to non 0 values. Always start from very small values of Position I-limit values.



NOTE

When operating with a resolver as the primary feedback, a second encoder can be connected to P4.



Operation

6.5 Enabling the PositionServo

Regardless of the selected operating mode, the PositionServo must be enabled before it can operate. A voltage in the range of 5-24 VDC connected between P3.26 and 3.29 (input IN_A3) is used to enable the drive. There is a difference in the behavior of input IN_A3 depending on how the “Enable switch function” is set.



TIP!

If using the onboard +5VDC power supply for this purpose, wire your switch between pins P3.6 and P3.29. Jumper P3.5 to P3.26. If doing this, all inputs in group A must be powered by P3.6.

When the “Enable switch function” is set to “RUN”:

IN_A3 acts as positive logic ENABLE or negative logic INHIBIT input depending on:

If user program is not running: Activating IN_A3 enables the drive
User program running: Activating IN_A3 acts as negative logic “Inhibit” and operates exactly as if parameter “Enable switch function” set to “Inhibit”

When the “Enable switch function” set to “Inhibit”:

IN_A3 acts as negative logic INHIBIT input regardless of mode or program status.

Activating input IN_A3 doesn’t enables the drive. The drive can be enabled from the user’s program or interface only when IN_A3 is active. Attempt to enable drive by executing the program statement “ENABLE” or from interface will cause the drive to generate a fault #36. Regardless of the mode of operation, if the input is deactivated while the drive is enabled, the drive will be disabled and will generate a fault #36.



WARNING!

Enabling the servo drive allows the motor to operate depending on the reference command. The operator must ensure that the motor and machine are safe to operate prior to enabling the drive and that moving elements are appropriately guarded. Failure to comply could result in damage to equipment and/or injury to personnel!

6.6 Drive Tuning

The PositionServo Drive will likely require some tuning of its gains parameters in order to achieve best performance in the application in which it is being applied. Only when the drive is placed in Torque Mode are the gain values not required to be tuned. The table herein lists the gains parameters that should be adjusted for each of the drive operating modes. These parameters are found within the ‘Compensation’ folder.

MotionView Parameter	Torque Mode	Velocity Mode	Positioning Mode
Velocity P Gain	No	Yes	Yes
Velocity I Gain	No	Yes	Yes
Position P Gain	No	No	Yes
Position I Gain	No	No	Yes
Position D Gain	No	No	Yes
Position I-Limit	No	No	Yes
Gain Scaling	No	Yes	Yes

Before using the tuning procedures detailed in the next sections, ensure that the system is in a safe condition for tuning to be carried out. It is often beneficial to first tune the motor off-load to obtain approximate gains setting before fine tuning in the application. Check that the drive output to the motor is disabled (via Input A3) and that the drive is powered up. Make sure any user program code previously entered into the [Indexer Program] folder in MotionView has been saved prior to tuning so it can be easily recalled after tuning is complete.



WARNING!

During both the Velocity and Position tuning procedures the PositionServo drive will perform rotation (motion) of the motor shaft in the forward and reverse directions at velocities based on the settings made by the user. Ensure that the motor and associated mechanics of the system are safe to operate in the way specified during these procedures.

6.6.1 Tuning the Drive in Velocity Mode

1) Parameter Setup

Set up the motor as per the instructions given in the relevant section of this manual. The motor must be configured correctly prior to tuning taking place.

The parameters Drive Mode, Reference and Enable Switch Function are configured automatically by the velocity tuning program. They are not required to be set at this stage.

2) Importing the Velocity Tuning Program

Before importing the Velocity Tuning Program, the example programs must be installed from the Documentation CD that shipped with the drive. If this has not been done then please do so now.

To load the TuneV program file to the drive, select [Indexer Program] in the MotionView Parameter Tree. Select [Import program from file] on the main toolbar. Navigate to [C:\Program Files\AC Technology\MotionView6.xx\Help\940Examples]. If during the installation of the Documentation CD files a different default directory was selected, then navigate to that directory. Click on the [TuneV.txt] file and select [Open].



3) Editing the Velocity Tuning Program

The Tune Velocity Program creates a step velocity demand in the forward and reverse directions that the drive will attempt to follow (based on its velocity gain settings). The drive will run for a set time in the forward direction and then reverse the reference and run for the same set time in the reverse direction, showing the acceleration, deceleration and steady state performance.

The speed and period (time for one complete cycle - forward and reverse) is set in the Indexer program with the following statements:

```
; Motion Parameters
Define SpeedReference 5 ; speed reference in Rps
Define Period 500 ; time in millisec
```

Adjust these parameters to values suitable to the application in which the drive is used before going to the next step.



Operation

4) Compile and Download Indexer Program to Drive

In the [Indexer program] folder in MotionView, select [Compile and Load with Source] from the pull down menu. The TuneV program will be compiled and sent to the drive. Select [Run] from the pull down menu to run the TuneV program. Do NOT enable the drive (via input A3) at this stage.

5) Oscilloscope Settings

Open the [Tools] folder in MotionView and select the [Oscilloscope] tool. Click the [Set on Top] box to place a checkmark in it and keep the scope on top.

In the Scope Tool Window make the following settings:

Channel 1: Signal = "Commanded Velocity"

Scale = appropriate to "SpeedReference" value set in Indexer Program

Channel 2: Signal = "Motor Velocity"

Scale = appropriate to "SpeedReference" value set in Indexer Program

Timebase: = as appropriate to "Period" value of Indexer Program

Trigger: = Channel 1, Rising Edge

Level: = 10 RPM

For better resolution, adjust these scaling factors during the tuning procedure.

6) Compensation Folder

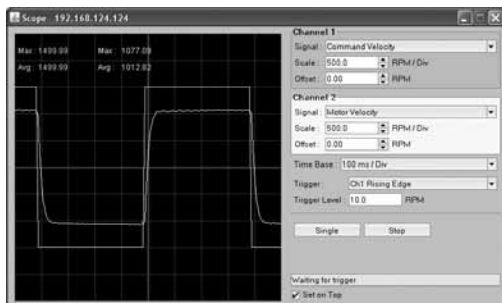
In MotionView, open the [Compensation] folder for the drive. Set [Gain Scaling] to a relatively low value, e.g. -6 for Encoder motor and -8 for a Resolver Motor. Set the [Velocity P-gain] to a mid-value (16000) and set the [Velocity I-Gain] to 0.

7) Gain Tuning

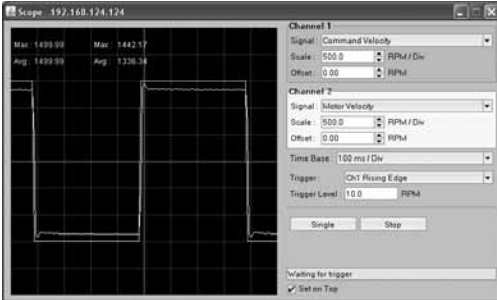
The system should now be ready to start tuning the velocity gains. Start the Oscilloscope by clicking [Run]. Apply the Enable input to Input A3 to enable the drive. At this point of the procedure it is desirable to have little to no motion until we start to increase the gain settings. If the motor vibrates uncontrollably disable the drive, lower the Gain Scaling parameter value and repeat the input enable.

Step 1: Setting the Gain Scaling Parameter

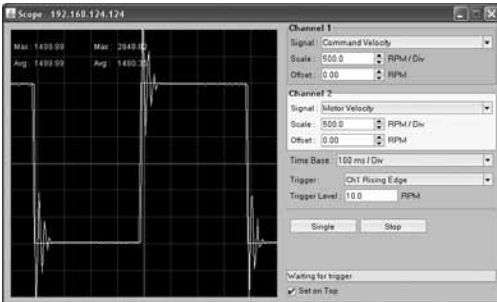
The gain scaling parameter is a 'course adjustment' of the other gain's parameter values. Steadily increase the value of the gain scaling parameter until a reasonable response is obtained from the motor (motor velocity starts to resemble the commanded velocity).



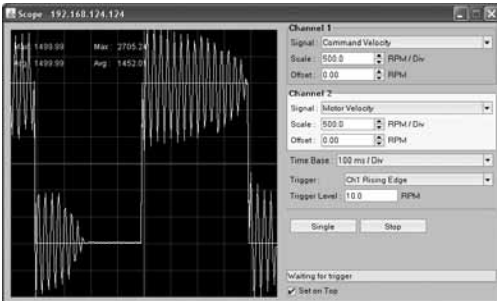
Gain Scaling set too LOW
Motor Velocity significantly different than
Commanded Velocity.



Gain Scaling set OK
Motor Velocity resembles Commanded Velocity. Motor Velocity is reasonably close with a slight overshoot.



Gain Scaling set too HIGH
Motor Velocity shows significant overshoot following the acceleration periods.



Gain Scaling set significantly too HIGH
Motor Velocity exhibits instability throughout the steady state Commanded Velocity.

Depending on the system begin tuned, the motor may go from stable operation (little to no overshoot with stable steady state velocity) to instability (continuous and pronounced oscillations during steady state command) very quickly as gains scaling is increased. The bandwidth for allowing some overshoot with a quick settle time may be very small and may only be achieved through adjustment of the Velocity P-Gain, as described in Step 2. Set the gain scaling parameter to the value preceding that where significant overshoot or continuous instability occurs. With the Gain scaling parameter set move onto tuning the velocity P and I gains.



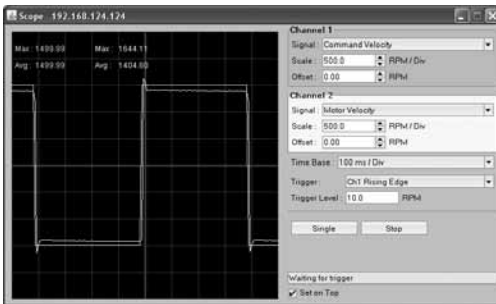
Operation

Step 2: Fine Tuning the Velocity P-Gain

Slowly alter the Velocity P-Gain (increase and decrease) and observe the motor velocity waveform on the oscilloscope. As the P-Gain increases the gradient of the velocity during acceleration and deceleration will also increase as will the final steady state velocity that is achieved. The application of too much P-Gain will eventually result in an overshoot in the motor velocity, and further increases will result in larger overshooting to the point that instability (continuous oscillation) occurs.

Increase the velocity P-gain until some overshoot occurs. Some overshoot is generally ok, and the objective is typically to achieve the shortest possible settle time (steady state velocity). When the system appears to have reached the shortest possible settle time, with acceptable overshoot, cease from increasing the P-Gain.

Scope traces will be similar to those shown in Step 1, however the P-gain will now be given a more precise adjustment in order to obtain the best possible tuning.

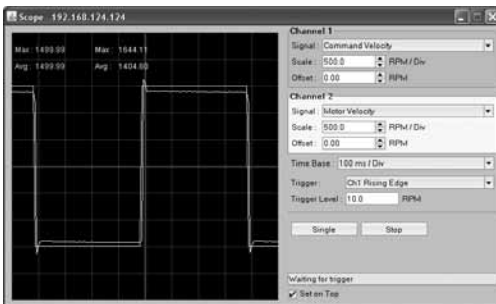


Good Fine Tuning of the P-Gain
Small overshoot with excellent settle time and steady state velocity regulation.

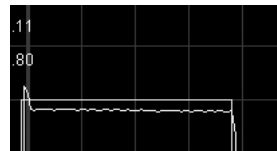
Step 3: Setting the Velocity I-Gain

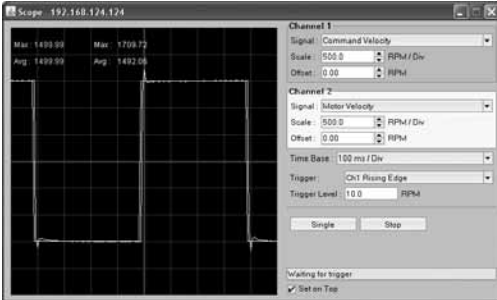
The purpose of the velocity I-gain is to correct any error that is present between the commanded velocity and the steady state velocity that could not be rectified by adjustment of the velocity P-gain. Adjustment of the velocity I-gain can also reduce the steady state ripple that may occur in the velocity waveform. Lastly, velocity I-gain has a positive effect on the holding torque produced by the motor.

Slowly increase the "Velocity I-Gain" and check for correction of the steady state error in the velocity waveform. Continuing to increase the velocity I-gain will eventually result in increased overshoot and instability in the motor velocity waveform. Stop increasing the I-Gain when additional overshoot or instability starts to occur.

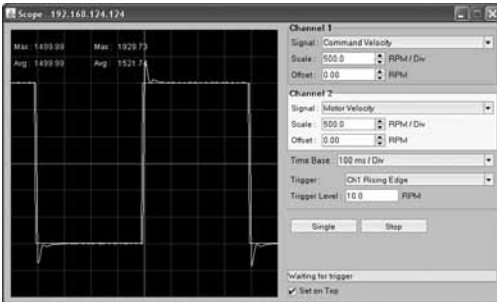
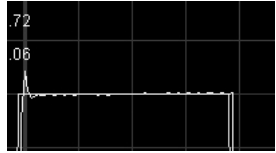


I-Gain set too LOW
Error exists between Commanded steady state velocity and Actual steady state velocity

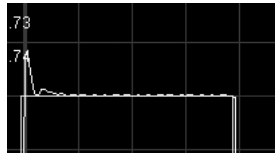




I-Gain set OK
No error between Commanded steady state velocity and Actual steady state velocity with excellent stability.



I-Gain set too HIGH
Additional overshoot and oscillations are starting to occur. Steady state velocity regulation



Step 4: Check Motor Currents

Finally check the motor currents on the Oscilloscope. Make the following settings to the oscilloscope.

Channel 1:

Signal = "Phase Current RMS"

Scale = as appropriate to peak current limit set in drive parameters (MotionView)

Timebase: = as appropriate to "Period" value of Indexer Program

Trigger: = Channel 2, Rising Edge

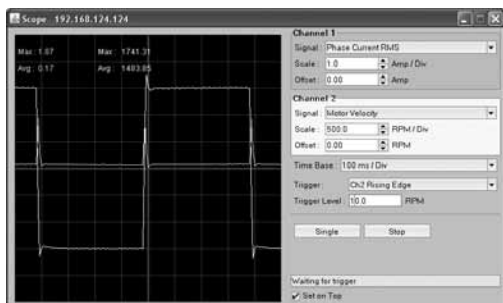
Level: = 10 RPM

Observe the waveforms to insure there are no significant oscillations. Reduce the gains values if necessary.

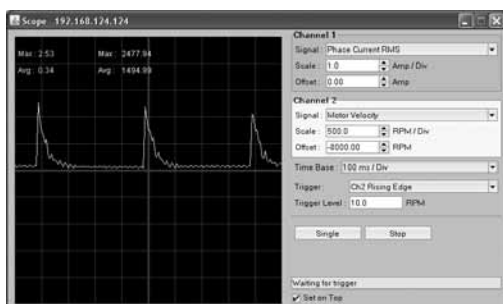
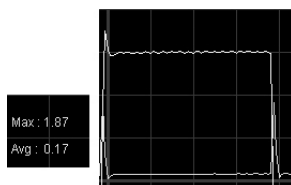
The current waveform should be showing spikes of current during acceleration / deceleration and steady state current during any steady state velocity. The maximum value (peak value) of the current waveform is shown at the top of the oscilloscope screen. This maximum value can be compared to the drive nominal current and peak current settings to check how much of the motors potential performance is being used and if optimum performance is being achieved.



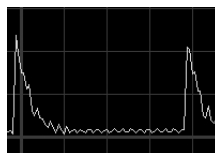
Operation



Good Current Trace
Uniform current pulses during accel/
deceleration and stable current during steady
state velocity.



Instability in Drive Output Current
(Note: Channel 2 trace has been removed for
clarity).



8) End Velocity Tuning

Remove the Enable Input from input A3 (disable the drive). In MotionView, click on the [Indexer] folder for the drive. Click [Reset] on the program toolbar. If the drive is to be run in just velocity mode then tuning is now complete. If the drive is to be used in Positioning mode continue with 'Tuning the Drive in Position Mode', section 6.6.2.

6.6.2 Tuning the Drive in Position Mode

Velocity Tuning should be carried out prior to the tuning of the position loop. Refer to the Velocity Tuning section, 6.6.1.

1) Parameter Set up

In MotionView, open the [Limits] folder and then the [Position Limits] sub-folder. Set the [Position Error] and [Max Error Time] parameters to their maximum values to effectively disable the position error trip while tuning takes place. Ensure the system is safe to operate in this manner.

Position Error = 32767

Max Error Time = 8000

The Drive Mode, Reference and Enable Switch Function parameters are automatically configured by the velocity tuning program. They do not require setting at this stage.



2) Importing the Position Tuning Program

Before importing the Position Tuning Program, the example programs must be installed from the Documentation CD that shipped with the drive. If this has not been done then please do so now.

To load the TuneP program file to the drive, select [Indexer Program] in MotionView. Select [Import program from file] on the main toolbar. Navigate to [C:\Program Files\AC Technology\MotionView6.x\Help\940Examples]. If during the installation of the Documentation CD files a different default directory was selected, then navigate to that directory. Click on the [TuneP.txt] file and select [Open].



3) Editing the Position Tuning Program

The Tune Position Program performs trapezoidal moves in the forward and reverse direction separated by a defined pause (or time delay).

The Accel, Decel, and MaxV variables within the TuneP program define the ramps and steady state velocity that will be used to execute the motion commands.

ACCEL = 500	;500 rps*s	Accel = Acceleration speed
DECEL = 500	;500 rps*s	Decel = Deceleration speed
MAXV = 20	;20 Rps	MaxV = Maximum

The size of each move and the pause between the moves is defined in the following lines of code. There are two moves and pauses for the forward and reverse moves to be performed.

MOVED 0.25	;move 1 rev	MoveD = Move distance
wait time 200	;wait time to analyze 'standstill' stability	wait time = Delay period
MOVED -0.25	;move opposite direction 1 rev	
wait time 200	;wait time to analyze 'standstill' stability	

Adjust these parameters if required to best suit the application before going to the next step.

4) Compile and Download Indexer Program to Drive

In the [Indexer Program] folder in MotionView, select [Compile and Load with Source] from the pull down menu. The TuneP program will be compiled and sent to the drive. Select [Run] from the pull down menu to run the TuneP program. Do NOT enable the drive (via input A3) at this stage.



Operation

5) Oscilloscope Settings

Open the [Tools] folder in MotionView and select the [Oscilloscope] tool. Click the [Set on Top] box to place a checkmark in it and keep the scope on top.

In the Scope Tool Window, make the following settings:

Channel 1:

Signal = "Position Error"

Scale = as appropriate to the Error that results once the TuneP program is run.

Channel 2:

Signal = "Target Position"

Scale = as appropriate to the position move generated by the TuneP program

Timebase: = as appropriate to the "Period" of the moves being generated.

Trigger: = Channel 1, Rising Edge.

Level: = 10 Pulses

6) Compensation Folder

Open the [Compensation] folder in MotionView.

Leave the Velocity P-Gain and Velocity I Gain unchanged, as they should already have been setup during velocity tuning. Do not adjust the Gain Scaling Parameter during this procedure.

Set the [Position P-gain] to a low value (e.g. 100) and set the [Position I-Gain] and [Position D-Gain] to 0.

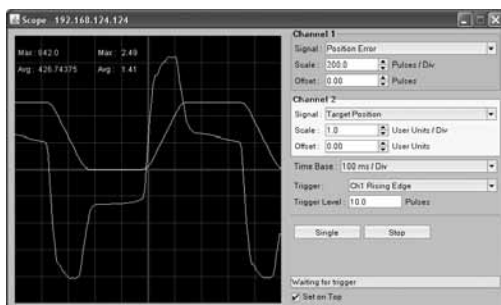
7) Gain Tuning

The system should now be ready to start tuning the position loop. Start the Oscilloscope by clicking [Run]. Apply the Enable input A3 to enable the drive.

The general goal in tuning the position loop is to achieve the minimum position error while maintaining system stability. Some experimentation with gain values will be required to achieve the best performance for the application.

Step 1: Setting the Position P-Gain

Slowly increase the Position P-Gain while watching the position error waveform on oscilloscope Channel 1. It is important to watch both the Max Error as well as the Average Error. While increasing Position P-gain, it should be apparent that both the Max Error as well as the Average Error decrease.



Position P-Gain set too LOW
Large Position Error occurring and large error
in final positioning achieved

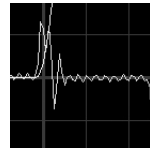


Increased Position P-Gain
Shows improvement to the maximum error
and the final positioning accuracy

At some point while increasing the P-Gain, additional oscillations (Average Error) will start to appear on the position error waveform.



Further Increased Position P-Gain
Shows very good reduction to the maximum
error but with additional oscillations starting
to occur.

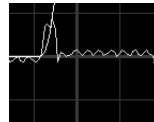


Step 2: Setting the Position D-Gain

Slowly increase the D-Gain while watching the position error waveform on oscilloscope Channel 1. As the D-Gain is increased, the position error oscillation caused by the P-Gain, should start to decrease. Continue to increase the D-Gain until oscillation is gone or until D-Gain is no longer having any apparent effect.



Adjustment of Position D-Gain
in conjunction with the P-Gain dampens
out additional oscillations while improving
position error.



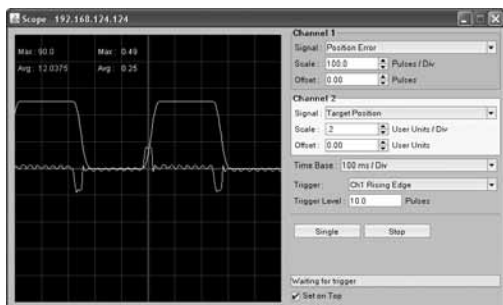
For optimum tuning, it is sometimes required to repeat the process of increasing the P-Gain until a slight oscillation occurs and then increase the D-Gain to suppress that oscillation. This procedure can be repeated until the increasing of D-Gain has negligible effect on the position error waveform.



Operation

Step 3: Setting the Position I-Gain and Position I-Gain Limit

The objective here is to minimize the position error during steady state operation and improve positioning accuracy. Start to increase the Position I-gain. Increasing the I-gain will increase the drive's reaction time while the I-Limit will set the maximum influence that the I-Gain can have on the Integral loop. When adjusting the I-gain start with a very small value for the I-gain (e.g. 1) then increase the I-gain parameter value until stand-still error is compensated and positioning accuracy is satisfactory. Remember that large values of Position I-limit can cause a large instability in the control loop and unsettled oscillation of the system mechanics.



Position Error trace following the tuning of Position P-, I- and D-Gains

Step 4: Check Motor Currents

Set the oscilloscope channel 2 to 'Phase Current RMS'

Channel 2:

Signal = "Phase Current RMS"

Scale = as appropriate to peak current limit set in drive parameters (MotionView)

Timebase: = as appropriate to the "Period" of the moves being generated

Trigger: = Ch1 Rising Edge

Level: = 10 Pulses

Observe the Current waveform to make sure that there are no significant oscillations during the steady state sections of the position profile (times when target position is not changing). If so then decrease the gains values until the oscillations are either removed or reduced to an acceptable level.



Minimal oscillation when motor positioned to target position.





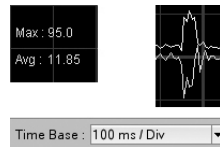
8) Setting the Position Error Limits

Look at the position error waveform on the oscilloscope. Note the maximum time that position errors exist (from the time axis of the scope) and the maximum peak errors being seen (from the value at the top of the screen). Use these values to set the position error limits to provide suitable position error protection for the application.

Open the 'Limits' folder and 'Position Limits' sub-folder within the MotionView node tree and set suitable values for the 'Position Error' and 'Max Error Time' parameters.



Maximum error and time period for error existing.



In this particular example maximum error in pulses is 95.0. The time this peak error occurs can be read from the oscilloscope at approximately $\frac{1}{2}$ of a division with each division equal to 100ms, hence the error pulse lasts approximately 50ms. Suitable settings for position error within this application might be as follows, although looser or tighter limits could be applied depending on the requirements of the application.

Description	Value
Position Error	100
Max Error Time	50

9) End Tuning

Remove the Enable Input from input A3 (disable the drive).

Click on the [Indexer Program] folder in MotionView. Click the [Reset] button at the top of the indexer programming screen.

Tuning is now complete.



7 Quick Start Reference

This section provides instructions for External Control, Minimum Connections and Parameter Settings to quickly setup a PositionServo drive for External Torque, Velocity or Positioning Modes. The sections are NOT a substitute for reading the entire PositionServo User Manual. Observe all safety notices in this manual.

7.1 Quick Start - External Torque Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector - Pin	Input Name	Description
P3-22	ACOM	Analog Common Reference from Controller
P3-24	AIN1+	Analog Torque Reference from Controller – Positive
P3-25	AIN1-	Analog Torque Reference from Controller – Negative
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (If required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z-	Buffered Encoder Output
P3-23	AO	Analog Output
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter

Mandatory Parameter Settings:

These Parameters are required to be set prior to running the drive

Folder / Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Torque]
	Reference	Set to [External]
IO / Analog IO	Analog Input (Current Scale)	Set to required current per 1V input from controller
	Analog Input Dead band	Set zero torque Dead band in mV
	Analog Input Offset	Set Analog Offset for Torque Reference
IO / Digital IO	Enable Switch Function	Set to [Run]



Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of a Zero Speed Output
	At Speed	Set Target Speed for activation of a At Speed Output
	Speed Window	Set bandwidth for activation of a At Speed Output

7.2 Quick Start - External Velocity Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector - Pin	Input Name	Description
P3-22	ACOM	Analog Common Reference from Controller
P3-24	AIN1+	Analog Velocity Reference from Controller – Positive
P3-25	AIN1-	Analog Velocity Reference from Controller – Negative
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (If required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z-	Buffered Encoder Output
P3-23	A0	Analog Output
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter



Reference

Mandatory Parameter Settings:

These parameters are required to be set prior to running the drive.

Folder/Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Velocity]
	Reference	Set to [External]
	Enable Velocity Accel / Decel Limits	Enable Ramp rates for Velocity Mode
	Velocity Accel Limit	Set required Acceleration Limit for Velocity command
	Velocity Decel Limit	Set required Deceleration Limit for Velocity command
IO / Analog IO	Analog Input (Velocity Scale)	Set to required velocity per 1 volt input from controller
	Analog Input Dead band	Set zero velocity Dead band in mV
	Analog Input Offset	Set Analog Offset for velocity Reference
IO / Digital IO	Enable Switch Function	Set to [Run]
Compensation (see tuning section)	Velocity P-Gain	Set P-Gain for Velocity loop
	Velocity I_Gain	Set I-Gain for Velocity loop
	Gain Scaling	Set Gain Scaling Parameter

Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of Zero Speed Output
	At Speed	Set Target Speed for activation of At Speed Output
	Speed Window	Set bandwidth for activation of At Speed Output



7.3 Quick Start - External Positioning Mode

Mandatory Signals:

These signals are required in order to achieve motion from the motor.

Connector-Pin	Input Name	Description
P3-1	MA+	Position Reference Input for Master Encoder / Step-Direction Input
P3-2	MA-	Position Reference Input for Master Encoder / Step-Direction Input
P3-3	MB+	Position Reference Input for Master Encoder / Step-Direction Input
P3-4	MB-	Position Reference Input for Master Encoder / Step-Direction Input
P3-26	IN_A_COM	Common Input for Enable Input
P3-29	IN_A3	Enable Input to Controller or switch

Optional Signals:

These signals may be required dependant on the control system being implemented.

Connector - Pin	Input Name	Description
P3-6	+5V	+5V Output for Enable Input (If required)
P3-7	A+	Buffered Encoder Output
P3-8	A-	Buffered Encoder Output
P3-9	B+	Buffered Encoder Output
P3-10	B-	Buffered Encoder Output
P3-11	Z+	Buffered Encoder Output
P3-12	Z-	Buffered Encoder Output
P3-22	ACOM	Analog Common Reference from Controller
P3-23	AO	Analog Output
P3-27	IN_A1	Positive Limit Switch: Required if Limit Switch Function is used
P3-28	IN_A2	Negative Limit Switch: Required if Limit Switch Function is used
P3-41	RDY+	Ready output Collector
P3-42	RDY-	Ready output Emitter
P3-43	OUT1-C	Programmable output #1 Collector
P3-44	OUT1-E	Programmable output #1 Emitter
P3-45	OUT2-C	Programmable output #2 Collector
P3-46	OUT2-E	Programmable output #1 Emitter
P3-47	OUT3-C	Programmable output #3 Collector
P3-48	OUT3-E	Programmable output #1 Emitter
P3-49	OUT4-C	Programmable output #4 Collector
P3-50	OUT4-E	Programmable output #1 Emitter



Reference

Mandatory Parameter Settings:

These parameters are required to be set prior to running the drive.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Drive Mode	Set to [Position]
	Reference	Set to [External]
	Step Input Type	Set to [S/D] or [Master Encoder]. (S/D = Step + Direction)
	System to Master Ratio	Set 'Master' and 'Slave' values to gear position input pulses to pulse revolution of the motor shaft
IO / Digital IO	Enable Switch Function	Set to [Run]
Limits / Position Limits	Position Error	Set Position Error Limit specific to application
	Max Error Time	Set Position Error Time specific to application
Compensation (see tuning section)	Velocity P-Gain	Set P-Gain for Velocity loop
	Velocity I_Gain	Set I-Gain for Velocity loop
	Position P-Gain	Set P-Gain for Position Loop
	Position I-Gain	Set I-Gain for Position Loop
	Position D-Gain	Set D-Gain for Position Loop
	Position I-Limit	Set I-Limit for Position Loop
	Gain Scaling	Set Gain Scaling Parameter

Optional Parameter Settings:

These parameters may require setting depending on the control system implemented.

Folder / Sub-Folder	Parameter Name	Description
Parameters	Resolver Track	PPR for simulated encoder on 941 Resolver drive
IO / Digital IO	Output 1 Function	Set to any pre-defined function required
	Output 2 Function	Set to any pre-defined function required
	Output 3 Function	Set to any pre-defined function required
	Output 4 Function	Set to any pre-defined function required
	Hard Limit Switch Actions	Set if Hard Limit Switches used in Application
IO / Analog IO	Adjust Analog Input	Tool that can be used to learn analog input level
	Analog Output	Set to any pre-defined function required
	Analog Output Current Scale	Set to scale analog output if current value is selected
	Analog Output Velocity Scale	Set to scale analog output if velocity value is selected
Limits / Velocity Limits	Zero Speed	Set bandwidth for activation of a Zero Speed Output
	At Speed	Set Target Speed for activation of a At Speed Output
	Speed Window	Set bandwidth for activation of a At Speed Output



8 Diagnostics

8.1 Display

The PositionServo 940 drives are equipped with a diagnostic LED display and 3 push buttons to select displayed information and to edit a limited set of parameter values.

Parameters can be scrolled by using the "UP" and "DOWN" (▲▼) buttons. To view a value, press "Enter" (↵). To return back to scroll mode press "Enter" again.

After pressing the "Enter" button on editable parameters, the yellow LED "C" (see figure in the next section) will blink indicating that parameter value can be changed. Use "UP" and "DOWN" buttons to change the value. Press "Enter" to store new setting and return back to scroll mode.

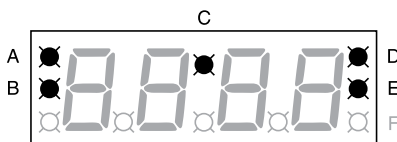
Display	Description
StAt	current drive status - ↵ to view: rUn - drive running dIS - drive disabled F_XX - drive fault. Where XX is the fault code (section 8.3.1)
Hx.xx	Hardware revision (e.g. H2.00)
Fx.xx	Firmware revision (e.g. F2.06)
bAud	RS232/RS485(normal mode) baud rate - ↵ to set ▲▼ selects from 2400 to 115200 baudrates
Adr	Drive's address - ↵ to set ▲▼ sets 0 - 31 drive's address
FLE5	Stored fault's history - ↵ to view ▲▼ scroll through stored faults F0XX to F7XX, where XX is the fault code (section 7.3.1)
Ht	Heatsink temperature - ↵ to view Shows heatsink temperature in °C if greater than 40°C. Otherwise shows "LO" (low).
EnC	Encoder activity - ↵ to view Shows primary encoder counts for encoder diagnostics activity
HALL	Displays motor's hall sensor states - ↵ to view Shows motor hall states in form XXX, where X is 1 or 0 - sensor logic states.
bUS	Displays drive DC bus voltage - ↵ to view Shows DC bus voltage value
Cur	Displays motor's phase current (RMS) Shows current value if drive is enabled, otherwise shows "dIS"



Diagnostics

8.2 LEDs

The PositionServo has five diagnostic LEDs mounted on the periphery of the front panel display as shown in the drawing below. These LEDs are designed to help monitor system status and activity as well as troubleshoot any faults.



S913

LED	Function	Description
A	Enable	Orange LED indicates that the drive is ENABLED (running).
B	Regen	Yellow LED indicates the drive is in regeneration mode.
C	Data Entry	Yellow LED will flash when changing.
D	Comm Fault	Red LED illuminates upon a communication fault. (available in CANbus only)
E	Comm Activity	Green LED flashes to indicate communication activity.

8.3 Faults

8.3.1 Fault Codes

Listed herein are fault codes caused mostly by hardware operations. Additional fault codes are listed in the PositionServo Programmer's manual.

Fault Code (Display)	Fault	Description
F_0U	Over voltage	Drive bus voltage reached the maximum level, typically due to motor regeneration
F_Fb	Feedback error	Invalid Hall sensors code; Resolver signal lost or at least one motor hall sensor is inoperable or not connected.
F_0C	Over current	Drive exceeded peak current limit. Software incapable of regulating current within 15% for more than 20mS. Usually results in wrong motor data or poor tuning.
F_0t	Over temperature	Drive heatsink temperature has reached maximum rating. Trip Point = 100°C for all drives except 480V 6A & 9A drives Trip Point = 108°C for 480V 6A & 9A drives
F_0S	Over speed	Motor has reached velocity above its specified limit
F_PE	Position Error Excess	Position error has exceeded maximum value.
F_bd	Bad motor data	Motor profile data is invalid or no motor is selected.
F_EP	EPM failure	EPM failure on power up
-EP-	EPM missing	EPM not recognized (connected) on power up
F_09	Motor over temperature	Motor over temperature switch activated; Optional motor temperature sensor (PTC) indicates that the motor windings have reached maximum temperature
F_10	Subprocessor failure	Error in data exchange between processors. Usually occurs when EMI level is high due to poor shielding and grounding.
F_14	Under voltage	Occurs when the bus voltage level drops below 50% of nominal bus voltage while drive is operating. An attempt to enable the drive with low bus voltage will also result in this fault
F_15	Hardware overload protection	Occurs when the phase current becomes higher than 400% of total drive's current capability for more than 5µs.
F_1B	Arithmetic Error Division by zero	Statement executed within the Indexer Program results in a division by 0 being performed. Drive programming error (error in drive source code).



Fault Code (Display)	Fault	Description
F_19	Arithmetic Error Register overflow	Statement executed within the Indexer Program results in a value being generated that is too big to be stored in the requested register. Drive programming error (error in drive source code).
F_20	Subroutine stack overflow	Exceeded 32 levels subroutines stack depth. Caused by executing excessive subroutine calls without a RETURN statement. Drive programming error (error in drive source code).
F_21	Subroutine stack underflow	Executing RETURN statement without preceding call to subroutine. Drive programming error (error in drive source code).
F_22	Arithmetic stack overflow	Variable evaluation stack overflow. Expression too complicated for compiler to process. Drive programming error (error in drive source code).
F_23	Motion Queue overflow	32 levels depth exceeded. Drive programming error (error in drive source code).
F_24	Motion Queue underflow	Relates to the MDV statements in the Indexer Program. Drive programming error (error in drive source code).
F_25	Unknown opcode	Byte code interpreter error; May occur when program is missing the closing END statement; when subroutine has no RETURN statement; or if data in EPM is corrupted at run-time
F_26	Unknown byte code	Byte code interpreter error; May occur when program is missing the closing END statement; when subroutine has no RETURN statement; or if data in EPM is corrupted at run-time
F_27	Drive disabled	Attempt to execute motion while drive is disabled. Drive programming error (error in drive source code).
F_28	Accel too high	Motion statement parameters calculate an Accel value above the system capability. Drive programming error (error in drive source code).
F_29	Accel too low	Motion statement parameters calculate an Accel value below the system capability. Drive programming error (error in drive source code).
F_30	Velocity too high	Motion statement parameters calculate a velocity above the system capability. Drive programming error (error in drive source code).
F_31	Velocity too low	Motion statement parameters calculate a velocity below the system capability. Drive programming error (error in drive source code).
F_32	Positive Limit Switch	Positive limit switch is activated.
F_33	Negative Limit Switch	Negative limit switch is activated.
F_34	Positive motion w/ Pos Lim Sw ON	Attempt at positive motion with engaged positive limit switch (Only available while drive is in position mode)
F_35	Negative motion w/ Neg Lim Sw ON	Attempt at negative motion with engaged negative limit switch (Only available while drive is in position mode)
F_36	Drive Disabled by User at Enable Input	The drive is disabled while operating or an attempt is made to enable the drive without deactivating "Inhibit input". "Inhibit" input has reverse polarity
F_39	Positive soft limit reached	Programmed (Soft) absolute limits reached during motion
F_40	Negative soft limit reached	Programmed (Soft) absolute limits reached during motion
F_41	Unknown Variable ID	Attempt to use variable with unknown ID from user program. Drive programming error (error in drive source code).
F_45	2nd Encoder Position Error	Second encoder position error has exceeded maximum value



Diagnosics

8.3.2 Fault Event

When drive encounters any fault, the following events occur:

- Drive is disabled
- Internal status is set to “Fault”
- Fault number is logged in the drive’s internal memory for later interrogation
- Digital output(s), if configured for “Run Time Fault”, are asserted
- Digital output(s), if configured for READY, are de asserted
- If the display is in the default status mode, the LEDs display F_XX where XX is current fault code.
- “Enable” LED turns OFF

8.3.3 Fault Reset

Fault reset is accomplished by disabling or re-enabling the drive depending on the setting of the “Reset option” parameter (section 5.3.10).

8.4 Troubleshooting



DANGER!

Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait at least 60 seconds before servicing drive. Capacitors retain charge after power is removed.

Before troubleshooting

Perform the following steps before starting any procedure in this section:

- Disconnect AC or DC voltage input from the PositionServo. Wait at least 60 seconds for the power to discharge.
- Check the PositionServo closely for damaged components.
- Check that no foreign material has become lodged on, or fallen into, the PositionServo.
- Verify that every connection is correct and in good condition.
- Verify that there are no short circuits or grounded connections.
- Check that the drive’s rated phase current and RMS voltage are consistent with the motor ratings.

For additional assistance, contact your local PositionServo® authorized distributor.

Problem	External line fuse blows
Possible Cause	Line fuses are the wrong size Motor leads or incoming power leads are shorted to ground. <i>Nuisance tripping caused by EMI noise spikes caused by poor grounding and/or shielding.</i>
Suggested Solution	<ul style="list-style-type: none"> • Check that line fuses are properly sized for the motor being used. • Check motor cable and incoming power for shorts. • Check that you follow recommendation for shielding and grounding listed in section 3.2 in this manual.



Problem	Ready LED is on but motor does not run
Suggested Solution	<p>If in Torque or Velocity mode: Reference voltage input signal is not applied. Reference signal is not connected to the PositionServo input properly; connections are open. In MotionView program check <Parameters> <Reference> set to <External></p> <p>For Velocity mode only: In MotionView check <Parameters> <Compensation><Velocity loop filter> P-gain must be set to value more then 0 in order to run. Without load motor will run with P-gain set as low as 20 but under load might not. If P-gain is set to 0 motor will not run at all.</p> <p>In Position mode with master encoder motion source (no program) Reference voltage input signal source is not properly selected. In MotionView program check <Parameters> <Reference> set to <External></p> <p>In Position mode using indexing program Variables ACCEL, DECEL,MAXV, UNITS are not set or set to 0. Before attempting the move set values of motion parameters ACCEL, DECEL,MAXV, UNITS</p>
Problem	In velocity mode, the motor runs away
Possible Cause	<ul style="list-style-type: none"> • Hall sensors or encoder mis-wired. • PositionServo not programmed for motor connected.
Suggested Solution	<ul style="list-style-type: none"> • Check Hall sensor and encoder connections. • Check that the proper motor is selected..

Notes

Notes

Lenze AC Tech Corporation
630 Douglas Street • Uxbridge, MA 01569 • USA
Sales: (800) 217-9100 • Service: (508) 278-9100
www.lenze-actech.com

(S94P01B2)

POSITIONServo



MotionView
Configuration and Programming
Software
USER'S MANUAL

Table of Contents

1	MOTIONVIEW SOFTWARE OVERVIEW	3
1.1	Installation and Package Revision	3
1.2	Main Screen	4
1.2.1	Node Tree	4
1.2.2	List View	4
1.2.3	Message Window	4
1.2.4	How to Change Parameters	5
1.2.5	Main Menu and Toolbar	6
1.3	Managing Projects	7
1.4	Connecting to the Drive	7
1.4.1	Connection using PPP over RS-232/RS485	8
1.4.2	Connection using 10/100 Ethernet	8
1.4.3	Disconnect or Remove a Drive	11
1.5	Build RS-485 Connection List	12
1.6	Build Ethernet Connection List	12
1.7	File Operations	13
1.7.1	Opening and Closing Parameter Files	13
1.7.2	Load Parameters from File to Drive	13
2	NODE TREE FOLDERS	13
2.1	Drive	13
2.2	Motor	14
2.3	Parameters	14
2.4	Communication	15
2.4.1	Ethernet	15
2.4.2	RS485 and Modbus	16
2.4.3	CAN	16
2.5	I/O	16
2.5.1	Digital I/O	16
2.5.2	Analog I/O	17
2.6	Limits	17
2.6.1	Velocity Limits	17
2.6.2	Position Limits	17
2.7	Compensation	17
2.8	Indexer Program	18
2.9	Tools	18
2.9.1	Oscilloscope	18
2.9.2	Run Panels	20
2.9.3	Diagnostic	20
2.10	Faults	20
2.11	Documents	20

Safety Warnings



WARNING!

- Hazard of unexpected motor starting! When using MotionView software, or otherwise operating the PositionServo drive over RS-232/485, CANopen or Ethernet, the motor may start unexpectedly, which may result in damage to equipment and/or injury to personnel. Make sure the equipment is free to operate in this manner, and that all guards and covers are in place to protect personnel.



DANGER!

- Hazard of electrical shock! Circuit potentials are at 115 VAC, 230 VAC or 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.



NOTE

- The symbol shown at left indicates additional information, shortcuts, or tips that do not affect the safe operation of the drive.

1 MotionView Software Overview

MotionView is the setup and management tool for SimpleServo and PositionServo Drives. The user interface is intuitive in the way information is arranged and is logically divided into groups for viewing and editing. This manual covers the concept and basic operations of the MotionView program, please refer to the corresponding product's User and Programmer Manuals for further details on MotionView features and capabilities.

1.1 Installation and Package Revision

MotionView software can be installed on Windows, Windows XP system. To locate the package revision check the MotionView CD label or open the [Help] folder then the [About MotionView] folder. Each time a file is revised on the MotionView CD, the package revision is increased even though the MotionView revision is not changed. As illustrated in Figure 1, MotionView revision (6.04), motor database revision (3.01) and package revision (MV94CD14) can be found by clicking [About MotionView].

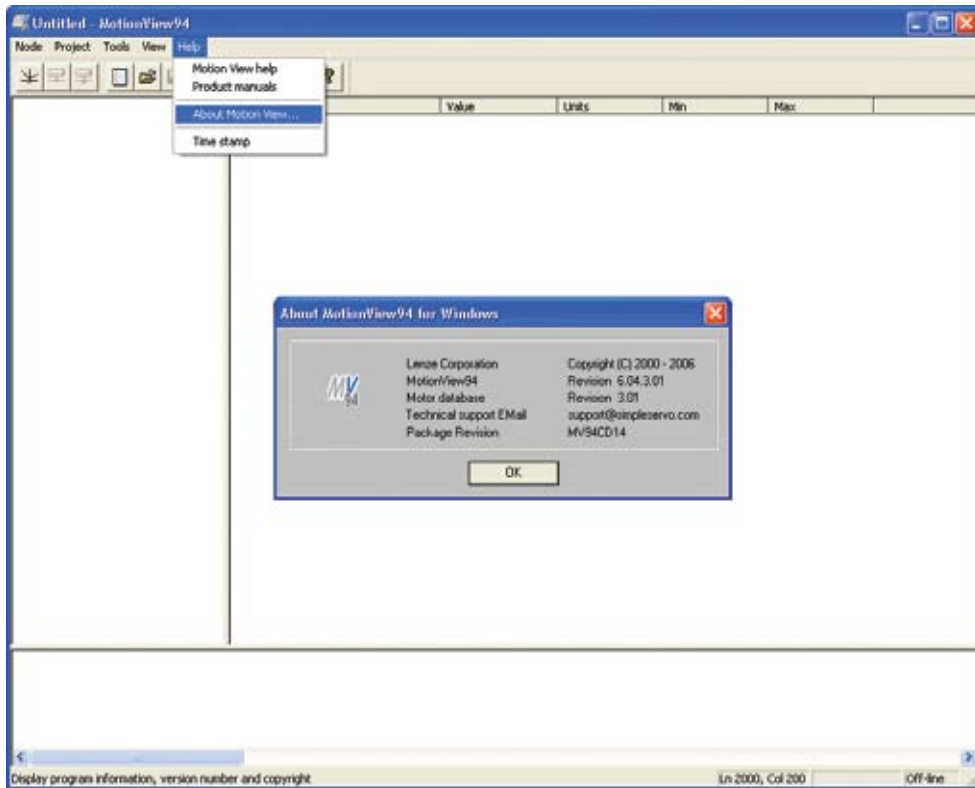


Figure 1: About MotionView

To obtain the latest revision of the MotionView software, visit the Technical Library at <http://www.actech.com>.

1.2 Main Screen

The user interface or Motion View main screen consists of 3 main panels: the Node Tree, the List View, and the Message Window as illustrated in Figure 2.

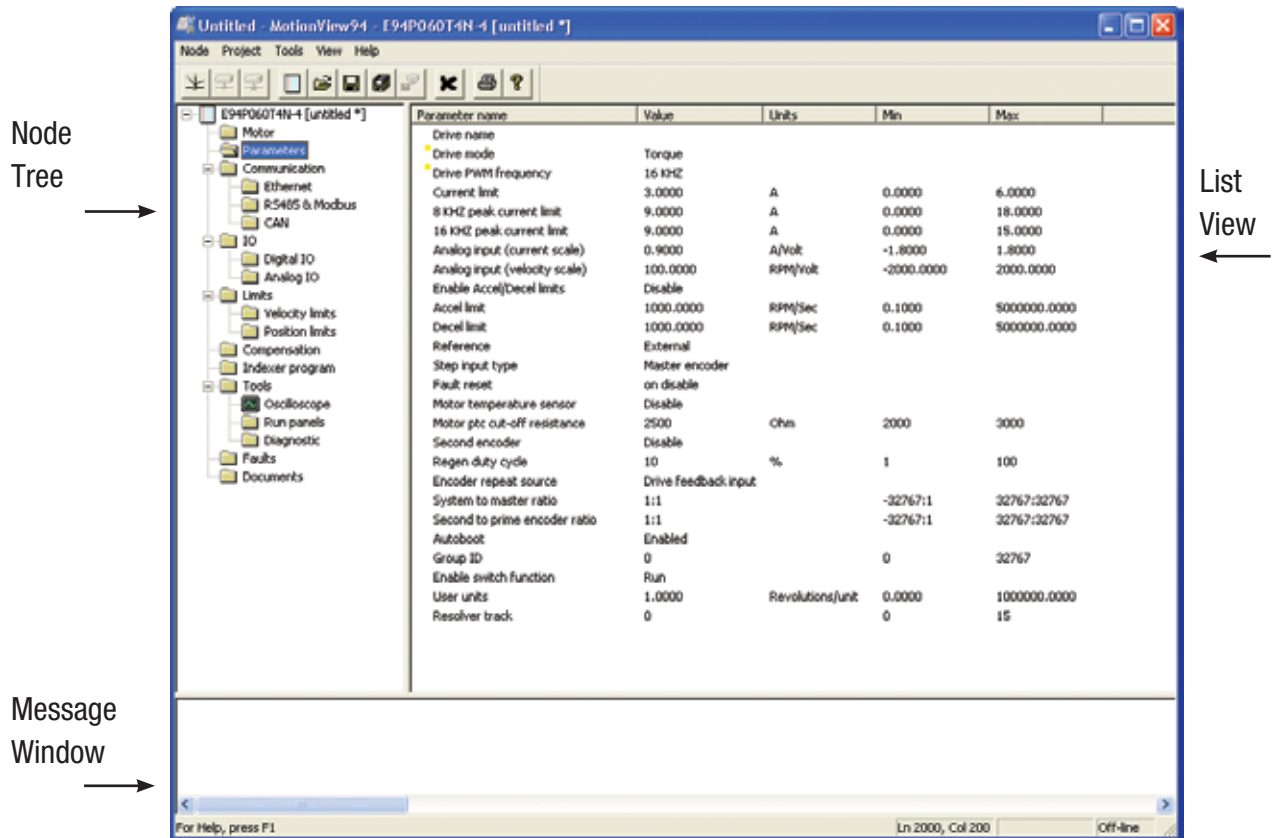


Figure 2: MotionView Screen

1.2.1 Node Tree

Drives and Parameter files appear in the Node Tree on the left hand side of the screen. The Drive and Parameter files contain sub folders (denoted by a + symbol) with parameter groups and different tools needed to work with the selected Drive (Parameter file). Drive and Parameter Files appear almost identical in the Node Tree and both operate in the same way. The main difference is that Drive files can have a connection and parameter files cannot. To expand a folder to view its subfolders double-click on the [+] symbol next to it. To collapse a folder double-click on the [-] symbol next to it.

1.2.2 List View

The right panel of the MotionView main screen is called the List View. When you click on a Node Tree file or sub-folder, the parameters belonging to this group are displayed in the List View. Every parameter can be viewed in detail; its current value, units and min and max values. When one navigates through the node tree, information in the list view changes automatically in sync with the node tree selection.

1.2.3 Message Window

The bottom or footer section of the MotionView main screen is called the Message Window. The Message Window gives information about communication status and supplies various information to make troubleshooting easier. To clear the message in the Message Window, double-click the message and then right click to reveal the [Clear] button. Click [Clear].

1.2.4 How to Change Parameters

To change any parameter, click on the parameter of interest on the list view. The dialog box opens and the user can then change the value. There are several different types of dialog boxes depending on the parameter being changed:

- A Numeric value parameter with an Entry Dialog Box.
- A Numeric value parameter with an Entry Dialog Box containing a Slider.
- A Selection type parameter with a Predefined Value(s) Dialog Box.

The Entry Dialog Box requires the user to input (type) the value of the parameter. The Entry Dialog Box with Slider permits the user to increase/decrease the value of the parameter by clicking on the slider. A Predefined Value Dialog Box contains a pull-down menu from which the user can select the appropriate value for his parameter setup.

All dialog boxes contain a set of [Apply] [OK] and [Cancel] buttons. Use the [Apply] button to accept the value changes but leave the dialog box open. Use the [OK] button to accept the changes and dismiss the dialog box. Use the [Cancel] button to dismiss the dialog box and make no changes to the parameter setup.

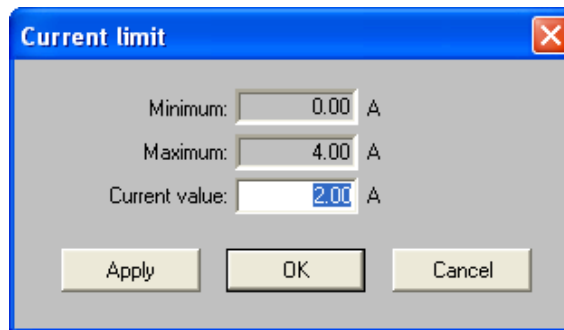


Figure 3: Numeric value parameter with an Entry Dialog Box

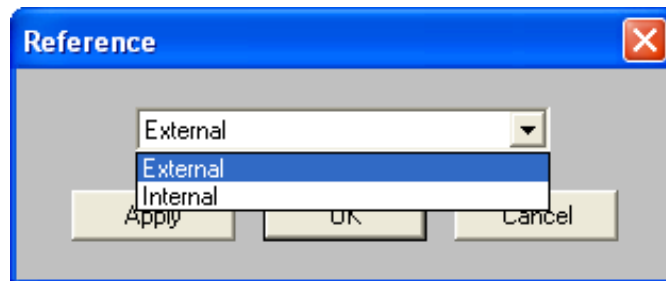


Figure 4: Numeric Entry Dialog Box with Pull-Down Predefined Values

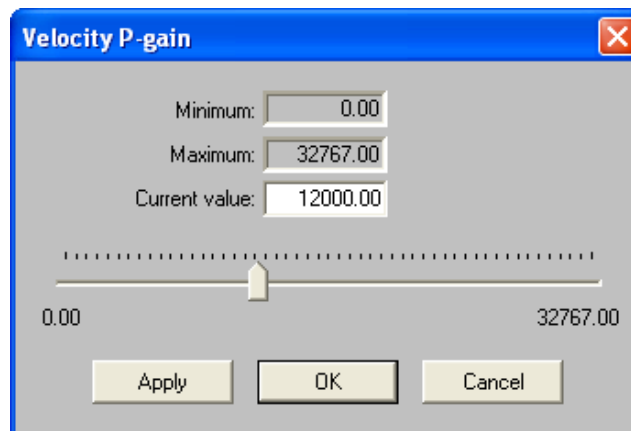


Figure 5: Numeric Entry Dialog Box containing a Slider

Some groups from the Node Tree have Action buttons in the List View. They will perform the action listed against the buttons in the list view. (Refer to Figure 6). Example: clicking “Load fault history” loads the fault history from the PositionServo drive.

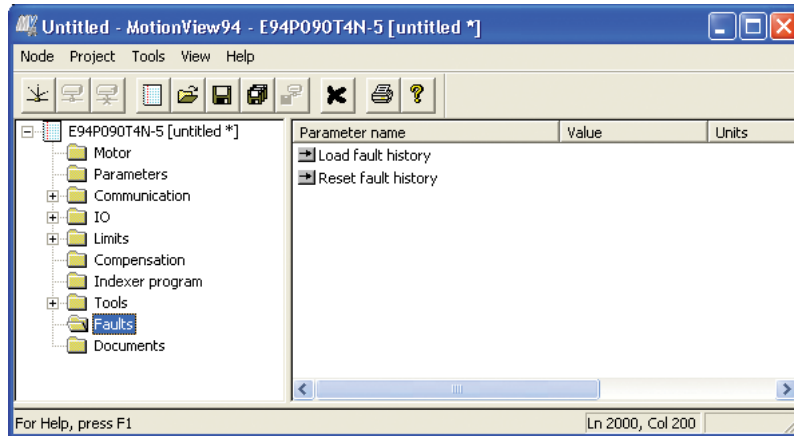


Figure 6: Action Buttons embedded in List View

1.2.5 Main Menu and Toolbar

The functions of MotionView are accessible in two ways: via the Main Menu or the Toolbar as illustrated in Figure 7. If a function in a pull-down menu or an icon is greyed out that denotes the function is unavailable. A function may be unavailable because a drive is not physically connected to the network or the drive is not configured the same as the communication link in MotionView.

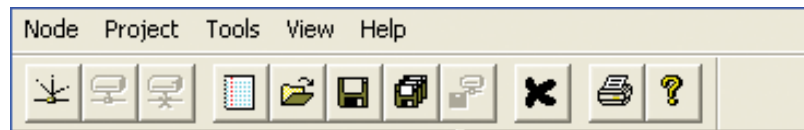


Figure 7: Main Menu and Toolbar

Table 1: Contents of Main Menu Pull-Down Folders

Main Menu				
Node	Project	Tools	View	Help
New configuration file	New project	Browse motor database	Toolbar	MotionView help
Open configuration file	Open project	Clear output window	Status Bar	Product manuals
Save configuration file	Close project			About MotionView
Load configuration file	Save project			Time stamp
Set all parameters to default	Save all configuration files			
Connect drive	Options			
Disconnect all connected drives	Connection setup			
Remove node from project	Recent file			

Table 2: Toolbar Icons

Toolbar Icons									
Connect	Connected	Disconnected	Add File	Open File	Save	Save As	Remove Node	Print	Help

1.3 Managing Projects

Multiple parameter files and drives can be opened at the same time. Information about which files and drives are open and the current window layout is defined as a Project. A Project can be saved as a file to the PC's hard drive. Future sessions will allow opening a project to automatically load the desired files and drives into the node tree and restore the windows layout. Note that MotionView will try to connect all drives listed in the project.

To save a project to a file, click [Project] on the main menu then [Save project] from the pull-down menu.

To start a new project and close all opened files and drives, click [Project] on the main menu then [New project] from the pull-down menu. When you click [New project] all drive connections and file will be closed. The Node Tree will be emptied.

To create a new project using an existing project as a template, click [Project] on the main menu then [Save project As] from the pull-down menu.



Note:

A Project file does not save parameter files or drive data. It saves the list of opened devices and window positions on the screen. Use [Project] [Save All configuration files] to save changes in all opened files and drives. Use [Node] [Save] to save each individual file or drive.

1.4 Connecting to the Drive

To be able to view or change a drive's parameters a connection must be made between the drive and the computer. When you make a connection you establish a communication link between the physical drive and the MotionView program. MotionView software supports RS-232, PPP over RS-485/RS-232 and UDP 10/100 Ethernet.



Note:

If the connection is successful, the drive will appear on the node tree.

If the connection is unsuccessful then the drive will not appear on the node tree. The Message Window will then explain why the connection failed.

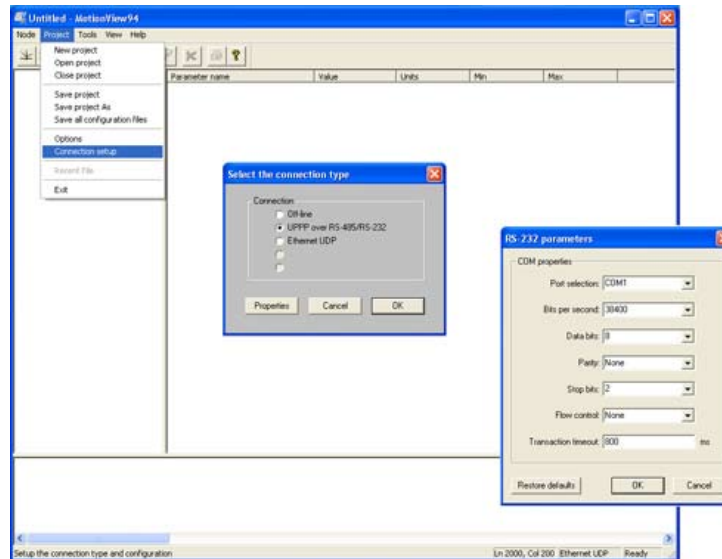




Figure 8: RS-232 Setup

1.4.1 Connection using PPP over RS-232/RS485

1. The first time in a session, click [Project] then [Connection Setup] or click the  icon and select the proper interface from the list.
2. Optionally click on the [Properties] button to change “COM Port” number and/or baud rate if necessary. Note that baud rate assigned in MotionView must be the same as the baud rate set on the drive.
3. Select [Node] then [Connect drive] or click the  icon. The Connection dialog box opens as shown in Figure 8.
4. Specify the address(es) of the drive(s) you want to connect. Refer to paragraph 1.5 “Build RS-485 Connection List” for details.
5. Click the [Connect] button.
6. Drive(s) with specified address(es) will be connected and appear on the left in the Node Tree.

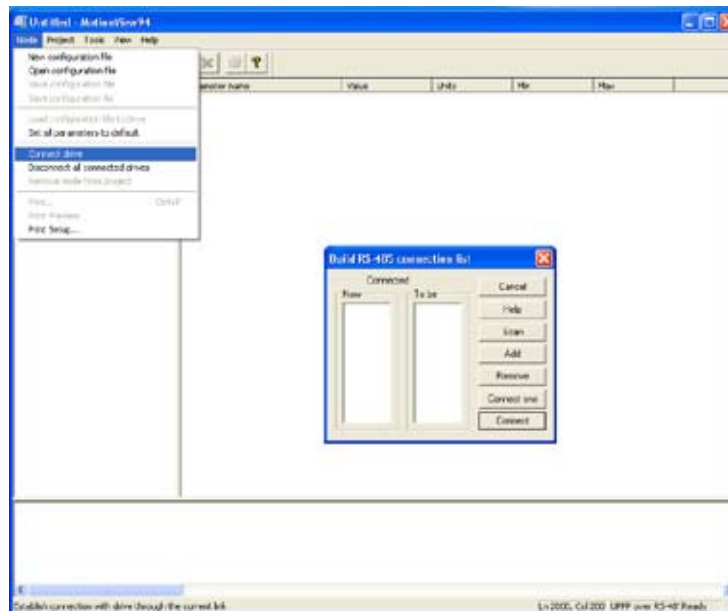


Figure 9: RS-485 Build List

1.4.2 Connection using 10/100 Ethernet

1. Configure IP Address of the PC
Click [Start] (In the bottom left-hand corner of your desktop).

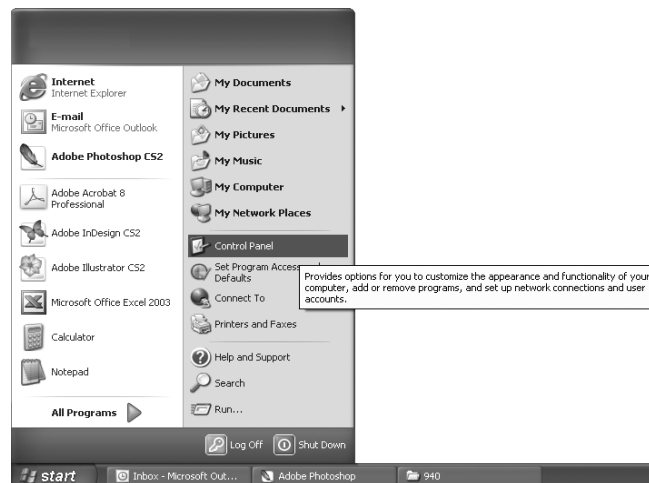


Figure 10a: Start

Select [Control Panel] from the Start menu.

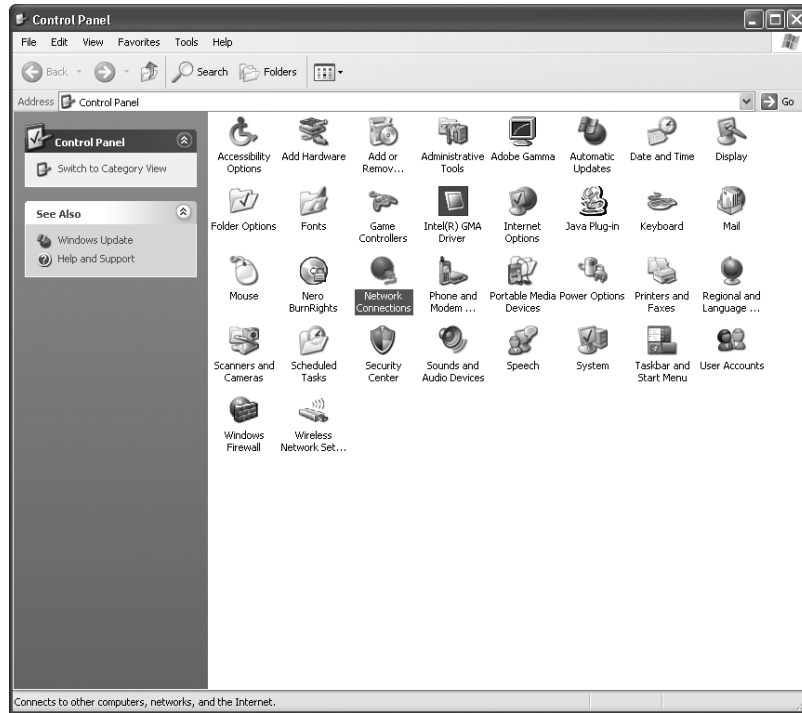


Figure 10b: Network Connections

Select [Network Connections] in the Control Panel menu.

Select the [Local Area Connection] with the number next to it.

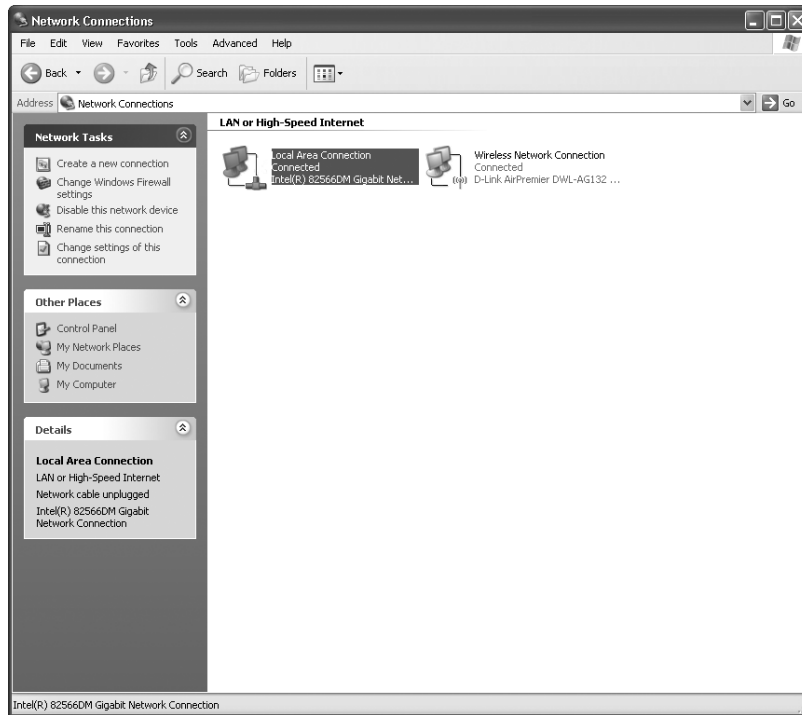


Figure 10c: Local Area Connection

Under the General tab, select [Internet Protocol (TCP/IP)].

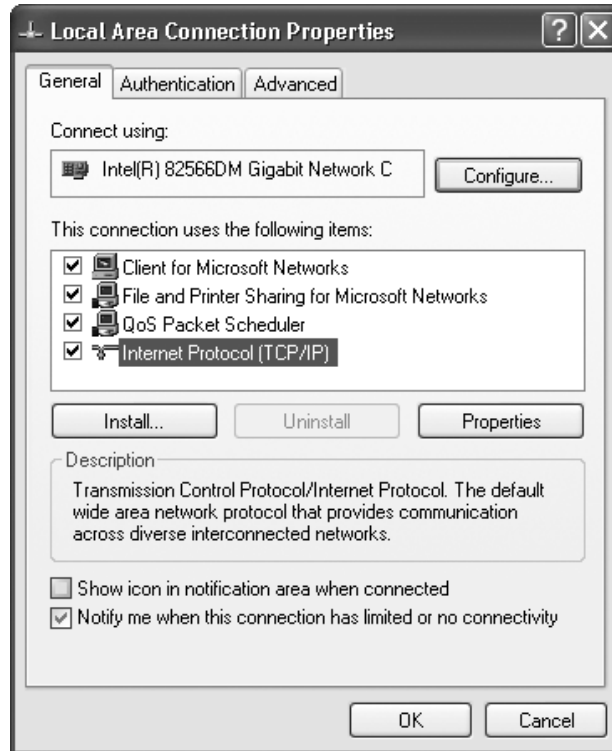


Figure 10d: Internet Protocol TCP/IP

Click [Properties].

Select [Use the following IP Address]

Type "192.168.124.1" in the IP address window.

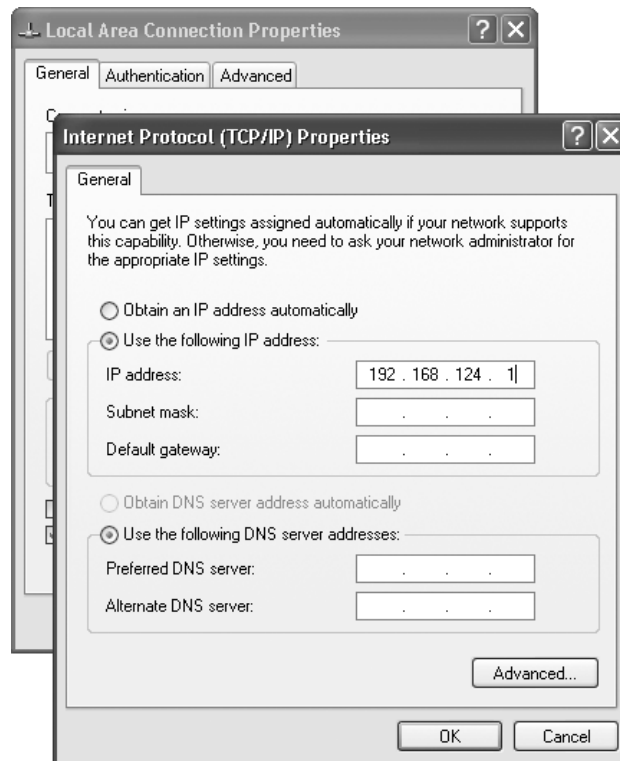




Figure 10e: Use the following IP address

The Subnet Mask window will automatically populate with 255.255.255.0. Click [OK].
The IP address of the PC has been configured.

2. Configure IP Address of Drive:
3. The first time in a session, click [Project] then [Connection Setup] or click the  icon and select the proper interface from the list.
4. Select [Node] then [Connect drive] or click the  icon. The connection dialog opens as illustrated in Figure 10.
5. Specify the IP address(es) of the drive(s) to connect to. Refer to paragraph 1.6 “Build Ethernet Connection List” for details.
6. Click the [Connect] button.
7. The Drive(s) with the specified address(es) will be connected and shown in the Node Tree.

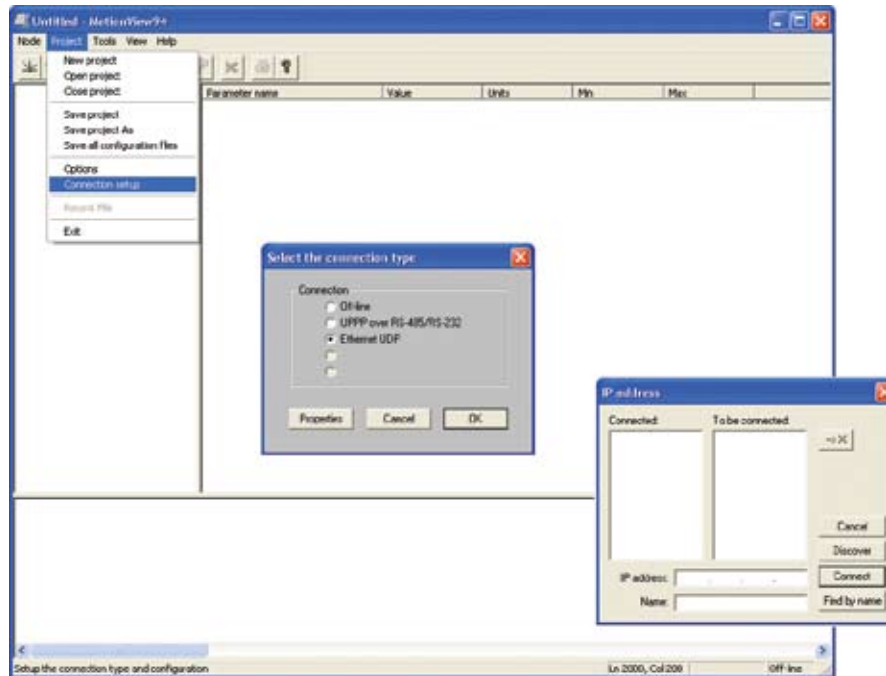



Figure 10e: Composite Screenshot - Connection using 10/100 Ethernet

1.4.3 Disconnect or Remove a Drive

To disconnect a drive: Select the drive by clicking on its icon on the left tree. Click [Node] on the main menu then [Disconnect] from the pull-down menu or click the  icon. MotionView will prompt to [save the current settings to file] before it disconnects the drive. The user can reconnect the drive later by selecting [Node] on the main menu then [Connect drive] from the pull-down menu.

To remove a drive or file from the Node Tree: Select the drive or file by clicking on its icon on the left tree. Select [Node] from the main menu then [Remove node from project] from the pull-down menu.

Note:



A connection does not need to be setup every time the user connects. A connection needs to be setup only once per session or any time the communication settings are changed.

If the work is saved to a project file then the connection does not need to be setup unless different communication settings are used. Re-save the project file if changes are made to it.

1.5 Build RS-485 Connection List



Figure 11: Build RS 485 Connection List

Now	This window shows a list of drives currently connected.
To be	This window shows a list of available drives that could be connected
Cancel	Dismiss dialog box without any action.
Help	Access contents of MotionView Help folder
Scan	Find and connect all drives on the network. A search is performed for all drives in address range 0-31.
Add	Add address to the connection list.
Remove	Removes highlighted address from the connection list.
Connect One	Specifies one drive's address to be connected
Connect	Puts the drive selected in the "To be" window into the "Now" window.

1.6 Build Ethernet Connection List

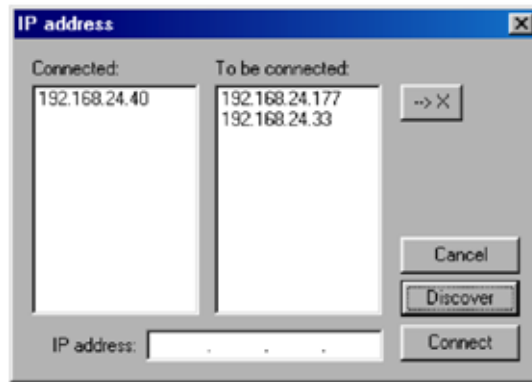


Figure 12: Build Ethernet Connection List

IP address	Specify a single device address in this field and then click [Connect] to this drive with this IP address.
Connected	Shows drives currently connected
To be connected	Shows drives MotionView found on the network and available for connection
- -> X	Removes highlighted addresses from the "To Be Connected" list
Discover	Locates every physically connected drive on the network
Connect	Connects every drive with an IP listed in the "To Be Connected" list box.
Cancel	Dismisses dialog box without any action.

1.7 File Operations

1.7.1 Opening and Closing Parameter Files.

To open an existing configuration file, click [Node] on the main menu then [Open configuration file] from the pull-down menu. Select the configuration file with extension “.dcf” in the open window. The File will appear in the Node Tree on the left.

To save the file, click [Node] on the main menu then [Save configuration file] from the pull-down menu.

To remove a file from the Node Tree, click [Node] on the main menu then [Remove node from project] from the pull-down menu.

1.7.2 Load Parameters from File to Drive.

To load data from a file to a drive, click [Node] on the main menu then [Load configuration file to drive] from the pull-down menu. The Drive data will be updated with the file data.



Note:

For this operation the drive must be connected and DISABLED.

Parameters will not be loaded when the drive is enabled.

2 Node Tree Folders



Note:

Refer to the PositionServo User's Manual (S94P01) and Programming Manual (PM94P01) for complete descriptions of motor features and instructions for programming each parameter. To access the PositionServo Programming Manual, click [Help] then click [Product manuals]. The information contained herein is a brief description of the folders in the Node Tree that populate once a drive file is opened.

2.1 Drive

Click the Drive name in the Node Tree. The drive ID string, device family, firmware revision, vector processor revision, hardware revision and serial number are displayed as illustrated in Figure 13. The drive identification parameters are fixed and are provided for information only.

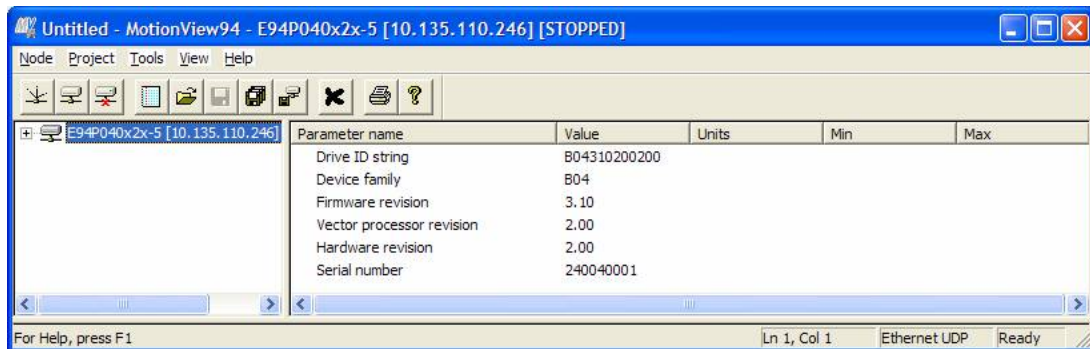


Figure 13: Drive ID String

2.2 Motor

After configuring the interface of the PositionServo drive, the motor needs to be setup if one is attached. To select a motor, click on the [Motor] folder. Click on the action button [Click here to change the motor] to bring up the motor parameters screen. Set the motor vendor and motor model number. If the motor is not in the list, click [Create File] to define a new motor setup.

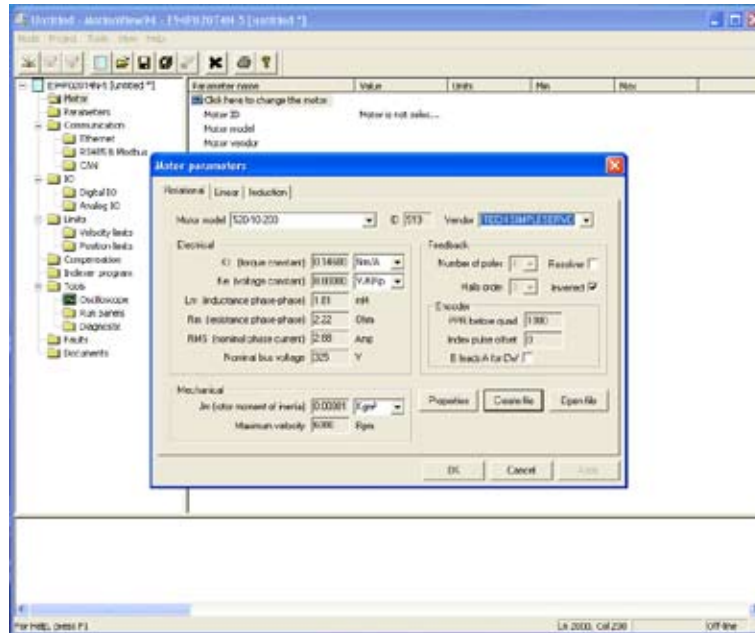


Figure 14: Motor Parameters

2.3 Parameters

To setup the drive's parameters click on the [Parameters] folder. To set any one of the parameters in the List View on the right, double-click the parameter name. A dialog box will open for changes to that specific parameter. The parameters that populate the List View are applicable to the Drive file at the top of the Node Tree. Table 1 lists the parameter name, value, units, and min/max values for the selected E94P090T4N Drive which is a 3-phase, 480V/9A, non-filtered PositionServo 940 encoder-based drive.

Parameter name	Value	Units	Min	Max
Drive name				
Drive mode	Torque			
Drive PWM frequency	15.000			
Current limit	2.8000	A	0.0000	6.0000
IKHG peak current limit	7.2000	A	0.0000	27.0000
14 OH peak current limit	7.2000	A	0.0000	27.0000
Analog input (current scale)	0.0000	A/Bit	-4.0000	1.8000
Analog input (velocity scale)	100.0000	SRM/Bit	-2000.0000	2000.0000
Enable Accel/Decel limits	Disable			
Accel limit	1000.0000	SRM/sec	0.1000	1000000.0000
Decel limit	1000.0000	SRM/sec	0.1000	1000000.0000
Reference	External			
Stop input type	Master encoder			
Fault reset	on disable			
Motor temperature sensor	Disable			
Major pit: cut-off resistance	2000	Ohm	2000	2000
Second encoder	Disable			
Regen duty cycle	13	%	1	200
Encoder repeat source	Drive feedback input			
Software to master ratio	1:1		-32767:1	32767:32767
Second to prime encoder ratio	1:1		-32767:1	32767:32767
Autolock	Enabled			
Group ID	0		0	32767
Enable switch function	Run			
User units	1.0000	Revolutions/Unit	0.0000	1000000.0000
Resolver track	0		0	25
Current Limit Flux Override	Disable			

Figure 15: Parameters (Drive)

Table 3: E94P09T4N Drive Parameters

Parameter Name	Value	Units	Min	Max
Drive name				
Drive mode	Torque			
Drive PWM frequency	16kHz			
Current limit	2.8	A	0.0000	9.0000
8kHz peak current limit	7.2000	A	0.0000	27.0000
16kHz peak current limit	7.2000	A	0.0000	22.500
Analog input (current scale)	0.9000	A/volt	-1.8000	1.8000
Analog input (velocity scale)	100.0000	RPM/volt	-2000.0000	2000.0000
Enable Accel/Decel limits	Disable			
Accel limit	1000.0000	RPM/sec	0.1000	5000000.0000
Decel limit	1000.0000	RPM/sec	0.1000	5000000.0000
Reference	External			
Step input type	Master encoder			
Fault reset	on disable			
Motor temperature sensor	Disable			
Motor ptc cut-off resistance	2500	Ohm	2000	3000
Second encoder	Disable			
Regen duty cycle	10	%	1	100
Encoder repeat source	Drive feedback input			
System to master ratio	1:1		-32767:1	32767:32767
Second to prime encoder ratio	1:1		-32767:1	32767:32767
Autoboot	Enabled			
Group ID	0		0	32767
Enable switch function	Run			
User units	1.0000	Revolutions/unit	0.0000	1000000.0000
Resolver track	0		0	15
Current Limit Max Overwrite	Disable			

The “Resolver track” parameter is only applicable to the PositionServo 941 Resolver-based drive. The Resolver track function sets the pulse per revolution (PPR) resolution of the buffered encoder outputs when a resolver motor is used. Refer to the PositionServo User’s Manual for more details. The Resolver track and Current Limit Max Overwrite functions are available with PositionServo drive firmware revision 3.06 or higher.

2.4 Communication

There are 3 sub-folders under the [Communications] folder in the MotionView Node Tree. The Ethernet folder contains an action button [IP setup] that permits the user to configure the Ethernet interface. The [RS485 and Modbus] folder contains the configuration data of the Modbus interface. The [CAN] folder contains the configuration data for the CAN interface.

2.4.1 Ethernet

The Ethernet folder contains an action button [IP setup] that permits the user to configure the Ethernet interface. Click on the [IP setup] button to view/change the Ethernet interface setup (The Ethernet interface may have been previously configured with [Project] [Configuration setup], paragraph 1.3.2). In [IP setup], the user can specify the IP address.

2.4.2 RS485 and Modbus

The [RS485 and Modbus] folder contains the configuration parameters of the Modbus interface. Click on any Modbus parameter to change it. Table 4 lists the range and default value of each RS485 Modbus parameter.

Table 4: RS 485 Modbus Parameters

Parameter	Range	Default Value
RS 485 Configuration	Normal, Modbus slave	Normal
Modbus baud rate	2400, 4800, 9600, 19200, 38400, 57600, 115200	19200
Modbus reply delay	0 - 1000ms	0
Modbus parity	No Parity, Odd, Even	No Parity
Modbus stop bits	1.0, 1.5, 2.0	1.0

2.4.3 CAN

The [CAN] folder contains the configuration parameters for the CAN interface. Click on Parameter name to change the setting of that parameter. Table 5 lists the range and default value for each CAN parameter.

Table 5: CAN Parameters

Parameter	Range	Default Value
CAN Control	Disabled, CANOpen Simple, CANOpen 402	Disabled
CAN baud rate	10k, 25k, 50k, 125k, 250k, 500k, 800k, 1000k	500k
CAN Address	1 - 127	1
CAN Bootup Mode	Pre-operational, Operational, Pseudo master mode	Pre-operational
CAN Bootup Delay	0-5 sec	5 sec

2.5 I/O

There are 2 sub-folders under the [I/O] folder in the MotionView Node Tree. The [Digital I/O] folder contains the values of the 4 outputs and debounce times for the 12 inputs. The [Analog I/O] folder contains the values of the one output and one input plus an action button [Adjust analog input zero offset] that permits the user to change the analog zero offset.

2.5.1 Digital I/O

The [Digital I/O] folder contains the values of the 4 outputs and debounce times for the 12 inputs (A1-A4, B1-B4, C1-C4).

Table 6: Digital Input/Output Parameters

Parameter	Range	Default Value
Output 1 function	Not assigned, Zero Speed, In speed window, Current limit, Run time fault, Ready, Brake, In position	Not assigned
Output 2 function	Not assigned, Zero Speed, In speed window, Current limit, Run time fault, Ready, Brake, In position	Not assigned
Output 3 function	Not assigned, Zero Speed, In speed window, Current limit, Run time fault, Ready, Brake, In position	Not assigned
Output 4 function	Not assigned, Zero Speed, In speed window, Current limit, Run time fault, Ready, Brake, In position	Not assigned
Input A1 debounce time	0-1000 ms	0
Input A2 debounce time	0-1000 ms	0
Input A3 debounce time	0-1000 ms	0
Input A4 debounce time	0-1000 ms	0
Input B1 debounce time	0-1000 ms	0
Input B2 debounce time	0-1000 ms	0
Input B3 debounce time	0-1000 ms	0
Input B4 debounce time	0-1000 ms	0
Input C1 debounce time	0-1000 ms	0
Input C2 debounce time	0-1000 ms	0
Input C3 debounce time	0-1000 ms	0
Input C4 debounce time	0-1000 ms	0
Hard limit switches action	Not assigned, Fault, Stop and fault	Not assigned

2.5.2 Analog I/O

The [Analog I/O] folder contains the parameters of one output and one input plus an action button [Adjust analog input zero offset] that permits the user to change the analog zero offset.

Table 7: Analog Input/Output Parameters

Parameter	Range	Default Value
Analog output	Not assigned, Phase current RMS, Phase current Peak, Motor velocity, Phase R current, Phase S current, Phase T current, Iq current, Id current	Not assigned
Analog output current scale	0.1000 - 10.000 Volt/Amp	1.0000
Analog output velocity scale	0.1000 - 5.0000 mV/RPM	1.0000
Analog input dead band	0 - 50 mV	10
Analog input offset	-1000 - 1000 mV	0

Note: Phases R, S and T are equivalent to phases U, V and W respectively.

2.6 Limits

There are 2 sub-folders under the [Limits] folder in the MotionView Node Tree for setting the velocity and position limits.

2.6.1 Velocity Limits

To set the velocity limits of the PositionServo drive in MotionView, double click on the [Limits] folder to expand it then click on the [Velocity limits] folder to open this function. Table 8 lists the range and default value of each of the Velocity limits parameters.

Table 8 Velocity Limits Parameters

Parameter	Range	Default Value
Zero speed	0 - 100 RPM	10
Speed window	10 - 10000 RPM	100
At speed	-10000 - 10000 RPM	10000

2.6.2 Position Limits

To set the position limits of the PositionServo drive in MotionView, double click on the [Limits] folder to expand it then click on the [Position limits] folder to open this function. Table 9 lists the range and default value of each of the Position limits parameters.

Table 9: Position Limits Parameters

Parameter	Range	Default Value
Position error	1 - 32767 counts	500
Max Error Time	0.2500 - 8000.0000 ms	500.0000
Second encoder Position error	1 - 32767 counts	500
Second encoder Max Error Time	0.2500 - 8000.0000 ms	500.0000

2.7 Compensation

To set the Compensation parameters, click the [Compensation] folder to open its contents in the List View window. To change a compensation parameter, click the Parameter name. Table 10 lists the range and default value of each compensation parameter.

Table 10: Compensation Parameters

Parameter	Range	Default Value
Velocity P-gain	0.0000 - 32767.0000	600.0000
Velocity I-gain	0.0000 - 16383.0000	0.0000
Position P-gain	0.0000 - 32767.0000	600.0000
Position I-gain	0.0000 - 16383.0000	0.0000
Position D-gain	0.0000 - 32767.0000	0.0000
Position I-limit	0.0000 - 20000.0000 RPM	200.0000
Gain scaling	-16 - 4	-4

2.8 Indexer Program

Click on the [Indexer program] folder to open the MotionView Studio (the List View window is gray when MotionView Studio is selected). The user can type program code in the MotionView Studio or import it from a file, compile the program, step in/over it, run it, then compile with the option of sending the compiled program directly to the drive.

The PositionServo Programming Manual contains full details on the MotionView Studio and the Indexer Program. To access the PositionServo Programming Manual, click [Help] then click [Product manuals].

2.9 Tools

There are 3 sub-folders under the [Tools] folder in the MotionView Node Tree. The Oscilloscope tool provides a real-time display of the PositionServo drive's behavior. The [Run panels] folder contains an action button [Check phasing] that permits the user to check the phase of the motor. The [Diagnostic] folder contains an action button [Show variables] that permits the user to access the list of variables.

2.9.1 Oscilloscope

The Oscilloscope tool provides a real-time display of the different electrical signals inside the PositionServo drive. Like a "real" oscilloscope, the Scope tool displays two channels simultaneously. The signals in Table 11 can be observed using the Oscilloscope tool. Click on the [Oscilloscope] folder to open the Oscilloscope in a separate window. Click on the [RUN]/[STOP] button to start or stop the scope display. Click on the [Close] button to close the Oscilloscope display.

Table 11: Oscilloscope Parameters

Parameter	Description
Phase Current (RMS)	Motor phase (RMS) current
Phase Current (Peak)	Motor phase peak current
Iq Current	Motor Iq (torque producing) current
Motor Velocity	Actual motor speed in RPM
Commanded Velocity	Desired motor speed in RPM (velocity mode only)
Velocity Error	Difference in RPM between actual and commanded motor speed
Position Error	Difference between actual and commanded position (Step and Direction mode only)
Bus voltage	DC bus voltage
Analog input	Voltage at the drive's analog input
Target position	Requested position
Absolute position	Absolute position (actual position)
Target position pulses	
Absolute position pulses	
Secondary absolute position	
Position increment	
Secondary position error	

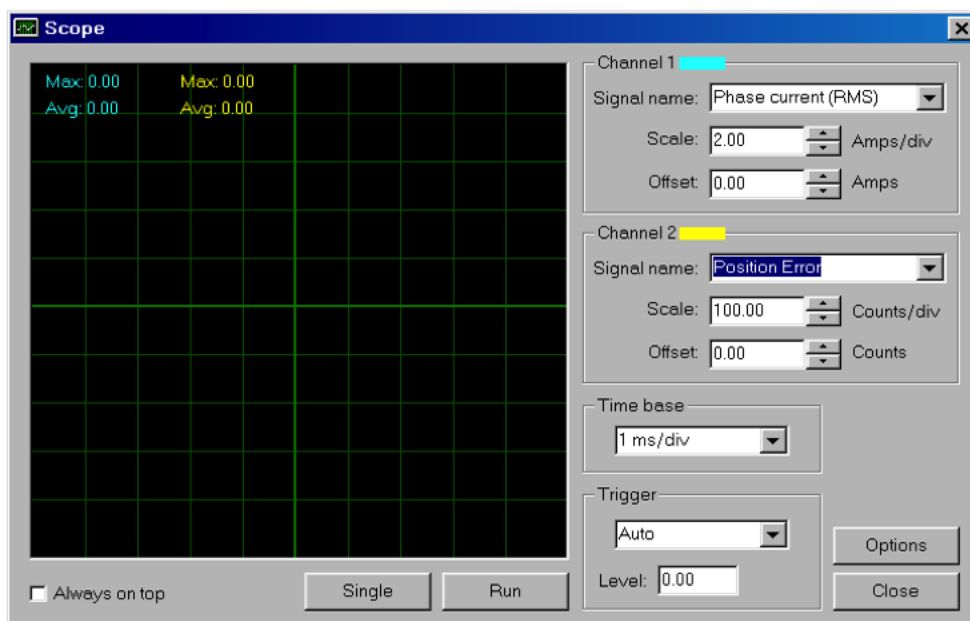


Figure 16: Oscilloscope Display

Signal Name

You can customize the information presented by the Scope tool by choosing the drop-down box in each channel. The set of available signal depends on the drive model. Refer to the User's Manual appropriate for your drive model to see the list of the signals.

Scale

Scale sets the sensitivity of the display. Each division is considered one unit of the selected scale. A scale of 100 RPM/div, for example, means that the signal will rise (or descend) by one division for every change of 100 RPM in the signal level. Thus, a 500-RPM signal would deflect the signal by five divisions from the central reference line.

Offset

Offset sets the vertical distance from the central base line to the signal trace. This is useful if you want to compare two signals. For example, if you wish to compare the actual vs. commanded motor velocity, you would enter an offset that would move the two signals to the same side of the central reference line.

Time Base

Time base sets the number of cycles displayed per division. Higher frequencies have a shorter time base than lower frequencies. If you wanted to display one cycle of a particular signal, your time base setting would therefore be lower for high-frequency signals than for low-frequency signals.

Trigger/Trigger Level

Trigger level specifies the signal level after which the scope starts acquiring data. You can also specify which channel will be a source for the trigger. The oscilloscope display will continue to run while the signal level crosses the specified level (above if the trigger is set for rising or leading edge, or below if the trigger is set for trailing or falling edge).

Single

Also called one-shot trigger. If Single Sweep is selected, data acquisition will be stopped after the scope buffer is filled and data displayed on the screen (frozen data). To repeat data acquisition, you will need to click the Single again.

Always on top

Select this button to display the oscilloscope window on top all other windows.

Options

Select this button to change the channel mode, display mode and channel width settings. The default settings are: channel mode: normal, display mode: connected lines and channel width: average maximum.

2.9.2 Run Panels

The [Run panels] folder permits the user to check the phasing of the motor. Click on the [Run panels] folder then click the action button [Check phasing] to check the phasing of the motor. The motor data is not modified when checking the phase.

2.9.3 Diagnostic

The [Diagnostic] folder permits the user to check the drive's variables. Click on the [Diagnostic] folder then click the action button [Show variables] to open the list of the drive's variables.

2.10 Faults

The [Faults] folder contains two action buttons. The [Load fault history] permits the user to load the fault history of the drive onto the computer. The sixteen most recent faults are displayed with the newer faults replacing the older faults in a first-in first-out manner. In all cases, fault #0 is the most recent fault. The [Reset fault history] permits the user to clear the fault history of the drive from within the MotionView program.

2.11 Documents

The [Documents] folder contains the Release notes for the installed version of MotionView software. Click on the document title to open a hypertext link to the AC Technology's Technical Library.



Note:

Refer to the PositionServo User's Manual (S94P01) and Programming Manual (PM94P01) for complete descriptions of motor features and instructions for programming each parameter. To access the PositionServo Programming Manual, click [Help] then click [Product manuals]. The information contained herein is a brief description of the folders in the Node Tree that populate once a drive file is opened.

AC Technology Corporation • 630 Douglas Street • Uxbridge, MA 01569 • USA
☎ +1 (508) 278-9100

```

***** HEADER *****
;Title:           Pick and Place example program
;Author:          Lenze - AC Technology
;Description:     This is a sample program showing a simple sequence that
;                picks up a part, moves to a set position and drops the part

;***** I/O List *****
;   Input A1      -   not used
;   Input A2      -   not used
;   Input A3      -   Enable Input
;   Input A4      -   not used
;   Input B1      -   not used
;   Input B2      -   not used
;   Input B3      -   not used
;   Input B4      -   not used
;   Input C1      -   not used
;   Input C2      -   not used
;   Input C3      -   not used
;   Input C4      -   not used
;   Output 1      -   Pick Arm
;   Output 2      -   Gripper
;   Output 3      -   not used
;   Output 4      -   not used

;***** Initialize and Set Variab
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
;V1 =
;V2 =

;***** Events *****
;Set Events handling here

;***** Main Program *****

RESET_DRIVE:                ;Place holder fo
WAIT UNTIL IN_A3:           ;Make sure tha
continuing
ENABLE
PROGRAM_START:
MOVEP 0                      ;Move to Pick po
OUT1 = 1                    ;Turn on output
WAIT TIME 1000              ;Delay 1 sec to
OUT2 = 1                    ;Turn on output
WAIT TIME 1000              ;Delay 1 sec to
OUT1 = 0                    ;Turn off output
MOVED -10                   ;Move 10 REVS to
OUT1 = 1                    ;Turn on output
WAIT TIME 1000              ;Delay 1 sec to
OUT2 = 0                    ;Turn off output
WAIT TIME 1000              ;Delay 1 sec to
OUT1 = 0                    ;Retract Pick ar
GOTO PROGRAM_START
END

;***** Sub-Routines *****
Enter Sub-Routine code here

;***** Fault Handler Routine *****
;
;   Enter Fault Handler code here
ON FAULT
ENDFAULT

```



PositionServo

Programming Manual for PC-based MotionView

Copyright ©2005 by AC Technology Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from AC Technology Corporation. The information and technical data in this manual are subject to change without notice. AC Technology makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. AC Technology assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView®, PositionServo®, and all related indicia are either registered trademarks or trademarks of Lenze AG in the United States and other countries.

Contents

1.	Introduction	4
1.1	Definitions	4
1.2	Programming Flowchart.....	5
1.3	MotionView / MotionView Studio	6
1.3.1	Main Toolbar.....	6
1.3.2	Program Toolbar.....	8
1.3.3	MotionView Studio - Indexer Program.....	9
1.4	Programming Basics.....	10
1.5	Using Advanced Debugging Features.....	17
1.6	Inputs and Outputs	17
1.7	Events.....	22
1.8	Variables and Define Statement.....	23
1.9	IF/ELSE Statements	24
1.10	Motion.....	25
1.10.1	Drive Operating Modes.....	26
1.10.2	Point To Point Moves.....	26
1.10.3	Segment Moves.....	27
1.10.4	Registration.....	28
1.10.5	S-Curve Acceleration.....	29
1.10.6	Motion Queue	29
1.11	Subroutines and Loops.....	30
1.11.1	Subroutines.....	30
1.11.2	Loops.....	31
2.	Programming	32
2.1	Program Structure	32
2.2	Variables.....	34
2.3	Arithmetic Expressions	36
2.4	Logical Expressions and Operators.....	36
2.4.1	Bitwise Operators	36
2.4.2	Boolean Operators.....	37
2.5	Comparison Operators	37
2.6	System Variables and Flags	38
2.7	System Variables Storage Organization.....	38
2.7.1	RAM File for User's Data Storage	38
2.7.2	Memory Access Through Special System Variables	39
2.7.3	Memory Access Through MEMSET, MEMGET Statements	40
2.8	System Variables and Flags Summary.....	41
2.8.1	System Variables.....	41
2.8.2	System Flags	42
2.9	Control Structures.....	43
2.9.1	DO/UNTIL Structure	43
2.9.2	WHILE Structure.....	43
2.9.3	Subroutines.....	44
2.9.4	IF Structure	45
2.9.5	IF/ELSE Structure.....	45
2.9.6	WAIT Statement	46
2.9.7	GOTO Statement & Labels.....	46
2.10	Scanned Event Statements	46

Contents

2.11	Motion	48
2.11.1	How Moves Work.....	48
2.11.2	Incremental (MOVED) and Absolute (MOVEP) Motion	48
2.11.3	Incremental (MOVED) Motion.....	49
2.11.4	Absolute (MOVEP) Move.....	49
2.11.5	Registration (MOVEDR MOVEPR) Moves	50
2.11.6	Segment Moves.....	50
2.11.7	MDV Segments.....	50
2.11.8	S-curve Acceleration.....	52
2.11.9	Motion SUSPEND/RESUME	52
2.11.10	Conditional Moves (MOVE WHILE/UNTIL)	52
2.11.11	Motion Queue and Statement Execution while in Motion	53
2.12	System Status Register (DSTATUS register)	55
2.13	Fault Codes (DFAULTS register).....	56
2.14	Limitations and Restrictions	57
2.15	Homing	58
2.15.1	What is Homing?	58
2.15.2	The Homing Function	58
2.15.3	Home Offset.....	59
2.15.4	Homing Velocity.....	59
2.15.5	Homing Acceleration.....	59
2.15.6	Homing Switch.....	59
2.15.7	Homing Start.....	59
2.15.8	Homing Method	60
2.15.9	Homing Methods.....	61
2.15.9.1	Homing Method 1: Homing on the Negative Limit Switch.....	62
2.15.9.2	Homing Method 2: Homing on the Positive Limit Switch	62
2.15.9.3	Homing Method 3: Homing on the Positive Home Switch & Index Pulse	63
2.15.9.4	Homing Method 4: Homing on the Positive Home Switch & Index Pulse	63
2.15.9.5	Homing Method 5: Homing on the Negative Home Switch & Index Pulse	64
2.15.9.6	Homing Method 6: Homing on the Negative Home Switch & Index Pulse	64
2.15.9.7	Homing Method 7: Homing on the Home Switch & Index Pulse.....	65
2.15.9.8	Homing Method 8: Homing on the Home Switch & Index Pulse.....	66
2.15.9.9	Homing Method 9: Homing on the Home Switch & Index Pulse.....	67
2.15.9.10	Homing Method 10: Homing on the Home Switch & Index Pulse.....	68
2.15.9.11	Homing Method 11: Homing on the Home Switch & Index Pulse.....	69
2.15.9.12	Homing Method 12: Homing on the Home Switch & Index Pulse.....	70
2.15.9.13	Homing Method 13: Homing on the Home Switch & Index Pulse.....	71
2.15.9.14	Homing Method 14: Homing on the Home Switch & Index Pulse.....	72
2.15.9.15	Homing Method 17: Homing without an Index Pulse.....	73
2.15.9.16	Homing Method 18: Homing without an Index Pulse.....	74
2.15.9.17	Homing Method 19: Homing without an Index Pulse.....	75
2.15.9.18	Homing Method 21: Homing without an Index Pulse.....	76
2.15.9.19	Homing Method 23: Homing without an Index Pulse.....	77
2.15.9.20	Homing Method 25: Homing without an Index Pulse.....	78
2.15.9.21	Homing Method 27: Homing without an Index Pulse.....	79
2.15.9.22	Homing Method 29: Homing without an Index Pulse.....	80
2.15.9.23	Homing Method 33: Homing to an Index Pulse	81
2.15.9.24	Homing Method 34: Homing to an Index Pulse	81
2.15.9.25	Homing Method 35: Using Current Position as Home	81
2.15.10	Homing Mode Operation example	82
3.	Reference	83
3.1	Program Statement Glossary	83
3.2	Variable List.....	103
3.3	Quick Start Examples	117
3.3.1	Quick Start - External Torque/Velocity.....	117
3.3.2	Quick Start - External Positioning	119
3.3.3	Quick Start - Internal Torque/Velocity.....	121
3.3.4	Quick Start - Internal Positioning	123
3.4	PositionServo Reference Diagrams.....	124

About These Instructions

This documentation applies to the programming of the PositionServo drive with model numbers ending in “EX” and “RX”. This documentation should be used in conjunction with the PositionServo User Manual (Document S94P01) that shipped with the drive. These documents should be read in their entirety as they contain important technical data and describe the installation and operation of the drive.

Safety Warnings

Take note of these safety warnings and those in the PositionServo User Manual and related documentation.



WARNING! Hazard of unexpected motor starting!

When using MotionView, or otherwise remotely operating the PositionServo drive, the motor may start unexpectedly, which may result in damage to equipment and/or injury to personnel. Make sure the equipment is free to operate and that all guards and covers are in place to protect personnel.

All safety information contained in these Programming Instructions is formatted with this layout including an icon, signal word and description:



Signal Word! (Characterizes the severity of the danger)

Safety Information (describes the danger and informs on how to proceed)

Table 1: Pictographs used in these Instructions

Icon		Signal Words	
	Warning of hazardous electrical voltage	DANGER!	Warns of impending danger . Consequences if disregarded: Death or severe injuries.
	Warning of a general danger	WARNING!	Warns of potential, very hazardous situations . Consequences if disregarded: Death or severe injuries.
	Warning of damage to equipment	STOP!	Warns of potential damage to material and equipment . Consequences if disregarded: Damage to the controller/drive or its environment.
	Information	NOTE	Designates a general, useful note. If the note is observed then handling the controller/drive system is made easier.

Related Documents

The documentation listed herein contains information relevant to the operation and programming of the Position Servo drive with model numbers ending in “EX” and “RX”. To obtain the latest documentation, visit the Technical Library at <http://www.lenze-actech.com>.

Table 2: Reference Documentation

Document #	Description
S94P01	PositionServo User Manual
PM94P01	PositionServo Programming Manual
P94MOD01	Position Servo ModBus RTU over RS485, ModBus TCP/IP
P94CAN01	PositionServo CANopen Communications Reference Guide

1. Introduction

1.1 Definitions

Included herein are definitions of several terms used throughout this programming manual and the PositionServo user manual.

PositionServo: The PositionServo is a programmable digital drive/motion controller, that can be configured as a stand alone programmable motion controller, or as a high performance torque and velocity drive for centralized control systems. The PositionServo family of drives includes the 940 Encoder-based drive and the 941 Resolver-based drive.

MotionView: MotionView is a universal communication and configuration software that is utilized by the PositionServo drive family. MotionView has an automatic self-configuration mechanism that recognizes what drive it is connected to and configures the tool set accordingly. The MotionView platform is divided up into three sections or windows, the "Parameter Tree Window", the "Parameter View Window" and the "Message Window". Refer to Section 1.3 for more detail.

SimpleMotion Language (SML): SML is the programming language utilized by MotionView. The SML software provides a very flexible development environment for creating solutions to motion applications. The software allows you to create complex and intelligent motion moves, process I/O, perform complex logic decision making, do program branching, utilize timed event processes, as well as a number of other functions found in PLC's and high end motion controllers.

User Program (or Indexer Program): This is the SML program, developed by the user to describe the programmatic behavior of the PositionServo drive. The User Program can be stored in a text file on your PC or in the PositionServo's EPM memory. The User Program needs to be compiled (translated) into binary form with the aid of the MotionView Studio tools before the PositionServo can execute it.

MotionView Studio: MotionView Studio is the front end interface of the MotionView platform. It is a tool suite containing all the software tools needed to program and debug a PositionServo. These tools include a full-screen text editor, a program compiler, status and monitor utilities, an online oscilloscope and a debugger function that allows the user to step through the program during program development.



WARNING!

- Hazard of unexpected motor starting! When using the MotionView software, or otherwise remotely operating the PositionServo drive, the motor may start unexpectedly, which may result in damage to equipment and/or injury to personnel. Make sure the equipment is free to operate in this manner, and that all guards and covers are in place to protect personnel.
 - Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the printed circuit board or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.
-

1.2 Programming Flowchart

MotionView utilizes a BASIC-like programming structure referred to as SimpleMotion Programming Language (SML). SML is a quick and easy way to create powerful motion applications.

With SML the programmer describes his system's logistics, motion, I/O processing and user interaction using the SML structured code. The program structure includes a full set of arithmetic and logical operator programming statements, that allow the user to command motion, process I/O and control program flow.

Before the PositionServo drive can execute the user's program, the program must first be compiled (translated) into binary machine code, and downloaded to the drive. Compiling the program is done by selecting the [Compile] button from the toolbar. The user can also compile and download the program at the same time by selecting the [Compile and Load] button from the toolbar. Once downloaded, the compiled program is stored in both the PositionServo's EPM memory and the internal flash memory. Figure 1 illustrates the flow of the program preparation process.

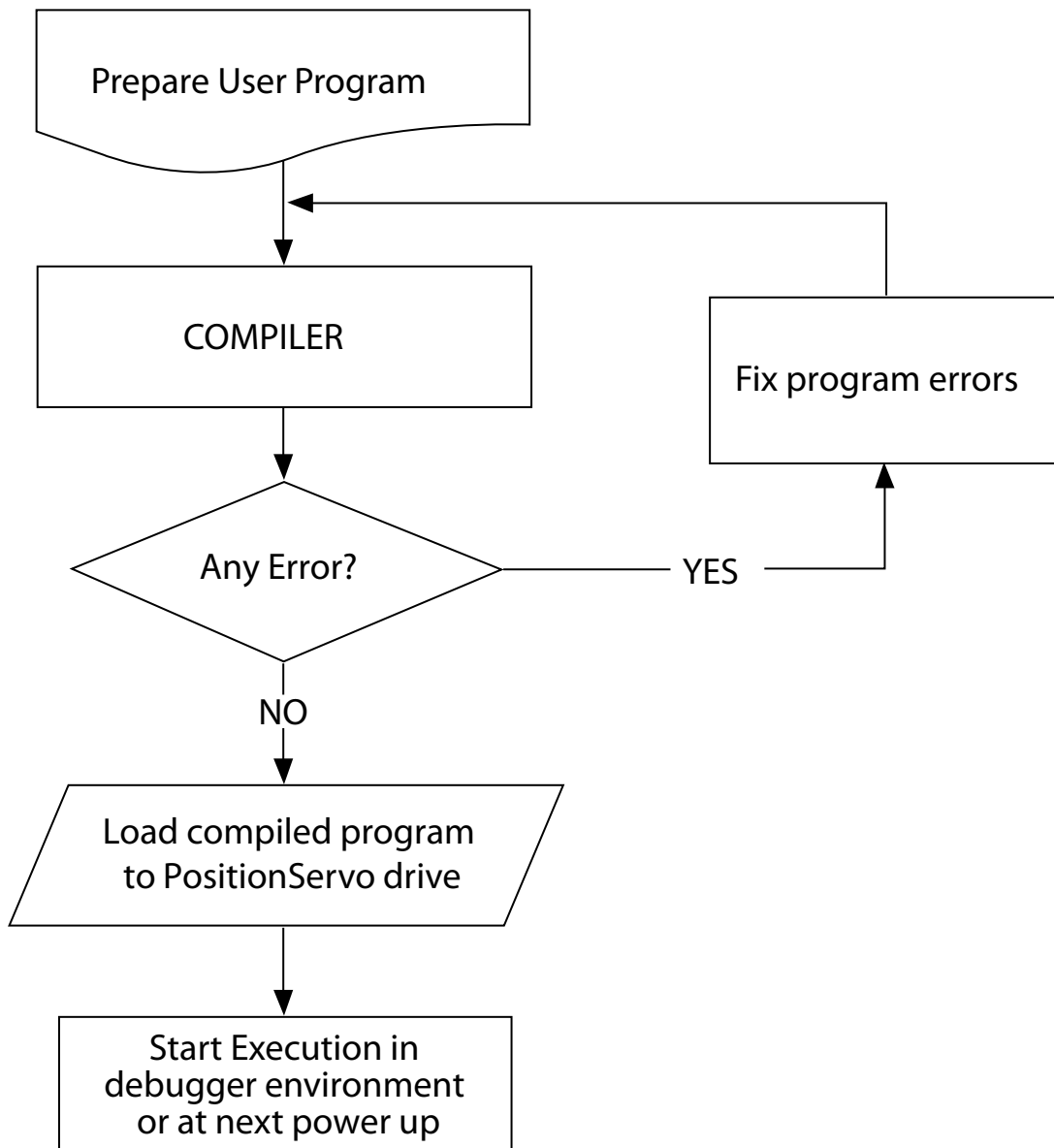


Figure 1: Program Preparation

Introduction

1.3 MotionView / MotionView Studio

There are two versions of MotionView Software: one which resides inside the drive's memory, referred to as "MotionView on Board" (MVOB); and one supplied as a PC-installed software package, referred to simply as MotionView. This manual describes the PC-installed MotionView software for PositionServo drives with P/N ending in EX or RX. The MotionView display is illustrated in Figure 2.



NOTE

For MotionView OnBoard (MVOB), refer to "PositionServo with MVOB Programming Manual" document number PM94M01

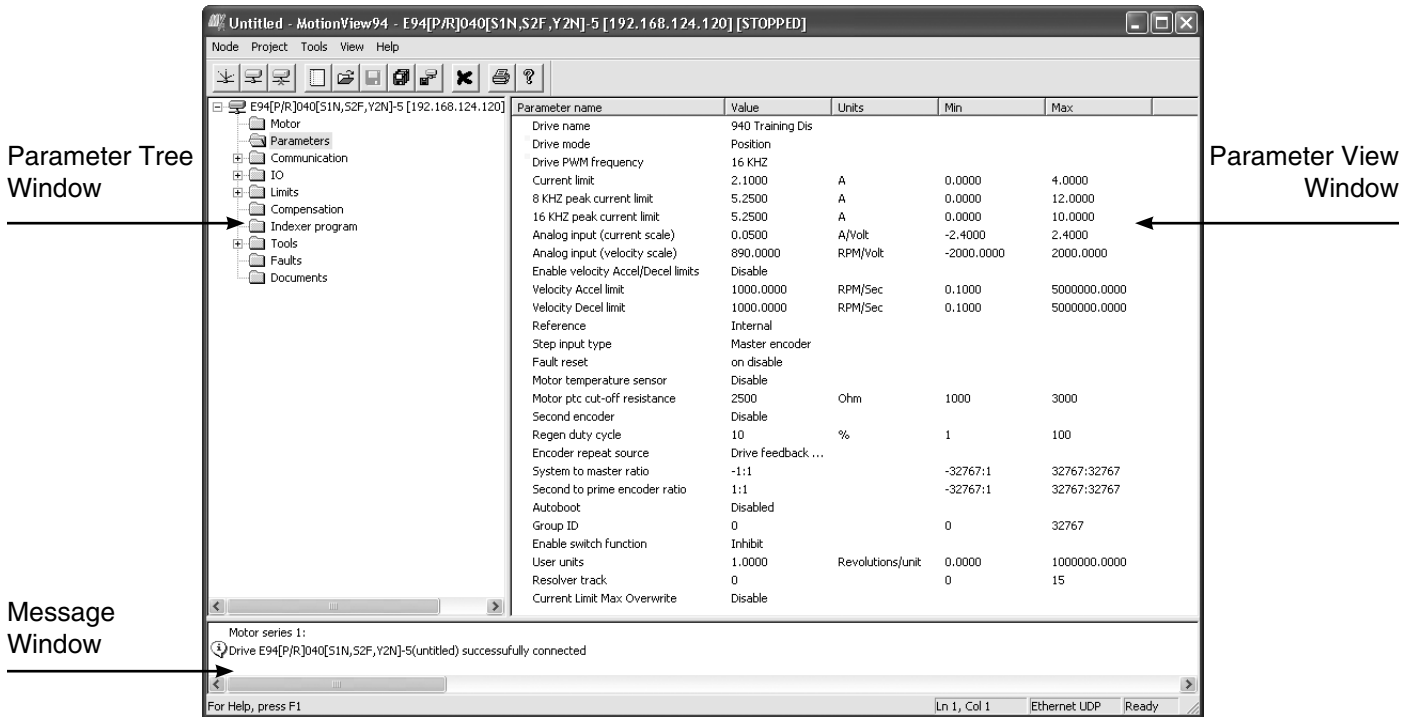


Figure 2: MotionView Parameters Display

MotionView is the universal programming software used to communicate with and configure the PositionServo drive. The MotionView platform is segmented into three windows. The first window is the "**Parameter Tree Window**". This window is used much like Windows Explorer. The various parameter groups for the drive are represented here as folders or files. Once the desired parameter group file is selected, all of the corresponding parameters within that parameter group will appear in the second window, the "**Parameter View Window**". The user can then enable, disable or edit drive features or parameters in the Parameter View window. The third window is the "**Message Window**". This window is located at the bottom of the screen and will display communication status and errors.

1.3.1 Main Toolbar

The functions of MotionView are accessible via the Main Toolbar as illustrated in Figure 3. If a function in a pull-down menu or an icon is greyed out that denotes the function is unavailable. A function may be unavailable because a drive is not physically connected to the network or the present set-up and operation of the drive prohibits access to that function.

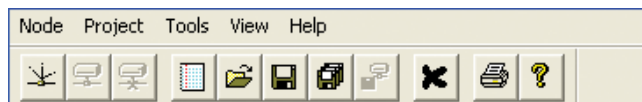













Figure 3: Main Menu and Toolbar

Introduction

Table 3a: Main Menu Text Pull-Down Folders

Main Menu				
Node	Project	Tools	View	Help
New configuration file	New project	Browse motor database	Toolbar	MotionView help
Open configuration file	Open project	Clear output window	Status Bar	Product manuals
Save configuration file	Close project			About MotionView
Load configuration file	Save project			Time stamp
Set all parameters to default	Save all configuration files			
Connect drive	Options			
Disconnect all connected drives	Connection setup			
Remove node from project	Recent file			

Table 3b: Main Menu Icon Functions

Icon	Function	Description
	Connect	Build a connection list of the drive(s) to communicate with on the network. Build the connection list by using any one of these three methods:
	Discover	[Discover] button discovers all drives on the network that are available for connectivity. Once drives have been discovered they are listed in the 'Connect to drive' list box. To connect one or a number of drives highlight their IP address in this window and press the [Connect] button. The 'Ctrl' key on your keyboard can be used to select multiple drives for connection.
	Connect	If the IP address on the drive is known, enter it in the IP address dialog box and then select [Connect] to access the drive.
	Find by name	If a drive has been assigned a specified "Drive Name", enter this name in the Name dialog box and then select [Find by name]. The IP address should then appear in the "Connect To Drive" list. The drive can now be connected by highlighting and pressing the [Connect] button.
	Connected	Drive is connected as a node on the network.
	Disconnect	Terminate connection to the drive selected (backlit) in the Node Tree.
	Add File	Import a configuration file to the drive
	Open File	Recall & open any previously saved configuration files and connection parameters.
	Save	Save the configuration file and the connection parameters of the drive selected in Node Tree.
	Save As	Save the configuration file and the connection parameters of the drive selected in Node Tree.
	Remove Node	Remove a node from the network
	Print	Print a report for the currently selected drive, containing all parameter set-up and programming information.
	Help	Open MotionView Help folder (from original installation CD)

Introduction

1.3.2 Program Toolbar

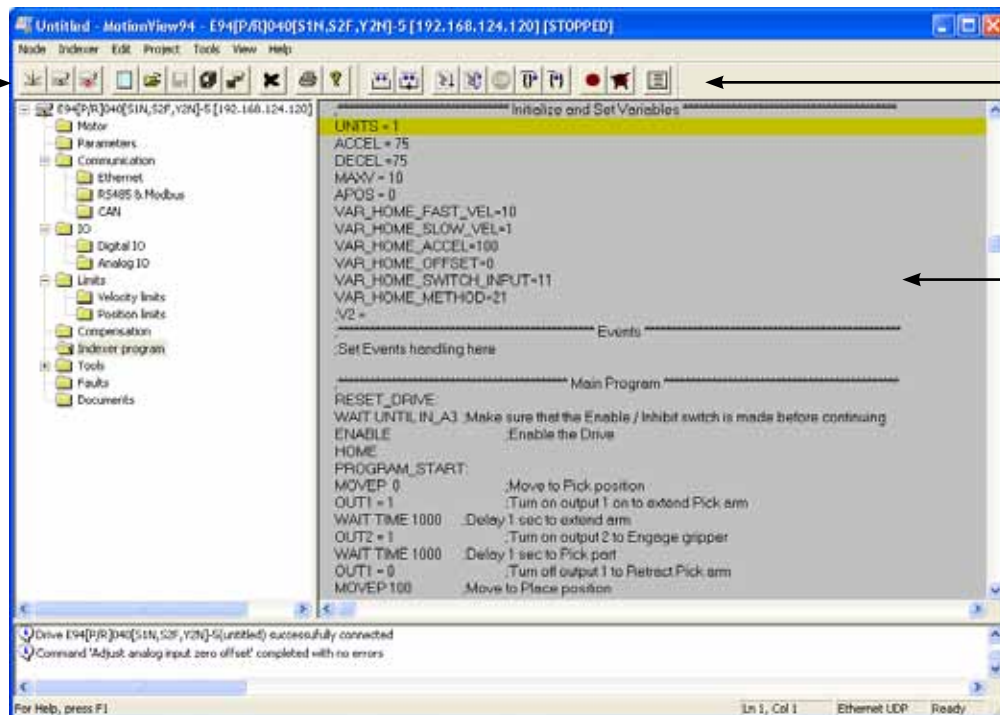
To view the Program Toolbar, click on the [Indexer Program] folder in the Node Tree. Click anywhere inside the gray Indexer program in the right-hand parameter window to bring up the program toolbar. This paragraph contains a brief description of the programming tools: Compile, Load with Source, Run, Reset, Stop, Step Over, Step Into, Set Breakpoint and Remove Breakpoint. For detailed descriptions of the program toolbar functions refer to paragraphs 1.3.3 and 1.4.



Figure 4: Program Toolbar

Icon	Function	Description
	Compile	Check compilation of the indexer program currently in the List View window.
	Compile & Load w/ Source	Load program including source code to the PositionServo drive listed in Node Tree.
	Run	Start / Continue program execution.
	Reset	Reset Drive. Disable drive, stop program execution, and return program processing to the beginning. Program will not restart program execution automatically.
	Stop	Stop program execution on completion of the current statement being executed. WARNING: Stop button does not place the drive in a disable state or prevent execution of motion commands waiting on the motion stack.
	Step Over	Execute each line of code in the program sequentially on each press on the [Step Over] button excluding subroutines.
	Step Into	Execute each line of code in the program sequentially on each press on the [Step Into] button, including subroutines.
	Set Breakpoint	Set breakpoint at current location of cursor in Indexer program.
	Remove Breakpoint	Remove breakpoint from current location of cursor in Indexer program.
	Watch Window	Display Parameter I/O window

Main
Toolbar



Program
Toolbar

User
Program
Area

Figure 5: MotionView - Indexer Program Display

1.3.3 MotionView Studio - Indexer Program

The MotionView Studio provides a tool suite used by MotionView to enter, compile, load and debug the user program. To view and develop the user program, select the [Indexer Program] folder in the Parameter (Node) Tree window. Once selected the program toolbar is displayed. The program displayed in the View window is uploaded from the drive when the connection is made between MotionView and the drive. This upload is always performed regardless of program running state. Click anywhere in the Parameter View Window to edit the Indexer program.

Common Programming Actions

Load User program from the PC to MotionView

- Select [**Indexer Program**] in the Node Tree.
- Select [**Import**] on the program toolbar.
Select the program to import from the PC folder where it is located. This procedure loads the program from the file to the editor window. It doesn't load the program to the drive's memory.

Compile program and **Load** to the drive

- Select [**Indexer Program**] in the Node Tree.
- Select [**Compile & Load W Source**] on the program toolbar to compile the program and load the source code and the compiled binary file to the PositionServo drive. The original source code contained in the drive can be viewed whenever the drive is accessed through MotionView and the Indexer Program folder is opened.
- Select [**Compile**] to check syntax errors without loading the program to drive. If the compiler finds any syntax error, compilation stops. Errors are reported in bottom portion of the screen in Message window.

Save User program from MotionView to PC.

- Select [**Indexer Program**] in the Node Tree.
- Select [**Export**] on the program toolbar.
The program will be saved to the Windows "My Documents" folder by default.

Run User program in drive.

- Select [**Indexer Program**] in the Node Tree.
- Select [**Run**] on the program toolbar.
If the program is already running, then first select [**Reset**] or [**Stop**] to stop the program.

Step Through the User program.

- Select [**Indexer Program**] in the Node Tree.
- Select [**Step**] or [**Step over**] on the program toolbar.
If [Step] is selected, the drive will execute the program one step at a time including subroutines. If [Step Over] is selected, the drive will execute the program one step at a time excluding subroutines. The program statement under execution will be highlighted. If the program is running, it will have to be either stopped or reset.

Set **Breakpoint(s)** in the program

- Select [**Indexer Program**] in the Node Tree.
- Place the cursor at the point in the program where the program will stop.
- Select [Set Breakpoint] or [Remove Breakpoint] on the program toolbar.
A convenient way to debug a user program is to insert breakpoints at critical junctions throughout the program. These breakpoints stop the drive from executing the program, but do not disable the drive and the position variables. Once the program has stopped, the user can continue to run the program, step through the program or reset the program.

Introduction

Stop program execution

- Select **[Indexer Program]** in the Node Tree.
- Select **[Stop]** on the program toolbar.

The program will stop after completing the current statement. Select **[Run]** to resume the program from the same point.



IMPORTANT!

The **[Stop]** button only stops the execution of the program code. It does **not** stop motion or disable the drive.

Restart Program execution

- Select **[Indexer Program]** in the Node Tree.
- Select **[Reset]** on the program toolbar.

The program will be reset and the drive will be disabled. Variables within the drive are not cleared (reset) when program execution is reset. It is important that any variables used by the programmer are set to safe values at the start of the user program.

1.4 Programming Basics

The user program consists of statements which when executed will not only initiate motion moves but also process the drives I/O and make decisions based on drive parameters. Before motion can be initiated, certain drive and I/O parameters must be configured. To configure these parameters perform the following procedure.

Parameter setup

Select **[Parameter]** folder in the Node Tree window and set the following parameters.

Set the “Drive” to “Position”:

- Select **[Drive mode]** from the Parameter View Window.
- Select **[Position]**, **[Velocity]**, or **[Torque]** from the drop down menu depending on the mode the drive is to be operated in. In order to execute the examples contained in this section of the manual the drive will need to be in **[Position]** mode.

Set the **[Reference]** to **[Internal]**:

- Select **[Reference]** from the Parameter View Window.
- Select **[Internal]** from the pull down menu to select the user program as the source of the Torque, Velocity, or Position Reference.

Set the **[Enable switch function]** to **[Inhibit]**:

- Select **[Enable switch function]** from the Parameter View Window.
- Select **[Inhibit]** from the menu to allow the user program control of the enable / disable status of the drive. Input A3 will now act as a hardware inhibit.

I/O Configuration

Input A3 is the Inhibit/Enable special purpose input. Refer to the PS User Manual (S94P01) for more information. Before executing any motion related statements, the drive must be enabled by executing “ENABLE” statement. “ENABLE” statement can only be accepted if input A3 is made. If at any time while drive is enabled A3 deactivates then the fault “F36” (“Drive Disabled”) will result. This is a hardware safety feature.

Introduction

Basic Motion Program

Select [**Indexer program**] from the Node Tree. The Parameter View window will display the current User Program stored in the drive. Note that if there is no valid program in the drive's memory the program area will be empty.



WARNING!

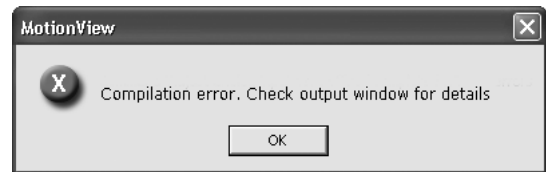
This program will cause motion. The motor should be disconnected from the application (free to rotate) or if a motor is connected, the shaft must be free to spin 10 revs forward and reverse from the location of the shaft at power up. Also, the machine must be capable of 10 RPS and an accel / decel of 5 RPSS.

In the program area, clear any existing program and replace it with the following program:

```
UNITS=1
ACCEL = 5
DECEL = 5
MAXV = 10
ENABLE
MOVED 10
MOVEDISTANCE -10
END
```



After the text has been entered into the program area, select the [Compile] icon from the toolbar. After compilation is done, a "Compilation Error" message should appear:



Click [OK] to dismiss the "Compilation error" dialog box. The cause of the compilation error will be displayed in the Message window, located at the bottom of the MotionView OnBoard window. MotionView will also highlight the program line where the error occurred.

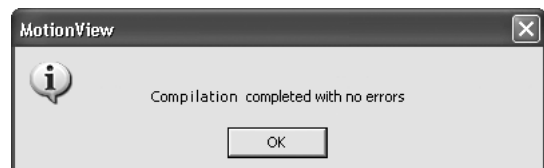
```
UNITS=1
ACCEL = 5
DECEL = 5
MAXV = 10 ;
ENABLE
MOVED 10 ;
MOVEDISTANCE -10
END
```

The problem in this example is that "**MOVEDISTANCE**" is not a valid command. Change the text "**MOVEDISTANCE**" to "**MOVED**".

```
UNITS=1
ACCEL = 5
DECEL = 5
ENABLE
MOVED 10
MOVED -10
END
```



After editing the program, select the [Compile] icon from the program toolbar. After compilation is done, the "Compilation Complete" message box should appear.



Introduction

The program has now been compiled without errors. Select [Compile & Load W Source] to load the program to the drive's memory. Click [OK] to dismiss the dialog box.



To **Run** the program, input A3 must be active to remove the hardware inhibit. Select the **[Run]** icon on the program toolbar. The drive will start to execute the User Program. The motor will spin 10 revolutions in the CCW direction and then 10 revolutions in the CW direction. After all the code has been executed, the program will stop and the drive will stay enabled.



To **Restart** the program, select the **[Reset]** icon on the program toolbar. This will disable the drive and reset the program to execute from the start. The program does not run itself automatically. To run the program again, select the **[Run]** icon on the toolbar.

Program Layout

When developing a program, structure is very important. It is recommended that the program be divided up into the following 7 segments:

- Header:** The header defines the title of the program, who wrote the program and description of what the program does. It may also include a date and revision number.
- I/O List:** The I/O list describes what the inputs and outputs of the drive are used for. For example input A1 might be used as a Start Switch.
- Init & Set Var:** Initialize and Set Variables defines the drives settings and system variables. For example here is where acceleration, deceleration and max speed might be set.
- Events:** An Event is a small program that runs independently of the main program. This section is used to define the Events.
- Main Program:** The Main Program is the area where the process of the drive is defined.
- Sub-Routines:** This is the area where any and all sub-routines should reside. These routines will be called out from the Main Program with a GOSUB command.
- Fault Handler:** This is the area where the Fault Handler code resides. If the Fault handler is utilized this code will be executed when the drive generates a fault.

The following is an example of a Pick and Place program divided up into the above segments.

```
***** HEADER *****
;Title:      Pick and Place example program
;Author:     Lenze - AC Technology
;Description: This is a sample program showing a simple sequence that
;            picks up a part, moves to a set position and places the part
;***** I/O List *****
;   Input A1 - not used
;   Input A2 - not used
;   Input A3 - Enable Input
;   Input A4 - not used
;   Input B1 - not used
;   Input B2 - not used
;   Input B3 - not used
;   Input B4 - not used
;   Input C1 - not used
;   Input C2 - not used
;   Input C3 - not used
;   Input C4 - not used
;   Output 1 - Pick Arm
;   Output 2 - Gripper
;   Output 3 - not used
;   Output 4 - not used
```

Introduction

```
;***** Initialize and Set Variables *****
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
;V1 =
;V2 =
;***** Events *****
;Set Events handling here
;No events are currently defined in this program
;***** Main Program *****
RESET_DRIVE:      ;Place holder for Fault Handler Routine
WAIT UNTIL IN_A3: ;Make sure that the Enable input is made before continuing
ENABLE           ;Enable output from drive to motor
PROGRAM_START:   ;Place holder for main program loop
MOVEP 0          ;Move to Pick position
OUT1 = 1         ;Turn on output 1 to extend Pick arm
WAIT TIME 1000   ;Delay 1 sec to extend arm
OUT2 = 1         ;Turn on output 2 to Engage gripper
WAIT TIME 1000   ;Delay 1 sec to Pick part
OUT1 = 0         ;Turn off output 1 to Retract Pick arm
MOVED -10        ;Move 10 REVs to Place position
OUT1 = 1         ;Turn on output 1 to extend Pick arm
WAIT TIME 1000   ;Delay 1 sec to extend arm
OUT2 = 0         ;Turn off output 2 to Disengage gripper
WAIT TIME 1000   ;Delay 1 sec to Place part
OUT1 = 0         ;Retract Pick arm
GOTO PROGRAM_START ;Loop back and continuous execute main program loop
END

;***** Sub-Routines *****
;Enter Sub-Routine code here
;***** Fault Handler Routine *****
;Enter Fault Handler code here
ON FAULT
;No Fault Handler is currently defined in this program
ENDFAULT
```

Saving Configuration File to PC

The “Configuration File” consists of all the parameter settings for the drive, as well as the User Program. Once you are done setting up the drive’s parameters and have written your User Program, you can save these setting to your computer. To save the settings, select **[Save All]** from the **Main** toolbar. Then simply assign your program a name, (e.g. Basic Motion), and click [Save] in the dialog box. The configuration file has a “.dcf” extension and by default will be saved to the “My Documents” folder.

Loading Configuration File to the Drive

There are times when it is desired to import (or export) the program to another drive. Other times the program was prepared off-line. In both scenarios, the program or configuration file needs to be loaded from the PC to the drive. To load the configuration file to the drive, select **[Load Configuration]** from the **Main** toolbar. Then simply select the program you want to load and click [Open] in the dialog box. MotionView will first compile the selected program. Once compiled, the [Compilation Complete] dialog box should appear. Click [OK] to dismiss this dialog box. MotionView will then load the selected file to the drive. When done, a “Parameters Successfully Loaded” or similar message will be displayed in the Message Window.

Introduction

Motion source (Reference)

The PositionServo can be set up to operate in one of three modes: Torque, Velocity, or Position. The drive must be given a command before it can initiate any motion. The source for commanding this motion is referred to as the "Reference". With the PositionServo you have two ways of commanding motion, or two types of References. When the drive's command signal is from an external source, for example a PLC or Motion Controller, it is referred to as an External Reference. When the drive is being given its command from the User program or through one of the system variables it is referred to as an Internal Reference.

Table 4: Setting the Reference

Mode	"Reference" Parameter Setting	
	External	Internal
Torque	Analog input AIN1	System variable "IREF"
Velocity	Analog input AIN1	System variable "IREF"
Position	Step/Direction Inputs Master Encoder Pulse Train Inputs User Program (Trajectory generator output)	User Program/Interface (Trajectory generator)

Units

All motion statements in the drive work with User units. The statement on the first line of the test program, UNITS=1, sets the relationship between User units and motor revolutions. For example, if UNITS=0.5 the motor will turn 1/2 of a revolution when commanded to move 1 Unit. When the UNITS variable is set to zero, the motor will operate with encoder counts as User units.

Time base

Time base is always in seconds i.e. all time-related values are set in USER UNITS/SEC.

Enable/Disable/Inhibit drive

Set "Enable switch function" to "Run".

When the "Enable switch function" parameter is set to Run, and the Input A3 is made, the drive will be enabled. Likewise, toggling input A3 to the off state will disable the drive.

- Select "**Parameter**" from the Parameter Tree Window.
- Select "**Enable switch function**" from the Parameter View Window.
- Select "**Run**" from the popup menu. This setting is primarily used when operating without any user's program in torque or velocity mode or as position follower with Step&Direction/Master Encoder reference.

Set "Enable switch function" to "Inhibit".

In the example of the Enable switch function being set to Run the decision on when to enable and disable the drive is determined by an external device, PLC or motion controller. The PositionServo's User Program allows the programmer to take that decision and incorporate it into the drive's program. The drive will execute the User Program whether the drive is enabled or disabled, however if a motion statement is executed while the drive is disabled, the F36 fault will occur. When the "**Enable switch function**" parameter is set to **Inhibit**, and Input A3 is on, the drive will be disabled and remain disabled until the ENABLE statement is executed by the User Program.

- Select "**Parameter**" from the Parameter Tree Window.
- Select "**Enable switch function**" from the Parameter View Window.
- Select "**Inhibit**" from the popup menu.

Faults

When a fault condition has been detected by the drive, the following actions will occur:

- Drive will Immediately be placed in a Disabled Condition.
- Motion Stack will be flushed of any Motion Commands
- Execution of the user program will be terminated and program control will be handed over to the Fault Handler section. If no Fault handler is described then program execution will terminate. See fault handler section.
- A fault code defining the nature of the drive trip will be written to the DFAULTS system variable and can be accessed by the fault handler. Refer to section 2.13 for a list of fault codes.
- The fault code will be displayed on the drive display.
- Dedicated “Ready” output will turn off.
- Any Output with assigned special function “Fault” will turn on.
- Any Output with assigned special function “ready/enabled” will turn off.
- The “enable” status indicator on the drive display will turn off indicating drive in disabled state.

Clearing a fault condition can be done in one of the following ways:



- Select the [**Reset**] button from the toolbar.
- Execute the **RESUME** statement at the end of the Fault Handler routine (see Fault Handler example).
- Send “Reset” command over the Host Interface.
- Cycle power (hard reset).

Fault Handler

The Fault Handler is a code segment that will be executed when the drive is experiencing a fault. The fault handler allows the programmer to analyze the type of fault and define a recovery process for the drive and permits the continuation of program execution. While the drive is executing the Fault Handler Routine the drive is disabled and therefore will not be able to detect any additional faults that might occur. Fault handler code should be treated as the drive’s first reaction on fault. While it executes, the drive will not respond to any I/O, interface commands etc. Therefore the user should use the fault handler to manipulate time critical and safety related I/O and variables and then exit the Fault Handler Routine by executing a “**RESUME**” statement for a full stop after statement. The Resume statement permits program execution to leave the fault handler and resume back in the main program section of the user code. Use the Resume statement to jump back to a section of the main program that designates the recovery process for the fault. Waiting in Fault handler for I/O state change or for interface command is not allowed. Do that in the code where you point the “RESUME” statement.

Without Fault Handler

To simulate a fault, restart the Pick and Place example program. While the program is running, switch the ENABLE input IN_A3 to the off state. This will cause the drive to generate an F_36 fault (Drive Disabled) and put the drive into Fault Mode. While the drive is in Fault Mode, any digital output currently active will remain active and any output deactivated will remain deactivated, excluding the dedicated ready output and any output that has been assigned special functionality. The program execution will stop and any motion moves will be terminated. In this example the Pick and Place arm may not be in a desired location when the program goes into the fault mode.

Introduction

With Fault Handler

Add the following code to the end of your sample program. While the program is running, switch the ENABLE input IN_A3, to the off state. This will cause the drive to generate an F_36 fault (Drive Disabled) and put the drive into a Fault Mode. From this point the Fault Handler Routine will take over.

```
F_PROCESS:
WAIT UNTIL IN_A4==1 ;Wait until reset switch is made
WAIT UNTIL IN_A4==0 ;and then released before
GOTO RESET_DRIVE ;returning to the beginning of the program
END
;***** Sub-Routines *****
Enter Sub-Routines here;
;***** Fault Handler Routine *****
ON FAULT ;Statement starts fault handler routine
;Motion stopped, drive disabled, and events no longer
;scanned while executing the fault handler routine.
OUT2 = 0 ;Output 1 off to Disengage gripper.
;This will drop the part in the gripper
OUT1 = 0 ;Retract Pick arm to make sure it is up and out of the way
RESUME F_PROCESS ;program restarts from label F_PROCESS
ENDFAULT ;fault handler MUST end with this statement
```



NOTE

The following statements can not be used inside the Fault Handler Routine:

- ENABLE
- WAIT
- MOVE
- MOVED
- MOVEP
- MOVEDR
- MOVEPR
- MDV
- MOTION SUSPEND
- MOTION RESUME
- GOTO, GOSUB
- JUMP
- ENABLE
- VELOCITY ON/OFF

Refer to section 2.1 for additional details and the Language Reference section for the statement "ON FAULT/ENDFAULT".

1.5 Using Advanced Debugging Features

To debug a program or view the I/O, open the Diagnostic window by clicking on the [Tools] in the Parmeter (Node) Tree list then click on the [Parameter & I/O View] button. The Diagnostic window will open. This window allows the programmer to monitor and set variables, and to view status of drive digital inputs and outputs.

Click on a variable in the variable list on the right-hand side to select that parameter



Use the left arrow button to add variables after selecting a variable.



Use the right arrow button to remove variables after selecting a variable.



Use the double right arrow button to remove all variables in left-hand Diagnostic window.



Use the [R] (Refresh) button to refresh variable values.

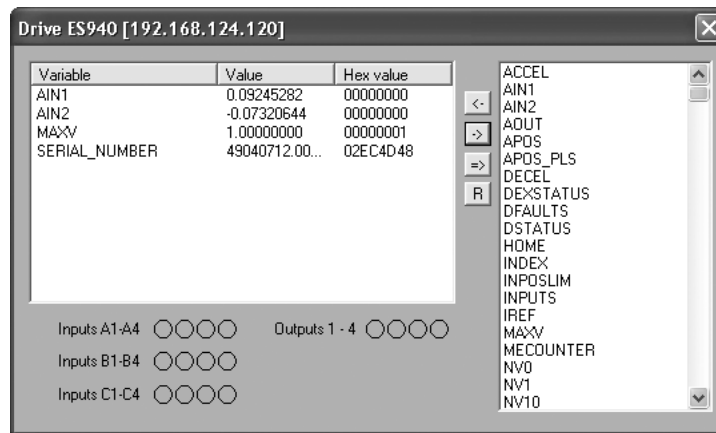


Figure 6: Variable Diagnostic Display



NOTE

Write-only variables cannot be read. Attempts to either display a write-only variable in the diagnostic window or to read a write-only variable via network communications can show erroneous data.

1.6 Inputs and Outputs

Analog Input and Output

- The PositionServo has two analog inputs. These analog inputs are utilized by the drive as System Variables and are labeled “AIN1” and “AIN2”. Their values can be directly read by the User Program or via a Host Interface. Their value can range from -10 to +10 and correlates to ± 10 volts analog input.
- The PositionServo has one analog output. This analog output is utilized by the drive as a System Variable and is labeled “AOUT”. It can be directly written by the User Program or via a Host Interface. Its value can range from -10 to +10 which correlates to ± 10 volts analog input.



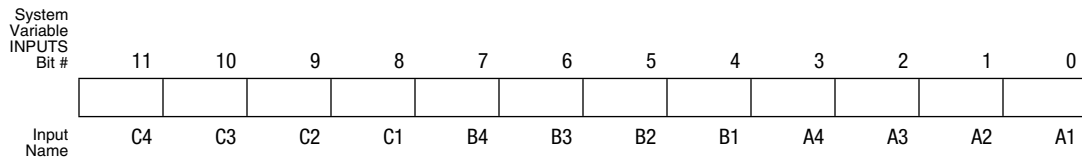
NOTE

If an analog output is assigned to any special function from MotionView, writing to AOUT from the User Program will not change its value. If an analog output is set to “Not assigned” then it can be controlled by writing to the AOUT variable.

Introduction

Digital Inputs

- The PositionServo has twelve digital inputs that are utilized by the drive for decision making in the User Program. Example uses: travel limit switches, proximity sensors, push buttons and hand shaking with other devices.
- Each input can be assigned an individual debounce time via MotionView. From the **Parameter Tree**, select [IO]. Then select the [**Digital Input**] folder. The debounce times will be displayed in the **Parameter View Window**. Debounce times can be set between 0 and 1000 ms (1ms = 0.001 sec). Debounce times can also be set via variables in the user program.
- The twelve inputs are separated into three groups: A, B and C. Each group has four inputs and share one common: Acom, Bcom and Ccom respectfully. The inputs are labeled individually as **IN_A1 - IN_A4, IN_B1 - IN_B4 and IN_C1 - IN_C4**.
- In addition to monitoring each input individually, the status of all twelve inputs can be represented as one binary number. Each input corresponds to 1 bit in the INPUTS system variable. Use the following format:



- Some inputs can have additional special functionality such as Travel Limit switch, Enable input, and Registration input. Configuration of these inputs is done from MotionView or through variables in the user program. Input special functionality is summarized in the table below and in the following sections. The current status of the drive's inputs is available to the programmer through dedicated System Flags or as bits of the System Variable INPUTS. Table 5 summarizes the special functions for the inputs.

Table 5: Input Functions

Input	Special Function
Input A1	negative limit switch
Input A2	positive limit switch
Input A3	Inhibit/Enable input
Input A4	N/A
Input B1	N/A
Input B2	N/A
Input B3	N/A
Input B4	N/A
Input C1	N/A
Input C2	N/A
Input C3	Registration sensor input
Input C4	N/A

Read Digital Inputs

The Pick and Place example program has been modified below to utilize the “WAIT UNTIL” inputs statements in place of the “WAIT TIME” statements. **IN_A1** and **IN_A4** will be used as proximity sensors to detect when the pick and place arm is extended and when it is retracted. When the arm is extended, **IN_A1** will be in an ON state and will equal “1”. When the arm is retracted, **IN_A4** will be in an ON state and will equal “1”.

```
;***** Main Program *****
RESET_DRIVE:                ;Place holder for Fault Handler Routine
WAIT UNTIL IN_A3            ;Make sure that the Enable input is made before continuing
ENABLE
PROGRAM_START:
WAIT UNTIL IN_A4==1        ;Make sure Arm is retracted
MOVEP 0                    ;Move to Pick position
OUT1 = 1                   ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1        ; Arm extend
OUT2 = 1                   ;Turn on output 2 to Engage gripper
WAIT TIME 1000             ;Delay 1 sec to Pick part
OUT1 = 0                   ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1        ;Make sure Arm is retracted
MOVED -10                  ;Move 10 REVs to Place position
OUT1 = 1                   ;Turn on output 1 on to extend Pick arm
WAIT UNTIL IN_A1==1        ; Arm is extended
OUT2 = 0                   ;Turn off output 2 to Disengage gripper
WAIT TIME 1000             ;Delay 1 sec to Place part
OUT1 = 0                   ;Retract Pick arm
WAIT UNTIL IN_A4==1        ;Arm is retracted
GOTO PROGRAM_START
END
```

Once the above modifications have been made, export the program to file and save it as “Pick and Place with I/O”, then compile, download and test the program.

ASSIGN & INDEX - Using inputs to generate predefined indexes

“INDEX” is a variable on the drive that can be configured to represent a certain group of inputs as a binary number. “ASSIGN” is the command that designates which inputs are utilized and how they are configured.

Below the Pick and Place program has been modified to utilize this “INDEX” function. The previous example program simply picked up a part and moved it to a place location. For demonstration purposes we will add seven different place locations. These locations will be referred to as Bins. What Bin the part is placed in will be determined by the state of three inputs, B1, B2 and B3.

Bin 1	-	Input B1 is made
Bin 2	-	Input B2 is made
Bin 3	-	Inputs B1 and B2 are made
Bin 4	-	Input B3 is made
Bin 5	-	Inputs B1 and B3 are made
Bin 6	-	Inputs B2 and B3 are made
Bin 7	-	Inputs B1, B2 and B3 are made

The “ASSIGN” command is used to assign the individual input to a bit in the “INDEX” variable. ASSIGN INPUT <input name> AS BIT <bit #>

```
;***** Initialize and Set Variables *****
ASSIGN INPUT IN_B1 AS BIT 0 ;Assign the Variable INDEX to equal 1 when IN_B1 is made
ASSIGN INPUT IN_B2 AS BIT 1 ;Assign the Variable INDEX to equal 2 when IN_B2 is made
ASSIGN INPUT IN_B3 AS BIT 2 ;Assign the Variable INDEX to equal 4 when IN_B4 is made
```

Introduction

Table 6: Bin Location, Inputs & Index Values

Bin Location	Input State	INDEX Value
Bin 1	Input B1 is made	1
Bin 2	Input B2 is made	2
Bin 3	Inputs B1 and B2 are made	3
Bin 4	Input B3 is made	4
Bin 5	Inputs B1 and B3 are made	5
Bin 6	Inputs B2 and B3 are made	6
Bin 7	Inputs B1, B2 and B3 are made	7

The Main program has been modified to change the end place position based on the value of the “INDEX” variable.

```

;***** Main Program *****
ENABLE
PROGRAM_START:
WAIT UNTIL IN_A4==1      ;Make sure Arm is retracted
MOVEP 0                  ;Move to (ABS) to Pick position
OUT1 = 1                 ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1      ;Arm extends
OUT2 = 1                 ;Turn on output 2 to Engage gripper
WAIT TIME 1000           ;Delay 1 sec to Pick part
OUT1 = 0                 ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==0      ;Make sure Arm is retracted

IF INDEX==1              ;In this area we use the If statement to
GOTO BIN_1               ;check and see what state inputs B1, B2 & B3
ENDIF                    ;are in.
IF INDEX==2              ;   INDEX = 1 when input B1 is made
GOTO BIN_2               ;   INDEX = 2 when input B2 is made
ENDIF                    ;   INDEX = 3 when input B1 & B2 are made.
.                         ;   INDEX = 4 when input B3 is made
.                         ;   INDEX = 5 when input B1 & B3 are made.
.                         ;   INDEX = 6 when input B2 & B3 are made.
IF INDEX==7              ;   INDEX = 7 when input B1, B2 & B3 are made
GOTO BIN_7               ;We can now direct the program to one of seven
ENDIF                    ;locations based on three inputs.

BIN_1:                   ;Set up for Bin 1
MOVEP 10                 ;Move to Bin 1 location
GOTO PLACE_PART          ;Jump to place part routine
BIN_2:                   ;Set up for Bin 2
MOVEP 20                 ;Move to Bin 2 location
GOTO PLACE_PART          ;Jump to place part routine
BIN_7:                   ;Set up for Bin 7
MOVEP 70                 ;Move to Bin 7 location
GOTO PLACE_PART          ;Jump to place part routine
PLACE_PART:
OUT1 = 1                 ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A4 == 1    ;Arm extends
OUT2 = 0                 ;Turn off output 2 to Disengage gripper
WAIT TIME 1000           ;Delay 1 sec to Place part
OUT1 = 0                 ;Retract Pick arm
WAIT UNTIL IN_A4 == 0    ;Arm is retracted
GOTO PROGRAM_START
END

```



NOTE

Any one of the 12 inputs can be assigned as a bit position within the INDEX variable. Only bits 0 through 7 can be used with the INDEX variable. Bits 8-31 are not used and are always set to 0. Unassigned bits in the INDEX variable are set to 0.

BITS 8-31 (not used)	A1	0	A2	A4	0	0	0	0
----------------------	----	---	----	----	---	---	---	---

Limit Switch Input Functions

Inputs A1 and A2 can be configured as special purpose inputs from the **[Digital IO]** folder in MotionView. They can be set to one of three settings:

- The **“Not assigned”** setting designates the inputs as general purpose inputs which can be utilized by the User Program.
- The **“Fault”** setting will configure A1 and A2 as Hard Limit Switches. When either input is made the drive will be disabled, the motor will come to an uncontrolled stop, and the drive will generate a fault. If the negative limit switch is activated, the drive will display an F-33 fault. If the positive limit switch is activated the drive will display an F32 fault.
- The **“Stop and fault”** setting will configure A1 and A2 as End of Travel limit switches. When either input is made the drive will initiate a rapid stop before disabling the drive and generating an F34 or F35 fault (refer to section 2.15 for details). The speed of the deceleration will be set by the value stored in the **“QDECEL”** System Variable.



NOTE

The “Stop and Fault” function is available in position mode only, (“Drive mode” is set to “Position”). In all other cases, the Stop and Fault function will act the same as the Fault function.

To set this parameter, select the **[IO]** folder from the Parameter Tree. Then select the **[Digital IO]** folder. From the Parameter View Window, use the pull-down menu next to **[Hard Limit Switches Action]** to select the status: Not Assigned, Fault or Stop and Fault.

Digital Outputs Control

- The PositionServo has 5 digital outputs. The **“RDY”** or READY output is dedicated and will only come on when the drive is enabled, i.e. in **RUN** mode. The other outputs are labeled **OUT1 - OUT4**.
- Outputs can be configured as Special Purpose Outputs. If an output is configured as a **Special Purpose Output** it will activate when the state assigned to it becomes true. For example, if an output is assigned the function **“Zero speed”**, the assigned output will come on when the motor is not in motion. To configure an output as a Special Purpose Output, select the **[IO]** folder from the Parameter Tree. Then select the **[Digital IO]** folder. From the Parameter View Window, select the **“Output function”** parameter you wish to set (1, 2, 3 or 4).
- Outputs that are configured as “Not assigned” can be activated either via the User Program or from a host interface. If an output is assigned as a Special Purpose Output, neither the user program nor the host interface can overwrite its status.
- The Systems Variable **“OUTPUTS”** is a read/write variable that allows the User Program, or host interface, to monitor and set the status of all four outputs. Each output allocates 1 bit in the OUTPUTS variable. For example, if you set this variable equal to 15 in the User Program, i.e. 1111 in binary format, then all 4 outputs will be turned on.
- The example below summarizes the output functions and corresponding System Flags. To set the output, write any non-0 value (TRUE) to its flag. To clear the output, write a 0 value (FALSE) to its flag. You can also use flags in an expression. If an expression is evaluated as TRUE then the output will be turned ON. Otherwise, it will be turned OFF.

```
OUT1 = 1           ;turn OUT1 ON
OUT2 = 10          ;any value but 0 turns output ON
OUT3 = 0           ;turn OUT3 OFF
OUT2 = APOS>3 && APOS<10 ;ON when position within window, otherwise OFF
```

Introduction

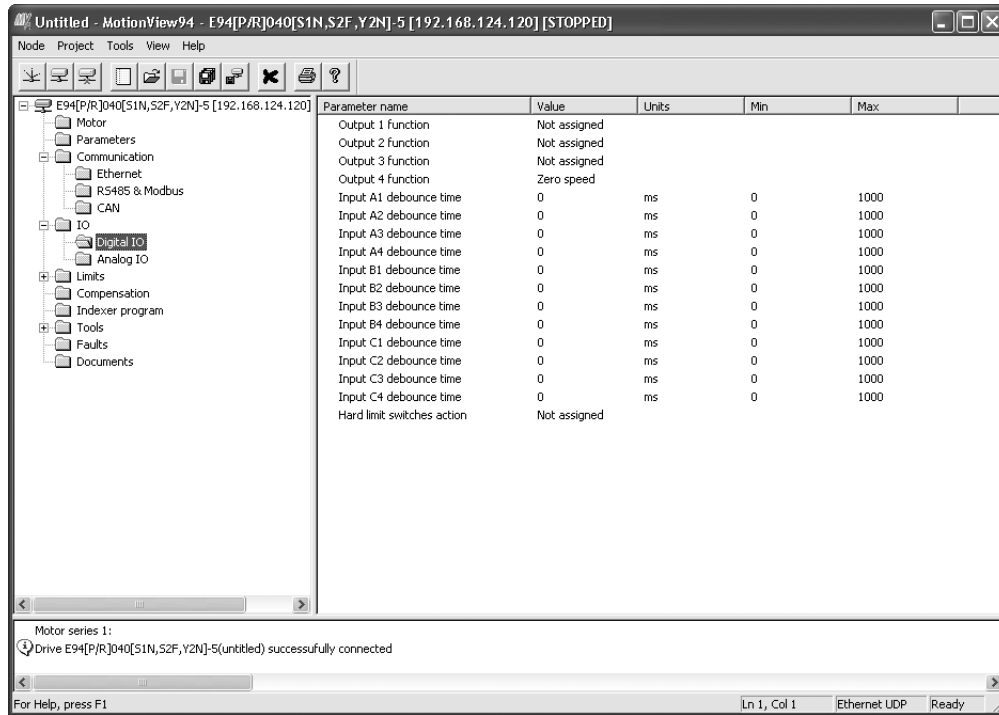


Figure 7: Digital IO Folder

1.7 Events

A Scanned Event is a small program that runs independently of the main program. An event statement establishes a condition that is scanned on a regular basis. Once established, the scanned event can be enabled and disabled in the main program. If condition becomes true and EVENT is enabled, the code placed between EVENT and ENDEVENT executes. Scanned events are used to trigger the actions independently of the main program.

In the following example the Event “**SPRAY_GUNS_ON**” will be setup to turn Output 3 on when the drive’s position becomes greater than 25. Note: the event will be triggered only at the instant when the drive position becomes greater than 25. It will not continue to execute while the position is greater than 25. (i.e. the event is triggered by the transition in logic from false to true). Note also that main program doesn’t need to be interrupted to perform this action.

```
;***** EVENT SETUP *****
EVENT SPRAY_GUNS_ON APOS>25 ;Event will trigger as position passes 25 in pos dir.
OUT3=1 ;Turn on the spray guns (out 3 on)
ENDEVENT ;End event
;*****
```

Enter the Event code in the EVENT SETUP section of the program. To Setup an Event, the “**EVENT**” command must be entered. This is followed by the Event Name “**SPRAY_GUNS_ON**” and the triggering mechanism, “**APOS>25**”. After that a sequence of programming statements can be entered once the event is triggered. In our case, we will turn on output 3. To end the Event, the “**ENDEVENT**” command must be used. Events can be activated (turned on) and deactivated (turned off) throughout the program. To turn on an Event, the “**EVENT**” command is entered, followed by the Event Name “**SPRAY_GUNS_ON**”. This is completed by the desired state of the Event, “**ON**” or “**OFF**”. Refer to Section 2.10 for more on Scanned Events.

```
;*****
EVENT SPRAY_GUNS_ON ON ;Enable 'spray guns on' event
;*****
```

Two Scanned Events have been added to the Pick and Place program below to trigger a spray gun on and off. The Event will be triggered after the part has been picked up and is passing in front of the spray guns (position greater than 25). Once the part is in position, output 3 is turned on to activate the spray guns. When the part has passed by the spray guns, (position greater than 75), output 3 is turned off, deactivating the spray guns.

Introduction

```
;***** Events *****
EVENT  SPRAY_GUNS_ON  APOS>25  ;Event will trigger as position passes 25 in pos dir.
OUT3=1  ;Turn on the spray guns (out 3 on)
ENDEVENT ;End event
EVENT  SPRAY_GUNS_OFF APOS>75  ;Event will trigger as position passes 75 in pos dir.
OUT3=0  ;Turn off the spray guns (out 3 off)
ENDEVENT ;End event
;***** Main Program *****
PROGRAM_START: ;Place holder for main program loop
ENABLE ;Enable output from drive to motor
EVENT  SPRAY_GUNS_ON  ON ;Enable 'spray guns on' event
EVENT  SPRAY_GUNS_OFF ON ;Enable 'spray guns off' event
WAIT UNTIL IN_A4==1 ;Make sure Arm is retracted
MOVEP 0 ;Move to Pick position
OUT1 = 1 ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1 ;Arm extends
OUT2 = 1 ;Turn on output 2 to Engage gripper
WAIT TIME 1000 ;Delay 1 sec to Pick part
OUT1 = 0 ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1 ;Make sure Arm is retracted
MOVEP 100 ;Move to Place position
OUT1 = 1 ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1 ;Arm extends
OUT2 = 0 ;Turn off output 2 to Disengage gripper
WAIT TIME 1000 ;Delay 1 sec to Place part
OUT1 = 0 ;Retract Pick arm
WAIT UNTIL IN_A4==1 ;Arm is retracted
GOTO PROGRAM_START ;Loop back and continuously execute main program loop
END
```

1.8 Variables and Define Statement

In the previous program for the pick and place machine constants were used for position limits to trigger the event and turn the spray gun ON and OFF. If limits must be calculated based on some parameters unknown before the program runs (like home origin, material width, etc.), then use the User Variables. The PositionServo provides 32 User Variables V0-V31 and 32 User Network Variables NV0-NV31. In the program below, the limit APOS (actual position) is compared to V1 for an ON event and V2 for an OFF event. The necessary limit values could be calculated earlier in the program or supplied by an HMI or host PC.

The DEFINE statement can be used to assign a name to a constant, variable or drive Input/Output. In the program below, constants 1 and 0 are defined as Output_On and Output_Off. DEFINE is a pseudo statement, i.e it is not executed by the program interpreter, but rather substitutes expressions in the subsequent program at the time of compilation.

```
DEFINE Value2      2
DEFINE Value10     10
V1 = Value2+Value10 ; result is 12
V1 = 2+10           ; does exactly same as above, the result is 12
```

Introduction

```
;***** Initialize and Set Variables *****
UNITS = 1 ;Define units for program, 1=revolution of motor shaft
ACCEL = 5 ;Set acceleration rate for motion command
DECEL = 5 ;Set deceleration rate for motion command
MAXV = 10 ;Maximum velocity for motion commands
V1 = 25 ;Set Variable V1 equal to 25
V2 = 75 ;Set Variable V2 equal to 75
DEFINE Output_On 1 ;Define Name for output On
DEFINE Output_Off 0 ;Define Name for output Off
;***** EVENTS *****
EVENT SPRAY_GUNS_ON APOS > V1 ;Event will trigger as position passes 25 in pos dir.
OUT3= Output_On ;Turn on the spray guns (out 3 on)
ENDEVENT ;End event

EVENT SPRAY_GUNS_OFF APOS > V2 ;Event will trigger as position passes 75 in pos dir.
OUT3= Output_Off ;Turn off the spray guns (out 3 off)
ENDEVENT ;End even
;***** Main Program *****
PROGRAM_START: ;Place holder for main program loop
ENABLE ;Enable output from drive to motor
EVENT SPRAY_GUNS_ON ON ;Enable the 'spray guns on' event
EVENT SPRAY_GUNS_OFF ON ;Enable the 'spray guns off' event
WAIT UNTIL IN_A4==1 ;Ensure Arm is retracted before running the program
MOVEP 0 ;Move to position 0 to pick part
OUT1 = Output_On ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1 ;Check input to make sure Arm is extended
OUT2 = Output_On ;Turn on output 2 to Engage gripper
WAIT TIME 1000 ;Delay 1 sec to Pick part
OUT1 = Output_Off ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1 ;Check input to make sure Arm is retracted
MOVED 100 ;Move to Place position
OUT1 = Output_On ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1 ;Check input to make sure Arm is extended
OUT2 = Output_Off ;Turn off output 2 to Disengage gripper
WAIT TIME 1000 ;Delay 1 sec to Place part
OUT1 = Output_Off ;Retract Pick arm
WAIT UNTIL IN_A4==1 ;Check input to make sure Arm is retracted
GOTO PROGRAM_START ;Loop back and continuously execute main program loop
END
```

1.9 IF/ELSE Statements

An IF/ELSE statement allows the user to execute one or more statements conditionally. The programmer can use an IF or IF/ELSE construct:

Single IF example:

This example increments a counter, Variable "V1", until the Variable, "V1", is greater than 10.

Again:

```
V1=V1+1
IF V1>10
    V1=0
ENDIF
GOTO Again
END
```

IF/ELSE example:

This example checks the value of Variable V1. If V1 is greater than 3, then V2 is set to 1. If V1 is not greater than 3, then V2 is set to 0.

```

IF V1>3
    V2=1
ELSE
    V2=0
ENDIF
    
```

Whether you are using an IF or IF/ELSE statement the construct must end with ENDIF keyword.

1.10 Motion

Figure 8 illustrates the Position and Velocity regulator of the PositionServo drive.

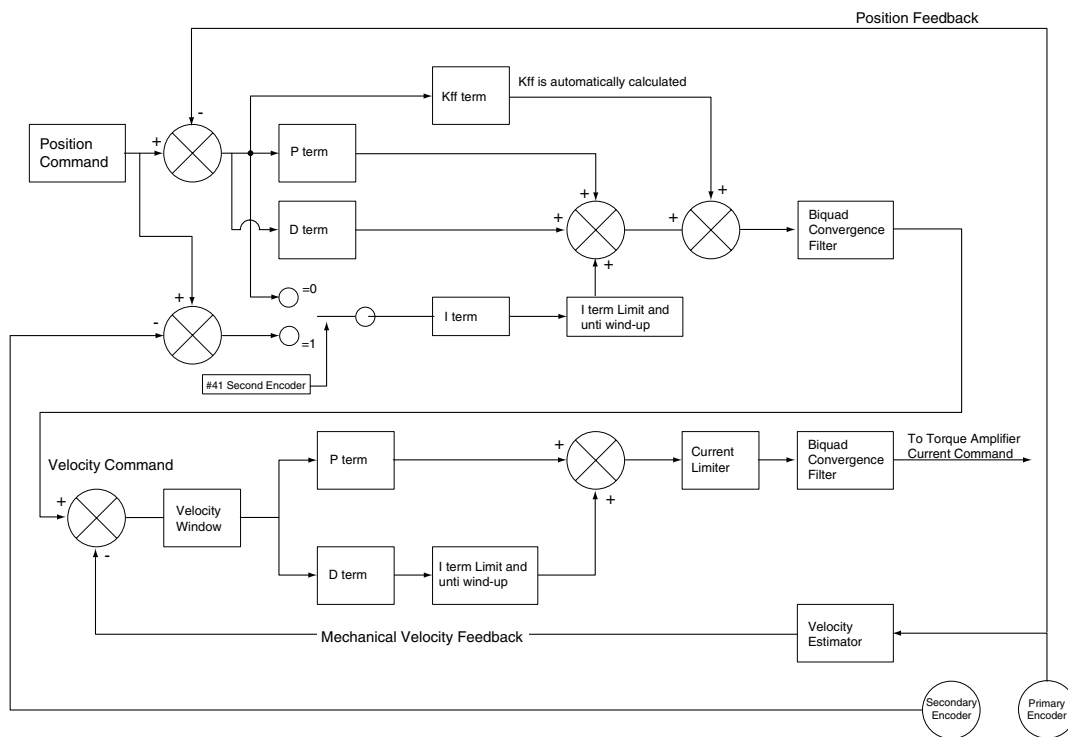


Figure 8: PositionServo Position and Velocity Regulator's Diagram

The “**Position Command**”, as shown in the regulator’s diagram (Figure 9), is produced by a **Trajectory Generator**. The Trajectory Generator processes the motion commands produced by the User’s program to calculate the position increment or decrement, also referred to as the “index” value, for every servo loop. This calculated target (or theoretical) position is then supplied to the **Regulator** input.

The main purpose of the **Regulator** is to set the motors position to match the target position created by the Trajectory Generator. This is done by comparing the input from the Trajectory Generator with the position feedback from the encoder or resolver, to control the torque and velocity of the motor. Of course there will always be some error in the position following. Such error is referred to as “Position Error” and is expressed as follows:

$$\text{Position Error} = \text{Target Position} - \text{Actual Position}$$

When the actual Position Error exceeds a certain threshold value a “Position Error limit”, fault (F_PE) will be generated. The Position Error limit and Position Error time can be set under the Node Tree “Limits”/ “Position Limits” in MotionView. The Position Error time specifies how long the actual position error can exceed the Position Error limit before the fault is generated.

Introduction

1.10.1 Drive Operating Modes

There are three modes of operation for the PositionServo: Torque, Velocity and Position. Torque and Velocity modes are generally used when the command reference is from an external device, (Ain). Position mode is used when the command comes from the drives User Program, or from an external device, encoder or a step and direction pulse. Setting the drive's mode is done from the [Parameter] folder in MotionView. To command motion from the user program the drive must be configured to internal reference mode. When the drive is in position mode, it can be placed into a velocity mode without the need to change operating mode to 'Velocity'. Velocity profiling from Positioning mode can be turned on and off from the User Program. Executing the VELOCITY ON statement is used to activate this mode while VELOCITY OFF will deactivate this mode. This mode is used for special case indexing moves. Velocity mode is the mode when the target position is constantly advanced with a rate set in the VEL system variable. The Reference arrangements for the different modes of operation are illustrated in Figure 9.

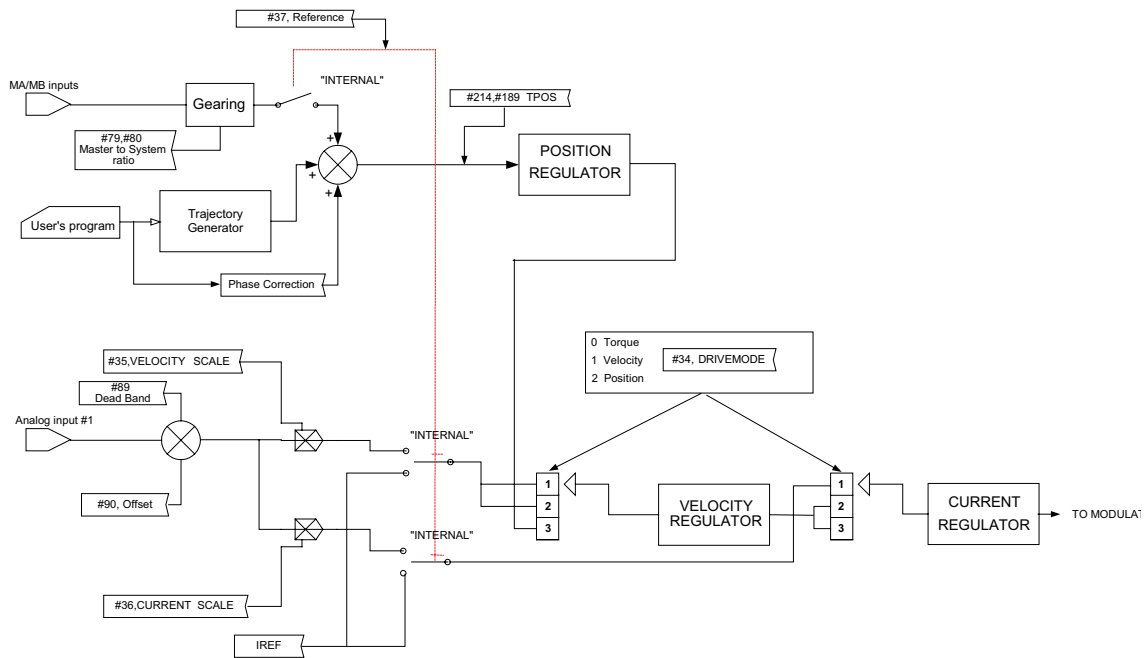


Figure 9: Reference Arrangement Diagram

1.10.2 Point To Point Moves

The PositionServo supports two types of moves: absolute and incremental. The statement MOVEP (Move to Position) is used to make an absolute move. When executing an absolute move, the motor is instructed to move to a known position. The move to this known position is always referenced from the motor's "home" or "zero" location. For example, the statement (MOVEP 0) will cause the motor to move to its zero or home position, regardless of where the motor is located at the beginning of the move. The statement MOVED (Move Distance) makes incremental, (or relative), moves from its current position. For example, MOVED 10, will cause the motor to move forward 10 user units from its current location.

MOVEP and MOVED statements generate what is called a trapezoidal point to point motion profile. A trapezoidal move is when the motor accelerates, using the current acceleration setting, (ACCEL), to a default top speed, (MAXV), it then maintains that speed for a period of time before decelerating to the end position using the deceleration setting, (DECCEL). If the distance to be moved is fairly small, a triangular move profile will be used. A triangular move is a move that starts to accelerate toward the Max Velocity setting but has to decelerate before ever achieving the max velocity in order to reach the desired end point.

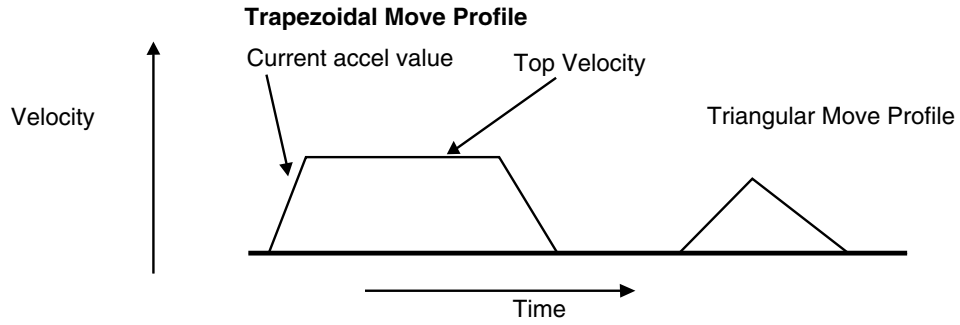


Figure 10: Trapezoidal Move

1.10.3 Segment Moves

MOVED and MOVEP commands facilitate simple motion to be commanded, but if the required move profile is more complex than a simple trapezoidal move, then the segment move MDV can be used.

The profile shown in Figure 11 is divided into 8 segments or 8 MDV moves. An MDV move (Move Distance Velocity) has two arguments. The first argument is the distance moved in that segment. This distance is referenced from the motor's current position in User Units. The second argument is the desired target velocity for the end of the segment move. That is the velocity at which the motor will run at the moment when the specified distance in this segment is completed.

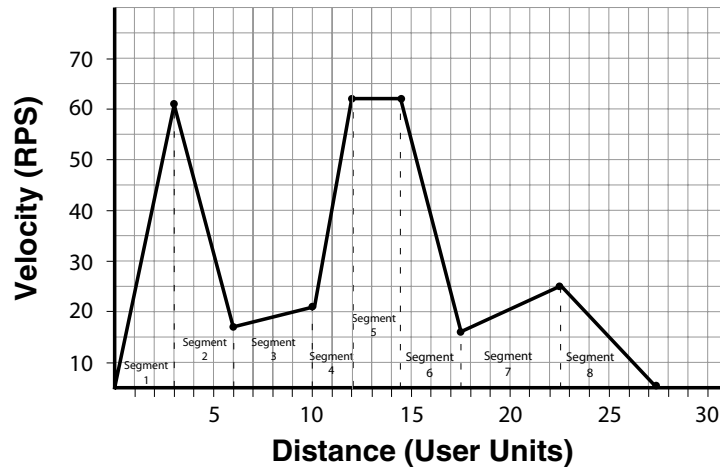


Figure 11: Segment Move

Table 7: Segment Move

Segment Number	Distance moved during segment	Velocity at the end of segment
1	3	56
2	3	12
3	4	16
4	2	57
5	2.5	57
6	3	11
7	5	20
8	5	0
-	-	-

Introduction

Here is the user program for the segment move example. The last segment move must have a "0" for the end velocity, (MDV 5 , 0). Otherwise, fault F_24 (Motion Queue Underflow), will occur.

```
;Segment moves
LOOP:
WAIT UNTIL IN_A4==0 ;Wait until input A4 is off before starting the move
MDV 3 , 56 ;Move 3 units accelerating to 56 User Units per sec
MDV 3 , 12 ;Move 3 units decelerating to 12 User Units per sec
MDV 4 , 16 ;Move 4 units accelerating to 16 User Units per sec
MDV 2 , 57 ;Move 2 units accelerating to 57 User Units per sec
MDV 2.5 , 57 ;Move 2.5 units maintaining 57 User Units per sec
MDV 3 , 11 ;Move 3 units decelerating to 11 User Units per sec
MDV 5 , 20 ;Move 5 units accelerating to 20 User Units per sec
MDV 5 , 0 ;Move 5 units decelerating to 0 User Units per sec
WAIT UNTIL IN_A4==1 ;Wait until input A4 is on before looping
GOTO LOOP
END
```



NOTE

When an MDV move is executed, the segment moves are stored to a Motion Queue. If the program loops on itself, then the queue will become full and an F_23 Fault Motion Queue Overflow will occur.

Since the MDV moves utilize a Motion Queue, the [Step] or [Step Over] debugging features can not be used.

1.10.4 Registration

Both absolute and incremental moves can be used for registration moves. The statements associated with these moves are MOVEPR and MOVEDR. These statements have two arguments. The first argument specifies the commanded move distance or position. The second argument specifies the move made after the registration input is seen. If the registration move is an absolute move, for MovePR, the first argument is absolute (referenced to the 0 position), the second argument is relative to the registration position. For MoveDR, both arguments are relative. The first is relative to the shaft position when motion is started and the second is relative to the registration position.

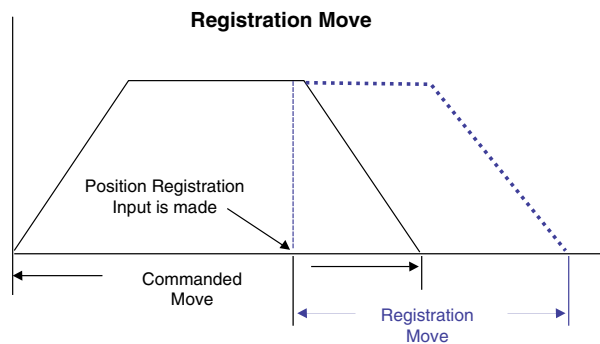


Figure 12: Registration Move

1.10.5 S-Curve Acceleration

Very often it is important for acceleration and deceleration of the motor to be as smooth as possible. For example, using a smooth acceleration/deceleration profile could minimize the wear and tear on a machine tool, smoothing the transition from accel/decel to steady state velocity. To perform smooth motion profiles, the PositionServo supports S-curve acceleration.

With normal straight line acceleration, the axis is accelerated to the target velocity in a linear fashion. With S-curve acceleration, the motor accelerates slowly at the first, then twice as fast as the middle straight line area, and then slowly stops accelerating as it reaches the target velocity. With straight line acceleration, the acceleration changes are abrupt at the beginning of the acceleration and again once the motor reaches the target velocity. With S-curve acceleration, the acceleration gradually builds to the peak value then gradually decreases to no acceleration. The disadvantage with S-curve acceleration is that for the same acceleration distance the peak acceleration is twice that of straight line acceleration, which often requires twice the peak torque. Note that the axis will arrive at the target position at the same time regardless of which acceleration method is used.

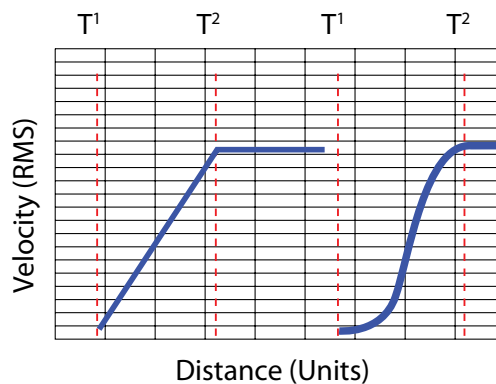


Figure 13: Sequential Move

To use S-curve acceleration in a MOVED, MOVEP or MDV statement requires only the additional “S” at the end of the statement.

Examples:

```
MOVED 10 , S
MOVEP 10 , S
MDV 10,20,S
MDV 10,0,S
```

1.10.6 Motion Queue

The PositionServo drive executes the User Program one statement at a time. When a move statement (MOVED or MOVEP) is executed, the move profile is stored to the Motion Queue. The program will, by default, wait on that statement until the Motion Queue has executed the move. Once the move is completed, the next statement in the program will be executed. By default motion commands (other than MDV statements) effectively suspend the program until the motion is complete.

A standard move (MOVED or MOVEP) is only followed by one argument. This argument references the distance or position to move the motor to. By adding the second argument “C”, (MOVEP 0,C) or (MOVED 100,C), the drive is allowed to continue executing the user program during the move. At this point, multiple move profiles can be stored to the queue. The Motion Queue can hold up to 32 profiles. The Continue “C” argument is very useful when it is necessary to trigger an action, e.g. handle I/O, while the motor is in motion. Below the Pick and Place Example Program has been modified to utilize the Continue, “C”, argument.

Introduction

```
;***** Main Program *****
PROGRAM_START:           ;Place holder for main program loop
ENABLE                   ;Enable output from drive to motor
WAIT UNTIL IN_A4==1     ;Make sure Arm is retracted before starting the program
MOVEP 0                  ;Move to position 0 to pick part
OUT1 = 1                 ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1     ;Check input to make sure Arm is extended
OUT2 = 1                 ;Turn on output 2 to Engage gripper
WAIT TIME 1000          ;Delay 1 sec to Pick part
OUT1 = 0                 ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1     ;Check input to make sure Arm is retracted
MOVED 100,C              ;Move to Place position and continue code execution
WAIT UNTIL APOS >25     ;Wait until pos is greater than 25
OUT3 = 1                 ;Turn on output 3 to spray part
WAIT UNTIL APOS >=75    ;Wait until pos is greater than or equal to 75
OUT3 = 0                 ;Turn off output 3 to shut off spray guns
WAIT UNTIL APOS >=95    ;Wait until move is almost done before extending arm
OUT1 = 1                 ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1     ;Check input to make sure Arm is extended
OUT2 = 0                 ;Turn off output 2 to Disengage gripper
WAIT TIME 1000          ;Delay 1 sec to Place part
OUT1 = 0                 ;Retract Pick arm
WAIT UNTIL IN_A4==1     ;Check input to make sure Arm is retracted
GOTO PROGRAM_START      ;Loop back and continuously execute main program loop
END
```

When the “C” argument is added to the standard MOVED and MOVEP statements, program execution is not interrupted by the execution of the motion command. NOTE: With an MDV move the execution of the program is never suspended.

The generated motion profiles are stored directly to the Motion Queue and are then executed in sequence. If the MOVED and MOVEP statements don't have the “C” modifier, then the motion profiles generated by these statements go to the motion stack and the program is suspended until each profile has been executed.

1.11 Subroutines and Loops

1.11.1 Subroutines

Often it is necessary to repeat a series of program statements in several places in a program. Subroutines can be useful in such situations. The syntax of a subroutine is simple. Subroutines must be placed after the main program, i.e. after the END statement, and must start with the subname: label (where subname is the name of subroutine), and must end with a statement RETURN.

Note that there can be more than one RETURN statement in a subroutine. Subroutines are called using the GOSUB statement.

1.11.2 Loops

SML language supports WHILE/ENDWHILE block statement which can be used to create conditional loops. Note that IF-GOTO statements can also be used to create loops.

The following example illustrates calling subroutines as well as how to implement looping by utilizing WHILE / ENDWHILE statements.

```
;***** Initialize and Set Variables *****
UNITS = 1 ;Units in Revolutions (R)
ACCEL = 15 ;15 Rev per second per second (RPSS)
DECEL = 15 ;15 Rev per second per second (RPSS)
MAXV = 100 ;100 Rev per second (RPS)/6000RPM
APOS = 0 ;Set current position to 0 (absolute zero position)
DEFINE LOOPCOUNT V1
DEFINE LOOPS 10
DEFINE DIST V2
DEFINE REPETITIONS V3
REPETITIONS = 0

;***** Main Program *****
PROGRAM_START: ;Place holder for main program loop
ENABLE ;Enable output from drive to motor
MAINLOOP:
    LOOPCOUNT=LOOPS ;Set up the loopcount to loop 10 times
    DIST=10 ;Set distance to 10
    WHILE LOOPCOUNT ;Loop while loopcount is greater than zero
        DIST=DIST/2 ;decrease dist by 1/2
        GOSUB MDS ;Call to subroutine
        WAIT TIME 100 ;Delay executes after returned from the subroutine
        LOOPCOUNT=LOOPCOUNT-1 ;decrement loop counter
    ENDWHILE
    REPETITIONS=REPETITIONS+1 ;outer loop
    IF REPETITIONS < 5
GOTO MAINLOOP
Wait Motioncomplete ;Wait for MDV segments to be completed
    ENDIF
END

;***** Sub-Routines *****
MDS:
    V4=dist/3
    MDV V4,10
    MDV V4,10
    MDV V4,0
RETURN
```



NOTE

Running this code as is will most likely result in F_23. There are 3 MDV statements that are executed 10 times = 30 moves. Then the condition set on the repetitions variable makes the program execute the above another 4 times. 4 x 30 = 120. The 120 moves, with no waits anywhere in the program will most likely produce an F_23 fault (Motion Queue overflow).

2. Programming

2.1 Program Structure

One of the most important aspects of programming is developing the program's structure. Before writing a program, first develop a plan for that program. What tasks must be performed? And in what order? What things can be done to make the program easy to understand and allow it to be maintained by others? Are there any repetitive procedures?

Most programs are not a simple linear list of instructions where every instruction is executed in exactly the same order each time the program runs. Programs need to do different things in response to external events and operator input. SML contains program control structure instructions and scanned event functions that may be used to control the flow of execution in an application program. Control structure instructions are the instructions that cause the program to change the path of execution. Scanned events are instructions that execute at the same time as the main body of the application program.

Header - Enter in program description and title information

```
***** HEADER *****
;Title:           Pick and Place example program
;Author:          Lenze - AC Technology
;Description:     This is a sample program showing a simple sequence that
;                picks up a part, moves to a set position and places the part
```

I/O List - Define what I/O will be used

```
***** I/O List *****
;   Input A1   -   not used
;   Input A2   -   not used
;   Input A3   -   Enable Input
;   Input A4   -   not used
;   Input B1   -   not used
;   Input B2   -   not used
;   Input B3   -   not used
;   Input B4   -   not used
;   Input C1   -   not used
;   Input C2   -   not used
;   Input C3   -   not used
;   Input C4   -   not used
;
;   Output 1   -   Pick Arm
;   Output 2   -   Gripper
;   Output 3   -   not used
;   Output 4   -   not used
```

Initialize and Set Variables - Define and assign variable values

```
***** Initialize and Set Variables *****
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
;V1 =
;V2 =
DEFINE Output_on 1
DEFINE Output_off 0
```

Introduction

Events - Define Event name, Trigger and Program Statements

```
;***** Events *****
EVENT SPRAY_GUNS_ON   APOS > V1 ;Event will trigger as position passes 25 in pos dir.
OUT3= Output_On      ;Turn on the spray guns (out 3 on)
ENDEVENT              ;End event
EVENT SPRAY_GUNS_OFF APOS > V2 ;Event will trigger as position passes 75 in pos dir.
OUT3= Output_Off      ;Turn off the spray guns (out 3 off)
ENDEVENT              ;End event
```

Main Program - Define the motion and I/O handling of the machine

```
;***** Main Program *****
RESET_DRIVE:          ;Place holder for Fault Handler Routine
WAIT UNTIL IN_A3      ;Make sure that the ENABLE input is made before continuing
ENABLE                ;Enable output from drive to motor
PROGRAM_START:       ;Place holder for main program loop
EVENT   SPRAY_GUNS_ON ON ;Enable the 'spray guns on' event
EVENT   SPRAY_GUNS_OFF ON ;Enable the 'spray guns off' event
WAIT UNTIL IN_A4==1   ;Make sure Arm is retracted before starting the program
MOVEP 0                ;Move to position 0 to pick part
OUT1 = Output_On      ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1   ;Check input to make sure Arm is extended
OUT2 = Output_On      ;Turn on output 2 to Engage gripper
WAIT TIME 1000        ;Delay 1 sec to Pick part
OUT1 = Output_Off     ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1   ;Check input to make sure Arm is retracted
MOVED 100             ;Move to Place position
OUT1 = Output_On      ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1   ;Check input to make sure Arm is extended
OUT2 = Output_Off     ;Turn off output 2 to Disengage gripper
WAIT TIME 1000        ;Delay 1 sec to Place part
OUT1 = Output_Off     ;Retract Pick arm
WAIT UNTIL IN_A4==1   ;Check input to make sure Arm is retracted
GOTO PROGRAM_START    ;Loop back and continuously execute main program loop
END
```

Sub-Routine - Any and all Sub-Routine code should reside here

```
;***** Sub-Routines *****
;   Enter Sub-Routine code here
```

Fault Handler - Define what the program should do when a fault is detected

```
;***** Fault Handler Routine *****
;   Enter Fault Handler code here
ON FAULT
ENDFAULT
```

The **header section** of the program contains description information, program name, version number, description of process and programmers name. The **I/O List section** of the program contains a listing of all the I/O used within the application. The **Initialize and Set Variables section** of the program defines the names for the user variables and constants used in the program and provides initial setting of these and other variables.

Programming

The **Events section** contains all scanned events. Remember to execute the **EVENT <eventname> ON** statement in the main program to enable the events. Please note that not all of the SML statements are executable from within the EVENT body. For more detail, reference “EVENT” and “ENDEVENT” in Section 3 of the manual. The GOTO statement can not be executed from within the Event body. However, the JUMP statement can be used to jump to code in the main program body. This technique allows the program flow to change based on the execution of an event. For more detail, reference “JUMP”, in Section 3.1 (Program Statement Glossary) of this manual.

The **main program** body of the program contains the main part of the program, which can include all motion and math statements, labels, I/O commands and subroutine calls. The main body should be finished with an END statement, however if the program loops indefinitely then the END statement can be omitted.

Subroutines are routines that are called from the main body of the program. When a subroutine is called, (GOSUB), the program’s execution is transferred from the main program to the called subroutine. It will then process the subroutine until a RETURN statement occurs. Once a RETURN statement is executed, the program’s execution will return back to the main program to the line of code immediately following the GOSUB statement.

Fault handler is a section of code that is executed when the drive detects a fault. This section of code begins with the “ON FAULT” statement and ends with an “ENDFAULT” statement. When a fault occurs, the normal program flow is interrupted, motion is stopped, the drive is disabled, Event scanning is stopped and the statements in the Fault Handler are executed, until the program exits the fault handler. The Fault handler can be exited in two ways:

- The “RESUME” statement will cause the program to end the Fault Handler routine and resume the execution of the main program. The location (label) called out in the “RESUME” command will determine where the program will commence.
- The “ENDFAULT” statement will cause the user program to be terminated.



While the Fault Handler is being executed, Events are not being processed and detection of additional faults is not possible. Because of this, the Fault Handler code should be kept as short as possible.

If extensive code must be written to process the fault, then this code should be placed in the main program and the “RESUME” statement should be utilized. Not all of SML statements can be utilized by the Fault Handler. For more details reference “ON FAULT/ENDFAULT”, in Section 3.1 (Program Statement Glossary) of this manual.

Comments are allowed in any section of the program and are preceded by a semicolon. They may occur on the same line as an instruction or on a line by themselves. Any text following a semicolon in a line will be ignored by the compiler.

2.2 Variables

Variables can be System or User. User variables do not have a predefined meaning and are available to the user to store any valid numeric value. System variables have a predefined meaning and are used to configure, control or monitor the operations of the PositionServo. (Refer to paragraph 2.6 for more information on System Variables).

All variables can be used in valid arithmetic expressions. All variables have their own corresponding index or identification number. Any variable can be accessed by their identification number from the User’s program or from a Host Interface. In addition to numbers some of the variables have predefined names and can be accessed by that name from the User’s program.

The following syntax is used when accessing variables by their identification number:

```
@102 = 20 ; set variable #102 to 20
@88=@100 ; copy value of variable #100 to variable #88
```

Variable @102 has the variable name ‘V2’; variable @88 has the variable name ‘VAR_AOUT’ and variable @100 has the variable name ‘V0’. Hence the program statements above could be written as:

```
V2 = 20
VAR_AOUT = V0
```

There are two types of variables in the PositionServo drive - **User Variables** and **System Variables**.

User Variables are a fixed set of variables that the programmer can use to store data and perform arithmetic manipulations. All variables are of a single type. Single type variables, i.e. typeless variables, relieve the programmer of the task of remembering to apply conversion rules between types, thus greatly simplifying programming.

User Variables

V0-V31 User defined variables. Variables can hold any numeric value including logic (Boolean 0 - FALSE and non 0 - TRUE) values. They can be used in any valid arithmetic or logical expressions.

NV0-NV31 User defined network variables. Variables can hold any numeric value including logic (Boolean 0 - FALSE and non 0 - TRUE) values. They can be used in any valid arithmetic or logical expressions. Variables can be shared across Ethernet network with use of statements SEND and SENDTO.

Since SML is a typeless language, there is no special type for Boolean type variables (variables that can be only 0 or 1). Regular variables are used to facilitate Boolean variables. Assigning a variable a "FALSE" state is done by setting it equal to "0". Assigning a variable a "TRUE" state is done by assigning it any value other than "0".

Scope

SML variables are accessible from several sources. Each of the variables can be read and set from any user program or Host communications interface at any time. There is no provision to protect a variable from change. This is referred to as global scope.

Volatility

User variables are volatile i.e. they don't maintain their values after the drive is powered down. After power up the values of the user variables are set to 0. Loading or resetting the program doesn't change variables values.

In addition to the user variables, system variables are also provided. System variables are dedicated variables that contain specific information relative to the setup and operation of the drive. For example, **APOS** variable holds actual position of the motor shaft. For more details refer to Section 2.9.

Flags, Resolution and Accuracy

Any variable can be used as a flag in a logical expression and as a condition in a conditional expression. Flags are often used to indicate that some event has occurred, logic state of an input has changed or that the program has executed to a particular point. Variables with non '0' values are evaluated as "TRUE" and variables with a "0" values are evaluated as "FALSE".

Variables are stored internally as 4 bytes (double word) for integer portion and 4 bytes (double word) for fractional portion. Every variable in the system is stored as 64 bit in 32.32 fixed point format. Maximum number can be represented by this format is +/- 2,147,483,648. Variable resolution in this format is 2.3E-10.

2.3 Arithmetic Expressions

Table 8 lists the four arithmetic functions supported by the Indexer program. Constants as well as User and System variables can be part of the arithmetic expressions.

Examples.

```
V1 = V1+V2           ;Add two user variables
V1 = V1-1           ;Subtract constant from variable
V2 = V1+APOS        ;Add User and System (actual position) variables
APOS = 20           ;Set System variable
V5 = V1*(V2+V3*5/2+1) ;Complicated expression
```

Table 8: Supported Arithmetic Expressions

Operator	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/

Result overflow for “*” and “/” operations will cause arithmetic overflow fault F_19. Result overflow/underflow for “+” and “-” operations does not cause an arithmetic fault.

2.4 Logical Expressions and Operators

Bitwise, Boolean, and comparison operators are considered as Logical Operators. They operate on logical values of the operands. There are two possible values for logical operand: TRUE and FALSE. Any value contained in a User variable, System variable or flag is treated as TRUE or FALSE with these types of the operators. If a variable value equals “0”, it is considered FALSE. All other values (non-0) including negative numbers are considered TRUE.

2.4.1 Bitwise Operators

Table 9 lists the bitwise operators supported by the Indexer program.

Table 9: Supported Bitwise Operators

Operator	Symbol
AND	&
OR	
XOR	^
NOT	!

Both User or System variables can be used with these operators. In order to perform a bitwise (Boolean) operation, the value must be referenced in hexadecimal format. Example: bit 22 alone would be referenced as 0x40000.

Examples:

```
V1 = V2 & 0xF        ;clear all bits but lowest 4
IF (INPUTS & 0x3)   ;check inputs 0 and 1
V1 = V1 | 0xFF      ;set lowest 8 bits
V1 = INPUTS ^ 0xF   ;invert inputs 0-3
V1 = !IN_A1         ;invert input A1
```

2.4.2 Boolean Operators

Table 10 lists the boolean operators supported by the Indexer program. Boolean operators are used in logical expressions.

Table 10: Supported Boolean Operators

Operator	Symbol
AND	&&
OR	
NOT	!

Examples:

```
IF APOS >2 && APOS <6 || APOS >10 && APOS <20
    {statements if true}
ENDIF
```

The above example checks if APOS (actual position) is within one of two windows; 2 to 6 units or 10 to 20 units. In other words:

If (APOS is more than 2 AND less than 6)

OR

If (APOS is more than 10 AND less than 20)

THEN the logical expression is evaluated to TRUE. Otherwise it is FALSE

2.5 Comparison Operators

Table 11 lists the comparison operators supported by the Indexer program.

Table 11: Supported Comparison Operators

Operator	Symbol
More	>
Less	<
Equal or more	>=
Equal or less	<=
Not Equal	<>
Equal	==

Examples:

```
IF APOS <=10 ; If Actual Position equal or less than 10
IF APOS > 20 ; If Actual Position greater than 20
IF V0 ==5 ; If V0 equal to 5
IF V1<2 && V2 <>4 ; If V1 less than 2 and V2 doesn't equal 4
```

2.6 System Variables and Flags

System variables are variables that have a predefined meaning. They give the programmer/user access to drive parameters and functions. Some of these variables can also be set via the parameters in MotionView. In most cases the value of these variables can be read and set in your program or via a Host Interface. Variables are either read only, write only or read and write. Read only variables can only be read and can't be set. For example, INPUTS = 5, is an illegal action because you can not set an input. Conversely, write-only variables cannot be read. Reading a write-only variable by either the variable watch window or network communications can result in erroneous data.

System Flags are System Variables that can only have values of 0 or 1. For example, IN_A1 is the system flag that reflects the state of digital input A1. Since inputs can only be ON or OFF, then the value of IN_A1 can only be 0 or 1.

2.7 System Variables Storage Organization

All system variables are located in drive's RAM memory and therefore are volatile. However, values for some of these system variables are also stored in EPM. When a system variable is changed in MotionView, its value changes in both RAM and EPM. When a system variable is changed from the user's program, its value is changed in RAM only and will be lost on power down.

Host interfaces have the capability to change the variable value in both the EPM and RAM. The user has a choice in memory to change a variable in RAM and EPM or in RAM only.

2.7.1 RAM File for User's Data Storage

In addition to the standard user variables (V0-V31 & NV0-NV31) PositionServo drives have a section of RAM memory (256k) allocated as data storage space and available to the programmer for storage of program data.

The RAM file data storage is often required in systems where it is desirable to store large amounts of data prepared by a host controller (PLC, HMI, PC, etc). This data might represent more complex Pick and Place coordinates, complicated trajectory coordinates, or sets of gains/limits specific for given motion segments.

RAM memory is also utilized in applications that require data collection during system operation. At the end of a period of time the collected data can be acquired by the host controller for analysis. For example, position errors and phase currents collected during the move are then analyzed by the host PLC/PC to qualify system tolerance to error free operation.

Implementation

There are 256K (262,144) bytes provided as RAM file for data storage. Since the basic data type in the drive is 64 bit (8 bytes) 32,768 data elements can be stored in the RAM file. The file is accessible from within the User's program or through any external communications interface (Ethernet, ModBUs, CAN etc.). Two statements and three system variables are provided for accessing the RAM file memory. The RAM file is volatile storage and is intended for "per session" usage. The data saved in the RAM file will be lost when the drive is powered off.

The three system variables provided to support file access are:

VAR_MEM_VALUE	(PID = 4)
VAR_MEM_INDEX	(PID = 5)
VAR_MEM_INDEX_INCREMENT	(PID = 6)

In addition, two statements are provided to allow access and storage to the RAM file direct from convenient statements within the user program. The statements MEMSET, MEMGET are described in paragraph 2.7.3 and Tables 44 & 45.

2.7.2 Memory Access Through Special System Variables

MEM_INDEX holds the value that will be read or written to the RAM file. MEM_INDEX points to the position in the RAM file (0 to 32767) and MEM_INDEX_INCREMENT holds the value that MEM_INDEX is going to modify after the read or write operation is completed.

The RAM memory access is illustrated with the example program herein.

```
-----
;User's program to read/write to RAM file.
;Advance index after writing/reading by 1
;Record position error to RAM file every 100 ms for 10 seconds. 10/0.1 = 100
;locations are needed
-----

#DEFINE      IndexStart      0
#DEFINE      MemIncrement    1
#DEFINE      RecordLength    100
#DEFINE      PELimit         0.1                ;0.1 user unit

VAR_MEM_INDEX = IndexStart                ;set start position
VAR_MEM_INDEX_INCREMENT=MemIncrement      ;set increment

-----
EVENT  StorePE TIME 100

      VAR_MEM_VALUE = VAR_POSEERROR        ;store in RAM file.

ENDEVENT

PROGRAMSTART:

      EVENT StorePE ON

      {
          Start some motion activity...
      }

;wait for data collection is over

WHILE  VAR_MEM_INDEX <  (IndexStart+RecordLength)
ENDWHILE
EVENT StorePE Off                ;turn off storage

;Analyze data collected. If PE > PELimit then signal system has low performance...
VAR_MEM_INDEX= IndexStart
WHILE  VAR_MEM_INDEX <  (IndexStart+RecordLength)
  IF (VAR_MEM_VALUE > PELimit)
    GOTO Label_SignalBad
  ENDIF
ENDWHILE

LabelSignalBad:

      {
          Signal that PE out of limits
          ...
      }
```

END

Programming

In the RAM memory access program example, the values of PE (position error) are stored sequentially in the RAM file every 100ms for 10 seconds. (100 samples). After collection is done the data is read from the file one by one and compared with limit.

Variable VAR_MEM_INDEX is incremented every read or write by the value stored in VAR_MEM_INDEX_INCREMENT. That value could be any value from -32767 to 32767. This way backwards reading is also possible. If the value is 0 (zero) no increment/decrement is produced. VAR_MEM_INDEX wraps around its min/max values. I.e. if the next read or write will result in a value more (less) than 32767 (-32767), the index will be adjusted by modulo 32767. This allows for the creation of circular arrays. This feature can be used for diagnostics when certain parameter(s) are stored in the memory continuously and then if the system fails the data array can be examined to simplify diagnostics.

2.7.3 Memory Access Through MEMSET, MEMGET Statements

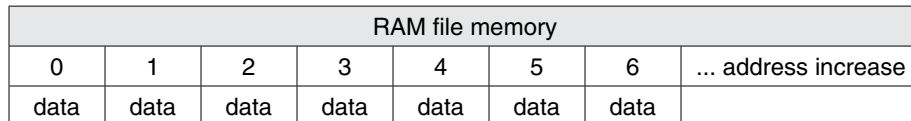
The memory access statements MEMSET and MEMGET are provided for simplified storage of data in the RAM memory to/from the user variables V0-V31. Using these statements any combinations of variables V0-V31 can be stored/retrieved with a single statement. This allows for efficient access to the RAM memory area. For example, in reading 10 variables of the user's program. Indeed for reading 10 variables V0-V10 it would normally require 10 read statements (Vx=VAR_MEM_VALUE). With the MEMGET statement all V0-V10 can be read in one step. The format of MEMSET/MEMGET is as follows:

```
MEMSET    <offset> [ <varlist>]
MEMGET    <offset> [ <varlist>]
<offset>  any valid expression that evaluates to a number between -32767 to 32767
           It specifies the offset in the RAM file where data will be stored or retrieved.
<varlist> any combinations of variables V0-V31
```

Examples for <offset> expression

```
5          constant
10+23+1   constant expression
V0         variable           Must hold values in -32767 to 32767 range
V0+V1+3   expression        Must evaluate to -32767 to 32767 range
```

Example: <offset> =5



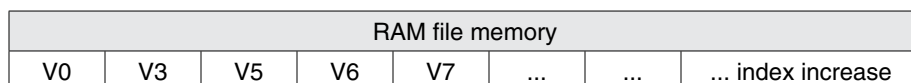
Examples for <varlist> instruction

```
[V0]      single variable will be stored/retrieved
[V0,V3,V2] variables V0,V3,V2 will be stored/retrieved
[V3-V7]   variables V3 to V7 inclusively will be stored
[V0,V2,V4-V8] variables V0,V2, V4 through V8 will be stored
```

Storage/Retrieval order with MEMSET/MEMGET

Variables in the list are always stored in order: the variable with smallest index first and the variables with highest index last regardless of the order they appear in the <varlist> instruction.

Example: [V0,V3, V5-V7] will be stored in memory in the order of increasing memory index as follows:



For comparison: [V5-V7, V0, V3] will have the same storage order as the above list regardless of the order in which the variables are listed.

Programming

When retrieving data with MEMGET statements memory locations will be sequentially copied to variables starting from the one with smallest index in the list to the last with biggest index. Consider the list for the MEMGET statement:
[V2,V3,V5-V7]

RAM file memory							
Data1	Data2	Data3	Data4	Data5	Data6 index increase

Here is how the data will be assigned to variables:

```
V2 <- Data1
V3 <- Data2
V5 <- Data3
V6 <- Data4
V7 <- Data5
```

2.8 System Variables and Flags Summary

2.8.1 System Variables

Section 3.2 provides a complete list of the system variables. Every aspect of the PositionServo can be controlled by the manipulation of the values stored in the System Variables. All System Variables start with a “VAR_” followed by the variable name. Alternatively, System Variables can be addressed as an @NUMBER where the number is the variable Index. The most frequently used variables also have alternate names listed in Table 12.

Table 12: System Variables

Index	Variable	Access	Variable Description	Units
181	ACCEL	R/W	Acceleration for motion commands	User Units/Sec ²
71	AIN1	R	Analog input. Scaled in volts. Range from -10 to +10 volts	V(olt)
72	AIN2	R	Analog input 2. Scaled in Volts. Range from -10 to +10 volts	V(olt)
88	AOUT	R/W	Analog output. Value in Volts. Valid range from -10 to +10 (V) ⁽²⁾	V(olt)
215	APOS	R/W	Actual motor position	User Units
190	APOS_PLS	R/W	Actual Motor Position	Encoder Counts
182	DECEL	R/W	Deceleration for motion commands	User Units/Sec ²
83	DEXSTATUS	R	Drive Extended Status Word	-
54	DSTATUS	R	Status flags register	-
	DFAULTS	R	Fault code register	-
245	HOME	W	Start Homing (pre-defined homing)	-
	INDEX	R	Lower 8 bits are used. See ASSIGN statement for details.	-
184	INPOSLIM	R/W	Maximum deviation of position for INPOSITION Flag to remain set	User Units
65	INPUTS	R	Digital Inputs states. The first 12 bits correspond to the 12 drive inputs	-
139	IREF	W	Internal Reference: Velocity / Torque	RPS/A
187	MECOUNTER	R	Master Encoder Counts (Master Encoder Input)	Encoder Counts
180	MAXV	R/W	Maximum velocity for motion commands	User Units/Sec
140-171	NV0 - NV31	R/W	User Network Variables	-
66	OUTPUTS	R/W	Digital outputs. Bits #0 to #4 represent outputs 1 through 5	-
216	PERROR	R	Position Error	Feedback Pls
191	PERROR_PLS	R	Position Error	User Units
48	PGAIN_D	R/W	Position loop D-gain	-
47	PGAIN_I	R/W	Position loop I-gain	-
49	PGAIN_ILIM	R/W	Position loop I gain limit	-
46	PGAIN_P	R/W	Position loop P-gain	-

Programming

Index	Variable	Access	Variable Description	Units
188	PHCUR	R	Motor phase current	A(mpere)
183	QDECEL	R/W	Quick Deceleration for STOP MOTION QUICK statement	User Units/Sec ²
213	RPOS	R	Registration position. Valid when system flag F_REGISTRATION set	User Units
212	RPOS_PLS	R	Registration position	Feedback Pls
218	TA	R	Commanded acceleration	User units/Sec ²
214	TPOS	R/W	Theoretical/commanded position	User Units
219	TPOS_ADV	W	Theoretical/commanded position advance	Feedback Pls
189	TPOS_PLS	R/W	Theoretical/commanded position	Feedback Pls
217	TV	R	Commanded velocity in	User Units/Sec
186	UNITS	R/W	User Units scale. ⁽¹⁾	UserUnits/Rev
185	VEL	R/W	Set Velocity when in velocity mode	User Units/Sec
44	VGAIN_P	R/W	Velocity loop P-gain	-
45	VGAIN_I	R/W	Velocity loop I-gain	-
100-131	V0 - V31	R/W	User Variables	

(1) When a "0", (Zero), value is assigned to the variable "UNITS", then "USER UNITS" is set to QUAD ENCODER COUNTS. This is the default setting at the start of the program before UNITS=<value> is executed.

(2) Any value outside +/- 10 range assigned to AOUT will be automatically trimmed to that range

Example:

AOUT=100 , AOUT will be assigned value of 10.

V0=236

VOUT=V0, VOUT will be assigned 10 and V0 will be unchanged.

2.8.2 System Flags

Flags don't have an Index number assigned to them. They are the product of a BIT mask applied to a particular system variable by the drive and are available to the user only from the User's program. Table 13 lists the System Flags with access rights and description.

Table 13: System Flags

Name	Access	Description
IN_A1-4, IN_B1-4, IN_C1-4	R	Digital inputs . TRUE if input active, FALSE otherwise
OUT1, OUT2, OUT3, OUT4, OUT5	W	Digital outputs OUTPUT1- OUTPUT5
F_ICONTROL OFF	R	Interface Control Status (ON/OFF) #27 in DSTATUS register
F_IN_POSITION	R	TRUE when Actual Position (APOS) is within limits set by INPOSLIM variable and motion completed
F_ENABLED	R	Set when drive is enabled
F_EVENTS OFF	R	Events Disabled Status (ON/OFF) #30 in DSTATUS register
F_MCOMPLETE	R	Set when motion is completed and there is no motion commands waiting in the Motion Queue
F_MQUEUE_FULL	R	Motion Queue full
F_MQUEUE_EMPTY	R	Motion Queue empty
F_FAULT	R	Set if any fault detected
F_ARITHMETIC_FLT	R	Arithmetic fault
F_REGISTRATION	R	Set when registration mark was detected. Content RPOS variable is valid when this flag is active. Flag resets by any registration moves MOVEPR,MOVEDR or by command REGISTRATION ON
F_MSUSPENDED	R	Set if motion suspended by statement MOTION SUSPEND

Flag logic is shown herein.

```
IF
    TPOS-INPOSLIM < APOS < TPOS+INPOSLIM  && F_MCOMPLETE && F_MQUEUE_EMPTY
    F_IN_POSITION = TRUE
ELSE
    F_IN_POSITION = FALSE
ENDIF
```

For VELOCITY mode F_MCOMPLETE and F_MQUEUE_EMPTY flags are ignored and assumed TRUE.

2.9 Control Structures

Control structures allow the user to control the flow of the program's execution. Most of the power and utility of any programming language comes from its ability to change statement order with structure and loops.

2.9.1 DO/UNTIL Structure

This statement is used to execute a block of code one time and then continue executing that block until a condition becomes true (satisfied). The difference between DO/UNTIL and WHILE statements is that the DO/UNTIL instruction tests the condition after the block is executed so the conditional statements are always executed at least one time. The syntax for DO/UNTIL statement is:

```
DO
    ...statements
UNTIL <condition>
```

The flowchart and code segment in Figure 14 illustrate the use of the DO/UNTIL statement.

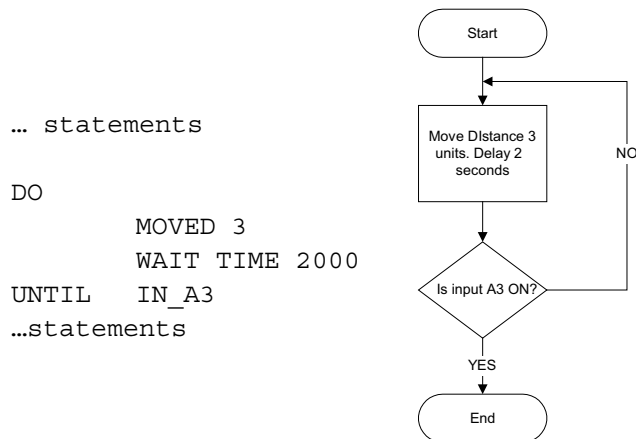


Figure 14: DO/UNTIL Code and Flowchart

2.9.2 WHILE Structure

This statement is used if you want a block of code to execute while a condition is true.

```
WHILE <condition>
    ...statements
ENDWHILE
```

Programming

The flowchart and code segment in Figure 15 illustrate the syntax for the WHILE instruction.

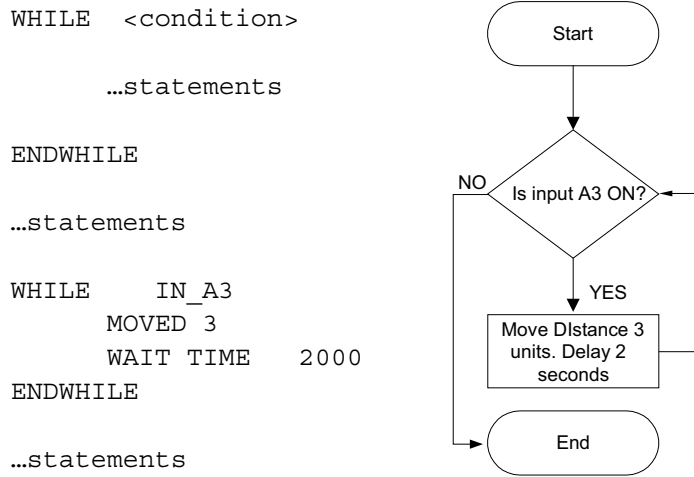


Figure 15: WHILE Code and Flowchart

2.9.3 Subroutines

A subroutine is a group of SML statements that is located at the end of the main body of the program. It starts with a label which is used by the GOSUB statement to call the subroutine and ends with a RETURN statement. The subroutine is executed by using the GOSUB statement in the main body of the program. Subroutines can not be called from an EVENT or from the FAULT handlers.

When a GOSUB statement is executed, execution is transferred to the first line of the subroutine. The subroutine is then executed until a RETURN statement is met. When the RETURN statement is executed, the program's execution returns to the program line, in the main program, following the GOSUB statement. Subroutines may have more than one RETURN statement in its body.

Subroutines may be nested up to 32 times. Only the main body of the program and subroutines may contain a GOSUB statement. Refer to Section 3.1 for more detailed information on the GOSUB and RETURN statements. The flowchart and code segment in Figure 16 illustrate the use of subroutines.

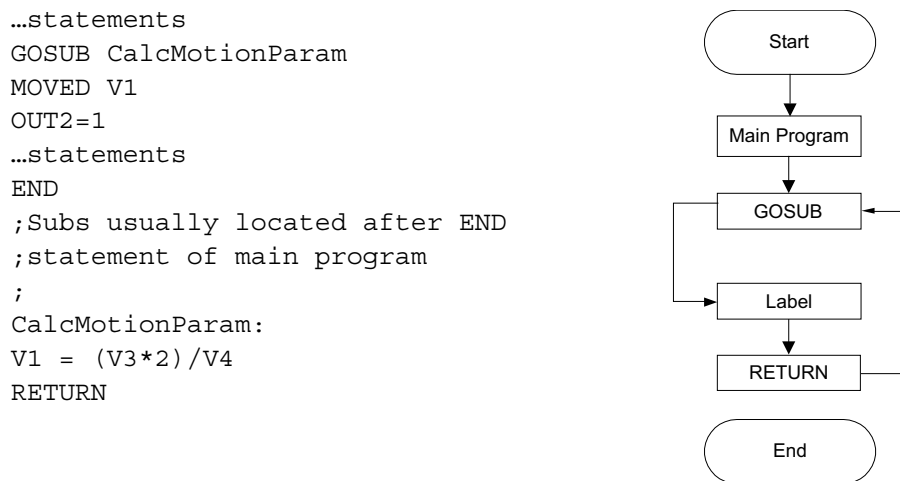


Figure 16: GOSUB Code and Flowchart

2.9.4 IF Structure

The "IF" statement is used to execute an instruction or block of instructions one time if a condition is true. The simplified syntax for the IF statement is:

```
IF condition
    ...statement(s)
ENDIF
```

The flowchart and code segment in Figure 17 illustrate the use of the IF statement.

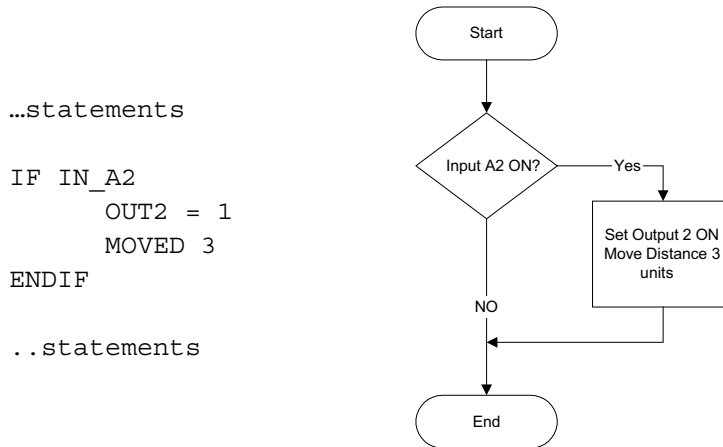


Figure 17: IF Code and Flowchart

2.9.5 IF/ELSE Structure

The IF/ELSE statement is used to execute a statement or a block of statements one time if a condition is true and a different statement or block of statements if condition is false.

The simplified syntax for the IF/ELSE statement is:

```
IF <condition>
    ...statement(s)
ELSE
    ...statement(s)
ENDIF
```

The flowchart and code segment in Figure 18 illustrate the use of the IF/ELSE instruction.

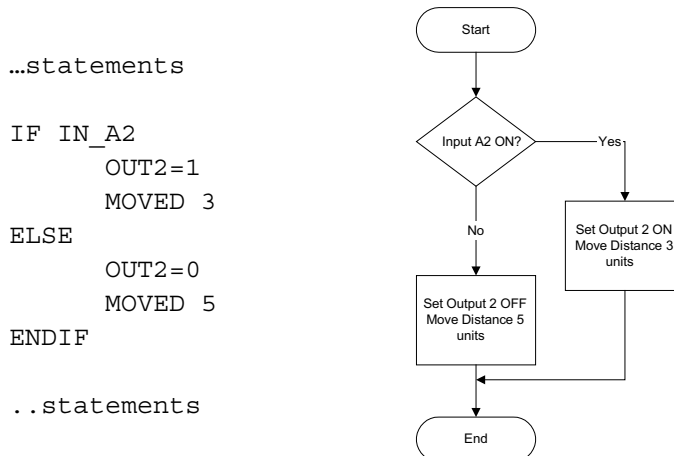


Figure 18: IF/ELSE Code and Flowchart

Programming

2.9.6 WAIT Statement

The WAIT statement is used to suspend program execution until or while a condition is true, for a specified time period (delay) or until motion has been completed. The simplified syntax for the WAIT statement is:

```
WAIT UNTIL <condition>
WAIT WHILE <condition>
WAIT TIME <time>
WAIT MOTION COMPLETE
```

2.9.7 GOTO Statement & Labels

The GOTO statement can be used to transfer program execution to a new point marked by a label. This statement is often used as the action of an IF statement. The destination label may be above or below the GOTO statement in the application program.

Labels may be any alphanumeric string 64 characters in length beginning with a letter and ending with a colon ":".

```
GOTO TestInputs
    ...statements
TestInputs:
    ...statements
IF (IN_A1) GOTO TestInputs
```

Table 14 provides a short description of the instructions used for program branching.

Table 14: Program Branching Instructions

Name	Description
GOTO	Transfer code execution to a new line marked by a label
DO/UNTIL	Do once and keep doing until conditions becomes true
IF and IF/ELSE	Execute if condition is true
RETURN	Return from subroutine
WAIT	Wait fixed time or until condition is true
WHILE	Execute while a condition is true
GOSUB	Go to subroutine

2.10 Scanned Event Statements

A Scanned Event is a small program that runs independently of the main program. SCANNED EVENTS are very useful when it is necessary to trigger an action , i.e. handle I/O, while the motor is in motion. When setting up Events, the first step is to define both the action that will trigger the event as well as the sequence of statements to be executed once the event has been triggered. Events are scanned every 512µs. Before an Event can be scanned however it must first be enabled. Events can be enabled or disabled from the user program, from another event or from itself (see explanations below). Once the Event is defined and enabled, the Event will be constantly scanned until the trigger condition is met, this scan rate is independent of the main program's timing. Once the trigger condition is met, the Event statements will be executed independently of the user program.

Scanned events are used to record events and perform actions independent of the main body of the program. For example, if you want output 3 to come ON when the position is greater than 4 units, or if you need to turn output 4 ON whenever input 2 and 3 are ON, you may use the following scanned event statements.

```
EVENT      PositionIndicator APOS > 4
           OUT3=1
ENDEVENT

EVENT      Inputs3and4          IN_A4 & IN_B1
           OUT4=1
ENDEVENT
```

Programming

Scanned events may also be used with a timer to perform an action on the periodic time basis.

The program statements contained in the action portion of the scanned event can be any legal program statement except the following statements: Subroutine calls (GOSUB), DO/WHILE, WHILE, WAIT, GOTO and also motion commands: MOVED, MOVEP, MDV, STOP, MOTION SUSPEND/RESUME.

EVENT <name> INPUT <inputname> RISE

This scanned event statement is used to execute a block of code each time a specified input <inputname> changes its state from low to high.

EVENT <name> INPUT <inputname> FALL

This scanned event statement is used to execute a block of code each time a specified input <inputname> changes its state from high to low.

EVENT <name> TIME <timeout>

This scanned event statement is used to execute a block of code with a repetition rate specified by the <timeout> argument. The range for "timeout" is 0 - 50,000ms (milliseconds). Specifying a timeout period of 0 ms will result in the event running every event cycle (256µs).

EVENT <name> expression

This scanned event statement is used to execute a block of code when the expression evaluates as true.

EVENT <name> ON/OFF

This statement is used to enable/disable a scanned event. Statement can be used within event's block of code.

Scanned Event Statements Summary

Table 15 contains a summary of instructions that relate to scanned events. Refer to Section 3 "Language Reference" for more detailed information.

Table 15: Scanned Events Instructions

Name	Description
EVENT <name> ON/OFF	enable / disable event
EVENT <name> INPUT <inputname> RISE	Scanned event when <input name> goes low to high
EVENT <name> INPUT <inputname> FALL	Scanned event when <input name> goes high to low
EVENT <name> TIME <value>	Periodic event with <input name> repetition rate.
EVENT <name> expression	Scanned event on expression = true

2.11 Motion

2.11.1 How Moves Work

The position command that causes motion to be generated comes from the profile generator or profiler for short. The profile generator is used by the MOVE, MOVED, MOVEP, MOVEPR, MOVEDR and MDV statements. MOVE commands generate motion in a positive or negative direction, while or until certain conditions are met. For example you can specify a motion while a specific input remains ON (or OFF). MOVEP generates a move to specific absolute position. MOVED generates incremental distance moves, i.e. move some distance from its current position. MOVEPR and MOVEDR are registration moves. MDV commands are used to generate complicated profiles. Profiles generated by these commands are put into the motion stack which is 32 levels deep. By default when one of these statements (except for MDV) is executed, the execution of the main User Program is suspended until the generated motion is completed. Motion requests generated by an MDV statement or MOVE statement with the "C" modifier do not suspend the program. All motion statements are put into the motion stack and executed by the profiler in the order in which they were loaded. The Motion Stack can hold up to 32 moves. The SML language allows the programmer to load moves into the stack and continue on with the program. It is the responsibility of the programmer to check the motion stack to make sure there is room available before loading new moves. This is done by checking the appropriate bits in the System status register or the appropriate system flag.

2.11.2 Incremental (MOVED) and Absolute (MOVEP) Motion

MOVED and MOVEP statements are used to create incremental and absolute moves respectively. The motion that results from these commands is by default a trapezoidal velocity move or an S-curved velocity move if the ",S" modifier is used with the statement,

For example:

```
MOVEP 10 ;will result in a trapezoidal move
```

But

```
MOVEP 10,S ;will result in an S-curved move
```

In the above example, (MOVEP 10), the length of the move is determined by the argument following the MOVEP command, (10). This argument can be a number, a variable or any valid arithmetic expression. The maximum velocity of the move is determined by setting the system variable MAXV. The acceleration and deceleration are determined by setting the system variables ACCEL and DECEL respectively.

If values for velocity, acceleration and deceleration, for a specified distance, are such that there is not enough time to accelerate to the specified velocity, the motion profile will result in triangular or double S profile as illustrated in Figure 19.

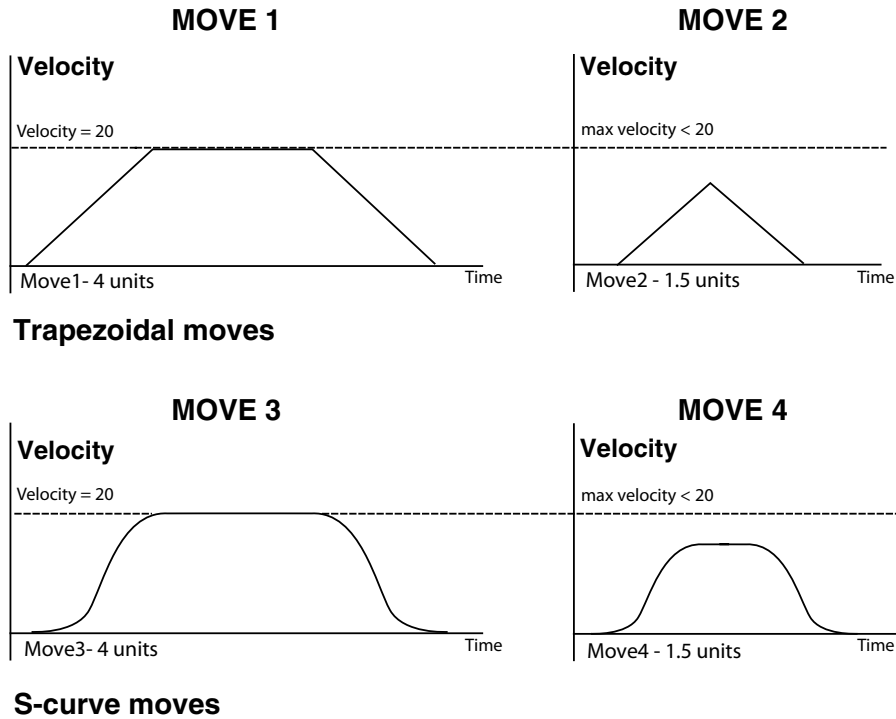


Figure 19: Move Illustration

```

ACCEL = 200
DECEL = 200
MAXV = 20
MOVED 4 ;Move 1
MOVED 1.5 ;Move 2
MOVED 4 , S ;Move 3
MOVED 1.5 , S ;Move 4
    
```

All four of the moves shown in Figure 19 have the same Acceleration, Deceleration and Max Velocity values. Moves 1 and 3 have a larger value for the move distance than Moves 2 and 4. In Moves 1 and 3 the distance is long enough to allow the motor to accelerate to the profiled max velocity and maintain that velocity before decelerating down to a stop. In Moves 2 and 4 the distance is so small that while the motor is accelerating towards the profiled Max Velocity it has to decelerate to a stop before it can ever obtain the profiled Max Velocity.

2.11.3 Incremental (MOVED) Motion

Incremental motion is defined as a move of some distance from the current position. 'Move four revolutions from the current position' is an example of an incremental move.

MOVED is the statement used to create incremental moves. The simplified syntax is:

```
MOVED <+/-distance>
```

+/- sign will tell the motor shaft what direction to move.

2.11.4 Absolute (MOVEP) Move

Absolute motion is defined as a motion to some fixed position from the current position. The fixed position is defined as a position relative to a fixed zero point. The zero point for a system is normally established during the homing cycle, typically performed immediately after power-up.

During a homing cycle, the motor will make incremental moves while checking for a physical input, index mark, or both.

2.11.5 Registration (MOVEDR MOVEPR) Moves

MOVEPR and MOVEDR are used to move to position or distance respectively just like MOVEP and MOVED. The difference is that while the statements are being executed they are looking for a registration signal or registration input (C3). If during the motion a registration signal is detected, then a new end position is generated. With both the MoveDR and MovePR statements the drive will increment the distance called out in the registration argument. This increment will be referenced from the position where the registration input has seen.

Example:

```
MOVEDR 5, 1 ;Statement move a distance of 5 user units or registration position +  
           ;1 user units if registration input is activated during motion.
```

There are two exceptions to this behavior:

Exception one:

The move will not be modified to “Registration position +displacement” if the registration was detected while system was decelerating to complete the motion.

Exception two:

Once the registration input is seen, there must be enough room for the motor to decelerate to a stop using the profiled Decel Value. If the new registration move is smaller than the distance necessary to come to a stop, then the motor will overshoot the new registration position.

2.11.6 Segment Moves

In addition to the simple moves that can be generated by MOVED and MOVEP statements, complex profiles can be generated using segment moves. A segment move represents one portion of a complete move. A complete move is constructed out of two or more segments, starting and ending at zero velocity.

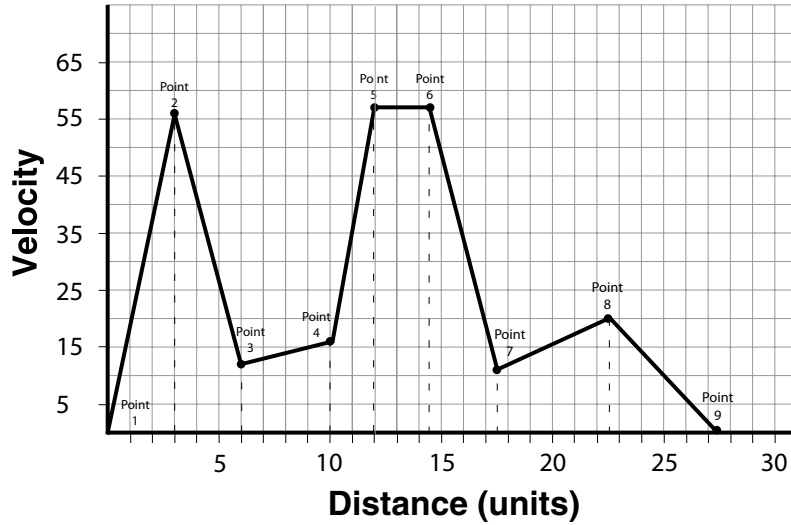
2.11.7 MDV Segments

Segments are created using a sequence of MDV statements. The simplified syntax for the **MDV (Move Distance with Velocity)** statement is:

```
MDV <distance>,<velocity>
```

The <distance> is the length of the segment move. The <velocity> is the final velocity for the segment move. The starting velocity is either zero or the final velocity of the previous segment. The final segment in a complete move must have a velocity of zero. If the final segment has a final velocity other than zero, a motion stack underflow fault will occur (F_24).

The profile shown in Figure 20 can be broken up into 8 MDV moves. The first segment defines the distance between point 1 and point 2 and the velocity at point 2. So, if the distance between point 1 and 2 was 3 units and the velocity at point 2 was 56 Units/S, the command would be: MDV 3 , 56. The second segment gives the distance between point 2 and 3 and the velocity at point 3, and so on.



S824

Figure 20: MDV Segment Example

Table 16 lists the supporting data for the graph in Figure 20.

Table 16: MDV Segment Example

Segment Number	Distance moved during segment	Velocity at the end of segment
1	3	56
2	3	12
3	4	16
4	2	57
5	2.5	57
6	3	11
7	5	20
8	5	0
-	-	-

;Segment moves

```
MDV 3 , 56
MDV 3 , 12
MDV 4 , 16
MDV 2 , 57
MDV 2.5 , 57
MDV 3 , 11
MDV 5 , 20
MDV 5 , 0
END
```

The following equation can be used to calculate the acceleration/deceleration that results from a segment move.

$$\text{Accel} = (V_f^2 - V_0^2) / [2 * D]$$

V_f = Final velocity
 V_0 = Starting velocity
 D = Distance

2.11.8 S-curve Acceleration

Instead of using a linear acceleration, the motion created using segment moves (MDV statements) can use S-curve acceleration. The syntax for MDV move with S-curve acceleration is:

```
MDV <distance>,<velocity>,S
```

Segment moves using S-curve acceleration will take the same amount of time as linear acceleration segment moves. S-curve acceleration is useful because it is much smoother at the beginning and end of the segment, however, the peak acceleration of the segment will be twice as high as the acceleration used in the linear acceleration segment.

2.11.9 Motion SUSPEND/RESUME

At times it is necessary to control the motion by preloading the motion stack with motion profiles. Then, based on the User Program, execute those motion profiles at some predetermined instance. The statement "MOTION SUSPEND" will suspend motion until the statement "MOTION RESUME" is executed. While motion is suspended, any motion statement executed by the User Program will be loaded into the motion stack. When the "MOTION RESUME" statement is executed, the preloaded motion profiles will be executed in the order that they were loaded.

Example:

```
MOTION SUSPEND
MDV 10,2 ;placed in stack
MDV 20,2 ;placed in stack
MDV 2,0 ;placed in stack
MOVED 3,C ;must use ",C "modifier. Otherwise program will hang.
MOTION RESUME
```

Caution should be taken when using MOVED,MOVEP and MOVE statements. If any of the MOVE instructions are written without the "C" modifier, the program will hang or lock up. The "MOTION SUSPEND" command effectively halts all execution of motion. In the example, as the program executes the "MDV" and "MOVED" statements, those move profiles are loaded into the motion stack. If the final "MOVED" is missing the "C" modifier then the User Program will wait until that move profile is complete before continuing on. Because motion has been suspended, the move will never be complete and the program will hang on this instruction.

2.11.10 Conditional Moves (MOVE WHILE/UNTIL)

The statements "MOVE UNTIL <expression>" and "MOVE WHILE <expression>" will both start their motion profiles based on their acceleration and max velocity profile settings. The "MOVE UNTIL <expression>" statement will continue the move until the <expression> becomes true. The "MOVE WHILE <expression>" will also continue its move while it's <expression> is true. Expression can be any valid arithmetic or logical expressions or their combination.

Examples:

```
MOVE WHILE APOS<20 ;Move while the position is less then 20, then
;stop with current deceleration rate.
MOVE UNTIL APOS>V1 ;Move positive until the position is greater than
;the value in variable V1
MOVE BACK UNTIL APOS<V1 ;Move negative until the position is less than the
;value in variable V1
MOVE WHILE IN_A1 ;Move positive while input A1 is activated.
MOVE WHILE !IN_A1 ;Move positive while input A1 is not activated.
;The exclamation mark (!) in front of IN_A1 inverts
;(or negates) the value of IN_A1.
```

This last example is a convenient way to find a sensor or switch.

2.11.11 Motion Queue and Statement Execution while in Motion

By default when the program executes a MOVE, MOVED or MOVEP statement, it waits until the motion is complete before going on to the next statement. This effectively will suspend the program until the requested motion is done. Note that "EVENTS" are not suspended however and continue executing in parallel with the User Program. The Continue "C" argument is very useful when it is necessary to trigger an action (handle I/O) while the motor is in motion. Below is an example of the Continue "C" argument.

```
;This program monitors I/O in parallel with motion:
START:
    MOVED 100,C           ;start moving max 100 revs
WHILE F_MCOMPLETE=0     ;while moving
    IF IN_A2 == 1       ;if sensor detected
        OUT1=1          ;turn ON output
        WAIT TIME 500   ;500 mS
        OUT1=0          ;turn output OFF
        WAIT TIME 500   ;wait 500 ms
    ENDIF
ENDWHILE
MOVED -100              ;Return back
WAIT TIME 1000          ;wait time
GOTO START              ;and start all over
END
```

This program starts a motion of 100 revolutions. While the motor is in motion, input A2 is monitored. If Input A2 is made during the move, then output 1 is turned on for 500ms and then turned off. The program will continue to loop in the WHILE statement, monitoring input A2, until the move is completed. If input 2 remains ON, or made, during the move, then Output 1 will continue to toggle On and Off every 500ms until the move is complete. If input A2 is only made while the motion passes by a sensor wired to the input, then output 1 will stay on for 500ms only. By adding the "Continue" argument "C" to the MOVE statement, the program is able to monitor the input while executing the motion profile. Without this modifier the program would be suspended until all motion is done making it impossible to look for the input during the move. After the motor has traveled the full distance it then returns back to its initial position and the process repeats. This program could be used for a simple paint mechanism which turns ON a paint spray gun as soon as the part's edge (or part guide) crosses the sensor(s) because delays, such as the one created by the 'Wait Time 500' statement are not allowed in an events task, this processor needs to be executed from the main program with the 'C' modifier on the move statement as shown.

Figure 21 illustrates the structure and operation of the Motion Queue. All moves are loaded into the Motion Queue before they are executed. If the move is a standard move, "MOVEP 10" or "MOVED 10", then the move will be loaded into the queue and the execution of the User Program will be suspended until the move is completed. If the move has the continue argument, e.g. "MOVEP 10,C" or "MOVED 10,C", or if it is an "MDV" move, then the moves will be loaded into Motion Queue and executed simultaneously with the User Program.

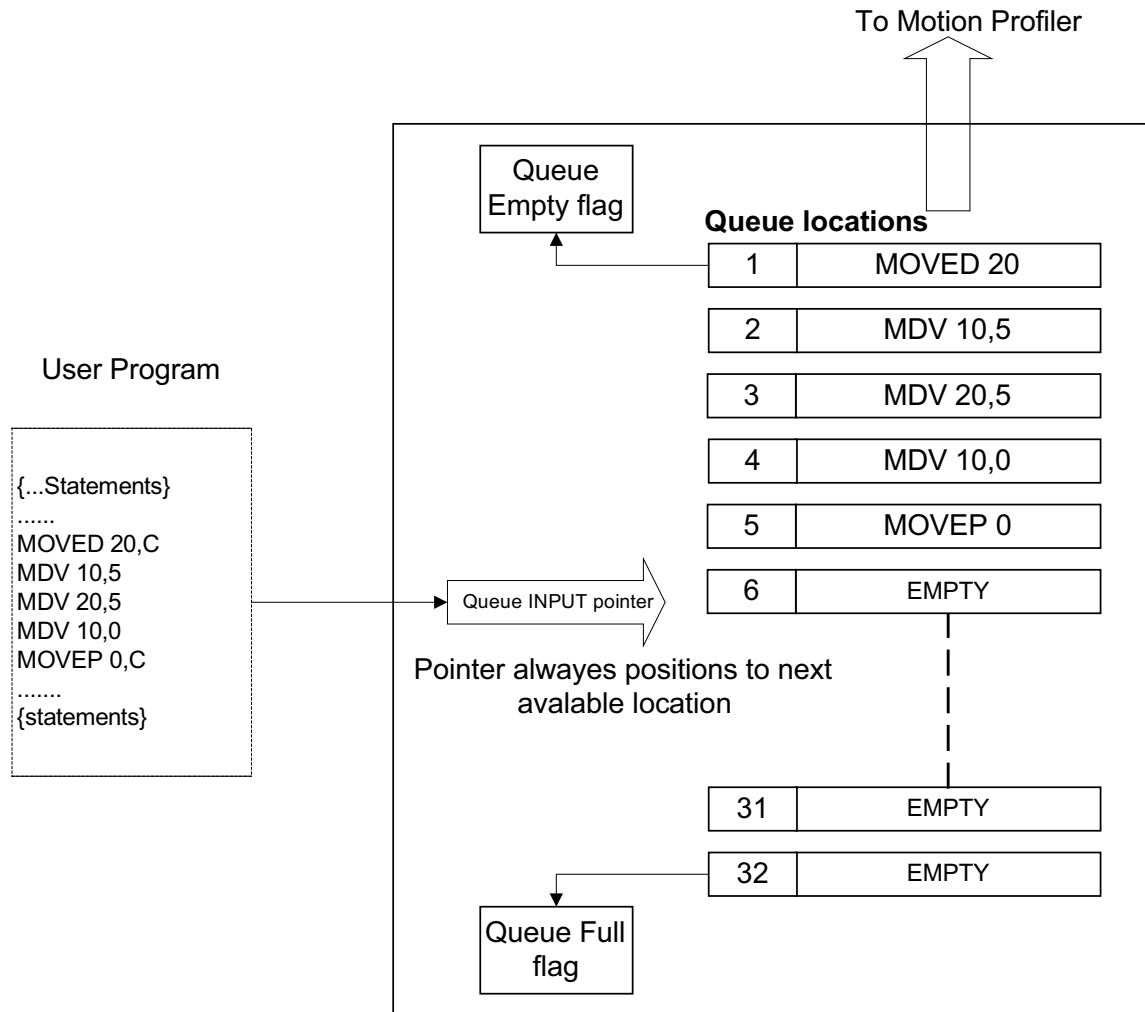


Figure 21: Motion Queue

The Motion Queue can hold a maximum of 32 motion profiles. The System Status Register contains bit values that indicate the state of the Motion Queue. Additionally, system flags (representing individual bits of the status register) are available for ease of programming. If the possibility of overflow exists, the programmer should check the Motion Queue full flag before executing any MOVE statements, especially in programs where MOVE statements are executed in a looped fashion. Attempts to execute a motion statement while the Motion Queue is full will result in fault #23. MDV statements don't have the "C" option and therefore the program is never suspended by these statements. If last MDV statement in the Queue doesn't specify a 0 velocity Motion, a Stack Underflow fault #24 will occur.

The "MOTION SUSPEND" and "MOTION RESUME" statements can be utilized to help manage the User Program and the Motion Queue. If the motion profiles loaded into the queue are not managed correctly, the Motion Queue can become overloaded which will cause the drive to fault.

2.12 System Status Register (DSTATUS register)

System Status Register, (DSTATUS), is a Read Only register. Its bits indicate the various states of the PositionServo's subsystems as listed in Table 17. Some of the flags are available as System Flag Variables and summarized in Table13.

Table 17: DSTATUS Register

Bit in register	Description
0	Set when drive enabled
1	Set if DSP subsystem at any fault
2	Set if drive has a valid program
3	Set if byte-code or system or DSP at any fault
4	Set if drive has a valid source code
5	Set if motion completed and target position is within specified limits
6	Set when scope is triggered and data collected
7	Set if motion stack is full
8	Set if motion stack is empty
9	Set if byte-code halted
10	Set if byte-code is running
11	Set if byte-code is set to run in step mode
12	Set if byte-code is reached the end of program
13	Set if current limit is reached
14	Set if byte-code at fault
15	Set if no valid motor selected
16	Set if byte-code at arithmetic fault
17	Set if byte-code at user fault
18	Set if DSP initialization completed
19	Set if registration has been triggered
20	Set if registration variable was updated from DSP after last trigger
21	Set if motion module at fault
22	Set if motion suspended
23	Set if program requested to suspend motion
24	Set if system waits completion of motion
25	Set if motion command completed and motion Queue is empty
26	Set if byte-code task requested reset
27	If set interface control is disabled. This flag is set/clear by ICONTROL ON/OFF statement.
28	Set if positive limit switch reached
29	Set if negative limit switch reached
30	Events disabled. All events disabled when this flag is set. After executing EVENTS ON all events previously enabled by EVENT EventName ON statements become enabled again

PositionServo variable #83 provides Extended Status Bits, the encoding of which is listed in Table 18.

Programming

Table 18: Encoding for Extended Status Bits (Variable #83 EXSTATUS):

Bit #	Function	Comment
0	Reserved	
1	Velocity in specified window	Velocity in limits as per parameter #59: VAR_VLIMIT_SPEEDWND
2-4	Reserved	
5	Velocity at 0 (zero)	Velocity 0: Zero defined by parameter #58: VAR_VLIMIT_ZEROSPEED
6,7	Reserved	
8	Bus voltage below under-voltage limit	Utilized to indicate drive is operating from +24V keep alive and a valid DC bus voltage level is not present.
9,10	Reserved	
11	Regen circuit is on	Drive regeneration circuit is active. Drive will be dissipating power through the braking resistor (if fitted).
12-20	Reserved	
21	Set if homing operation in progress	Drive executing Pre-defined homing function (see section 2.15).
22	Set if system homed	Drive completed Pre-defined homing function (see section 2.15).
23	If set then last fault will remain on the display until re-enabled.	User can set this bit to retain fault code on the display until re-enabled. It is useful if there is a fault handler routine. When the fault handler is exited, the fault number on the display will be replaced by current status (usually DiS if bit #24 is not set). Setting bit #24 retains diagnostics on the display.
24	Set if EIP IO exclusive owner connection is established. Cleared if closed.	Checks if drive is controlled by EthernetIP master. Use bit #24 and bit #25 to process "lost of connection" condition (if needed) in the user's program
25	Set if EIP IO exclusive owner connection times out. Cleared if exc. owner conn exists.	Checks if connection with Ethernet/IP master is lost. Use bit #24 and bit #25 to process "lost of connection" condition (if needed) in the user's program
26-31	Reserved	

2.13 Fault Codes (DFAULTS register)

Whenever a fault occurs in the drive, a record of that fault is recorded in the Fault Register (DFAULTS). In addition, specific flags in the System Status Register will be set helping to indicate what class of fault the current fault belongs to. Table 19 summarizes the possible fault codes. Codes from 1 to 16 are used for DSP subsystem errors. Codes above that range are generated by various subsystems of the PositionServo.

Table 19: DFAULTS Register

Fault ID	Associated flags in status register	Description
1	1, 3	Over voltage
2	1, 3	Invalid Hall sensors code
3	1, 3	Over current
4	1, 3	Over temperature
5	1, 3	The drive is disabled by the EN954-1 Safety Function
6	1, 3	Over speed. (Over speed limit set by motor capability in motor file)
7	1, 3	Position error excess.
8	1, 3	Attempt to enable while motor data array invalid or motor was not selected.
9	1,3	Motor over temperature switch activated
10	1,3	Sub processor error
11-13	-	Reserved
14	1,3	Under voltage
15	1,3	Hardware current trip protection
16	-	Reserved
17	3	Unrecoverable error.
18	16	Division by zero
19	16	Arithmetic overflow

Programming

Fault ID	Associated flags in status register	Description
20	3	Subroutine stack overflow. Exceeded 16 levels subroutines stack depth.
21	3	Subroutine stack underflow. Executing RETURN statement without preceding call to subroutine.
22	3	Variable evaluation stack overflow. Expression too complicated for compiler to process.
23	21	Motion Queue overflow. 32 levels depth exceeded
24	21	Motion Queue underflow. Last queued MDV statement has non 0 target velocity
25	3	Unknown opcode. Byte code interpreter error
26	3	Unknown byte code. Byte code interpreter error
27	21	Drive disabled. Attempt to execute motion while drive is disabled.
28	16, 21	Accel too high. Motion statement parameters calculate an Accel value above the system capability.
29	16, 21	Accel too low. Motion statement parameters calculate an Accel value below the system capability.
30	16, 21	Velocity too high. Motion statement parameters calculate a velocity above the system capability.
31	16, 21	Velocity too low. Motion statement parameters calculate a velocity below the system capability.
32	3,21	Positive limit switch engaged
33	3,21	Negative limit switch engaged
34	3,21	Attempt at positive motion with engaged positive limit switch
35	3,21	Attempt at negative motion with engaged negative limit switch
36	3	Hardware disable (enable input not active when attempting to enable drive from program or interface)
37	3	Undervoltage
38	3	EPM loss
39	3,21	Positive soft limit reached
40	3,21	Negative soft limit reached
41	3	Attempt to use variable with unknown ID from user program
45	1,3	Secondary encoder position error excess

2.14 Limitations and Restrictions

Communication Interfaces Usage Restrictions

Simultaneous connection to the RS485 port is allowed for retransmitting (conversion) between interfaces.



WARNING!

Usage of the RS485 simultaneously with Ethernet may lead to unpredictable behavior since the drive will attempt to perform commands from both interfaces concurrently.

Motion Parameters Limitation

Due to a finite precision in the calculations there are some restrictions for acceleration/deceleration and max velocity for a move. If you receive arithmetic faults during your programs execution, it is likely due to these limitations. Min/Max values are expressed in counts or counts/sample, where the sample is a position loop sample interval (512µsec).

Table 20: Motion Parameter Limits

Parameter	MIN	MAX	Units
Accel / Decel	65/(2 ³²)	512	counts/sample ²
MaxV (maximum velocity)	0	2048	counts/sample
Max move distance	0	+/- 2 ³¹	counts

Stacks and Queues Depth Limitations

Table 21: Stack Depth Limit

Stack/Queue	Motion Queue	Subroutines Stack	Number of Events
Depth	32	32	32

2.15 Homing

2.15.1 What is Homing?

Predefined (firmware based) homing functionality is available on PositionServo drives with firmware 3.03 or later. In addition custom homing functionality can be created by the programmer within the user program by utilizing the programming command set available.

Examples of custom homing routine creation as well as user program code to replicate each of the predefined homing routines is available from technical support.

Homing is the method by which a drive seeks the home position (also called the datum, reference point, or zero point). There are various methods of achieving this using:

- limit switches at the ends of travel, or
- a dedicated home switch, or
- an Index Pulse or zero reference from the motor feedback device, or
- a combination of the above.

In order to use home methods involving Motor Index Pulse (zero pulse), the index pulse of the motor **MUST** be connected to the drive registration input (C3). For encoder motors this connection can be made directly. Connect the 0V ref for the encoder to P3-36 (IN_C_COM) and the Z+ line from the encoder to P3-39 (IN_C3).

For convenience of wiring and for Resolver motors the Z pulse output from the simulated encoder can be looped back into the C3 registration input. Connect P3-36 (IN_C_COM) to the digital ground terminal P3-5 and P3-39 (IN_C3) to P3-11 (BZ+). For Resolver motors the Z Pulse is created by the simulated encoder at 0 degrees of the motor shaft.

Establish the time period that the Z pulse must be present on the input in order for it to be reliably detected (back thru C3), by calculating the maximum homing speed for the specific application. A 1k Ω pull-up resistor is available for those with issues picking up the index pulse.

2.15.2 The Homing Function

The homing function provides a set of trajectory parameters to the position loop, as shown in Figure 22. They are calculated based on user supplied variable values such as:

VAR_HOME_OFFSET
VAR_HOME_METHOD
VAR_HOME_SWITCH_INPUT
VAR_HOME_FAST_VEL
VAR_HOME_SLOW_VEL
VAR_HOME_ACCEL
VAR_START_HOMING

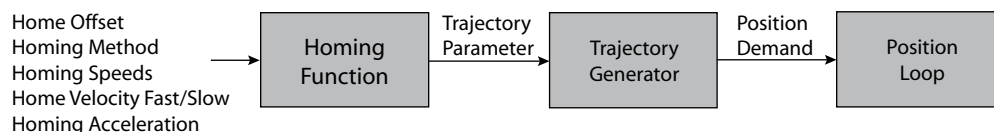


Figure: 22: Homing Function

Homing Function Monitoring:

The extended drive status variable (#83 EXSTATUS variable) contains bit values for monitoring the homing function over the communications interface.

Bit 21 of #83 indicates homing procedure in progress and is set to logic 1 while homing is being executed.

Bit 22 of #83 indicates homing complete. It is set to 1 upon the successful completion of the homing routine.

2.15.3 Home Offset

The home offset is the difference between the zero position for the application and the machine home position (found during homing). During homing the home position is found and once the homing is completed the zero position is offset from the home position by adding the home offset to the home position. All subsequent absolute moves shall be taken relative to this new zero position. This is illustrated in Figure 23. Offset can either be set in User Units (UU) by writing to variable #240, or in encoder counts by writing to variable #241. Setting a value for either variable #240 or #241 will result in the value being automatically calculated for the respective variable.

VAR_HOME_OFFSET (#240)
VAR_HOME_OFFSET_PULSES (#241)

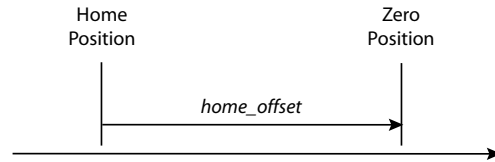


Figure 23: Home Offset

2.15.4 Homing Velocity

There are two homing velocities: fast and slow. These velocity variables are used to find the home switch and to find the index pulse. Which velocity (fast or slow) is used to locate the home switch and the index pulse depends on the homing routine selected.

VAR_HOME_FAST_VEL (#242)
VAR_HOME_SLOW_VEL (#243)

2.15.5 Homing Acceleration

Homing acceleration establishes the velocity ramp rate to be used for all accelerations and decelerations within the standard homing modes. Note that in homing, it is not possible to program a separate deceleration rate.

VAR_HOME_ACCEL (#239)

2.15.6 Homing Switch

The homing switch variable enables the user to select the PositionServo input used for the Home Switch connection. The Homing Switch Input Assignment range is 0 - 11. Inputs A1-A4 are assigned 0 to 3, respectively; inputs B1-B4 are assigned 4 to 7, respectively; and inputs C1-C4 are assigned 8 to 11, respectively.

VAR_HOME_SWITCH_INPUT (#246)



WARNING!

- Setting inputs A1 and A2 as the home switch in methods that do **NOT** use limit switches can cause the drive to behave in an unexpected manner.
- Input A3 is a dedicated hardware enable input and should **never** be assigned as the homing switch input.
- Input C3 can be used as the homing switch input only in methods that do not home to an index pulse from an encoder. Methods that use an index pulse automatically use Input C3 for capture of the index pulse, as described previously.

2.15.7 Homing Start

The homing operation is initiated using the home start variable. Start Homing range is: 0 or 1. When set to 0, no action occurs. When set to 1, the homing operation is started. It is recommended to directly write to VAR_START_HOMING solely via network communications.

VAR_START_HOMING (#245)

'HOME' is the logical command to set VAR_START_HOMING. Writing the word 'HOME' within the user program will result in the homing operation commencing. After initiating the HOME command with firmware 3.60 (and later) the user program will not execute subsequent lines of code until after homing is completed (similar to MOVE P). If either using firmware prior to 3.60 or if user initiates homing in the indexer program via the statement VAR_START_HOMING=1, then it is recommended to immediately follow that statement with the following code:

```
WAIT UNTIL VAR_EXSTATUS & 0x400000 == 0x400000.
```

Doing this ensures no further lines of code will be executed until homing is complete.

Programming

2.15.8 Homing Method

VAR_HOME_METHOD (#244)

The Home Method establishes the method that will be used for homing. All supported methods are summarized in Table 22 and described in sections 2.15.9.1 through 2.15.9.25. These homing methods define the location of the home position. The zero position is always the home position adjusted by the homing offset.

Table 22: Homing Methods

Method	Home Position
0	No operation/reserved. An attempt to execute 0 will result in execution of method 1.
1	Location of first encoder index pulse is on the positive side of the negative limit switch.
2	Location of first encoder index pulse is on the negative side of the positive limit switch.
3	Location of first index pulse is on the negative side of a positive home switch. ¹
4	Location of first index pulse is on the positive side of a positive home switch. ¹
5	Location of first index pulse is on the positive side of a negative home switch. ²
6	Location of first index pulse is on the negative side of a negative home switch. ²
7	Location of first index pulse is on the negative side of the negative edge of an intermittent home switch. ³
8	Location of first index pulse is on the positive side of the negative edge of an intermittent home switch. ³
9	Location of first index pulse is on the negative side of the positive edge of an intermittent home switch. ³
10	Location of first index pulse is on the positive side of the positive edge of an intermittent home switch. ³
11	Location of first index pulse is on the positive side of the positive edge of an intermittent home switch. ³
12	Location of first index pulse is on the negative side of the positive edge of an intermittent home switch. ³
13	Location of first index pulse is on the positive side of the negative edge of an intermittent home switch. ³
14	Location of first index pulse is on the negative side of the negative edge of an intermittent home switch. ³
15	Reserved for future use.
16	Reserved for future use
17	The edge of a negative limit switch.
18	The edge of a positive limit switch.
19	The edge of a positive home switch.
20	Reserved for future use.
21	The edge of a negative home switch.
22	Reserved for future use.
23	Positive edge of an intermittent home switch.
24	Reserved for future use.
25	The negative edge of an intermittent home switch.
26	Reserved for future use.
27	Negative edge of an intermittent home switch.
28	Reserved for future use.
29	The positive edge of an intermittent home switch.
30	Reserved for future use.
31	Reserved for future use.
32	Reserved for future use.
33	The first index pulse on the negative side of the current position.
34	The first index pulse on the positive side of the current position.
35	Current position becomes home position. Home offset is also active and will be added to current position to form the final value.

1 - A positive home switch is one that goes active at some position, and remains active for all positions greater than that one.

2 - A negative home switch is one that goes active at some position, and remains active for all positions less than that one.

3 - An intermittent home switch is one that is only active for a limited range of travel.

2.15.9 Homing Methods

There are several types of homing methods but each method establishes the:

- Homing signal (positive limit switch, negative limit switch, home switch, or index pulse)
- Direction of actuation and, where appropriate, the direction of the index pulse.

The homing method descriptions and diagrams in this manual are based on those in the CANopen Profile for Drives and Motion Control (DSP 402). As illustrated in Figure 24, each homing method diagram shows the motor in the starting position on a mechanical stage. The arrow line indicates direction of motion and the circled number indicates the homing method (the mode selected by the Homing Method variable).

The location of the circled method number indicates the home position reached with that method. The text designators (A, B) indicate the logical transition required for the homing function to complete its current phase of motion. Dashed lines overlay these transitions and reference them to the relevant transitions of limit switches, homing sensors, or index pulses.

Definitions

Positive home switch: goes active at some position, and remains active for all positions greater than that one.

Negative home switch: goes active at some position, and remains active for all positions less than that one.

Intermittent home switch: is one that is only active for a limited range of travel.

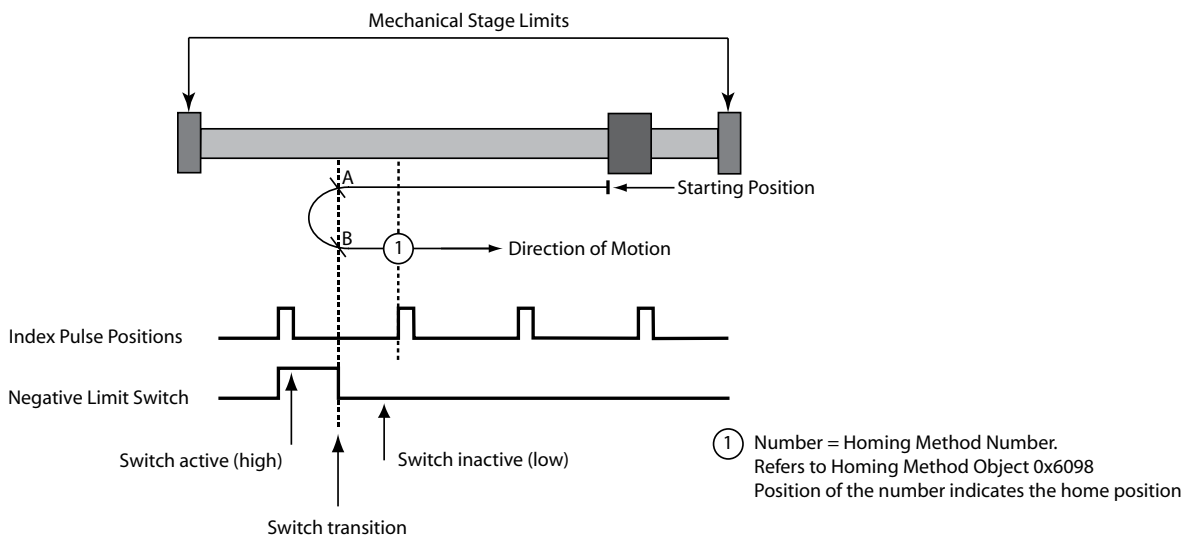


Figure 24: Homing Terms



NOTE

In the homing method descriptions, negative motion is leftward and positive motion is rightward

BLUE lines indicate fast velocity moves

GREEN lines indicate slow velocity moves

RED lines indicate slow velocity/100 moves

2.15.9.1 Homing Method 1: Homing on the Negative Limit Switch

Using this method, the initial direction of movement is negative if the negative limit switch is inactive (here shown as low). The home position is at the first index pulse to the positive of the position where the negative limit switch becomes active.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Negative Limit Switch (A1) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity. If the negative limit switch is already active when the homing routine commences then this initial move is not executed. Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until first the falling edge of the negative limit switch is detected (position B) and then the rising edge of the first index pulse (position 1) is detected.

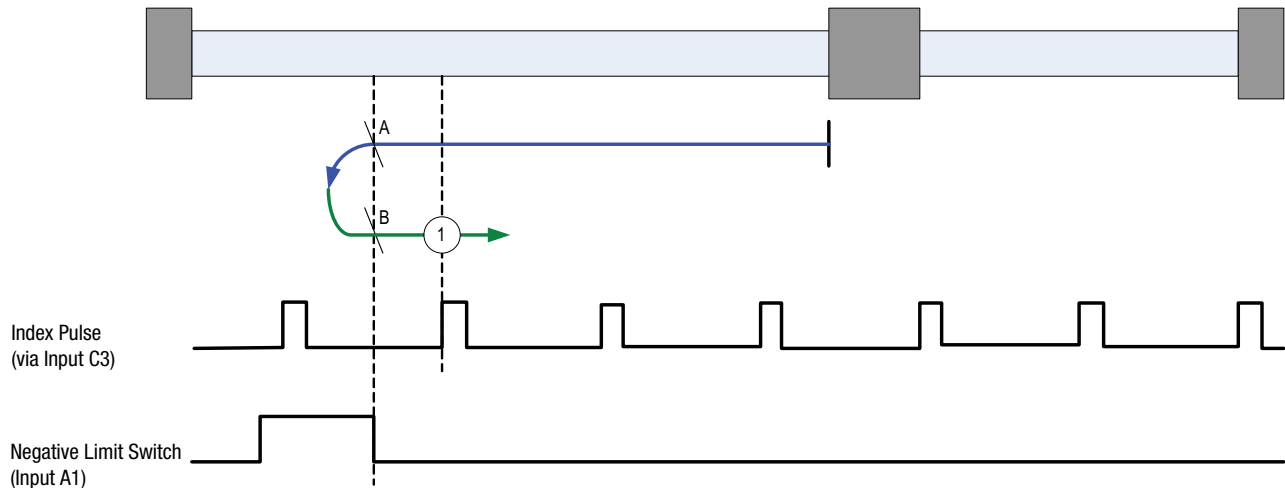


Figure 25: Homing Method 1

2.15.9.2 Homing Method 2: Homing on the Positive Limit Switch

Using this method the initial direction of movement is positive if the positive limit switch is inactive (here shown as low). The position of home is at the first index pulse to the negative of the position where the positive limit switch becomes active.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Positive Limit Switch (A2) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity. If the positive limit switch is already active when the homing routine commences then this initial move is not executed. Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until first the falling edge of the positive limit switch is detected (position B) and then the rising edge of the first index pulse (position 2) is detected.

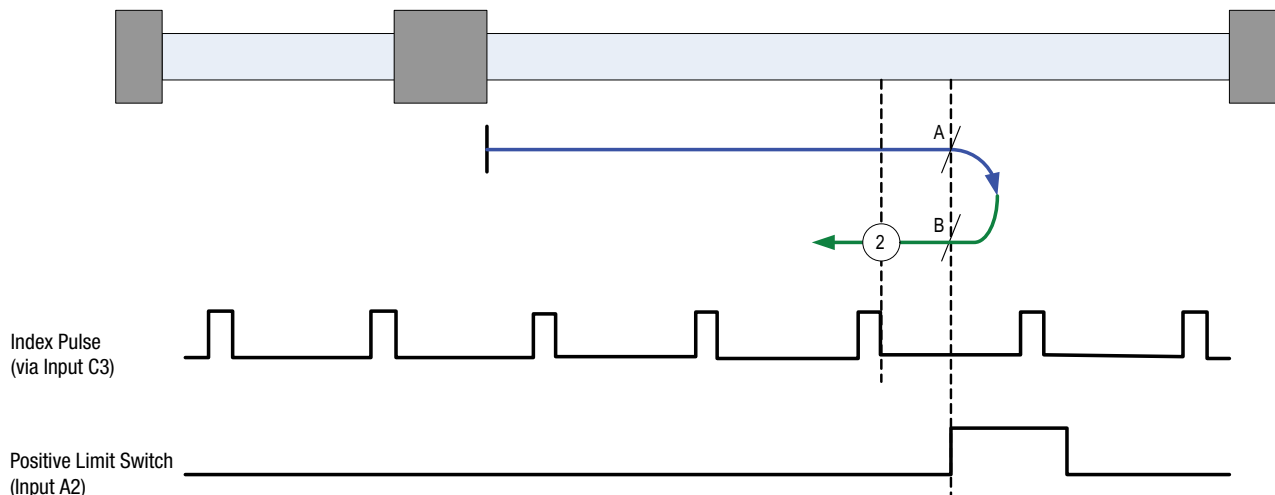


Figure 26: Homing Method 2

2.15.9.3 Homing Method 3: Homing on the Positive Home Switch & Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is inactive). The home position is the first index pulse to the negative of the position where the homing switch becomes active.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity. If the homing switch is already active when the homing routine commences then this initial move is not executed. Axis will then accelerate to **fast** homing velocity in negative direction. Motion will continue until first the falling edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 3) is detected.

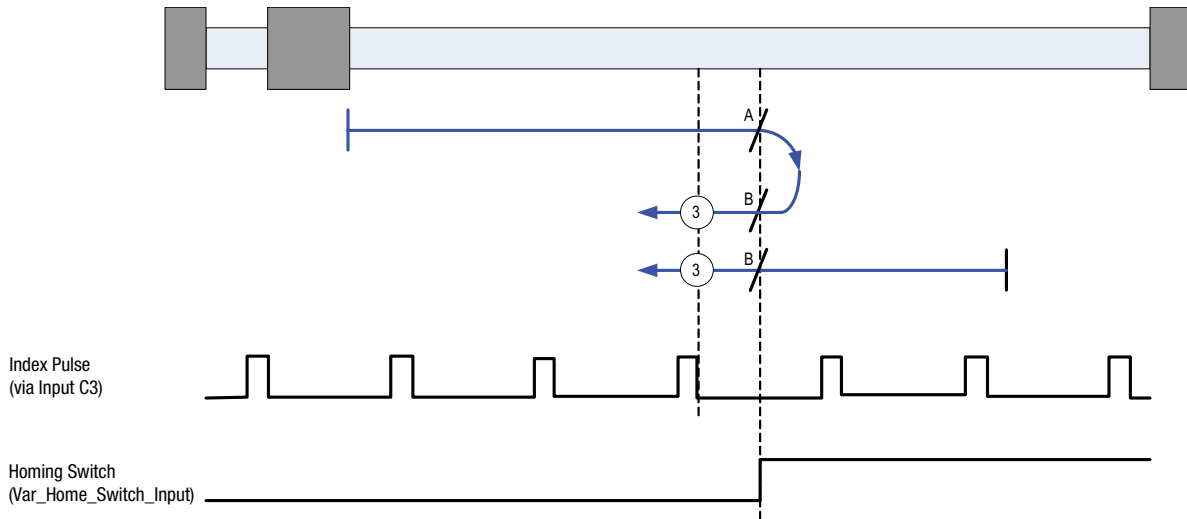


Figure 27: Homing Method 3

2.15.9.4 Homing Method 4: Homing on the Positive Home Switch & Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is active). The home position is the first index pulse to the positive of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity. If the homing switch is already inactive when the homing routine commences then this initial move is not executed. Axis will then accelerate to **fast** homing velocity in positive direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 4) is detected.

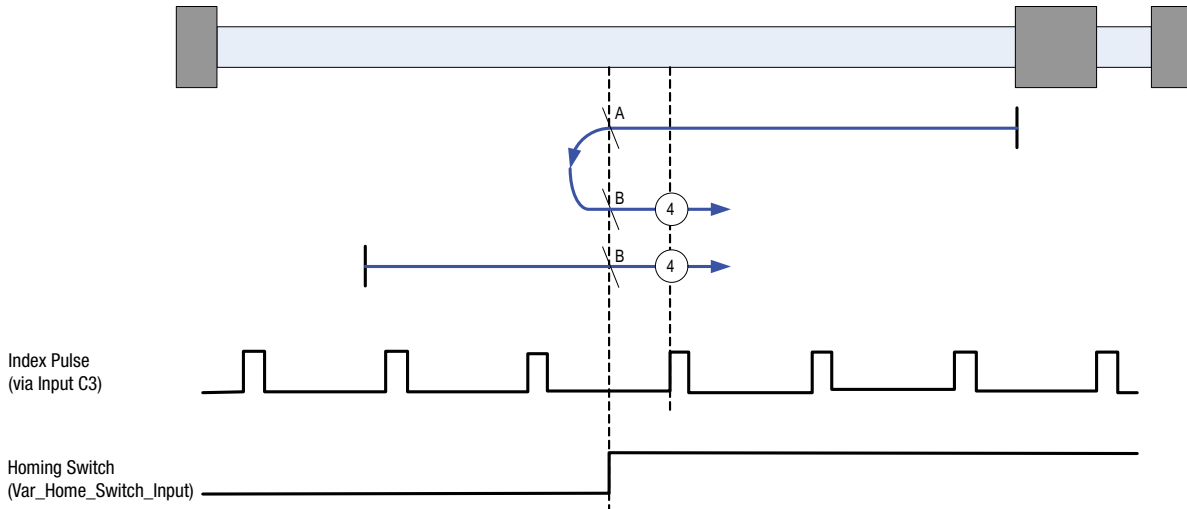


Figure 28: Homing Method 4

2.15.9.5 Homing Method 5: Homing on the Negative Home Switch & Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is inactive). The home position is the first index pulse to the positive of the position where the homing switch becomes active.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity. If the homing switch is already active when the homing routine commences then this initial move is not executed. Axis will then accelerate to **fast** homing velocity in positive direction. Motion will continue until first the falling edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 5) is detected.

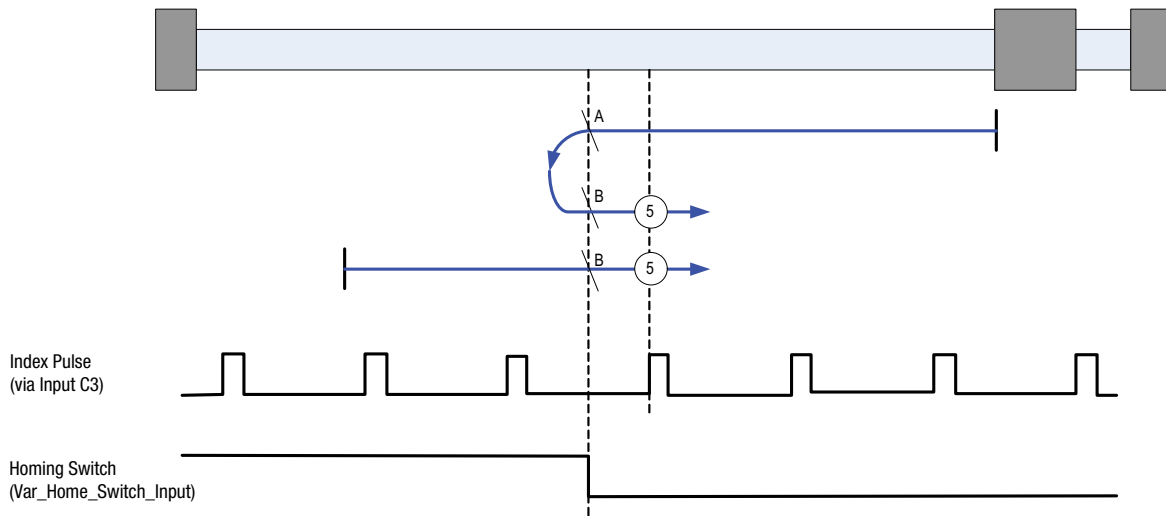


Figure 29: Homing Method 5

2.15.9.6 Homing Method 6: Homing on the Negative Home Switch & Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is active). The home position is the first index pulse to the negative of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity. If the homing switch is already inactive when the homing routine commences then this initial move is not executed. Axis will then accelerate to **fast** homing velocity in negative direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 6) is detected.

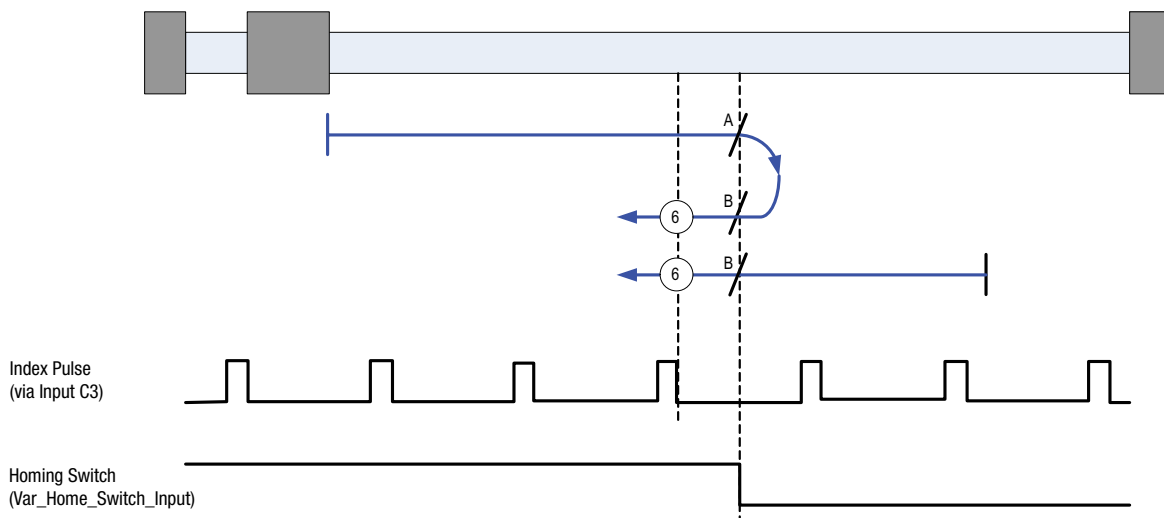


Figure 30: Homing Method 6

2.15.9.7 Homing Method 7: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is inactive). The home position is the first index pulse to the negative of the position where the homing switch becomes active.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in negative direction. Motion will continue until first the falling edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 7) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) following the procedure as detailed above, but moving negative instead of positive and without stopping on detection of the homing switch rising edge.

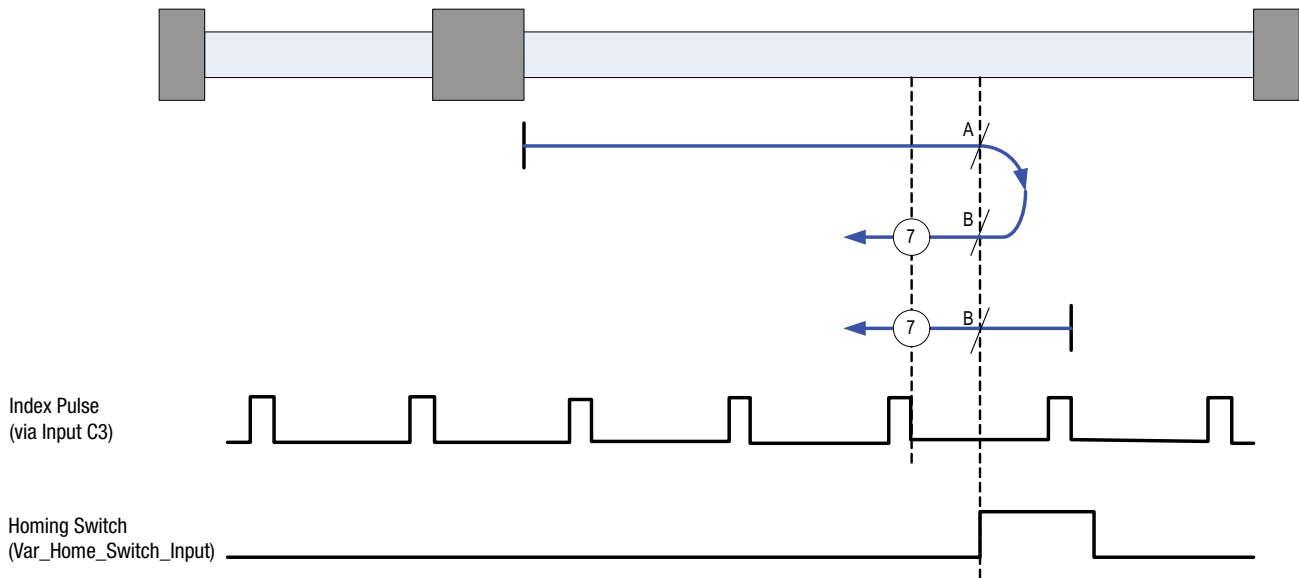


Figure 31: Homing Method 7

2.15.9.8 Homing Method 8: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is active). The home position is the first index pulse to the positive of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already inactive when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in positive direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 8) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) following the procedure as detailed above.

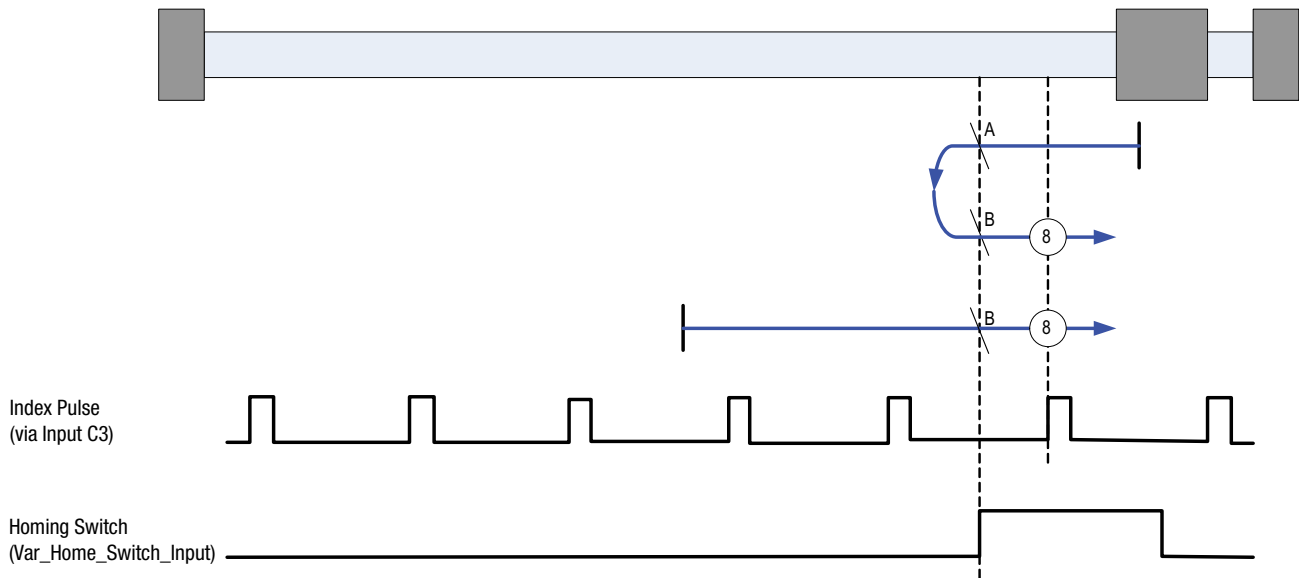


Figure 32: Homing Method 8

2.15.9.9 Homing Method 9: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is positive. The home position is the first index pulse to the negative of the position where the homing switch becomes inactive on its negative edge.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will then accelerate to **fast** homing velocity in negative direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 9) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) following the procedure as detailed above but ignoring the initial move in the positive direction.

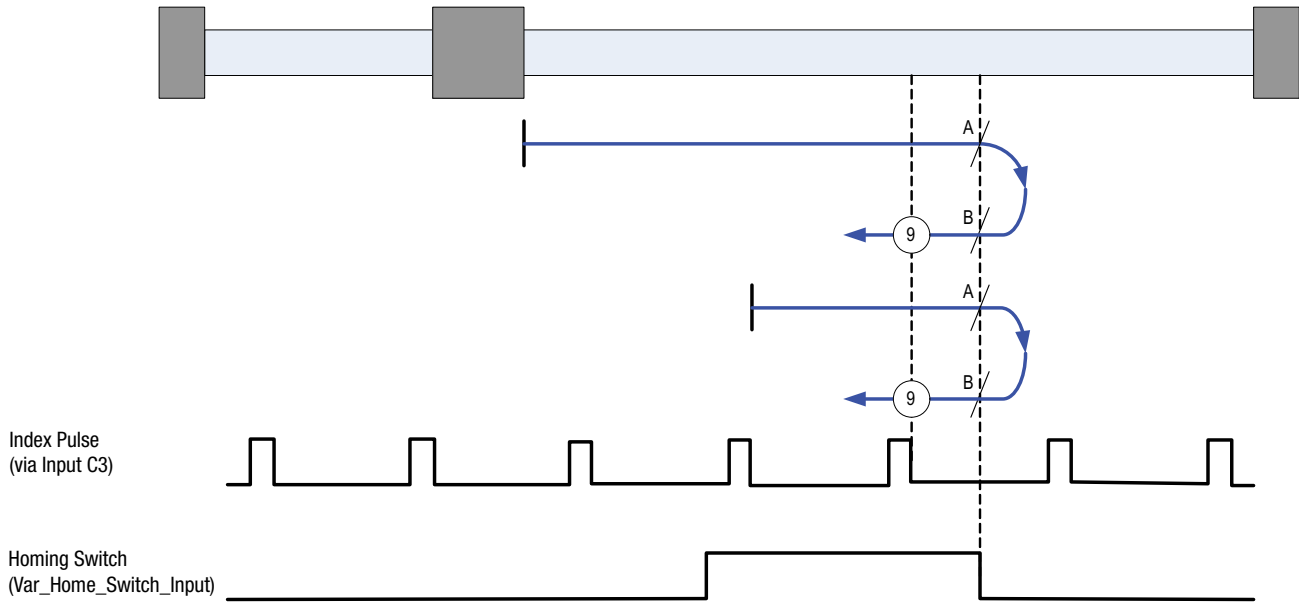


Figure 33: Homing Method 9

2.15.9.10 Homing Method 10: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is positive. The home position is the first index pulse to the positive of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will continue running at **fast** homing velocity in positive direction until the rising edge of the first index pulse (position 10) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) continuing motion until it sees the rising edge of the homing switch. The axis will then stop and follow the procedure as detailed above.

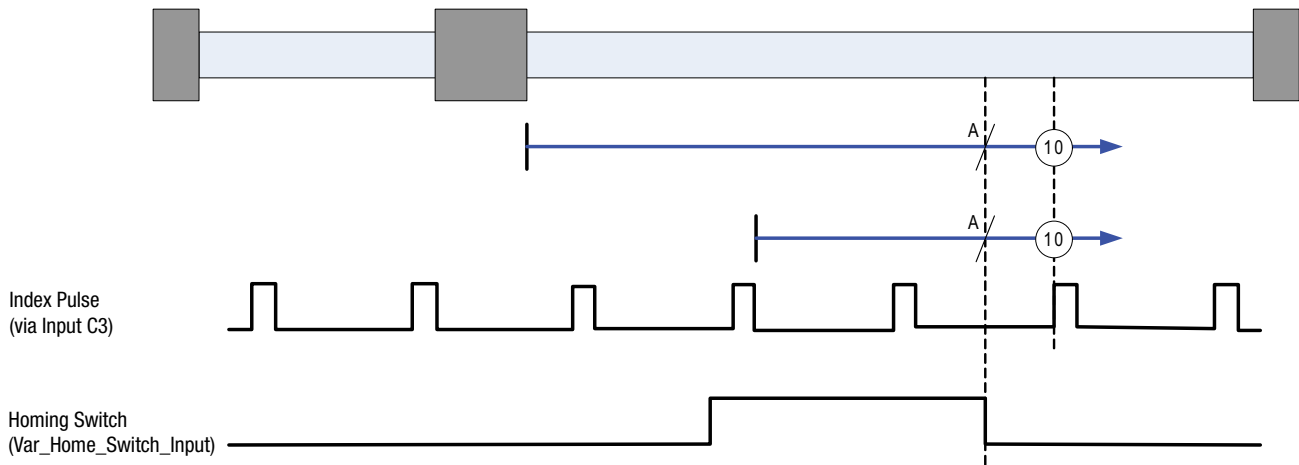


Figure 34: Homing Method 10

2.15.9.11 Homing Method 11: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is inactive). The home position is the first index pulse to the positive of the position where the homing switch becomes active.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in positive direction. Motion will continue until first the falling edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 11) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) following the procedure as detailed above, but moving positive instead of negative and without stopping on detection of the homing switch rising edge.

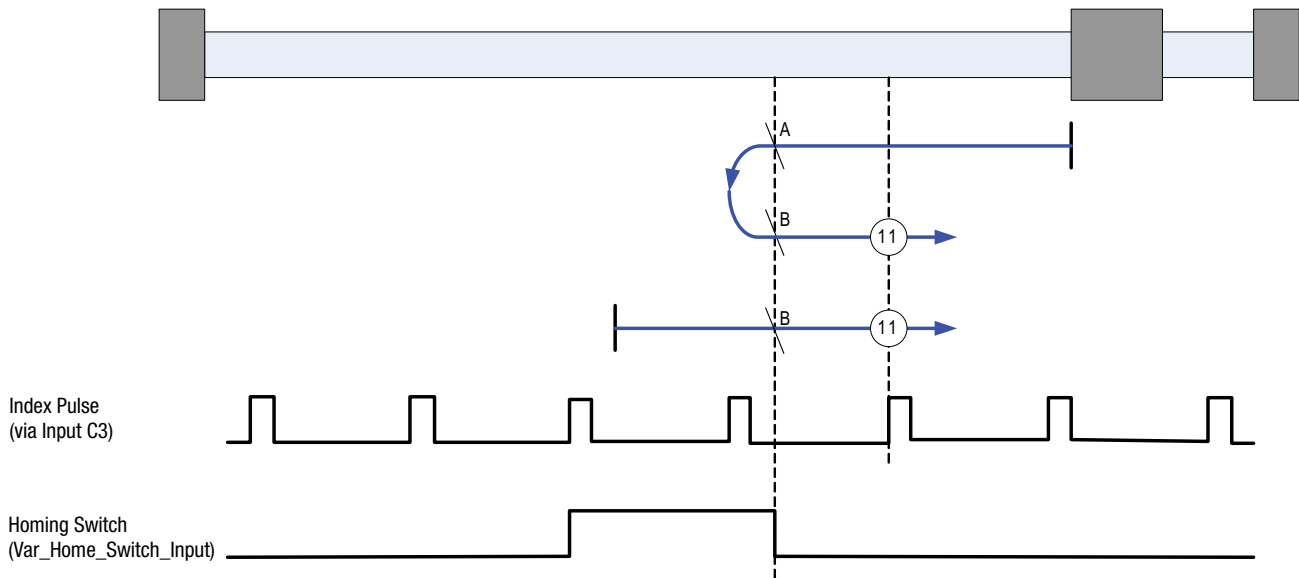


Figure 35: Homing Method 11

2.15.9.12 Homing Method 12: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is active). The home position is the first index pulse to the negative of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already inactive when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in negative direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 12) is detected.

NOTE: if it the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) following the procedure as detailed above.

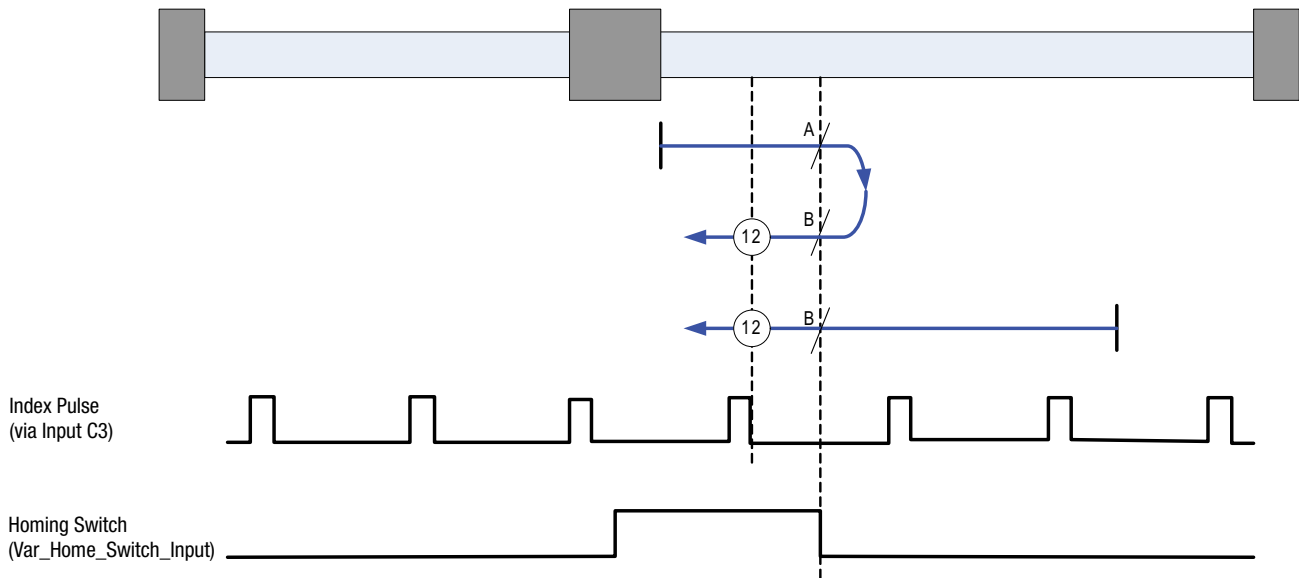


Figure 36: Homing Method 12

2.15.9.13 Homing Method 13: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is negative. The home position is the first index pulse to the positive of the position where the homing switch becomes inactive on its positive edge.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will then accelerate to **fast** homing velocity in positive direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 13) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) following the procedure as detailed above but ignoring the initial move in the negative direction.

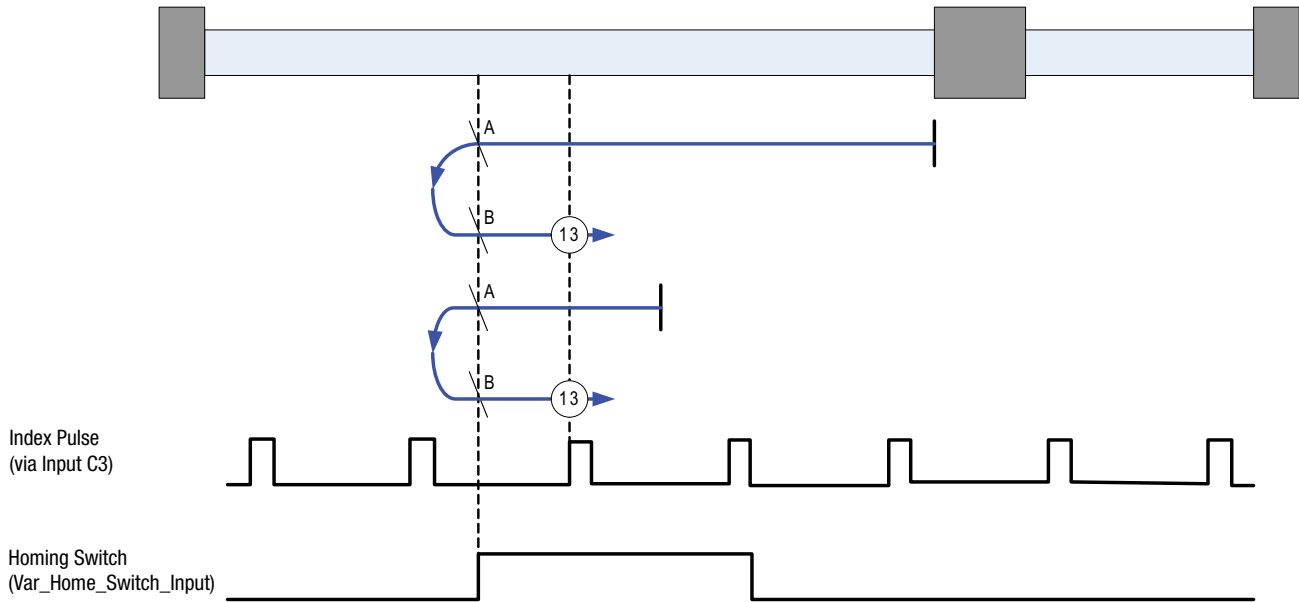


Figure 37: Homing Method 13

2.15.9.14 Homing Method 14: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is negative. The home position is the first index pulse to the negative of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will continue running at **fast** homing velocity in negative direction until the rising edge of the first index pulse (position 14) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) continuing motion until it sees the rising edge of the homing switch. The axis will then stop and follow the procedure as detailed above.

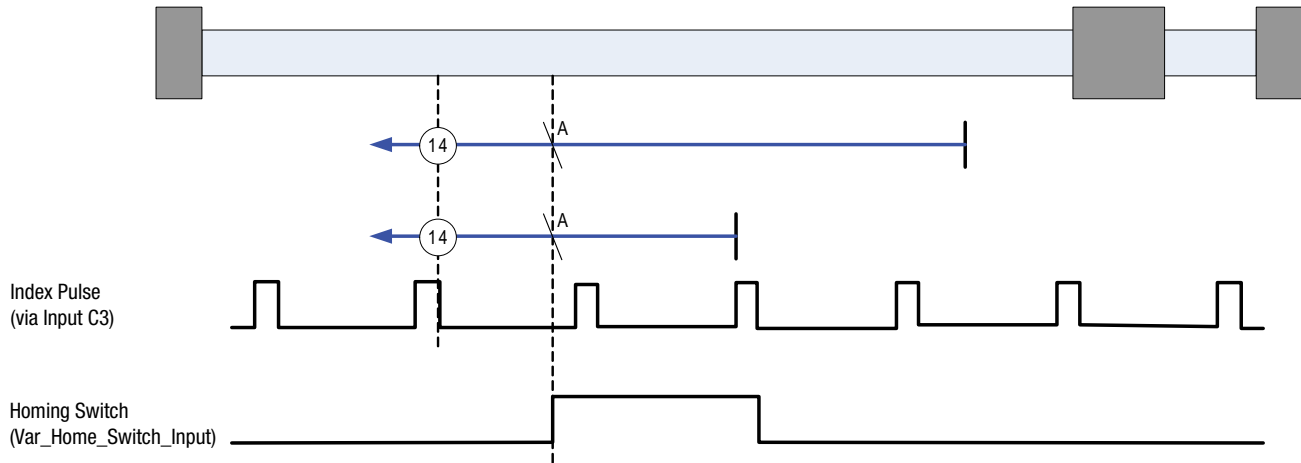


Figure 38: Homing Method 14

2.15.9.15 Homing Method 17: Homing without an Index Pulse

Method 17 is similar to method 1, except that the home position is not dependent on the index pulse but only on the negative limit switch translation.

Using this method the initial direction of movement is negative. The home position is the leading edge of the Negative limit switch.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Negative Limit Switch (A1) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the negative limit switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until the falling edge of the negative limit switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the rising edge of the negative limit switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity **divided by 100** in the positive direction. Motion will continue until the falling edge of the negative limit switch is detected (position 17). This is the home position (excluding offset).

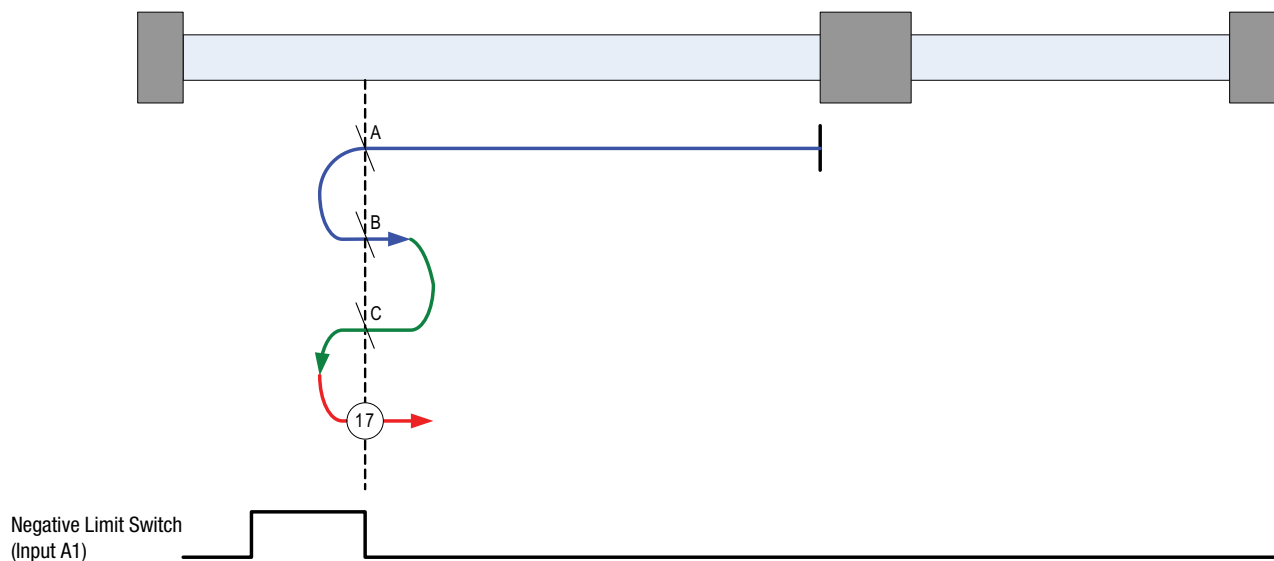


Figure 39: Homing Method 17

2.15.9.16 Homing Method 18: Homing without an Index Pulse

Method 18 is similar to method 2, except that the home position is not dependent on the index pulse but only on the Positive limit switch translation.

Using this method the initial direction of movement is positive. The home position is the leading edge of the Positive limit switch.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Positive Limit Switch (A2) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the positive limit switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until the falling edge of the positive limit switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the rising edge of the positive limit switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity **divided by 100** in the negative direction. Motion will continue until the falling edge of the positive limit switch is detected (position 18). This is the home position (excluding offset).

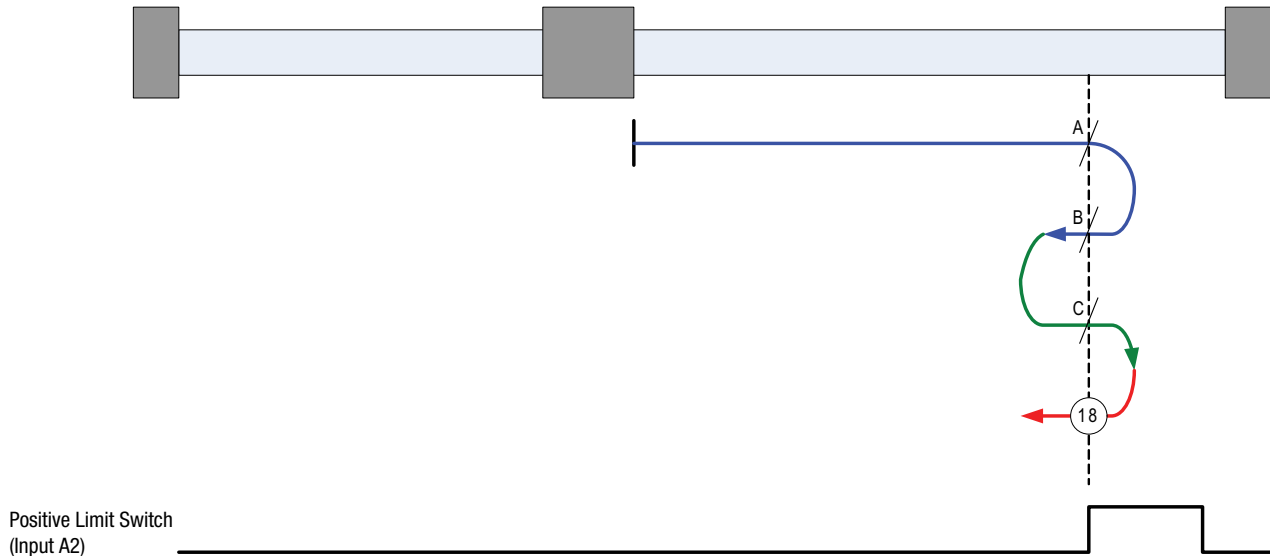


Figure 40: Homing Method 18

2.15.9.17 Homing Method 19: Homing without an Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is inactive). The home position is the leading edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until the homing switch is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the rising edge of the homing switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position 19). This is the home position (excluding offset).

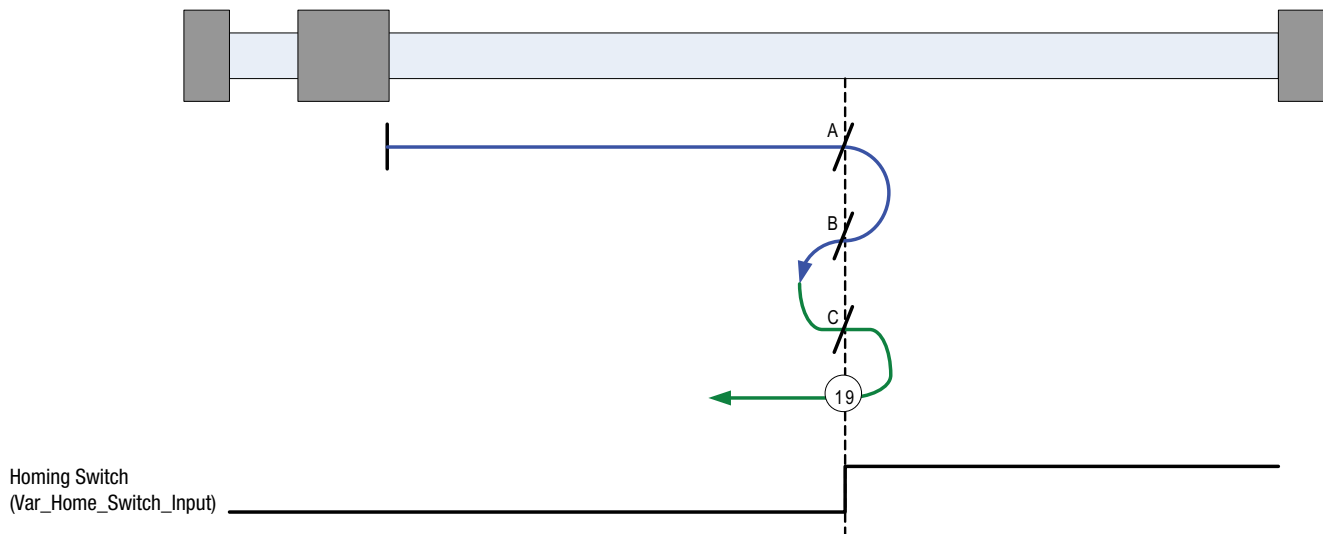


Figure 41: Homing Method 19

2.15.9.18 Homing Method 21: Homing without an Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is inactive). The home position is the leading edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until the homing switch is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the rising edge of the homing switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position 21). This is the home position (excluding offset).

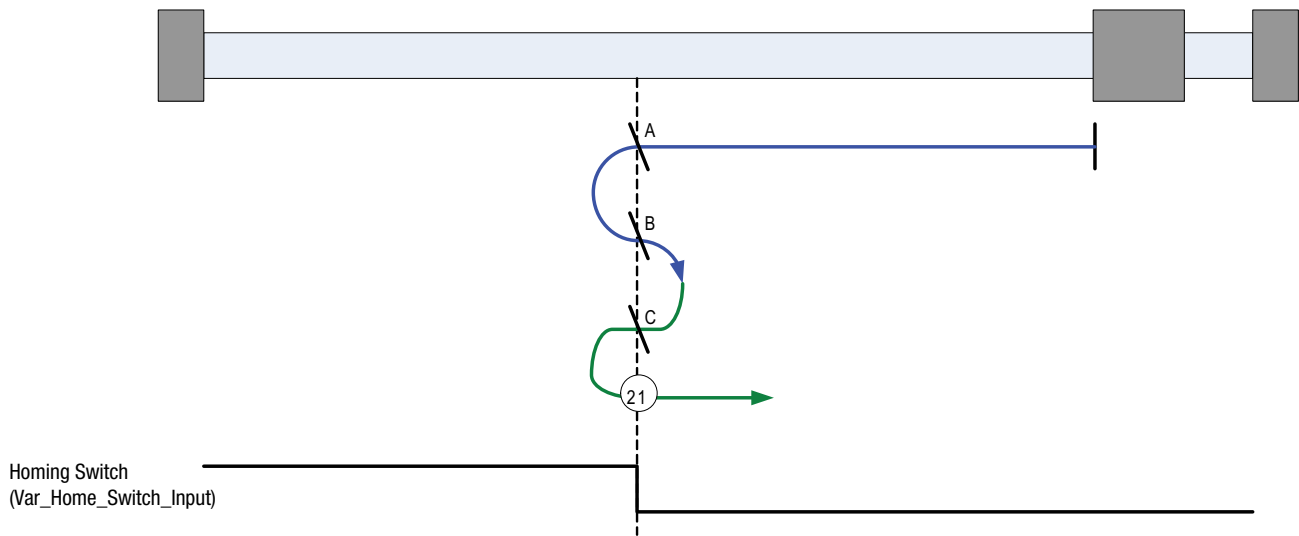


Figure 42: Homing Method 21

2.15.9.19 Homing Method 23: Homing without an Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is inactive). The home position is the leading edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until the homing switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the rising edge of the homing switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position 23). This is the home position (excluding offset).

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) following the procedure as detailed above but ignoring the initial move in the positive direction.

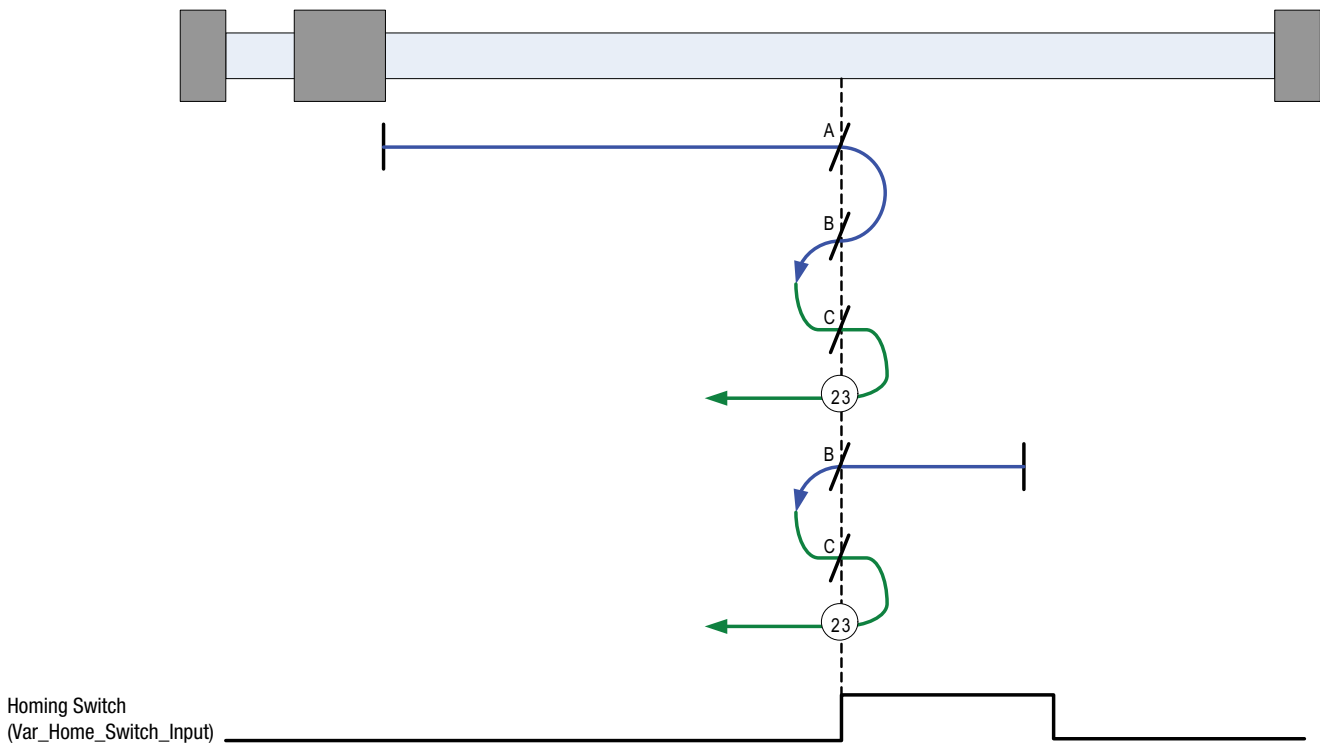


Figure 43: Homing Method 23

Programming

2.15.9.20 Homing Method 25: Homing without an Index Pulse

Using this method the initial direction of movement is positive. The home position is the negative edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the rising edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position 25). This is the home position (excluding offset).

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) continuing motion until it sees the rising edge of the homing switch. The axis will then stop and follow the procedure as detailed above.

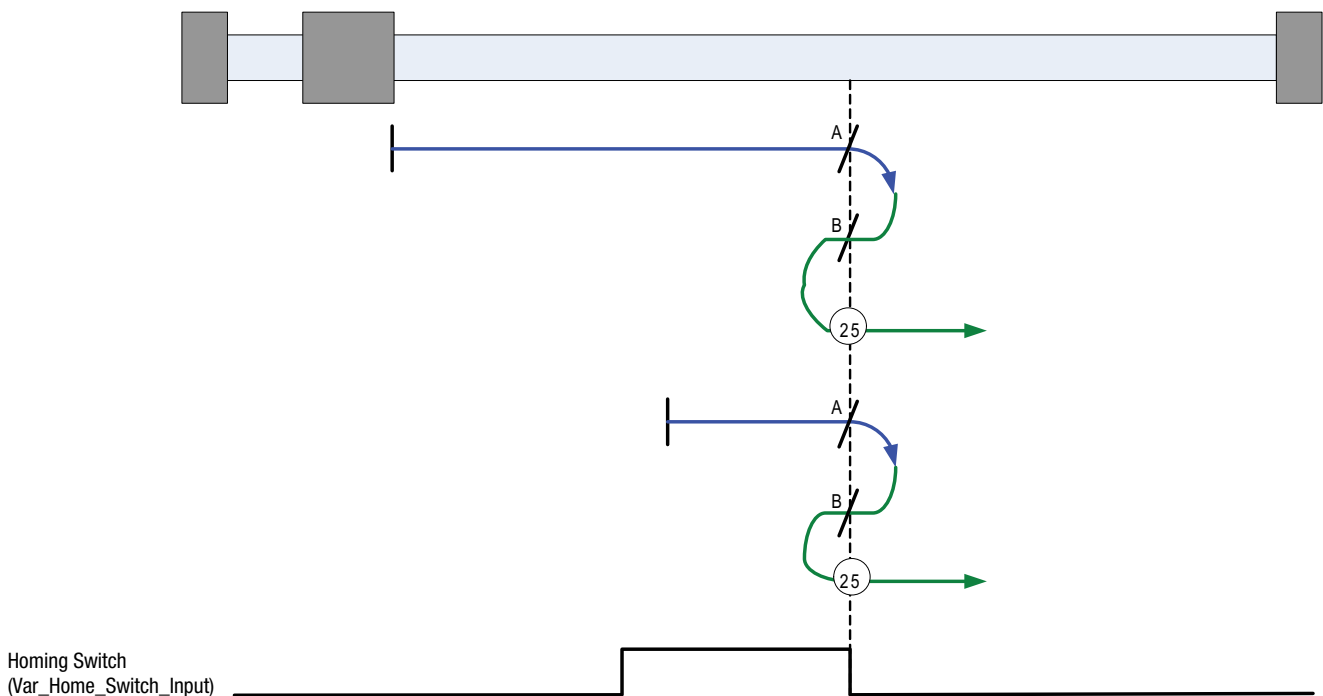


Figure 44: Homing Method 25

2.15.9.21 Homing Method 27: Homing without an Index Pulse

Using this method the initial direction of movement is negative. The home position is the negative edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the rising edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position 27). This is the home position (excluding offset).

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) continuing motion until it sees the rising edge of the homing switch. The axis will then stop and follow the procedure as detailed above.

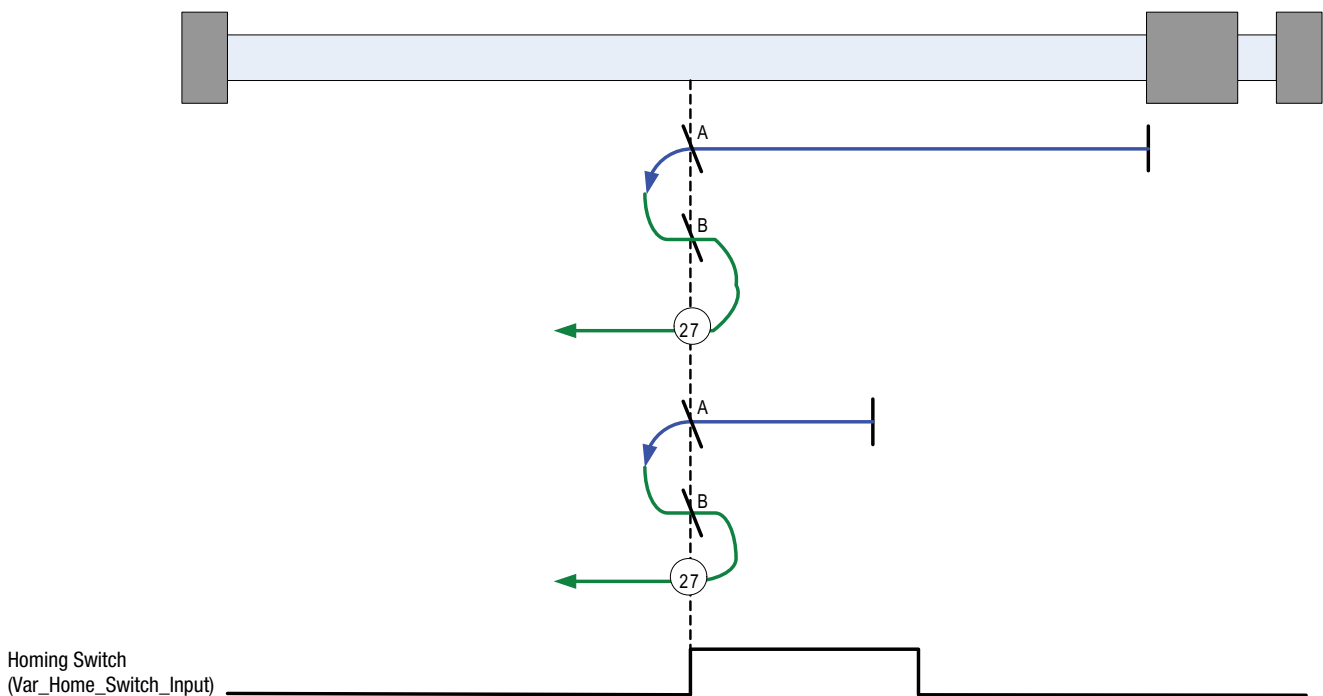


Figure 45: Homing Method 27

Programming

2.15.9.22 Homing Method 29: Homing without an Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is inactive). The home position is the leading edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until the homing switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the rising edge of the homing switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position 29). This is the home position (excluding offset).

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) following the procedure as detailed above but ignoring the initial move in the negative direction.

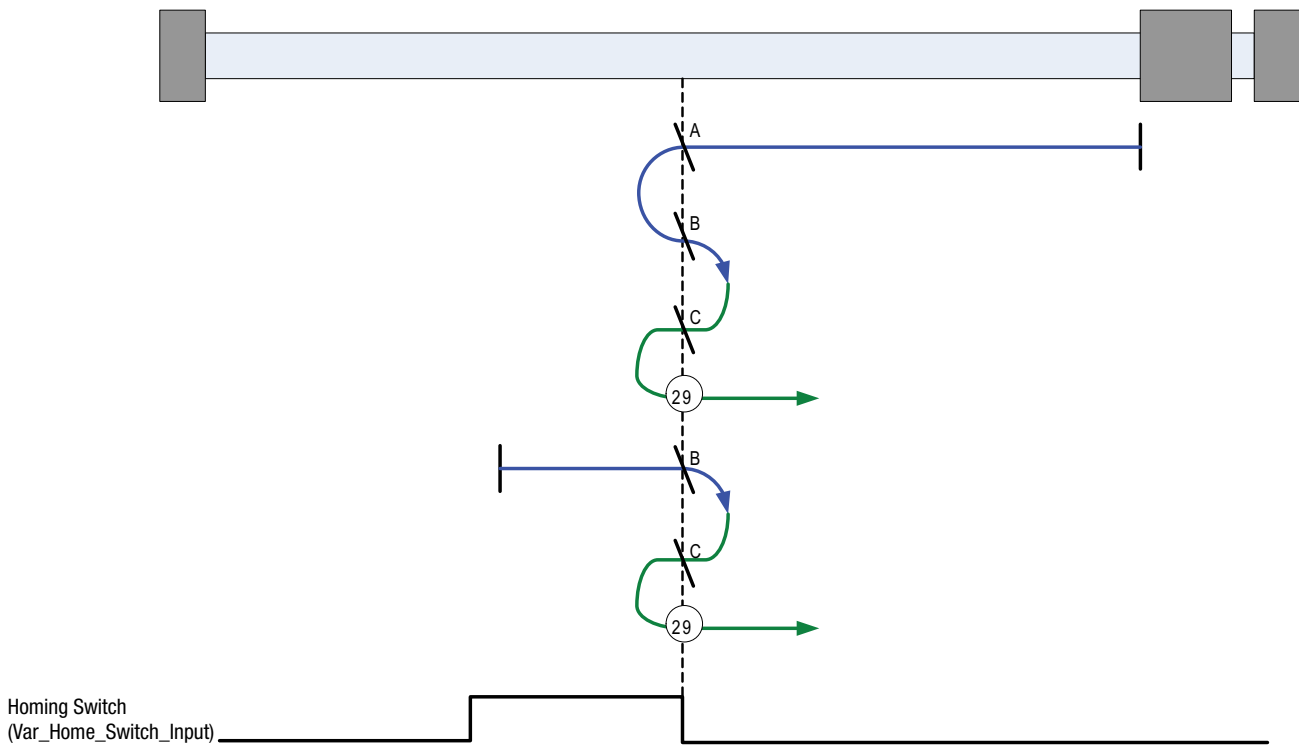


Figure 46: Homing Method 29

2.15.9.23 Homing Method 33: Homing to an Index Pulse

Using this method the initial direction of movement is negative. The home position is the first index pulse to the negative of the shaft starting Position. Axis will accelerate to **fast** homing velocity in the negative direction and continue until the rising edge of the first index pulse (position 33) is detected.

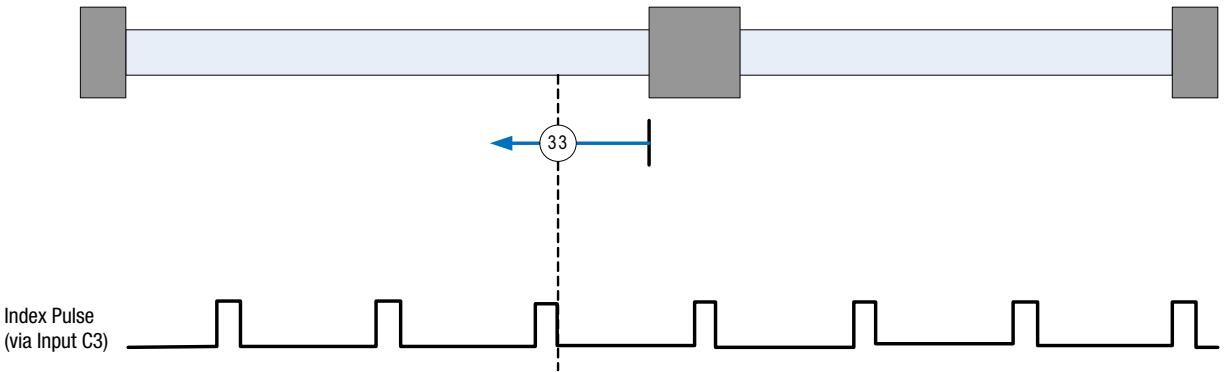


Figure 47: Homing Method 33

2.15.9.24 Homing Method 34: Homing to an Index Pulse

Using this method the initial direction of movement is positive. The home position is the first index pulse to the positive of the shaft starting Position. Axis will accelerate to **fast** homing velocity in the positive direction and continue until the rising edge of the first index pulse (position 34) is detected.

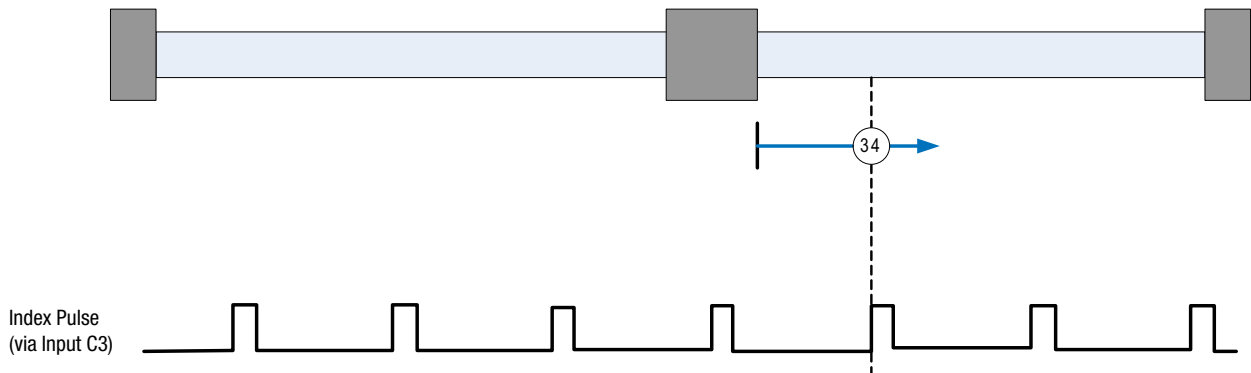


Figure 48: Homing Method 34

2.15.9.25 Homing Method 35: Using Current Position as Home

Using this method the current position of the axis is taken as the home position. There is no motion of the motor shaft during this procedure. Any offset specified (via the Var_Home_Offset Variable) will be added to the stored home position.

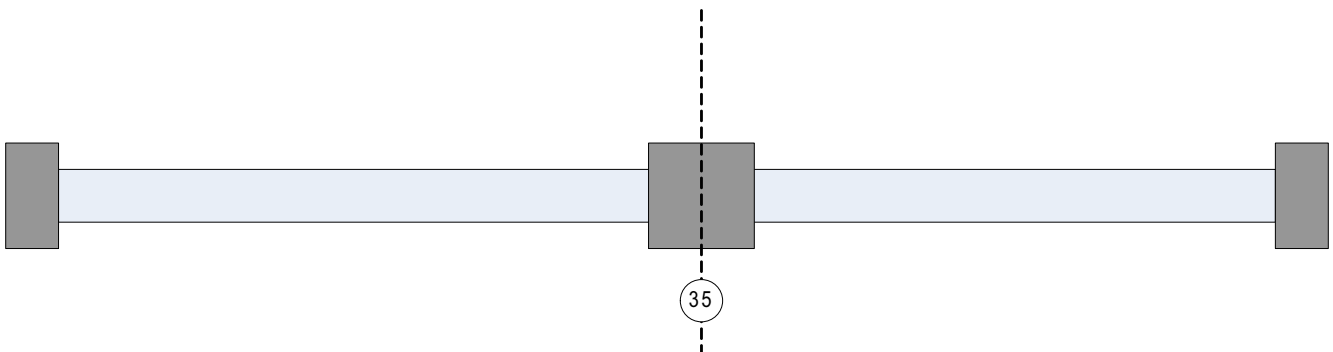


Figure 49: Homing Method 35

Programming

2.15.10 Homing Mode Operation example

The following steps are needed to execute the homing operation from the user program or under interface control.

1. Set Fast homing speed: Variable #242
2. Set Slow homing speed: Variable #243
3. Set Homing accel/decel: Variable #239
4. Set home offset:
 - a. In User Units Variable #240
 - b. In encoder pulses Variable #241
5. Set Home Switch Input Variable #246
6. Select Home Method Variable #244

'HOME' is the logical command to set VAR_START_HOMING. Writing the word 'HOME' within the user program will result in the homing operation commencing. After initiating the HOME command with firmware 3.60 (and later) the user program will not execute subsequent lines of code until after homing is completed (similar to MOVE P). If either using firmware prior to 3.60 or if user initiates homing in the indexer program via the statement VAR_START_HOMING=1, then it is recommended to immediately follow that statement with the following code:

```
WAIT UNTIL VAR_EXSTATUS & 0x400000 == 0x400000.
```

Doing this ensures no further lines of code will be executed until homing is complete.

```
;Program start-----  
;  
;  
  UNITS=1                               ;rps  
  
  Accel=1000  
  Decel=1000  
  MaxV =20  
  
;some program statements..  
;  
;  
;Homing specific set up..  
  VAR_HOME_FAST_VEL=                    10           ;rps  
  VAR_HOME_SLOW_VEL=                    1           ;rps  
  VAR_HOME_ACCEL=                        100         ;rps/sec^2  
  VAR_HOME_OFFSET=                      0           ;no offset from sensor  
  VAR_HOME_SWITCH_INPUT=                4           ;input B1 (0-A1, 1-A2...3-A4,4-B1,...11-C4)  
  VAR_HOME_METHOD=                      4           ;see table 22  
  ENABLE  
  HOME                                  ;starts homing sequence  
  WAIT UNTIL VAR_EXSTATUS & 0x400000 == 0x400000 ;wait for homing complete  
;Drive homed  
  
;Program statements..  
END
```

3. Reference

3.1 Program Statement Glossary

Each statement, system variable or operand is documented using the tabular format shown in Tables 23 and 24. The field label is still shown even if there is no information for a particular field. The individual program statements are listed in this section in alphabetical order with detailed descriptions in Tables 25 through 60.

Table 23: Language Format

KEYWORD	Long Name	Type
Purpose		
Syntax	KEYWORD <ARGUMENTS> ,[MODIFIERS]	
Remarks		
See Also		
Example		

Table 24: Field Descriptions

Field	Descriptions
KEYWORD:	The KEYWORD is the name of the programming statement as it would appear in a program.
Description:	The description is an interpretation of the keyword. For example: MOVEP is the keyword and Move to Position would be a description. The description is provided only as an aid to the reader and may not be used in a program.
Type:	The type field will identify the Keyword as either a Statement or a Pseudo statement. Statements are actual instructions converted to machine code by the compiler and form executable commands within the drive programming. Pseudo statements add convenience to the programmer but do not form instructions in their own right. They are therefore not executable code and are effectively removed when the program is compiled to it's native state by the compiler.
Purpose:	Purpose or Function of the Keyword (Programming Statement).
Syntax:	This field shows proper usage of the keyword. Arguments will be written in < > brackets. Optional arguments will be contained within [] brackets.
Arguments:	The data that is supplied with a statement that modifies the behavior of the statement. For example, MOVED=100. MOVED is the statement and 100 is the argument.
Remarks:	The remark field contains additional information about the use of the statement.
See Also:	This field contains a list of statements that are related to the purpose of the keyword.
Example:	The example field contains a code segment that illustrates the usage of the keyword
Reference	

Reference

Table 25: ASSIGN

ASSIGN	Assign Input As Index Bit	Statement
Purpose	Assign keyword causes a specified input to be assigned to a particular bit of system variable INDEX. Up to 8 digital inputs can be assigned to the first eight bits (bits 0 - 7) of the INDEX system variable in any order or combination. The purpose of the Assign Keyword and INDEX system Variable is to allow the creation of a custom input word for inclusion in the user program. Good examples of it's use are for implementing easy selection of preset torque, velocity or position values within the user program.	
Syntax	ASSIGN INPUT <input name> AS BIT <bit #> Input name (IN_A1..IN_A2 etc.) Bit# INDEX variable bit number from 0 to 7	
Remarks	Assign statements typically appear at the start of the program (Initialize and set Variables section) but can be included in other code sections with the exception of Events and the Fault Handler.	
See Also	VAR_IOINDEX Variable (#220)	

Example:

```
ASSIGN INPUT IN_B1 AS BIT 0 ;index bit 0 state matches state of input B1
ASSIGN INPUT IN_B2 AS BIT 1 ;index bit 1 state matches state of input B2
```

Program Start:

```
; <statements>
```

```
If Index == 0 ; If neither IN_B1 or IN_B2 is on
    MoveP 0 ; Move to Absolute Position 0
```

```
Endif
```

```
If Index == 1 ; If IN_B1 is on and IN_B2 is off
    MoveP 10 ; Move to Absolute Position 10
```

```
Endif
```

```
; If Index == 2 .....

```

Table 26: DEFINE

DEFINE	Define name	Pseudo-statement
Purpose	DEFINE is used to define symbolic names for User Variables, constants, and Digital I/O for programming convenience. Define statements greatly enhance program understanding by allowing the user to program using symbolic strings (names) relevant to their application. DEFINE can be used also to substitute a symbolic string.	
Syntax	DEFINE <name> <string> name any symbolic string string User Variable, constant, or Digital I/O Flag that symbolic string will represent	
Remarks:	DEFINE statements can be located anywhere within the user program (with the exception of events and the fault handler). Normally practice however is to place definitions at the start of the program prior to any executable code.	
See Also		
Example:	<pre> Define Start_Button IN_B1 ; Define a Digital Input Define System_Stop Out2 ; Define a Digital Output Define Loop_Counter V5 ; Define a User Variable Define Loop_Increment 1 ; Define a Constant Value Program_Start: ; Label Program Start If Start_Button == 0 ; If input B1 is off Disable ; Disable Servo System_Stop = 1 ; Turn on Output 2 Else ; Otherwise System_Stop = 0 ; Turn off Output 2 Enable ; Enable Servo MoveD 10 ; Move (increment) Distance 10 Loop_Counter = Loop_Counter + Loop_Increment ; Increment Variable V5 by 1 Endif Goto Program_Start ; Goto Label Program_Start </pre>	

Table 27: DISABLE

DISABLE	Disables the drive	Statement
Purpose	DISABLE turns OFF the power to the motor. Drive shows 'Dis' on display when in a disabled state.	
Syntax	DISABLE	
Remarks	Once the DISABLE statement is executed, the power to the motor is turned off and the motor can move freely. When disabled the drive will continue to monitor feedback and the actual position variable (APOS) will continue to update with the current position of the motor. The target position variable (TPOS) will be updated with the value of the actual position variable (APOS) on Enable to prevent unexpected motion from the motor shaft.	
See Also	ENABLE	
Example:	<pre> If Start_Button == 0 ; If input B1 is off Disable ; Disable Servo Else ; Otherwise Enable ; Enable Servo MoveD 10 ; Move (increment) Distance 10 Endif </pre>	

Reference

Table 28: DO UNTIL

DO UNTIL	Do/Until	Statement
Purpose	The DO / UNTIL statement is used to execute a statement or set of statements repeatedly until a logical condition becomes true. The Do / Until statements enclose the program code to be repeatedly executed with the UNTIL statement containing the logical statement for exit of the loop.	
Syntax	DO {statement(s)}... UNTIL <condition> {statement(s)} any valid statement(s) <condition> The condition to be tested.	
Remarks	The loop statement or statements contained within a DO / UNTIL loop will always be executed at least once because the logical condition to be tested is contained within the UNTIL statement in the last statement of the loop.	
See Also	WHILE, IF	
Example:	<pre> V0 = 0 ; Set V0 to Value 0 ; Create Loop to perform Move command 12 times DO ; Start of Do Loop V0 = V0 + 1 ; Add 1 to Variable V0 Moved 5 ; Move (incremental) distance 5 Until V0 == 12 ; Loop back to DO Statement, Repeat Until Logic True </pre>	

Table 29: ENABLE

ENABLE	Enables the drive	Statement
Purpose	Enable turns on power to the motor. Drive shows 'Run' on display when in the enabled state.	
Syntax	ENABLE	
Remarks	Once a drive is enabled motion can be commanded from the user program. Commanding motion while the drive is disabled will result in fault trip (F_27).	
See Also	DISABLE	
Example:	<pre> If Start_Button == 0 ; If input B1 is off Disable ; Disable Servo Else ; Otherwise Enable ; Enable Servo MoveD 10 ; Move (increment) Distance 10 Endif </pre>	

Table 30: END

END	END program	Statement
Purpose	This statement is used to terminate (finish) user program and its events.	
Syntax	END	
Remarks	END can be used anywhere in program	
See Also	DISABLE	
Example:	<pre> END ;end user program </pre>	

Table 31: EVENT

EVENT	Starts Event handler	Statement								
Purpose	EVENT keyword is used to create scanned events within the user program. Statement also sets one of 4 possible types of events.									
Syntax	<p>Any one of the 4 syntax examples herein may be used:</p> <ol style="list-style-type: none"> 1. EVENT <name> INPUT <inputname> RISE 2. EVENT <name> INPUT <inputname> FALL 3. EVENT <name> TIME <period> 4. EVENT <name> <expression> <table style="margin-left: 20px; border: none;"> <tr> <td style="padding-right: 10px;">name</td> <td>any valid alphanumeric string</td> </tr> <tr> <td>inputname</td> <td>any valid input "IN_A1 - IN_C4"</td> </tr> <tr> <td>period</td> <td>any integer number. Expressed in ms</td> </tr> <tr> <td>expression</td> <td>any arithmetic or logical expression</td> </tr> </table> <p>The following statements can not be used within event's handler:</p> <p>MOVE, MOVED, MOVEP, MOVEDR, MOVEPR, MDV MOTION SUSPEND MOTION RESUME STOP MOTION DO UNTIL GOTO GOSUB HALT VELOCITY ON/OFF WAIT WHILE</p>		name	any valid alphanumeric string	inputname	any valid input "IN_A1 - IN_C4"	period	any integer number. Expressed in ms	expression	any arithmetic or logical expression
name	any valid alphanumeric string									
inputname	any valid input "IN_A1 - IN_C4"									
period	any integer number. Expressed in ms									
expression	any arithmetic or logical expression									

While GOTO or GOSUB are restricted, a special JUMP statement can be used for program flow change from within event handler. See JUMP statement description in Language Reference section.

Remarks

For syntax 1 and 2:

The Event will occur when the input with the <name/number> transition from L(Low) to H (High), for syntax 1 (RISE) and from H (High) to L(Low) for syntax 2 (FALL).

For syntax 3:

The Event will occur when the specified , <period>, period of time has expired. This event can be used as periodic event to check for some conditions.

For syntax 4

The Event will occur when the expression, <expression>, evaluates to be true. The expression can be any valid arithmetic or logical expression or combination of the two. This event can be used when implementing soft limit switches or when changing the program flow based on some conditions. Any variable, (user and system), or constants can be used in the expression. The event will only trigger when the logic transitions from False to True. Further occurrence of the event will not occur while the condition remains true.

See Also ENDEVENT, EVENT ON, EVENT OFF

Example:

```

EVENT InEvent IN_A1 RISE
  V0 = V0+1           ;V0 increments by 1 each time IN_A1 transitions from low to high
ENDEVENT
EVENT period TIME 1000 ;1000 ms = 1Sec
  V3=V0-V1           ;Event subtracts V1 from V0 and stores result in V3 every second (1000mS)
ENDEVENT
;-----
EVENT InEvent ON
EVENT period ON
  {program statements}
END

```

Reference

Table 32: ENDEVENT

ENDEVENT	END of Event handler	Statement
Purpose	Indicates end of the scanned event code	
Syntax	ENDEVENT	
Remarks		
See Also	EVENT, EVENT ON, EVENT OFF	
Example:	<pre>EVENT InputRise IN_B4 RISE V0=V0+1 ENDEVENT</pre>	

Table 33: EVENT ON/OFF

EVENT ON/OFF	Turn events on or off	Statement
Purpose	Turns ON or OFF events created by an EVENT handler statement	
Syntax	EVENT <name> ON EVENT <name> OFF <name> Event handler name	
Remarks		
See Also	EVENT	
Example:	<pre>EVENT InputRise ON EVENT InputRise OFF</pre>	

Table 34: EVENTS ON/OFF

EVENTS OFF/ON	Globally Disables/enables events	Statement
Purpose	EVENTS OFF command when executed will disable any events currently enabled (running). EVENTS ON Command re-enables any events previously disabled through the events off command. EVENTS ON is not a global enable of all declared events. Events status is indicated through bit #30 of the DSTATUS register or by system flag 'F_EVENTSOFF'. EVENTS OFF/ON allows for easy disable and re-activation of events in sections of the main program or subroutines that the programmer doesn't want interrupted by event code.	
Syntax	EVENTS OFF Disables execution of all events EVENTS ON Restores execution of previously enabled events.	
Remarks	Events are globally disabled after a reset is made. Events are re-enable by executing the individual EVENT <name> ON statement.	
See Also	EVENT	

Example:

```

*****
EVENT SKIPOUT IN_B4 RISE      ;check for rising edge of input B4
  JUMP TOGGLE                 ;redirect code execution to TOGGLE
ENDEVENT                     ;end the event
EVENT OVERSHOOT IN_B3 RISE   ;check for rising edge of input B3
  JUMP SHUTDOWN              ;redirect code execution to SHUTDOWN
ENDEVENT                     ;end the event
*****
EVENT SKIPOUT ON
EVENT OVERSHOOT ON
*****
.....User code.....
EVENTS OFF                   ;turns off all events
.....User code.....
EVENTS ON                    ;turns on any event previously activated

```

Table 35: FAULT

FAULT	User generated fault	Statement
Purpose	Allows the user program to set a custom system fault. This is useful when the programmer needs to define a fault code and fault process for custom conditions like data supplied by interface out of range etc. Custom fault numbers must be in region of 128 to 240 (decimal)	
Syntax	FAULT FaultNumber	Sets system fault. Faultnumber - constant in range 128-240 Variables are not allowed in this statement.
Remarks	Custom fault will be processed as any regular fault. There will be a record in the fault log.	
See Also	ON FAULT	

Example:

```

FAULT 200           ;Sets fault #200
V0=200
FAULT V0           ;Not valid. Variables are not allowed here

```


Reference

Table 36: GOSUB

GOSUB	Go To subroutine	Statement
Purpose	GOSUB transfers control to subroutine.	
Syntax	GOSUB <subname>	
	<subname>	a valid subroutine name
Remarks	After return from subroutine program resumes from next statement after GOSUB	
See Also	GOTO, JUMP, RETURN	
Example:		
<pre> DO GOSUB CALCMOVE ;Go to CALCMOVE Subroutine MOVED V1 ;Move distance calculated in Subroutine UNTIL INA1 END SUB CALCMOVE: V1=(V2+V3)/2 ;Subroutine statement, Calculates value for V1 RETURN ;Return to main program execution </pre>		

Table 37: GOTO

GOTO	Go To	Statement
Purpose	Transfer program execution to label following the GOTO instruction.	
Syntax	GOTO <label>	
Remarks		
See Also	GOSUB, JUMP	
Example:		
<pre> GOTO Label2 {Statements...} Label2: {Statements...} </pre>		

Table 38: HALT

HALT	Halt the program execution	Statement
Purpose	Used to halt main program execution. Events are not halted by the HALT statement. Execution will be resumed by the RESET statement or by executing a JUMP to code from the EVENT handler.	
Syntax	HALT	
Remarks	This statement is convenient when writing event driven programs.	
See Also	RESET	
Example:		
<pre> {Statements...} HALT ;halt main program execution and wait for event </pre>		

Table 39: HOME

HOME	Execute homing routine	Statement
Purpose	Used to initiate homing.	
Syntax	HOME	
Remarks	This statement is convenient when writing event driven programs.	
See Also		
Example:	<pre>{Statements...} HOME ;initiate homing routine</pre>	

Table 40: ICONTROL ON/OFF

ICONTROL ON/OFF	Enables interface control	Statement
Purpose	Enables/Disables interface control. Effects bit #27 in DSTATUS register and system flag F_ICONTROLOFF. All interface motion commands and commands changing any outputs will be disabled. See Host interface commands manual for details. This command is useful when the program is processing critical states (example limit switches) and can't be disturbed by the interface.	
Syntax	ICONTROL ON ICONTROL OFF	Enables Interface control Disables interface control
Remarks	After reset interface control is enabled by default.	
See Also		
Example:	<pre>EVENT LimitSwitch IN_A1 RISE ;limit switch event Jump LimitSwitchHandler ;jump to process limit switch ENDEVENT V0=0 ;V0 will be used to indicate fault condition EVENT LimitSwitch ON ;Turn on event to detect limit switch activation Again: HALT ;system controlled by interface LimitSwitchHandler: EVENTS OFF ;turn off all events ICONTROL OFF ;disable interface control STOP MOTION QUICK DISABLE ;DISABLE V0=1 ;indicate fault condition to the interface ICONTROL ON ;Enable Interface Control EVENTS ON ;turn on events turned off by 'EVENTS OFF' GOTO AGAIN</pre>	

Reference

Table 41: IF

IF	IF/ENDIF	Statement
Purpose	The IF statement tests for a condition and then executes the specific action(s) between the IF and ENDIF statements if the condition is found to be true. If the condition is false, no action is taken and the instructions following the ENDIF statement are executed. Optionally, using the ELSE statement, a second series of statements may be specified to be executed if the condition is false.	
Syntax	<pre> IF <condition> {statements 1} ELSE {statements 2} ENDIF </pre>	
Remarks		
See Also	WHILE, DO	
Example:		
	<pre> IF APOS > 4 ;If actual position is greater than 4 units V0=2 ELSE ;otherwise... (actual position equal or less than 4) V0=0 ENDIF ;----- If V1 <> V2 && V3>V4 ;If V1 doesn't equal V2 AND V3 if greater than V4 V2=9 ENDIF </pre>	

Table 42: JUMP

JUMP	Jump to label from Event handler	Statement
Purpose	This is a special purpose statement to be used only in the Event Handler code. When the EVENT is triggered and this statement is processed, execution of the main program is transferred to the <label> argument called out in the "JUMP" statement. The Jump statement is useful when there is a need for the program's flow to change based on some event(s). Transfer program execution to the instruction following the label.	
Syntax	<pre> JUMP <label> <label> is any valid program label </pre>	
Remarks	Can be used in EVENT handler only.	
See Also	EVENT	
Example:		
	<pre> EVENT ExternalFault INPUT IN_A4 RISE ;activate Event when IN_A4 goes high JUMP ExecuteStop ;redirect program to <ExecuteStop> ENDEVENT StartMotion: EVENT ExternalFault ON ENABLE MOVED 20 MOVED -100 {statements} END ExecuteStop: STOP MOTION ;Motion stopped here DISABLE ;drive disabled Wait Until !IN_A4 ;Wait Until Input A4 goes low GOTO StartMotion </pre>	

Table 43: MDV

MDV	Segment Move	Statement
Purpose	MDV defines individual motion segment by specifying distance and final velocity (for each segment) in User Units. Acceleration (or deceleration) is calculated automatically based on these two parameters. This technique allows complicated moves to be created that consist of many segments. Each MDV sequence (series of MDV segments) starts and ends with a velocity of 0. Based on this an MDV sequence must have at least two segments. The MDV statement doesn't suspend execution of the main program. Each segment is loaded into the Motion Queue and the sequence executed immediately. If the last segment in the Motion Queue doesn't have a final velocity of 0, the drive will generate a "Motion Queue Empty" fault #24. If the "S" modifier is used in the statement, then the velocity acceleration/deceleration will be S-curved as opposed to be linear.	
Syntax	MDV <[-]segment distance>,<segment final velocity> [,S] S[-curve] optional modifier specifies S-curve acceleration / deceleration.	
See Also	MOVE, MOVEP, MOVEPR, MOVED, MOVEDR, MOTION SUSPEND, MOTION RESUME	
Example:		
{Statements...}		
MDV 5, 10 ;Move 5 user units and accelerate to a velocity of 10		
MDV 10,10 ;Move 10 user units and maintain a velocity of 10		
MDV 10,5 ;Move 10 user units and decelerate to velocity of 5		
MDV 5,;0 ;Move 5 user units and decelerate to velocity 0.		
;The last MDV must have a final velocity of 0.		
{Statements...}		

Table 44: MEMGET

MEMGET	Memory access statements MEMGET	Statement
Purpose	MEMGET provides command for simplified retrieval of data from the drives RAM memory file through transfer of data to the variables V0-V31. Using this statement any combinations of variables V0-V31 can be retrieved from the RAM file with a single statement.	
Syntax	MEMGET <offset> [<varlist>] <offset> It specifies offset in RAM file where data will be retrieved. Range: -32767 to 32767 <varlist> any combinations of variables V0-V31	
See Also	MEMSET	
Example:		
MEMGET 5 [V0]	;single variable will be retrieved from location 5	
MEMGET V1 [V0,V3,V2]	;variables V0,V3,V2 will be retrieved from ;memory location starting at value held in V1	
MEMGET 10 [V3-V7]	;variables V3 to V7 inclusively will be retrieved	
MEMGET V1 [V0,V2,V4-V8]	;variables V0,V2, V4 through V8 will be retrieved	

Reference

Table 45: MEMSET

MEMSET	Memory access statements MEMSET	Statement
Purpose	MEMSET provides command for simplified storage of data to the drives RAM memory file through transfer of data from variables V0-V31. Using this statement any combinations of variables V0-V31 can be stored in the RAM file with a single statement.	
Syntax	MEMSET	<offset> [<varlist>]
	<offset>	It specifies offset in RAM file where data will be stored. Range: -32767 to 32767
	<varlist>	any combinations of variables V0-V31
See Also	MEMGET	
Example:		
	MEMSET 5 [V0]	;single variable will be stored in location 5
	MEMSET V1 [V0,V3,V2]	;variables V0,V3,V2 will be stored in memory ;location starting at value held in V1
	MEMSET 10 [V3-V7]	;variables V3 to V7 inclusively will be stored
	MEMSET V1 [V0,V2,V4-V8]	;variables V0,V2, V4 through V8 will be stored.

Table 46: MOTION RESUME

MOTION RESUME	Resume Motion	Statement
Purpose	Statement resumes motion previously suspended by MOTION SUSPEND. If motion was not previously suspended, this has no effect on operation.	
Syntax	MOTION RESUME	
See Also	MOVE, MOVEP, MOVEDR, MOVED, MOVEPR ,MDV, MOTION SUSPEND	
Example:		
	...{statements}	
	MOTION RESUME	;Motion is resumed from first command in motion Queue (if any)
	...{statements}	

Table 47: MOTION SUSPEND

MOTION SUSPEND	Suspend	Statement
Purpose	This statement is used to temporarily suspend motion without flushing the Motion Queue's contents. If this statement is executed while a motion profile is being processed, then the motion will not be suspended until after the completion of the move. If executing a series of segment moves, motion will not be suspended until after all the MDV segments have been processed. If the Motion Queue is empty then any subsequent motion statement will be loaded into the queue and will remain there until the "Motion Resume" statement is executed. Any motion statements without the "C" modifier (except MDV statements) will lock-up the User Program.	
Syntax	MOTION SUSPEND	
Remarks	Performing any MOVEx commands without "C" modifier will lock-up the user program. You will be able to unlock it only by performing a Reset or Host Interface command "Motion Resume"	
See Also	MOVE, MOVEP, MOVEDR, MOVED, MOVEPR ,MDV, MOTION RESUME	
Example:		
	...{statements}	
	MOTION SUSPEND	;Motion will be suspended after current motion ;command is finished.
	...{statements}	

Table 48: MOVE

MOVE	Move	Statement
Purpose	MOVE UNTIL performs motion until condition becomes TRUE. MOVE WHILE performs motion while conditions stays TRUE. The statement suspends the programs execution until the motion is completed, unless the statement is used with C modifier.	
Syntax	MOVE [BACK] UNTIL <condition> [,C] MOVE [BACK] WHILE <condition> [,C] BACK Changes direction of the move. C (optional) C[ontinue] - modifier allows the program to continue while motion is being performed. If a second motion profile is executed while the first profile is still in motion, the second profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. The programmer should check the "F_MQUEUE_FULL" system flag to make sure that there is available space in the queue. If the queue becomes full, or overflows, then the drive will generate a fault. <condition> The condition to be tested. The condition may be a comparison, an input being TRUE or FALSE (H or L) system flag or a variable is used as flag (if 0 - false, else - true).	
Remarks		
See Also	MOVEP, MOVED, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME	
Example:	<pre> {Statements...} MOVE UNTIL V0<3 ;Move until V0 is less than 3 MOVE BACK UNTIL V0>4 ;Move back until V0 is greater than 4 MOVE WHILE V0<3 ;Move While V0 is less than 3 MOVE BACK WHILE V0>4 ;Move While V0 is greater than 4 MOVE WHILE V0<3,C ;Move While V0 < 3, continue program execution </pre>	

Table 49: MOVED

MOVED	Move Distance	Statement
Purpose	MOVED performs incremental motion (distance) specified in User Units. The commanded distance can range from -231 to 231. This statement will suspend the programs execution until the motion is completed, unless the statement is used with the "C" modifier. If the "S" modifier is used then S-curve accel is performed during the move.	
Syntax	MOVED <distance>[,S] [,C] C[ontinue] The "C" argument is an optional modifier which allows the program to continue executing while the motion profile is being executed. If the drive is in the process of executing a previous motion profile the new motion profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. The programmer should check the "F_MQUEUE_FULL" system flag to make sure that there is available space in the queue. If the queue becomes full, or overflows, then the drive will generate a fault. S[-curve] optional modifier specifies S-curve acceleration/deceleration.	
See Also	MOVE, MOVEP, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME	
Example:	<pre> {Statements...} MOVED 3 ;moves 3 user units forward MOVED BACK 3 ;moves 3 user units backward {Statements...} </pre>	

Reference

Table 50: MOVEDR

MOVEDR	Registered Distance Move	Statement
Purpose	MOVEDR performs incremental motion, specified in User Units. If during the move the registration input becomes activated (goes high) then the current position is recorded, and the displacement value (the second argument in the MOVEDR statement) is added to this position to form a new target position. The end of the move is then altered to this new target position. This statement suspends execution of the program until the move is completed, unless the statement is used with the “C” modifier.	
Syntax	MOVEDR <distance>,<displacement> [,S] [,C]	
	C[ontinue]	The “C” argument is an optional modifier which allows the program to continue executing the User Program while a motion profile is being processed. If a new motion profile is requested while the drive is processing a move the new motion profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. The programmer should check the “F_MQUEUE_FULL” system flag to make sure that there is available space in the queue. If the queue becomes full, or overflows, then the drive will generate a fault.
	S[-curve]	optional modifier specifies S-curve acceleration/deceleration.
See Also	MOVE, MOVEP, MOVEPR, MOVED, MDV, MOTION SUSPEND, MOTION RESUME	
Example:	This example moves the motor 3 user units and checks for the registration input. If registration isn’t detected then the move is completed.	
	{Statements...}	If registration is detected, the registration position is recorded and a displacement value of 2 is added to the recorded registration position to calculate the new end position.
	MOVEDR 3, 2	
	{Statements...}	

Table 51: MOVEP

MOVEP	Move to Position	Statement
Purpose	MOVEP performs motion to a specified absolute position in User Units. The command range for an Absolute move is from -2^{31} to 2^{31} User Units. This statement will suspend the program’s execution until the motion is completed unless the statement is used with the “C” modifier. If the “S” modifier is used then an S-curve accel is performed during the move.	
Syntax	MOVEP <absolute position>[,S] [,C]	
	C[ontinue]	The “C” argument is an optional modifier which allows the program to continue executing while the motion profile is being executed. If the drive is in the process of executing a previous motion profile the new motion profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. The programmer should check the “F_MQUEUE_FULL” system flag to make sure that there is available space in the queue. If the queue becomes full, or overflows, then the drive will generate a fault.
	S[-curve]	optional modifier specifies S-curve acceleration/deceleration.
See Also	MOVE, MOVED, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME	
Example:		
	{Statements...}	
	MOVEP 3	;moves to 3 user units absolute position
	{Statements...}	

Table 52: MOVEPR

MOVEPR	Registered Distance Move	Statement
Purpose	MOVEPR performs absolute position moves specified in User Units. If during a move the registration input becomes activated, i.e., goes high, then the end position of the move is altered to a new target position. The new position is generated from the second argument in the MOVEPR statement, (displacement). This statement suspends the execution of the program until the move is completed, unless the statement is used with the C modifier.	
Syntax	MOVEPR <distance>,<displacement> [,S] [,C]	
	C[ontinue]	The "C" argument is an optional modifier which allows the program to continue executing the User Program while a motion profile is being processed. If a new motion profile is requested while the drive is processing a move the new motion profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. The programmer should check the "F_MQUEUE_FULL" system flag to make sure that there is available space in the queue. If the queue becomes full, or overflows, then the drive will generate a fault.
	S[-curve]	optional modifier specifies S-curve acceleration/deceleration.
See Also	MOVE, MOVEP, MOVEDR, MOVED, MDV, MOTION SUSPEND, MOTION RESUME	
Example:	This example moves the motor to the absolute position of 3 user units while checking for the registration input.	
{Statements...}	If registration isn't detected, then the move is completed .	
MOVEPR 3, 2	If registration is detected, the registration position is recorded and a displacement value of 2 is added to the recorded registration position to calculate the new end position.	
{Statements...}		

Reference

Table 53: ON FAULT/ENDFAULT

ON FAULT/ ENDFAULT	Defines Fault Handler	Statement
Purpose	<p>This statement initiates the Fault Handler section of the User Program. The Fault Handler is a piece of code which is executed when a fault occurs in the drive. The Fault Handler program must begin with the "ON FAULT" statement and end with the "ENDFAULT" statement. If a Fault Handler routine is not defined, then the User Program will be terminated and the drive disabled upon the drive detecting a fault. Subsequently, if a Fault Handler is defined and a fault is detected, the drive will be disabled, all scanned events will be disabled, and the Fault Handler routine will be executed. The RESUME statement can be used to redirect the program execution from the Fault Handler back to the main program. If this statement is not utilized then the program will terminate once the ENDFault statement is executed.</p> <p>The following statements can't be used in fault handler: MOVE, MOVED, MOVEP, MOVEDR, MOVEPR, MDV, MOTION SUSPEND, MOTION RESUME, GOTO, GOSUB, JUMP, ENABLE, WAIT and VELOCITY ON/OFF</p>	
Syntax	<pre>ON FAULT {...statements} ENDFAULT</pre>	
See Also	RESUME	
Example:		
<pre>...{statements} ;User program FaultRecovery: ;Recovery procedure ...{statements} END ON FAULT ;Once fault occurs program is directed here ...{statements} ;Any code to deal with fault RESUME FaultRecovery ;Execution of RESUME ends Fault Handler and directs ;execution back to User Program. ENDFAULT ;If RESUME is omitted the program will terminate here ;Fault routine must end with a ENDFault statement</pre>		

Table 54: REGISTRATION ON

REGISTRATION ON	Registration On	Statement
Purpose	<p>This statement arms the registration input, (input IN_C3). When the registration input is activated, the Flag Variable "F_REGISTRATION" is set and the current position is captured and stored to the "RPOS" System Variable. Both of these variables are available to the User Program for decision making purposes. The "REGISTRATION ON" statement, when executed will reset the "F_REGISTRATION" flag ready for detection of the next registration input.</p>	
Syntax	REGISTRATION ON	Flag "F_REGISTRATION" is reset and registration input is armed
See Also	MOVEDR, MOVEPR	
Example:		
<pre>; Moves until input is activated and then come back to the sensor position. ...{statements} REGISTRATION ON ;Arm registration input MOVE UNTIL F_REGISTRATION ;Move until input is activated, (sensor hit) MOVEP RPOS ;Absolute move to the position of the sensor ...{statements}</pre>		

Table 55: RESUME

RESUME	Resume	Statement
Purpose	This statement redirects the code execution from the Fault Handler routine back to in the User Program. The specific line in the User Program to be directed to is called out in the argument <label> in the "RESUME" statement. This statement is only allowed in the fault handler routine.	
Syntax	RESUME <label> <label> Label address in User Program to recommence program execution	
See Also	ON FAULT	
Example:		
<pre> ...{statements} FaultRecovery: ...{statements} END ON FAULT ;Once fault occurs program is directed here ...{statements} ;Any code to deal with fault RESUME FaultRecovery ;Execution of RESUME ends Fault Handler and directs ;execution back the "FaultRecovery" label in the User ;Program. ENDFAULT ;If RESUME is omitted the program will terminate here. ;Fault routine must end with a ENDFault statement </pre>		

Table 56: RETURN

RETURN	Return from subroutine	Statement
Purpose	This statement will return the code execution back from a subroutine to the point in the program from where the subroutine was called. If this statement is executed without a previous call to subroutine, (GOSUB), fault #21 "Subroutine stack underflow" will result.	
Syntax	RETURN	
See Also	GOTO, GOSUB	
Example:		
<pre> ...{statements}... GOSUB MySub ;Program jumps to Subroutine "MySub" MOVED 10 ;Move to be executed once the Subroutine has executed ;the RETURN statement. ...{statements} END ;main program end MySub: ;Subroutine called out from User Program ...{statements} ;Code to be executed in subroutine RETURN ;Returns execution to the line of code under the "GOSUB" ;command, (MOVED 10 statement). </pre>		

Reference

Table 57: SEND / SEND TO

SEND/SEND TO	Send network variable(s)	Statement
Purpose	This statement is used to share the value of Network Variables between drives on an Ethernet network. Network Variables are variables NV0 through NV31. The variables to be sent out or synchronized with, are called out in the "SEND" statement. For example, "SEND [NV5]" will take the current value of variable NV5 and load it into the NV5 variable of every drive on the network. The SENDTO statement only updates network variables of the drives with the same group ID listed in the command.	
Syntax	SEND [NVa,NVb, NVx-NVy], SENDTO GroupID [NVa,NVb, NVx-NVy]	a,b,x,y Any number from 0 to 31 GroupID GroupID of the drives whose variables will be affected (synchronized)
See Also		
Example:	<pre> ...{statements}... NV1=12 ;Set NV1 equal to 12 SEND [NV1] ;Set the NV1 variable to 12 in every drive in the Network. SEND [NV5-NV10] ;Sets the NV5 through NV10 variable in all drives on the Network. NV20=25 ;Set NV20 equal to 25 SENDTO 2 [NV20] ;Set the NV20 variable to 25 only in drives with GroupID=2 ...{statements} END ;End main program </pre>	

Table 58: STOP MOTION

STOP MOTION [Quick]	Stop Motion	Statement
Purpose	This statement is used to stop all motion. When the "STOP MOTION" statement is executed all motion profiles stored in the Motion Queue are cleared, and motion will immediately be stopped via the deceleration parameter set in the "DECEL" variable. If the "QUICK" modifier is used, then the deceleration value will come from the "QDECEL" variable. The main use for this command is to control an emergency stop or when the End Of Travel sensor is detected. Note that the current position will not be lost after this statement is executed.	
Syntax	STOP MOTION STOP MOTION QUICK	Stops using DECEL deceleration rate Stops using QDECEL deceleration rate
See Also	MOTION SUSPEND	
Example:	<pre> ...{statements}... DECEL = 100 QDECEL = 10000 ...{statements} STOP MOTION QUICK </pre>	

Table 59: VELOCITY ON/OFF

VELOCITY ON/OFF	Velocity Mode	Statement
Purpose	The VELOCITY ON statement enables velocity mode in the drive. The VELOCITY OFF statement disables velocity mode and returns drive to its default mode. (Default mode is Positioning). The velocity value for this mode is set by writing to the System Variable "VEL". All position related variables are valid in this mode.	
Syntax	VELOCITY ON VELOCITY OFF	
Remarks	The "VELOCITY ON" statement is considered one of the motion related commands. It has to be implemented when the drive is enabled. If the "VELOCITY ON" statement is executed while the drive is disabled, fault # 27-"Drive disabled" will occur. Execution of any motion related profiles while the drive is in Velocity mode will be loaded into the Motion Queue. When the "VELOCITY OFF" statement is executed the drive defaults back to Position mode and immediately begins to execute the motion profiles stored in the Motion Queue. Please note that the "VEL" variable can be set on the fly, allowing dynamic control of the velocity.	
See Also		
Example:		
	VEL=0	;Set velocity to 0
	VELOCITY ON	;Turn on Velocity Mode
	VEL = 10	;Set velocity to 10
	...{statements}	
	VELOCITY OFF	;Turn off Velocity Mode

Table 60: WAIT

WAIT	Wait	Statement
Purpose	This statement suspend the execution of the program until some condition(s) is(are) met. Conditions include Expressions TRUE or FALSE, Preset TIME expiration, MOTION COMPLETE.	
Syntax	WAIT UNTIL <expression>	wait until expression becomes TRUE
	WAIT WHILE <expression>	wait while expression is TRUE
	WAIT TIME <time delay>	wait until <time delay> in mS is expired
	WAIT MOTION COMPLETE	wait until last motion in Motion Queue completes
Remarks		
See Also		
Example:		
	WAIT UNTIL (APOS>2 && APOS<3)	;Wait until Apos is > 2 and APOS < 3
	WAIT WHILE (APOS <2 && APOS>1)	;Wait while Apos is <2 and >1
	WAIT TIME 1000	;Wait 1 Sec (1 Sec=1000mS)
	WAIT MOTION COMPLETE	;Wait until motion is done

Reference

Table 61: WHILE / ENDWHILE

WHILE/ ENDWHILE	While	Statement
Purpose	The WHILE <expression> executes statement(s) between keywords WHILE and ENDWHILE repeatedly while the expression evaluates to TRUE.	
Syntax	WHILE <expression> {statement (s)}... ENDWHILE	
Remarks	WHILE block of statements has to end with ENDWHILE keyword.	
See Also	DO/UNTIL	
Example:	<pre>WHILE APOS<3 ;Execute the statements while Apos is <3 {statement (s)}.. ENDWHILE</pre>	

3.2 Variable List

Table 62 provides a complete list of the accessible PositionServo variables. These variables can be accessed from the user's program or any supported communications interface protocol. From the user program, any variable can be accessed by either its variable name or by its index value (using the syntax: @<VARINDEX> , where <VARINDEX> is the variable index from Table 62). From the communications interface any variable can be accessed by its index value.

The column "**Type**" indicates the type of variable:

mtr	Motor: denotes a motor value
mtn	Motion: writing to an "mtn" variable could cause the start of motion ⚠
vel	Velocity: denotes a velocity or velocity scaling value

The column "**Format**" provides the native format of the variable:

W	32 bit integer
F	float (real)

When setting a variable via an external device the value can be addressed as floating or integer. The value will automatically adjusted to fit it's given form.

The column "**EPM**" shows if a variable has a non-volatile storage space in the EPM memory:

Y	Variable has non-volatile storage Space in EPM
N	Variable does not exist in EPM memory

The user's program uses a RAM (volatile) 'copy' of the variables stored on the EPM. At power up all RAM copies of the variables are initialized with the EPM values. The EPM's values are not affected by changing the variables in the user's program. When the user's program reads a variable it always reads from the RAM (volatile) copy of the variable. Communications Interface functions can change both the volatile and non-volatile copy of the variable. If the host interface requests a change to the EPM (non-volatile) value, this change is done both in the user program's RAM memory as well as in the EPM. Interface functions have the choice of reading from the RAM (volatile) or from the EPM (non-volatile) copy of the variable.

The column "**Access**" lists the user's access rights to a variable:

R	read only
W	write only
R/W	read/write

Writing to an R (read-only) variable or reading from a W (write-only) variable will not work.

The column "**Units**" shows units of the variable. Units unique to this manual that are used for motion are:

UU	user units
EC	encoder counts
S	seconds
PPS	pulses per sample. Sample time is 255µs - servo loop rate
PPSS	pulses per sample per sample. Sample time is 255µs - servo loop rate

Reference

Table 62: PositionServo Variable List

Index	Name	Type	Format	EPM	Access	Description	Units
1	VAR_IDSTRING			N	R	Drive's identification string	
2	VAR_NAME			Y	R/W	Drive's symbolic name	
3	VAR_SERIAL_NUMBER				R	Drive's serial number	
4	VAR_MEM_INDEX				R/W	Position in RAM file (0 - 32767)	
5	VAR_MEM_VALUE				R/W	Value to be read or written to the RAM file	
6	VAR_MEM_INDEX_INCREMENT				R/W	Holds value the MEM_INDEX will modify once the R/W operation is complete	
7	RESERVED						
8	VAR_RSVD_2						
9	VAR_DFAULT Short Name: DFAULTS				R	Drive Default Settings	
10	VAR_M_ID	mtr		Y	R/W*	Motor ID	
11	VAR_M_MODEL	mtr		Y	R/W*	Motor model	
12	VAR_M_VENDOR	mtr		Y	R/W*	Motor vendor	
13	VAR_M_ESET	mtr		Y	R/W*	Motor Feedback Resolver: 'Positive for CW' 1 - Positive for CW 0 - none	
14	VAR_M_HALLCODE	mtr		Y	R/W*	Hallcode index Range: 0 - 5	
15	VAR_M_HOFFSET	mtr		Y	R/W*	Reserved	
16	VAR_M_ZOFFSET	mtr		Y	R/W*	Resolver Offset Range: 0 - 360	
17	VAR_M_ICTRL	mtr		Y	R/W*	Reserved	
18	VAR_M_JM	mtr		Y	R/W*	Motor moment of inertia, Jm Range: 0 - 0.1	Kgm2
19	VAR_M_KE	mtr		Y	R/W*	Motor voltage or back EMF constant, Ke Range: 1 - 500	V/Krpm
20	VAR_M_KT	mtr		Y	R/W*	Motor torque or force constant, Kt Range: 0.01 - 10	Nm/A
21	VAR_M_LS	mtr		Y	R/W*	Motor phase-to-phase inductance, Lm Range: 0.1 - 500	mH
22	VAR_M_RS	mtr		Y	R/W*	Motor phase-to-phase resistance, Rm Range: 0.01 - 500	[Ohm]
23	VAR_M_MAXCURRENT	mtr		Y	R/W*	Motor's max current(RMS) Range: 0.5 - 50	[A]mp
24	VAR_M_MAXVELOCITY	mtr		Y	R/W*	Motor's max velocity Range: 500 - 20,000	RPM
25	VAR_M_NPOLES	mtr		Y	R/W*	Motor's poles number Rnage: 2 - 200	
26	VAR_M_ENCODER	mtr		Y	R/W*	Encoder resolution Range: 256 - 65536 * 12/Npoles	PPR
27	VAR_M_TERMVOLTAGE	mtr		Y	R/W*	Nominal Motor's terminal voltage Range: 50 - 800	[V]olt
28	VAR_M_FEEDBACK	mtr		Y	R/W*	Feedback type 1 - Encoder 2 - Resolver	

* These are all R/W variables but they only become active after variable 247 is set

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
29	VAR_ENABLE_SWITCH_TYPE		W	Y	R/W	Enable switch function 0 - inhibit only 1 - Run	Bit
30	VAR_CURRENTLIMIT		F	Y	R/W	Current limit	[A]mp
31	VAR_PEAKCURRENTLIMIT16		F	Y	R/W	Peak current limit for 16kHz operation	[A]mp
32	VAR_PEAKCURRENTLIMIT		F	Y	R/W	Peak current limit for 8kHz operation	[A]mp
33	VAR_PWMFREQUENCY		W	Y	R/W	PWM frequency selection 0 - 16kHz 1 - 8kHz	
34	VAR_DRIVEMODE		W	Y	R/W	Drive mode: 0 - torque 1 - velocity 2 - position  WARNING! You can change operating modes when required during program execution but do not change modes on the fly (with drive enabled), as this may cause unexpected behavior of the motor.	
35	VAR_CURRENT_SCALE		F	Y	R/W	Analog input #1 current reference scale Range: Model dependent	A/V
36	VAR_VELOCITY_SCALE	vel	F	Y	R/W	Analog input #1 velocity reference scale Range: -10,000 to +10,000	RPM/V
37	VAR_REFERENCE		W	Y	R/W	Reference selection: 1 - internal source 0 - external	
38	VAR_STEPINPUTTYPE		W	Y	R/W	Selects how position reference inputs operating: 0 - Quadrature inputs (A/B) 1 - Step & Direction	
39	VAR_MOTORTHERMALPROTECT		W	Y	R/W	Motor thermal protection function: 0 - disabled 1 - enabled	
40	VAR_MOTORPTCRESISTANCE		F	Y	R/W	Motor thermal protection PTC cut-off resistance in Ohms	[Ohm]
41	VAR_SECONDENCODER		W	Y	R/W	Second encoder: 0 - Disabled 1 - Enabled	
42	VAR_REGENDUTY		W	Y	R/W	Regen circuit PWM duty cycle in % Range: 1-100%	%
43	VAR_ENCODERREPEATSRC		W	Y	R/W	Selects source for repeat buffers: 0 - Model 940 - Encoder Port P4 0 - Model 941 - 2nd Encoder Option Bay 1 - Model 940 - 2nd Encoder Option Bay 1 - Model 941 - Resolver Port P4	
44	VAR_VP_GAIN Short Name: VGAIN P	vel	W	Y	R/W	Velocity loop Proportional gain Range: 0 - 32767	
45	VAR_VI_GAIN Short Name: VGAIN I	vel	W	Y	R/W	Velocity loop Integral gain Range: 0 - 32767	
46	VAR_PP_GAIN Short Name: PGAIN P		W	Y	R/W	Position loop Proportional gain Range: 0 - 32767	
47	VAR_PI_GAIN Short Name: PGAIN I		W	Y	R/W	Position loop Integral gain Range: 0 - 16383	
48	VAR_PD_GAIN Short Name: PGAIN D		W	Y	R/W	Position loop Differential gain Range: 0 - 32767	






Reference

Index	Name	Type	Format	EPM	Access	Description	Units
49	VAR_PI_LIMIT Short Name: PGAIN_ILIM		W	Y	R/W	Position loop integral gain limit Range: 0 - 20000	
50	VAR_SEI_GAIN					Not Used	
51	VAR_VREG_WINDOW	vel	W	Y	R/W	Gains scaling coefficient Range: -16 to +4	
52	VAR_ENABLE		W	N	W	Software Enable/Disable 0 - disable 1 - enable	
53	VAR_RESET		W	N	W	Drive's reset (Disables drive, Stops running program if any, reset active fault) 0 - no action 1 - reset drive	
54	VAR_STATUS Short Name: DSTATUS		W	N	R	Drive's status register	
55	VAR_BCF_SIZE		W	Y	R	User's program Byte-code size	Bytes
56	VAR_AUTOBOOT		W	Y	R/W	User's program autostart flag. 0 - program has to be started manually (MotionView or interface) 1 - program started automatically after drive booted	
57	VAR_GROUPID		W	Y	R/W	Network group ID Range: 1 - 32767	
58	VAR_VLIMIT_ZEROSPEED		F	Y	R/W	Zero Speed window Range: 0 - 100	Rpm
59	VAR_VLIMIT_SPEEDWND		F	Y	R/W	At Speed window Range: 10 - 10000	Rpm
60	VAR_VLIMIT_ATSPEED		F	Y	R/W	Target Velocity for At Speed window Range: -10000 - +10000	Rpm
61	VAR_PLIMIT_POSEERROR		W	Y	R/W	Position error Range: 1 - 32767	EC
62	VAR_PLIMIT_ERRORTIME		F	Y	R/W	Position error time (time which position error has to remain to set-off position error fault) Range: 0.25 - 8000	mS
63	VAR_PLIMIT_SEPOSEERROR		W	Y	R/W	Second encoder Position error Range: 1 - 32767	EC
64	VAR_PLIMIT_SEERRORTIME		F	Y	R/W	Second encoder Position error time (time which position error has to remain to set-off position error fault) Range: 0.25 - 8000	mS
65	VAR_INPUTS Short Name: INPUTS		W	N	R	Digital inputs states. A1 occupies Bit 0, A2- Bit 1 ... C4 - bit 11.	
66	VAR_OUTPUT Short Name: OUTPUTS		W	N	R/W	Digital outputs states. Writing to this variables sets/resets digital outputs, except outputs which has been assigned special function. Output 1 Bit0 Output 2 Bit 1 Output 3 Bit 2 Output 4 Bit 3	
67	VAR_IP_ADDRESS		W	Y	R/W	Ethernet IP address. IP address changes at next boot up. 32 bit value	
68	VAR_IP_MASK		W	Y	R/W	Ethernet IP NetMask. Mask changes at next boot up. 32 bit value	
69	VAR_IP_GATEWAY		W	Y	R/W	Ethernet Gateway IP address. Address changes at next boot up. 32 bit value	



Reference

Index	Name	Type	Format	EPM	Access	Description	Units
70	VAR_IP_DHCP		W	Y	R/W	Use DHCP 0-manual 1- use DHCP service	
71	VAR_AIN1 Short Name: AIN1		F	N	R	Analog Input AIN1 current value	[V]olt
72	VAR_AIN2 Short Name: AIN2		F	N	R	Analog Input AIN2 current value	[V]olt
73	VAR_BUSVOLTAGE		F	N	R	Bus voltage	[V]olt
74	VAR_HTEMP		F	N	R	Heatsink temperature Returns: 0 - for temperatures < 40C and actual heat sink temperature for temperatures >40 C	[c]
75	VAR_ENABLE_ACCELDECEL	vel		Y	R/W	Enable Accel/Decel function for velocity mode 0 - disable 1 - enable	
76	VAR_ACCEL_LIMIT System variable for ramp parameters in MotionView	vel	F	Y	R/W	Accel value for velocity mode Range: 0.1 - 5000000	Rpm*Sec
77	VAR_DECEL_LIMIT System variable for ramp parameters in MotionView	vel	F	Y	R/W	Decel value for velocity mode Range: 0.1 - 5000000	Rpm*Sec
78	VAR_FAULT_RESET		W	Y	R/W	Reset fault configuration: 1 - on deactivation of Enable/Inhibit input (A3) 0 - on activation of Enable/Inhibit input (A3)	
79	VAR_M2SRATIO_MASTER		W	Y	R/W	Master to system ratio. Master counts range: -32767 - +32767 Value will be applied upon write to PID #80. Write to this PID followed by writing to PID#80 to apply new ratio pair	
80	VAR_M2SRATIO_SYSTEM		W	Y	R/W	Master to system ratio. System counts range: 1 - 32767 Writing to this PID also applies value currently held in PID #79. If you need to change both values, set #79 first then write to this PID the desired value to apply new ratio.	
81	VAR_S2PRATIO_SECOND		W	Y	R/W	Secondary encoder to prime encoder ratio. Second counts range: -32767 - +32767 Value will be applied upon write to PID #82. Write to this PID followed by writing to PID#82 to apply new ratio pair	
82	VAR_S2PRATIO_PRIME		W	Y	R/W	Secondary encoder to prime encoder ratio. Prime counts range: 1 - 32767 Writing to this PID also applies value currently held in PID #81. If you need to change both values, set #81 first then write to this PID the desired value to apply new ratio.	
83	VAR_EXSTATUS Short Name: DEXSTATUS		W	N	R	Extended status. Lower word copy of DSP status flags.	
84	VAR_HLS_MODE		W	Y	R/W	Hardware limit switches. 0 - not used 1 - stop and fault 2 - fault	



Reference

Index	Name	Type	Format	EPM	Access	Description	Units
85	VAR_AOUT_FUNCTION		W	Y	R/W	Analog output function range: 0 - 8 0 - Not assigned 1 - Phase Current (RMS) 2 - Phase Current (Peak Value) 3 - Motor Velocity 4 - Phase Current R 5 - Phase Current S 6 - Phase Current T 7 - Iq current 8 - Id current	
86	VAR_AOUT_VELSCALE		F	Y	R/W	Analog output scale for velocity quantities. Range: 0 - 10	mV/Rpm
87	VAR_AOUT_CURSCALE		F	Y	R/W	Analog output scale for current related quantities. Range: 0 - 10	V/A
88	VAR_AOUT Short Name: AOUT		F	N	W	Analog output value.(Used if VAR #85 is set to 0 - no function) Range: 0 - 10	V
89	VAR_AIN1_DEADBAND		F	Y	R/W	Analog input #1 dead-band. Applied when used as current or velocity reference. Range: 0 - 100	mV
90	VAR_AIN1_OFFSET			Y	R/W	Analog input #1 offset. Applied when used as current/velocity reference Range: -10,000 to +10,000	mV
91	VAR_SUSPEND_MOTION		W	N	R/W	Suspend motion. Suspends motion produced by trajectory generator. Current move will be completed before motion is suspended. 0 - motion suspended 1 - motion resumed	
92	VAR_MOVEP	 mtn	W	N	W	Target position for absolute move. Writing value executes Move to position as per MOVEP statement using current values of acceleration, deceleration and max velocity.	UU
93	VAR_MOVED	 mtn	W	N	W	Incremental position. Writing value <0> executes Incremental move as per MOVED statement using current values of acceleration, deceleration and max velocity.	UU
94	VAR_MDV_DISTANCE		F	N	W	Distance for MDV move	UU
95	VAR_MDV_VELOCITY	 mtn	F	N	W	Velocity for MDV move. Writing to this variable executes MDV move with Distance value last written to variable #94	UU
96	VAR_MOVE_PWI1	 mtn	W	N	W	Writing value executes Move in positive direction while input true (active). Value specifies input # 0 - 3 : A1 - A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	
97	VAR_MOVE_PWIO	 mtn	W	N	W	Writing value executes Move in positive direction while input false (not active). Value specifies input # 0 - 3 : A1 - A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
98	VAR_MOVE_NWI1	 mtn	F	N	W	Writing value executes Move negative direction while input true (active). Value specifies input # 0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	
99	VAR_MOVE_NWIO	 mtn	F	N	W	Writing value executes Move negative direction while input false (not active). Value specifies input # 0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	
100	VAR_V0 Short Name: V0		F	Y	R/W	User variable General purpose user defined variable	
101	VAR_V1 Short Name: V1		F	Y	R/W	User variable General purpose user defined variable	
102	VAR_V2 Short Name: V2		F	Y	R/W	User variable General purpose user defined variable	
103	VAR_V3 Short Name: V3		F	Y	R/W	User variable General purpose user defined variable	
104	VAR_V4 Short Name: V4		F	Y	R/W	User variable General purpose user defined variable	
105	VAR_V5 Short Name: V5		F	Y	R/W	User variable General purpose user defined variable	
106	VAR_V6 Short Name: V6		F	Y	R/W	User variable General purpose user defined variable	
107	VAR_V7 Short Name: V7		F	Y	R/W	User variable General purpose user defined variable	
108	VAR_V8 Short Name: V8		F	Y	R/W	User variable General purpose user defined variable	
109	VAR_V9 Short Name: V9		F	Y	R/W	User variable General purpose user defined variable	
110	VAR_V10 Short Name: V10		F	Y	R/W	User variable General purpose user defined variable	
111	VAR_V11 Short Name: V11		F	Y	R/W	User variable General purpose user defined variable	
112	VAR_V12 Short Name: V12		F	Y	R/W	User variable General purpose user defined variable	
113	VAR_V13 Short Name: V13		F	Y	R/W	User variable General purpose user defined variable	
114	VAR_V14 Short Name: V14		F	Y	R/W	User variable General purpose user defined variable	
115	VAR_V15 Short Name: V15		F	Y	R/W	User variable General purpose user defined variable	
116	VAR_V16 Short Name: V16		F	Y	R/W	User variable General purpose user defined variable	
117	VAR_V17 Short Name: V17		F	Y	R/W	User variable General purpose user defined variable	
118	VAR_V18 Short Name: V18		F	Y	R/W	User variable General purpose user defined variable	
119	VAR_V19 Short Name: V19		F	Y	R/W	User variable General purpose user defined variable	


Reference

Index	Name	Type	Format	EPM	Access	Description	Units
120	VAR_V20 Short Name: V20		F	Y	R/W	User variable General purpose user defined variable	
121	VAR_V21 Short Name: V21		F	Y	R/W	User variable General purpose user defined variable	
122	VAR_V22 Short Name: V22		F	Y	R/W	User variable General purpose user defined variable	
123	VAR_V23 Short Name: V23		F	Y	R/W	User variable General purpose user defined variable	
124	VAR_V24 Short Name: V24		F	Y	R/W	User variable General purpose user defined variable	
125	VAR_V25 Short Name: V25		F	Y	R/W	User variable General purpose user defined variable	
126	VAR_V26 Short Name: V26		F	Y	R/W	User variable General purpose user defined variable	
127	VAR_V27 Short Name: V27		F	Y	R/W	User variable General purpose user defined variable	
128	VAR_V28 Short Name: V28		F	Y	R/W	User variable General purpose user defined variable	
129	VAR_V29 Short Name: V29		F	Y	R/W	User variable General purpose user defined variable	
130	VAR_V30 Short Name: V30		F	Y	R/W	User variable General purpose user defined variable	
131	VAR_V31 Short Name: V31		F	Y	R/W	User variable General purpose user defined variable	
132	VAR_MOVEDR_DISTANCE		F	N	W	Registered move distance. Incremental motion as per MOVEDR statement	UU
133	VAR_MOVEDR_DISPLACEMENT	 mtn	F	N	W	Registered move displacement Writing to this variable executes the move MOVEDR using value set by #132	UU
134	VAR_MOVEPR_DISTANCE		F	N	W	Registered move distance. Absolute motion as per MOVEPR statement	UU
135	VAR_MOVEPR_DISPLACEMENT	 mtn	F	N	W	Registered move displacement Writing to this variable makes the move MOVEPR using value set by #134	UU
136	VAR_STOP_MOTION		W	N	W	Stops motion: 1 - stops motion 0 - no action	
137	VAR_START_PROGRAM		W	N	W	Starts user program 1 - starts program 0 - no action	
138	VAR_VEL_MODE_ON		W	N	W	Turns on Profile Velocity for Internal Position Mode. (Acts as statement VELOCITY ON) 0 - normal operation 1 - velocity mode on	
139	VAR_IREF Short Name: IREF		F	N	W	Reference for Internal Torque or Velocity Mode. 0 - Internal Velocity mode 1 - Internal Torque mode	RPS Amps
140	VAR_NV0 Short Name: NV0		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
141	VAR_NV1 Short Name: NV1		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
142	VAR_NV2 Short Name: NV2		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
143	VAR_NV3 Short Name: NV3		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
144	VAR_NV4 Short Name: NV4		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
145	VAR_NV5 Short Name: NV5		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
146	VAR_NV6 Short Name: NV6		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
147	VAR_NV7 Short Name: NV7		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
148	VAR_NV8 Short Name: NV8		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
149	VAR_NV9 Short Name: NV9		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
150	VAR_NV10 Short Name: NV10		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
151	VAR_NV11 Short Name: NV11		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
152	VAR_NV12 Short Name: NV12		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
153	VAR_NV13 Short Name: NV13		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
154	VAR_NV14 Short Name: NV14		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
155	VAR_NV15 Short Name: NV15		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
156	VAR_NV16 Short Name: NV16		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
157	VAR_NV17 Short Name: NV17		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
158	VAR_NV18 Short Name: NV18		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
159	VAR_NV19 Short Name: NV19		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
160	VAR_NV20 Short Name: NV20		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
161	VAR_NV21 Short Name: NV21		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
162	VAR_NV22 Short Name: NV22		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
163	VAR_NV23 Short Name: NV23		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
164	VAR_NV24 Short Name: NV24		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
165	VAR_NV25 Short Name: NV25		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
166	VAR_NV26 Short Name: NV26		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
167	VAR_NV27 Short Name: NV27		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
168	VAR_NV28 Short Name: NV28		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
169	VAR_NV29 Short Name: NV29		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
170	VAR_NV30 Short Name: NV30		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
171	VAR_NV31 Short Name: NV31		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
172	VAR_SERIAL_ADDRESS		W	Y	R/W	RS485 drive ID. Range: 0 - 254	
173	VAR_MODBUS_BAUDRATE		W	Y	R/W	Baud rate for ModBus operations: 0 - 2400 3 - 19200 1 - 4800 4 - 38400 2 - 9600 5 - 57600 6 - 115200	
174	VAR_MODBUS_DELAY		W	Y	R/W	ModBus reply delay in mS Range: 0 - 1000	mS
175	VAR_RS485_CONFIG		W	Y	R/W	Rs485 configuration: 0 - normal IP over PPP 1 - ModBus	
176	VAR_PPP_BAUDRATE NOTE: Does NOT apply to MVOB.		W	Y	R/W	RS232/485 (normal mode) baud rate: 1 - 4800 2 - 9600 3 - 19200 4 - 38400 5 - 57600 6 - 115200	
177	VAR_MOVEPS		F	N	W	Same as variable #92 but using S-curve acceleration/deceleration	
178	VAR_MOVEDS		F	N	W	Same as variable #93 but using S-curve acceleration/deceleration	
179	VAR_MDVS_VELOCITY	 mtn		N	W	Velocity for MDV move using S-curve accel/deceleration. Writing to this variable executes MDV move with Distance value last written to variable #94 (unless motion is suspended by #91).	UU
180	VAR_MAXVEL Short Name: MAXV		F	N	R/W	Max velocity for motion profile	UU/S
181	VAR_ACCEL Short Name: ACCEL		F	N	R/W	Accel value for indexing	UU/S ²
182	VAR_DECEL Short Name: DECEL		F	N	R/W	Decel value for indexing	UU/S ²
183	VAR_QDECEL Short Name: QDECEL		F	N	R/W	Quick decel value	UU/S ²
184	VAR_INPOSLIM Short Name: INPOSLIM		W	N	R/W	Sets window for "In Position" Limits	UU
185	VAR_VEL Short Name: VEL		F	N	R/W	Velocity reference for "Profiled" velocity	UU/S
186	VAR_UNITS Short Name: UNITS		F	Y	R/W	User units	
187	VAR_MEOUNTER Short Name: MEOUNTER		W	N	R/W	A/B inputs reference counter value	Count
188	VAR_PHCUR Short Name: PHCUR		F	N	R	Phase current	A
189	VAR_POS_PULSES Short Name: TPOS PLS		W	N	R/W	Target position in encoder pulses	EC

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
190	VAR_APOS_PULSES Short Name: APOS_PLS		W	N	R/W	Actual position in encoder pulses	EC
191	VAR_POSEERROR_PULSES Short Name: PERROR_PLS		W	N	R	Position error in encoder pulses	EC
192	VAR_CURRENT_VEL_PPS		F	N	R	Set-point (target) velocity in PPS	PPS
193	VAR_CURRENT_ACCEL_PPSS		F	N	R	Set-point (target) acceleration (demanded value) value	PPSS
194	VAR_IN0_DEBOUNCE		W	Y	R/W	Input A1 de-bounce time in mS Range: 0 - 1000	mS
195	VAR_IN1_DEBOUNCE		W	Y	R/W	Input A2 de-bounce time in mS Range: 0 - 1000	mS
196	VAR_IN2_DEBOUNCE		W	Y	R/W	Input A3 de-bounce time in mS Range: 0 - 1000	mS
197	VAR_IN3_DEBOUNCE		W	Y	R/W	Input A4 de-bounce time in mS Range: 0 - 1000	mS
198	VAR_IN4_DEBOUNCE		W	Y	R/W	Input B1 de-bounce time in mS Range: 0 - 1000	mS
199	VAR_IN5_DEBOUNCE		W	Y	R/W	Input B2 de-bounce time in mS Range: 0 - 1000	mS
200	VAR_IN6_DEBOUNCE		W	Y	R/W	Input B3 de-bounce time in mS Range: 0 - 1000	mS
201	VAR_IN7_DEBOUNCE		W	Y	R/W	Input B4 de-bounce time in mS Range: 0 - 1000	mS
202	VAR_IN8_DEBOUNCE		W	Y	R/W	Input C1 de-bounce time in mS Range: 0 - 1000	mS
203	VAR_IN9_DEBOUNCE		W	Y	R/W	Input C2 de-bounce time in mS Range: 0 - 1000	mS
204	VAR_IN10_DEBOUNCE		W	Y	R/W	Input C3 de-bounce time in mS Range: 0 - 1000	mS
205	VAR_IN11_DEBOUNCE		W	Y	R/W	Input C4 de-bounce time in mS Range: 0 - 1000	mS
206	VAR_OUT1_FUNCTION		W	Y	R/W	Programmable Output function 0 - Not Assigned 1 - Zero Speed 2 - In Speed Window 3 - Current Limit 4 - Run time fault 5 - Ready 6 - Brake 7 - In position	
207	VAR_OUT2_FUNCTION		W	Y	R/W	Programmable Output Function. See range (settings) for Variable #206	
208	VAR_OUT3_FUNCTION		W	Y	R/W	Programmable Output Function. See range (settings) for Variable #206	
209	VAR_OUT4_FUNCTION		W	Y	R/W	Programmable Output Function. See range (settings) for Variable #206	
210	VAR_HALLCODE		W	N	R	Current hall code Bit 0 - Hall 1 Bit 1 - Hall 2 Bit 2 - Hall 3	
211	VAR_ENCODER		W	N	R	Primary encoder current value	EC
212	VAR_RPOS_PULSES Short Name: RPOS_PLS		W	N	R	Registration position	EC

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
213	VAR_RPOS Short Name: RPOS		F	N	R	Registration position	UU
214	VAR_POS Short Name: TPOS		F	N	R/W	Target position	UU
215	VAR_APOS Short Name: APOS		F	N	R/W	Actual position	UU
216	VAR_POSEERROR Short Name: PERROR		W	N	R	Position error	EC
217	VAR_CURRENT_VEL Short Name: TV		F	N	R	Set-point (target) velocity (demanded value)	UU/S
218	VAR_CURRENT_ACCEL Short Name: TA		F	N	R	Set-point (target) acceleration (demanded value)	UU/S ²
219	VAR_TPOS_ADVANCE Short Name: TPOS_ADV		W	N	W	Target position advance. Every write to this variable adds value to the Target position summing point. Value gets added once per write. This variable useful when loop is driven by Master encoder signals and trying to correct phase. Value is in encoder counts	EC
220	VAR_IOINDEX Short Name: INDEX		W	N	R/W	Same as INDEX variable in user's program. See "INDEX" in Language Reference section Of this manual.	
221	VAR_PSLIMIT_PULSES		W	Y	R/W	Positive Software limit switch value in Encoder counts	EC
222	VAR_NSLIMIT_PULSES		W	Y	R/W	Negative Software limit switch value in Encoder counts	EC
223	VAR_SLS_MODE		W	Y	R/W	Soft limit switch action code: 0 - no action 1- Fault. 2- Stop and fault (When loop is driven by trajectory generator only. With all the other sources same action as 1) --	
224	VAR_PSLIMIT		F	Y	R/W	Same as var 221 but value in User Units	UU
225	VAR_NSLIMIT		F	Y	R/W	Same as var 222 but value in User Units	UU
226	VAR_SE_APOS_PULSES		W	N	R	2nd encoder actual position in encoder counts	EC
227	VAR_SE_POSEERROR_PULSES		W	N	R	2nd encoder position error in encoder counts	EC
228	VAR_MODBUS_PARITY		W	Y	R/W	Parity for Modbus Control: 0 - No Parity 1 - Odd Parity 2 - Even Parity	
229	VAR_MODBUS_STOPBITS		W	Y	R/W	Number of Stopbits for Modbus Control: 0 - 1.0 1 - 1.5 2 - 2.0	
230	VAR_M_NOMINALVEL		F	Y	R/W	Induction Motor Nominal Velocity Range: 500 - 20000 RPM	RPM
231	VAR_M_COSPHI		F	Y	R/W	Induction Motor Cosine Phi Range: 0 - 1.0	
232	VAR_M_BASEFREQUENCY		F	Y	R/W	Induction Motor Base Frequency: Range: 0 - 400Hz	Hz
233	VAR_M_SERIES					Induction Motor Series	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
234	VAR_CAN_BAUD_EPM		W	Y	R/W	CAN Bus Parameter: Baud Rate: 1 - 8 1 - 10k 2 - 20k 3 - 50k 4 - 125k 5 - 250k 6 - 500k 7 - 800k 8 - 1000k	
235	VAR_CAN_ADDR_EPM		W	Y	R/W	CAN Bus Parameter: Address: 1-127	
236	VAR_CAN_OPERMODE_EPM		W	Y	R/W	CAN Bus Parameter: Boot-up Mode: 0 - 2 (Operational State Control) 0 - enters into pre-operational state 1 - enters into operational state 2 - pseudo NMT: sends NMT Start Node command after delay (set by variable 237)	
237	VAR_CAN_OPERDELAY_EPM		W	Y	R/W	CAN Bus Parameter: pseudo NMT mode delay time in seconds (refer to variable 236)	sec
238	VAR_CAN_ENABLE_EPM		W	Y	R/W	CAN Bus Parameter: Mode Control: 0, 1, 2 0 - Disable CAN interface 1 - Enable CAN interface in DS301 mode Concurrent user's program execution possible 2 - Enable CAN interface in DS402 mode Concurrent user's program execution possible 3 - Enable DeviceNet 4 - Enable PROFIBUS DP	
239	VAR_HOME_ACCEL		F	Y	R/W	Homing Mode: ACCEL rate: 0 - 10000000.0	UU/sec ²
240	VAR_HOME_OFFSET		F	Y	R/W	Homing Mode: Home Position Offset Range: -32767 to +32767	UU
241	VAR_HOME_OFFSET_PULSES		W	Y	R/W	Homing Mode: Home Position Offset in encoder counts Range: +/- 2,147,418,112	EC
242	VAR_HOME_FAST_VEL		F	Y	R/W	Homing Mode: Fast Velocity Range: -10,000 to +10,000	UU/sec
243	VAR_HOME_SLOW_VEL		F	Y	R/W	Homing Mode: Slow Velocity Range: -10,000 to +10,000	UU/sec
244	VAR_HOME_METHOD		W	Y	R/W	Homing Mode: Homing Method Range: 1 - 35	
245	VAR_START_HOMING Short Name: HOME		W	N	W	Homing Mode: Start Homing: 0, 1 0 - No action 1 - Start Homing	
246	VAR_HOME_SWITCH_INPUT		W	Y	R/W	Homing Mode: Switch Input Assignment: Range: 0 - 11 0 - 3: A1 - A4 4 - 7: B1 - B4 8 - 11: C1 - C4 Warning: If using A1, A2, A3, or C3 refer to the homing section	
247	VAR_M_VALIDATE_MOTOR		W	N	W	Makes Drive accept Motor's parameters entered in motor data PIDs. Motor parameters are variables whose identifier starts with VAR_M_xxxxxx 0 - No Action 1 - Validate Motor Data	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
248	VAR_M_I2T		F	Y	R/W	Motor	
249	VAR_M_EABSOLUTE		F	Y	R/W	Indicates type of ABS encoder for models with ABS encoder support. Otherwise ignored	
250	VAR_M_ABSWAP		F	Y	R/W	Motor Encoder Feedback: B leads A 0 - No Action 1 - B leads A for forward checked (active)	
251	VAR_M_HALLS_INVERTED		F	Y	R/W	Motor Encoder Feedback: Halls 0 - No Action 1 - Inverted Halls Box checked (active)	
252	RESERVED					Do NOT Use	
253	RESERVED					Do NOT Use	
254	RESERVED					Do NOT Use	
255	RESERVED					Do NOT Use	
256	RESERVED					Do NOT Use	
257	RESERVED					Do NOT Use	
258	RESERVED					Do NOT Use	
259	RESOLVER_EMU_TRK		W	Y	R/W	Resolver Emulation Track Number Range: 0 - 15 0 - 1024 1 - 256 2 - 360 3 - 400 4 - 500 5 - 512 6 - 720 7 - 800 8 - 1000 9 - 1024 10 - 2000 11 - 2048 12 - 2500 13 - 2880 14 - 250 15 - 4096	
260	VAR_VELOCITY_ACTUAL		F	N	R	Actual measured motor velocity	UU/sec



NOTE:

PIDs 261-413 are **not** applicable to PositionServo drives with the PC-installed version of MotionView.

3.3 Quick Start Examples

Contained in the following four paragraphs are the connections and parameter settings to quickly setup a PositionServo drive for External Torque/Velocity, External Positioning, Internal Torque/Velocity and Internal Positioning modes. These Quick Start reference tables are NOT a substitute for reading the PositionServo User Manual. Observe all safety notices in the PositionServo User and Programming Manuals.

3.3.1 Quick Start - External Torque/Velocity

Table 63: Connections for External Torque/Velocity Mode

I/O (P3)		
Pin	Name	Function
20	AIN2+	Positive (+) of Analog signal input
21	AIN2-	Negative (-) of Analog signal input
22	ACOM	Analog common
23	A01	Analog output
24	AIN1+	Positive (+) of Analog signal input
25	AIN1 -	Negative (-) of Analog signal input
26	IN_A_COM	Digital input group A COM terminal
27	IN_A1	Digital input A1
28	IN_A2	Digital input A2
29	IN_A3	Digital input A3
30	IN_A4	Digital input A4
31	IN_B_COM	Digital input group B COM terminal
32	IN_B1	Digital input B1
33	IN_B2	Digital input B2
34	IN_B3	Digital input B3
35	IN_B4	Digital input B4
36	IN_C_COM	Digital input group C COM terminal
37	IN_C1	Digital input C1
38	IN_C2	Digital input C2
39	IN_C3	Digital input C3
40	IN_C4	Digital input C4
41	RDY+	Ready output Collector
42	RDY-	Ready output Emitter
43	OUT1-C	Programmable output #1 Collector
44	OUT1-E	Programmable output #1 Emitter
45	OUT2-C	Programmable output #2 Collector
46	OUT2-E	Programmable output #2 Emitter
47	OUT3-C	Programmable output #3 Collector
48	OUT3-E	Programmable output #3 Emitter
49	OUT4-C	Programmable output #4 Collector
50	OUT4-E	Programmable output #4 Emitter

Note 1: Connections highlighted in BLUE are mandatory/necessary for operation in this mode.

Reference

Table 64: Parameter Settings for External Torque/Velocity Mode

MV Folder	Sub-Folder	Setting	
Parameters	--	Parameter Name	Description
		Drive Mode	Set to [Torque] for Torque Mode; [Velocity] for Velocity Mode
		Analog Input (Current Scale)	Torque Mode Only: Set to Required Amps per Volt
		Analog Input (Velocity Scale)	Velocity Mode Only: Set to Required RPM per Volt
		Enable Accel/Decel Limits	Velocity Mode Only: Set to [Enable] to switch on velocity ramp rates; Set to [Disable] to switch OFF (accelerate at current limit)
		Accel Limit	Velocity Mode Only: Set Acceleration Limit in RPM/Sec
		Decel Limit	Velocity Mode Only: Set Deceleration Limit in RPM/Sec
		Reference	Set to [External] for external Torque/Velocity Mode
		Enable Switch Input	Set to [Run] to allow Enable/Disable of the PositionServo to be controlled via Input A3 (Dedicated Enable)
IO	Digital IO	Parameter Name	Description
		Output 1 Function	Output # indicates Digital Output No. 1-4; Set value to select Output Functionality; Output Function Values: 1=Not Assigned; 2=Zero Speed; 3=In Speed Window; 4=Current Limit; 5=Run Time Fault; 6=Ready; 7=Brake; 8=In Position
		Output 2 Function	
		Output 3 Function	
		Output 4 Function	
IO	Analog IO	Parameter Name	Description
		Analog Input Dead Band	Set Zero Speed Dead Band in mV for Torque/Velocity Reference on Analog Input 1
		Analog Input Offset	Set Torque/Velocity Reference Input Offset on Analog Input 1 to match Controller Offset
		Adjust Analog Input Zero Offset	Tool to automatically learn the Analog Input Offset (of Analog Input 1)
Limits	Velocity Limits	Parameter Name	Description
		Zero Speed	Velocity Mode Only: Set a bandwidth (around 0RPM) for activation of the Zero Speed Output/Flag
		At Speed	Velocity Mode Only: Set a Target Speed for activation of the At Speed Output/Flag
		Speed Window	Velocity Mode Only: Set a bandwidth (around At Speed parameter) for activation of the At Speed Output/Flag
Compensation	--	Parameter Name	Description
		Velocity P-Gain	Velocity Mode Only: Set P-Gain for Velocity Loop
		Velocity I-Gain	Velocity Mode Only: Set I-Gain for Velocity Loop
		Gain Scaling	Velocity Mode Only: Apply Scaling Factor to Velocity Gain Set

Note 1: Parameters highlighted in BLUE are mandatory/necessary for operation in this mode.

3.3.2 Quick Start - External Positioning

Table 65: Connections for External Positioning Mode

I/O (P3)		
Pin	Name	Function
1	MA+	Master Encoder A+ / Step+ input
2	MA-	Master Encoder A- / Step- input
3	MB+	Master Encoder B+ / Direction+ input
4	MB-	Master Encoder B- / Direction- input
5	GND	Drive Logic Common
6	+5V	+5V Output (max 100mA)
7	BA+	Buffered Encoder Output: Channel A+
8	BA-	Buffered Encoder Output: Channel A-
9	BB+	Buffered Encoder Output: Channel B+
10	BB-	Buffered Encoder Output: Channel B-
11	BZ+	Buffered Encoder Output: Channel Z+
12	BZ-	Buffered Encoder Output: Channel Z-
26	IN_A_COM	Digital input group A COM terminal
27	IN_A1	Digital input A1
28	IN_A2	Digital input A2
29	IN_A3	Digital input A3
30	IN_A4	Digital input A4
41	RDY+	Ready output Collector
42	RDY-	Ready output Emitter
43	OUT1-C	Programmable output #1 Collector
44	OUT1-E	Programmable output #1 Emitter
45	OUT2-C	Programmable output #2 Collector
46	OUT2-E	Programmable output #2 Emitter
47	OUT3-C	Programmable output #3 Collector
48	OUT3-E	Programmable output #3 Emitter
49	OUT4-C	Programmable output #4 Collector
50	OUT4-E	Programmable output #4 Emitter

Note 1: Connections highlighted in BLUE are mandatory/necessary for operation in this mode.

Note 2: Connections highlighted in GREEN are frequently required in applications of this type.

Reference

Table 66: Parameter Settings for External Positioning Mode

MVOB Folder	Sub-Folder	Setting	
Parameters	--	Parameter Name	Description
		Drive Mode	Set to [Position] for Position Mode
		Reference	Set to [External] for external Position Mode
		Step Input Type	Set to either [Step and Direction] or [Master Encoder] to match the Position Controller
		System to Master Ratio	Set Electronic Gear Ratio on Reference Signal to the PositionServo Motor Output
		Enable Switch Input	Set to [Run] to allow Enable/Disable of the PositionServo to be controlled via Input A3 (Dedicated Enable)
		Resolver Track	If using Resolver Feedback, set value that represents the pulses per revolution required on the PositionServo simulated encoder. 0=1024ppr; 1=256ppr; 2=360ppr; 3=400ppr; 4=500ppr; 5=512ppr; 6=720ppr; 7=800ppr; 8=1000ppr; 9=1024ppr; 10=2000ppr; 11=2048ppr; 12=2500ppr; 13=2880ppr; 14=250ppr; 15=4096ppr
IO	Digital IO	Parameter Name	Description
		Output 1 Function	Output # indicates Digital Output No. 1-4; Set value to select Output Functionality; Output Function Values: 1=Not Assigned; 2=Zero Speed; 3=In Speed Window; 4=Current Limit; 5=Run Time Fault; 6=Ready; 7=Brake; 8=In Position
		Output 2 Function	
		Output 3 Function	
		Output 4 Function	
Hard Limit Switches Action	Set to Enable Inputs A1 and A2 to act as System Hard Limit Switches and define functionality in the event of an active input.		
Limits	Position Limits	Parameter Name	Description
		Position Error	Set Position Error Limit at which Position Error Timer starts counting
		Max Error Time	Set Maximum Error Time for Position Error Correction before position error trip occurs.
Compensation	--	Parameter Name	Description
		Velocity P-Gain	Set P-Gain for Velocity Loop
		Velocity I-Gain	Set I-Gain for Velocity Loop
		Position P-Gain	Set P-Gain for Position Loop
		Position I-Gain	Set I-Gain for Position Loop
		Position D-Gain	Set D-Gain for Position Loop
		Position I-Limit	The Position I-Limit will clamp the Position I-Gain compensator to prevent excessive torque overshoot caused by an over-accumulation of I-Gain.
Gain Scaling	Apply Scaling Factor to Velocity Gain Set		

Note 1: Parameters highlighted in BLUE are mandatory/necessary for operation in this mode.

3.3.3 Quick Start - Internal Torque/Velocity

Table 67: Internal Torque/Velocity Mode

Connections for Internal Torque/Velocity: I/O (P3)			Variable References for Internal Torque/Velocity				
Pin	Name	Function	Index	Name	EPM	R/W	Description
20	AIN2+	Positive (+) of Analog signal input	29	VAR_ENABLE_SWITCH_TYPE	Y	R/W	Enable switch function: 0-inhibit only, 1- Run
21	AIN2-	Negative (-) of Analog signal input	34	VAR_DRIVEMODE	Y	R/W	Drive mode selection: 0-torque 1-velocity, 2-position
22	ACOM	Analog common	37	VAR_REFERENCE	Y	R/W	Reference source: set to 1 - internal (for 'internal torque' or 'internal velocity' mode)
23	A01	Analog output	44	VAR_VP_GAIN	Y	R/W	Velocity loop Proportional gain Range: 0 - 32767
24	AIN1+	Positive (+) of Analog signal input	45	VAR_VI_GAIN	Y	R/W	Velocity loop Integral gain Range: 0 - 16383
25	AIN1 -	Negative (-) of Analog signal input	51	VAR_VREG_WINDOW	Y	R/W	Gains scaling coefficient Range: -5 - +4
26	IN_A_COM	Digital input group A COM terminal	52	VAR_ENABLE	N	W	Software Enable/Disable: 0 – disable, 1 - enable
27	IN_A1	Digital input A1	58	VAR_VLIMIT_ZEROSPEED	Y	R/W	Zero Speed value Range: 0 - 100
28	IN_A2	Digital input A2	59	VAR_VLIMIT_SPEEDWND	Y	R/W	Speed window Range: 10 - 10000
29	IN_A3	Digital input A3	60	VAR_VLIMIT_ATSPEED	Y	R/W	Target speed for velocity window Range: -10000 - +10000
30	IN_A4	Digital input A4	71	VAR_AIN1	N	R	Analog Input AIN1 current value
31	IN_B_COM	Digital input group B COM terminal	72	VAR_AIN2	N	R	Analog Input AIN2 current value
32	IN_B1	Digital input B1	75	VAR_ENABLE_ACCELDECCEL	Y	R/W	Enable Accel/Decel (velocity mode), 0 – disable, 1 - enable
33	IN_B2	Digital input B2	76	VAR_ACCEL_LIMIT	Y	R/W	Accel value for velocity mode Range: 0.1 - 5000000
34	IN_B3	Digital input B3	77	VAR_DECEL_LIMIT	Y	R/W	Decel value for velocity mode Range: 0.1 - 5000000
35	IN_B4	Digital input B4	139	VAR_IREF	N	R/W	Internal ref Current or Velocity mode
36	IN_C_COM	Digital input group C COM terminal	192	VAR_CURRENT_VEL_PPS	N	R	Current velocity in PPS (pulses per sample)
37	IN_C1	Digital input C1	193	VAR_CURRENT_ACCEL_PPSS	N	R	Current acceleration (demanded value) value
38	IN_C2	Digital input C2	217	VAR_CURRENT_VEL	N	R	Current velocity (demanded value)
39	IN_C3	Digital input C3	218	VAR_CURRENT_ACCEL	N	R	Current acceleration (demanded value)
40	IN_C4	Digital input C4	Positional Mode Language Reference - Enable/Disable				
41	RDY+	Ready output Collector	Command	Syntax	Long Name		
42	RDY-	Ready output Emitter	DISABLE	DISBALE	Turns OFF Servo output		
43	OUT1-C	Programmable output #1 Collector	ENABLE	ENABLE	Turns ON Servo output		
44	OUT1-E	Programmable output #1 Emitter					
45	OUT2-C	Programmable output #2 Collector					
46	OUT2-E	Programmable output #2 Emitter					
47	OUT3-C	Programmable output #3 Collector					
48	OUT3-E	Programmable output #3 Emitter					
49	OUT4-C	Programmable output #4 Collector					
50	OUT4-E	Programmable output #4 Emitter					

Note 1: Connections highlighted in BLUE are mandatory/necessary for operation in this mode.

Reference

Example Internal Torque Program

```
;Program slowly increases Motor Torque until nominal motor current is reached
VAR_DriveMode = 0 ;Set Drive to Torque mode
VAR_Reference = 1 ;Set Reference to Internal control
Program Start:
IREF = 0 ;Reset Torque Reference to 0(Amps)
Wait While !In_A3 ;Wait while Enable input is OFF
Enable ;Enable Drive
Torque_Loop:
Wait Time 500 ;Set time between step increases in Torque
If REF < VAR_CurrentLimit ;If Set Torque < Motor Nominal Torque
IREF = IREF+0.1 ;Then increase by 0.1(Amps)
GOTO Torque_Loop ;Loop to next torque increase
Else
Goto Program_Start ;Else restart program
Endif
END
```

Example Internal Velocity Program

```
;Program slowly increases and decreases Motor Velocity between Maximum Velocity Forward direction and
;Maximum Velocity Reverse direction producing a saw-tooth velocity profile.
Define MaxVelocityRPS 60 ;Enter Maximum Velocity (RPS) value here
Define VelocityStepRPS 1 ;Define Velocity INC/DEC per Step/Program Loop (RPS)
Define VelocityStepTime 200 ;Define Time for Velocity Steps in mS
Define Velocity_Inc_Dec V0 ;Define a Variable to identify if Velocity is currently INC/DECcreasing
VAR_DriveMode = 1 ;Set Drive to Velocity mode
VAR_Reference = 1 ;Set Reference to Internal control
VAR_Enable_AccelDecel = 1 ;Enable Accel/Decel Ramps
VAR_Accel_Limit = 3000 ;Set Accel Rate required in RPS^2
VAR_Decel_Limit = 3000 ;Set Decel Rate required in RPS^2
Program Start:
IREF = 0 ;Reset Velocity Reference to 0(RPS)
Wait While !In_A3 ;Wait while Enable input is OFF
Enable ;Enable Drive
Velocity_Loop:
Wait Time VelocityStep Time ;Set Time between Step Increases/Decreases in Velocity (mS)
If REF <= MaxVelocityRPS ;If Current Motor Velocity < MaxVelocityRPS
IREF = IREF+VelocityStepRPS ;Then increase Velocity by VelocityStepRPS
Else
Velocity_Inc_Dec = 1 ;Set Variable to start decreasing velocity
Endif
Else ;If Speed Decreasing
If REF >= -1* MaxVelocityRPS ;If Current Motor Velocity > -MaxVelocityRPS
IREF = IREF-VelocityStepRPS ;Then decrease Velocity by VelocityStepRPS
Else
Velocity_Inc_Dec = 0 ;Set Variable to start increasing velocity
Endif
Endif
Goto Velocity_Loop ;Loop to next Velocity Increase/Decrease
END ;End Code - Never Reached
On Fault ;Fault Handler
Resume Program_Start ;Resume at Program Start
EndFault
```

3.3.4 Quick Start - Internal Positioning

Table 68: Internal Positioning

Connections: I/O (P3)		
Pin	Name	Function
26	IN_A_COM	Digital input group A COM terminal
27	IN_A1	Digital input A1
28	IN_A2	Digital input A2
29	IN_A3	Digital input A3
30	IN_A4	Digital input A4
31	IN_B_COM	Digital input group B COM terminal
32	IN_B1	Digital input B1
33	IN_B2	Digital input B2
34	IN_B3	Digital input B3
35	IN_B4	Digital input B4
36	IN_C_COM	Digital input group C COM terminal
37	IN_C1	Digital input C1
38	IN_C2	Digital input C2
39	IN_C3	Digital input C3
40	IN_C4	Digital input C4
41	RDY+	Ready output Collector
42	RDY-	Ready output Emitter
43	OUT1-C	Programmable output #1 Collector
44	OUT1-E	Programmable output #1 Emitter
45	OUT2-C	Programmable output #2 Collector
46	OUT2-E	Programmable output #2 Emitter
47	OUT3-C	Programmable output #3 Collector
48	OUT3-E	Programmable output #3 Emitter
49	OUT4-C	Programmable output #4 Collector
50	OUT4-E	Programmable output #4 Emitter

Language Reference		
Enable/Disable		
Command	Syntax	Long Name
DISABLE	DISBALE	Turns OFF Servo output
ENABLE	ENABLE	Turns ON Servo output
Program Structure		
Command	Syntax	Long Name
STOP MOTION	STOP MOTION	Stop AA Motion - Clear
STOP MOTION QUICK	STOP MOTION QUICK	Motion Slack
WAIT	WAIT MOTION COMPLETE	Wait
Move / Motion Commands		
Command	Syntax	Long Name
MOVE	MOVE [BACK] UNTIL <condition> [,C]	Move
MOVED	MOVED <distance> [,S] [,C]	Move Distance
MOVEP	MOVEP <absolute position> [,S] [,C]	Move to Position
MOVEDR	MOVEDR <distance> , <displacement> [,C]	Registered Distance Move
MOVEPR	MOVEPR <distance> , <displacement> [,C]	Registered Position Move
MDV	MDV <[-]segment distance>,<segment final velocity>[,S]	Segmented Move
MOTION SUSPEND	MOTION SUSPEND	Temporarily Suspend Motion
MOTION RESUME	MOTION RESUME	Statement Resumes Motion

Reference

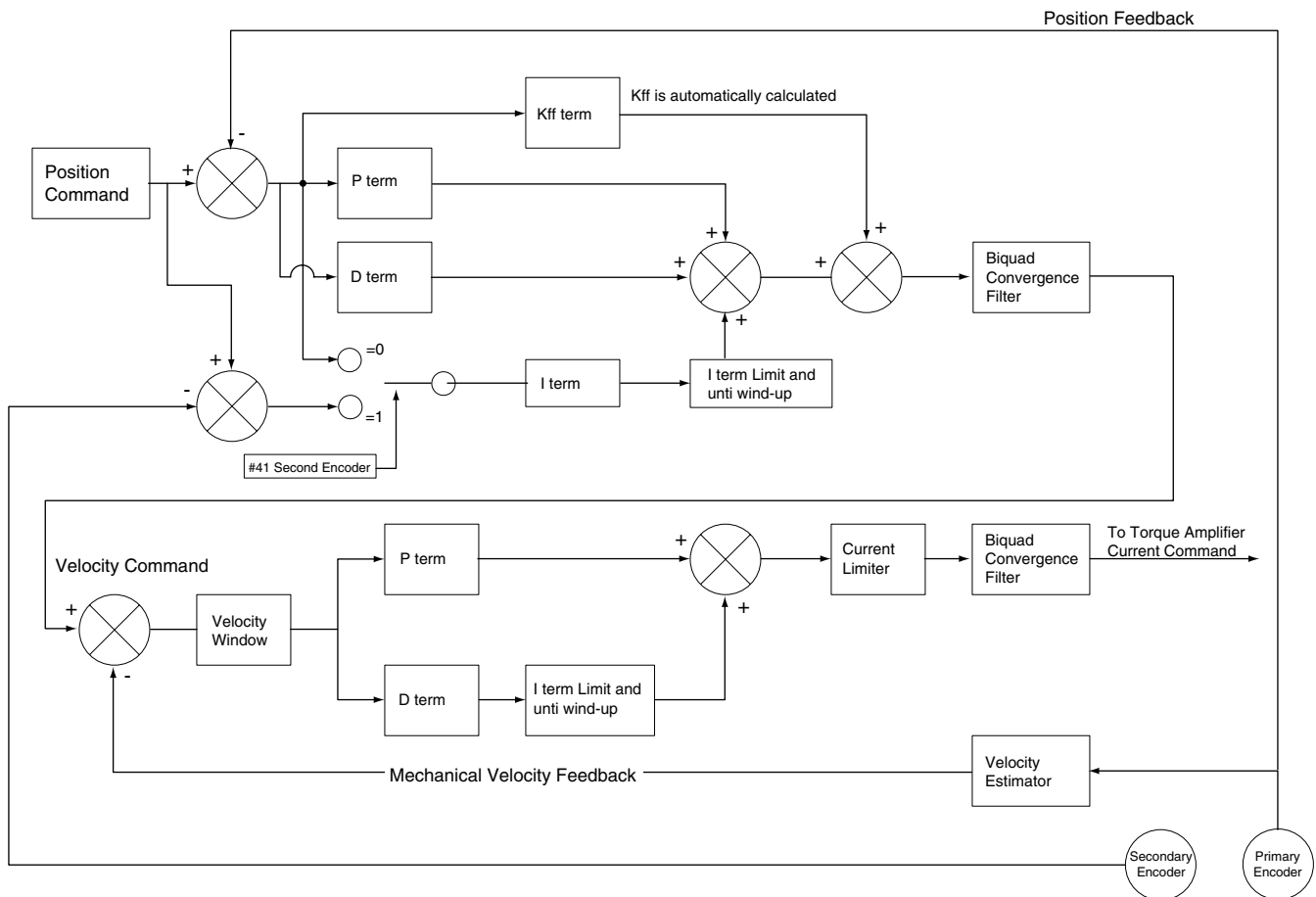
3.4 PositionServo Reference Diagrams

This section contains the process flow diagrams listed in Table 69. These diagrams are for reference only.

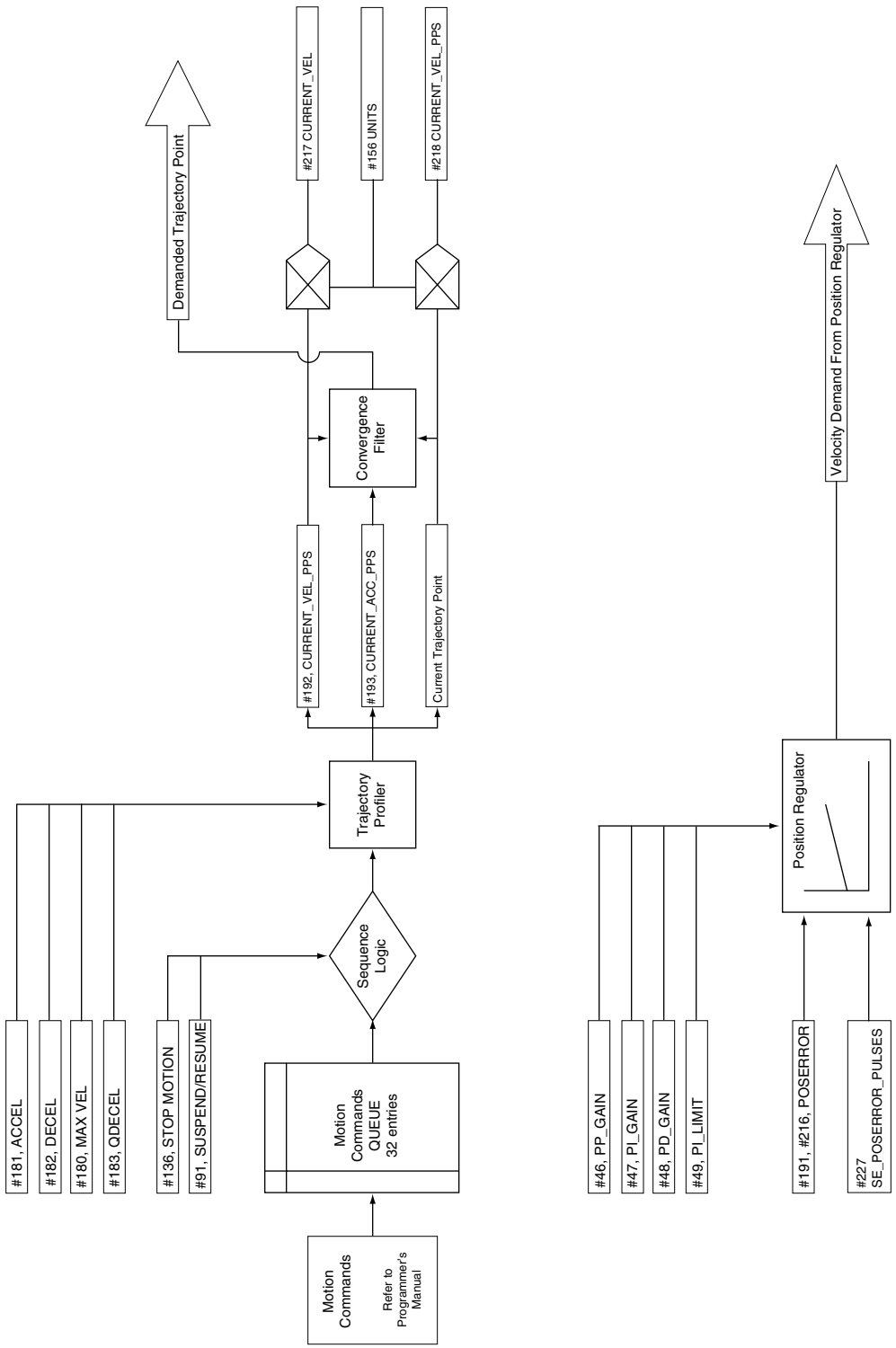
Table 69: PositionServo Process Flow Diagrams

Drawing #	Description
S999	Position and Velocity Regulator
S1000	Motion Commands -> Motion Queue -> Trajectory Generator
S1001	Current Command -> Motor
S1002	Encoder Inputs
S1003	Analog Inputs
S1004	Analog Outputs
S1005	Digital Inputs
S1006	Digital Outputs

Position and Velocity Regulators

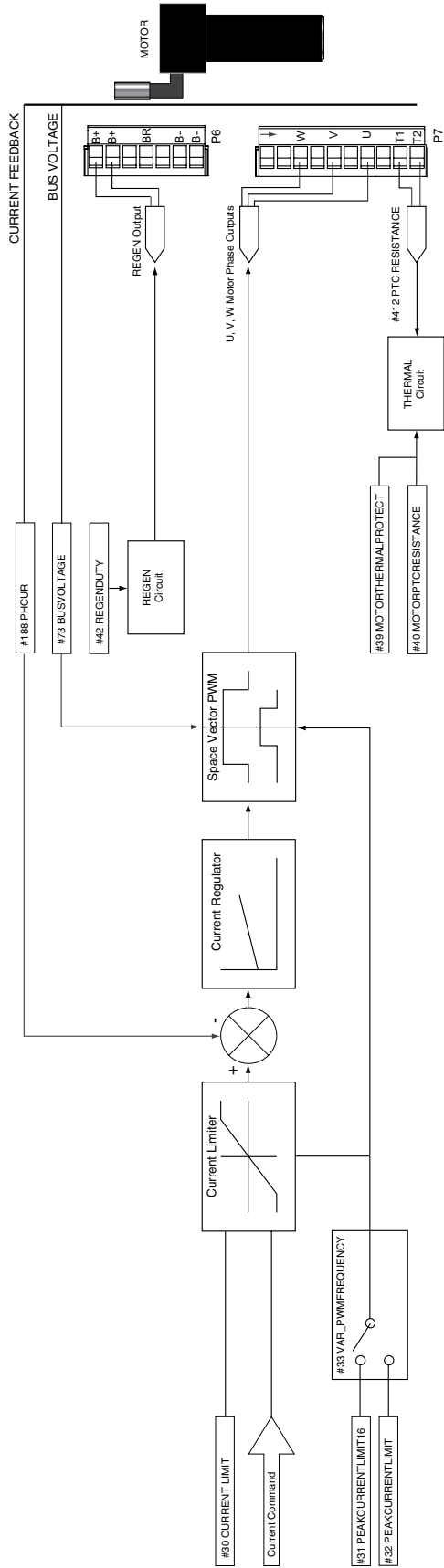


Motion Commands, Motion Queue & Trajectory Generator

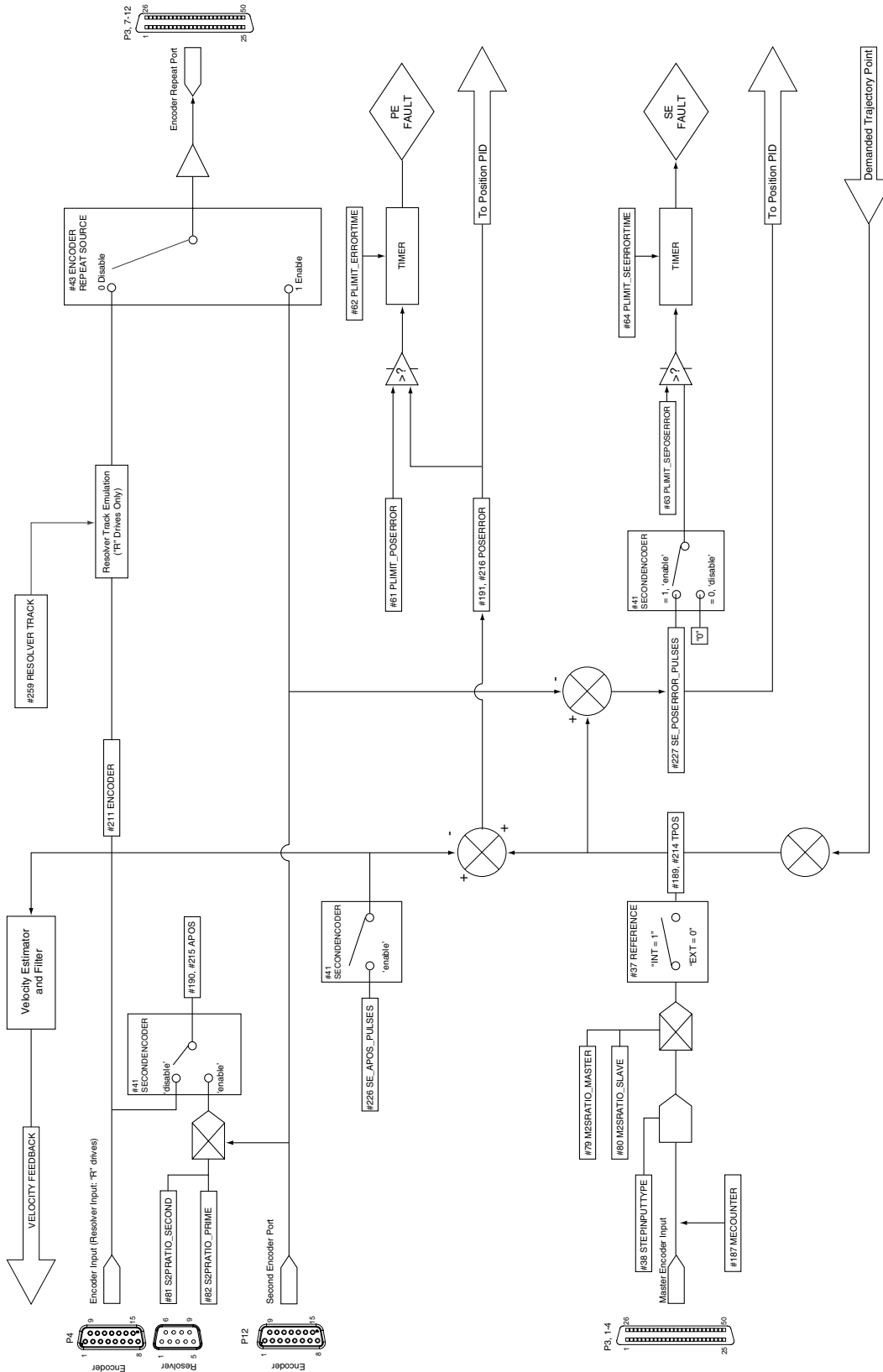


Reference

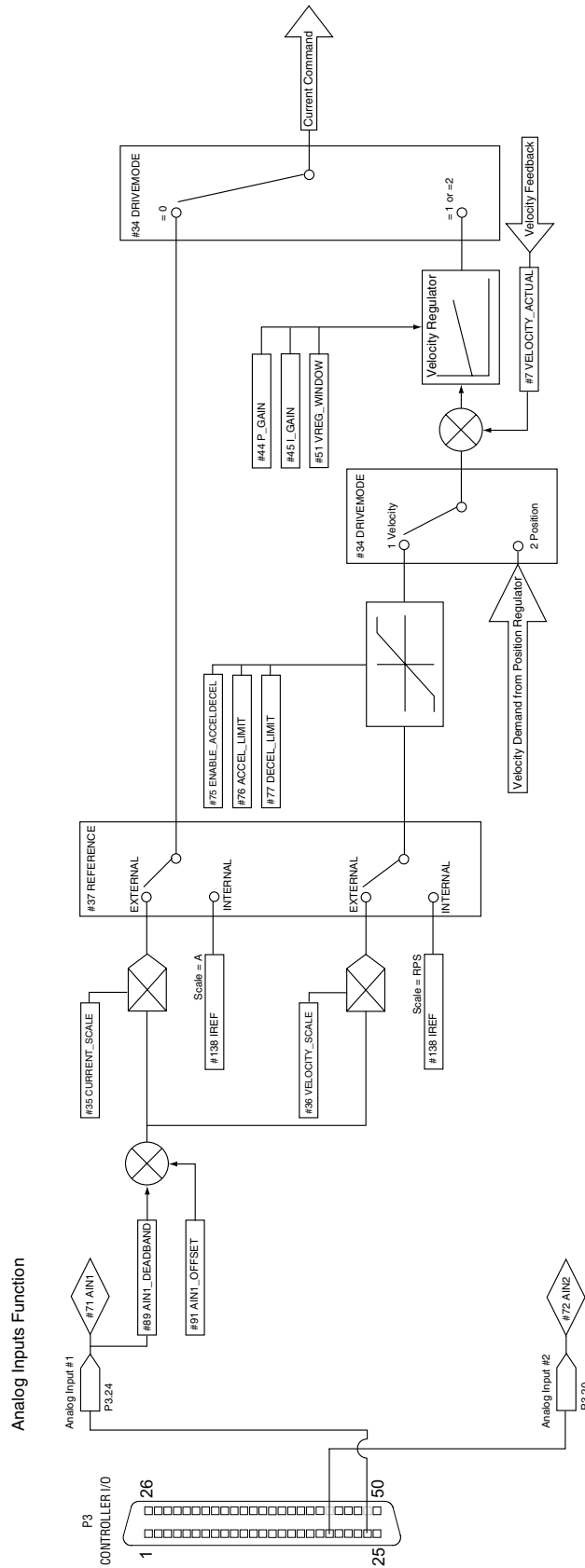
Current Command --> Motor



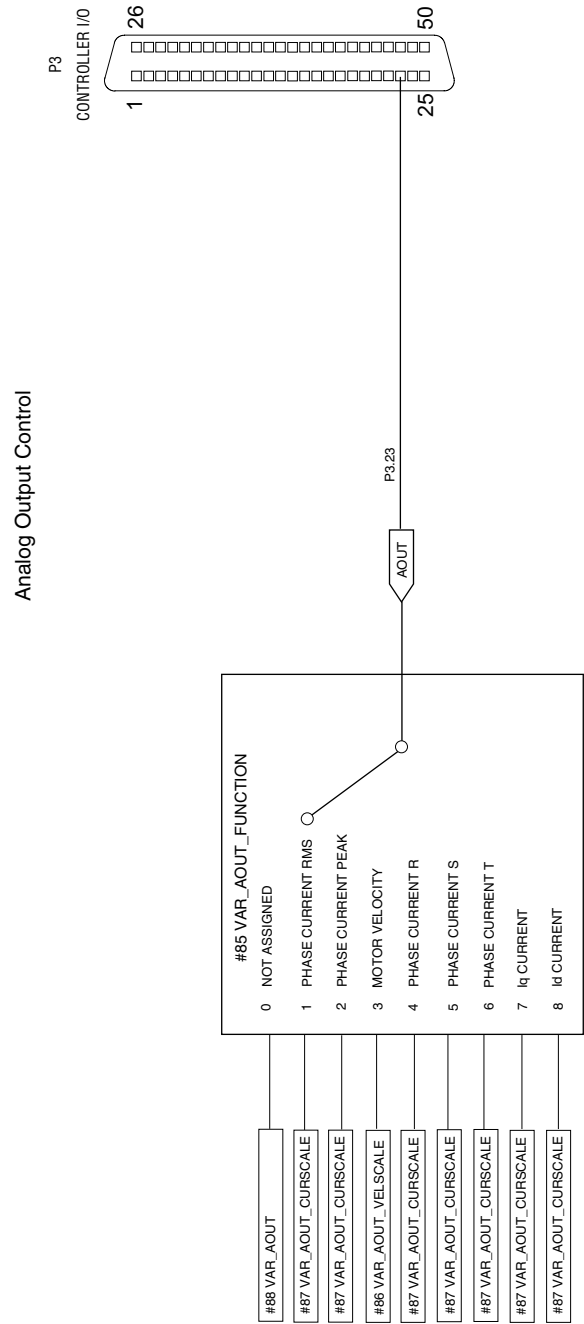
Encoder Inputs



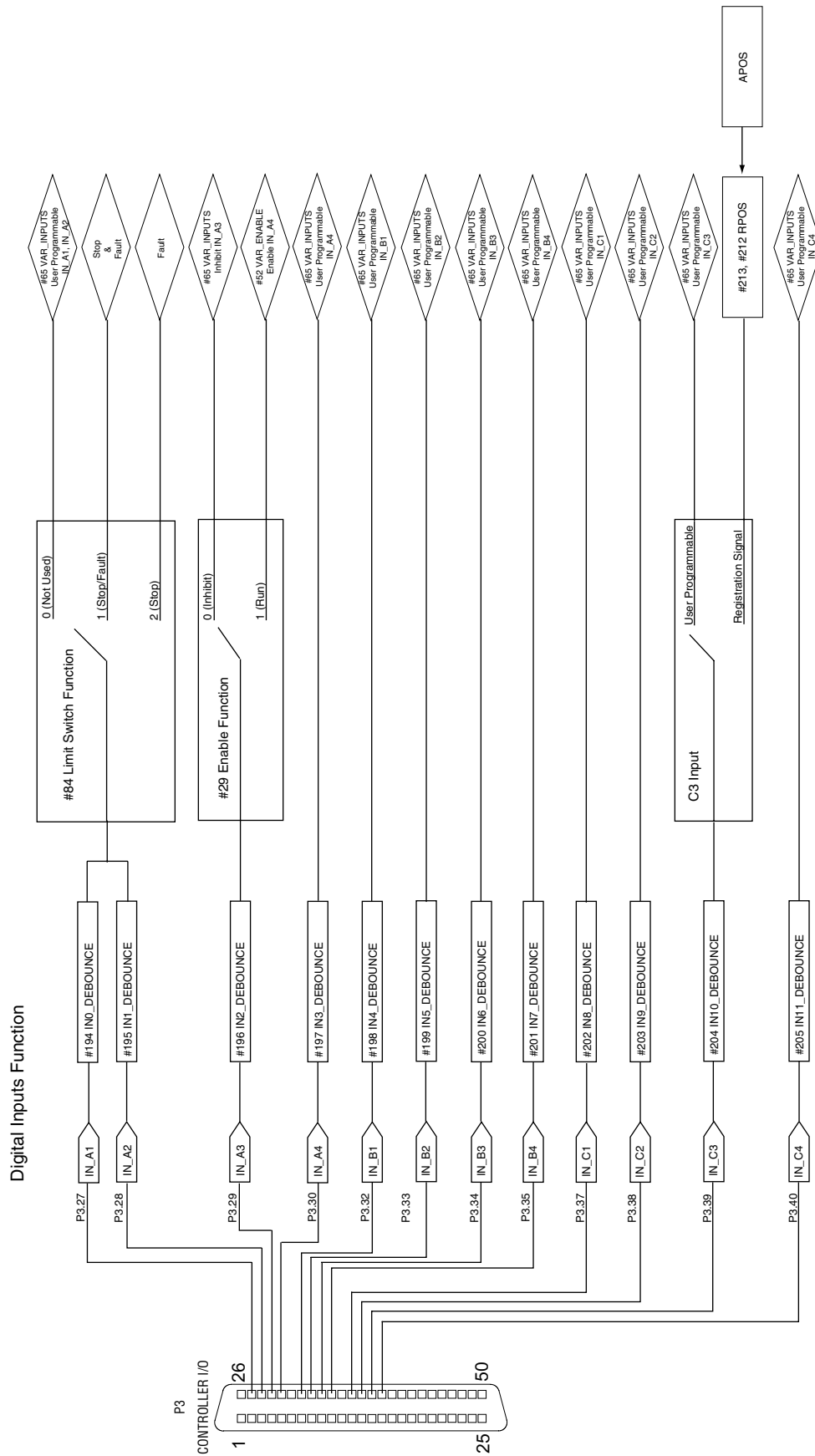
Analog Inputs



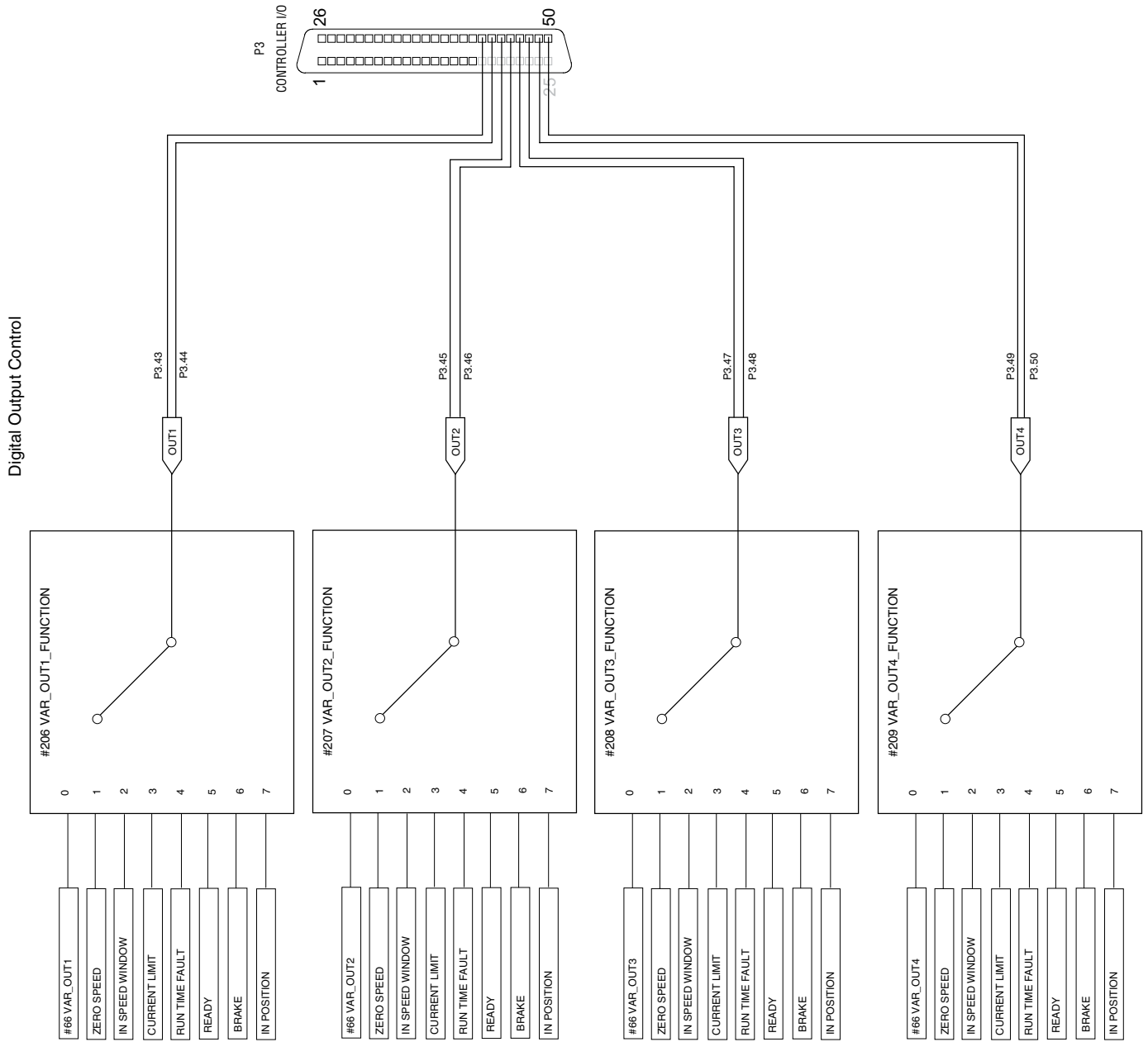
Analog Output



Digital Inputs



Digital Outputs



Lenze AC Tech Corporation

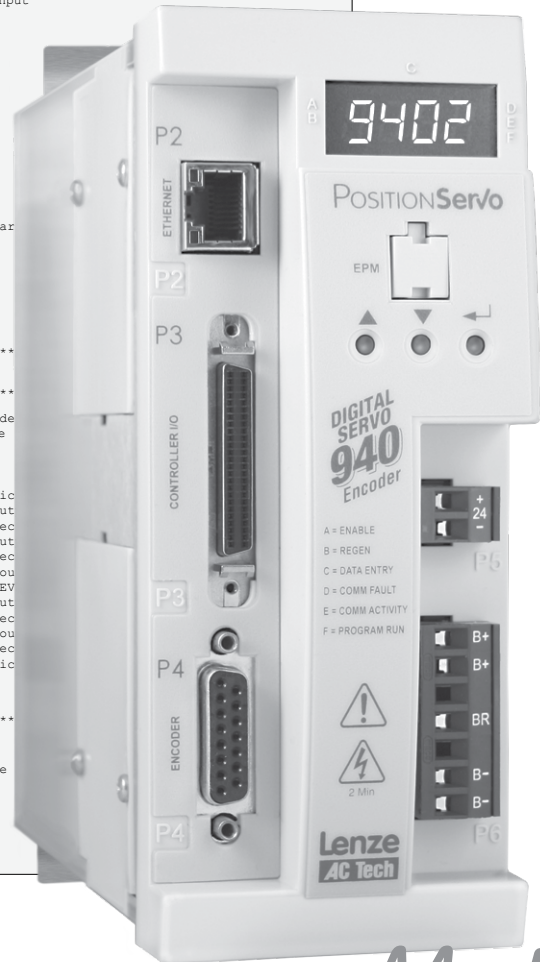
630 Douglas Street • Uxbridge, MA 01569 • USA
Sales 800 217 9100 • Service 508 278 9100
www.lenze-actech.com

PM94P01C

```

***** HEADER *****
;Title:          Pick and Place example program
;Author:         Lenze - AC Technology
;Description:    This is a sample program showing a simple sequence that
;               picks up a part, moves to a set position and drops the part
;
;***** I/O List *****
;               Input A1 - not used
;               Input A2 - not used
;               Input A3 - Enable Input
;               Input A4 - not used
;               Input B1 - not used
;               Input B2 - not used
;               Input B3 - not used
;               Input B4 - not used
;               Input C1 - not used
;               Input C2 - not used
;               Input C3 - not used
;               Input C4 - not used
;               Output 1 - Pick Arm
;               Output 2 - Gripper
;               Output 3 - not used
;               Output 4 - not used
;
;***** Initialize and Set Variables *****
UNITS = 1
ACCEL = 75
DECEL = 75
MAXV = 10
;V1 =
;V2 =
;
;***** Events *****
;Set Events handling here
;***** Main Program *****
RESET_DRIVE:                ;Place holder
WAIT UNTIL IN_A3:          ;Make sure
                           continuing
ENABLE
PROGRAM START:
MOVEP 0                    ;Move to Pick
OUT1 = 1                   ;Turn on out
WAIT TIME 1000             ;Delay 1 sec
OUT2 = 1                   ;Turn on out
WAIT TIME 1000             ;Delay 1 sec
OUT1 = 0                   ;Turn off ou
MOVED -10                  ;Move 10 REV
OUT1 = 1                   ;Turn on out
WAIT TIME 1000             ;Delay 1 sec
OUT2 = 0                   ;Turn off ou
WAIT TIME 1000             ;Delay 1 sec
OUT1 = 0                   ;Retract Pic
GOTO PROGRAM_START
END
;***** Sub-Routines *****
Enter Sub-Routine code here
;***** Fault Handler Routine *****
;               Enter Fault Handler code here
ON FAULT
ENDFAULT

```



MotionView[®]
On Board

PositionServo with MVOB Programming Manual

Valid for Hardware Version 2

Copyright © 2013 - 2010 by Lenze AC Tech Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from Lenze AC Tech Corporation. The information and technical data in this manual are subject to change without notice. Lenze AC Tech Corporation makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. Lenze AC Tech Corporation assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView[®], PositionServo[®], and all related indicia are either registered trademarks or trademarks of Lenze AG in the United States and other countries.

Contents

1.	Introduction	4
1.1	Definitions	4
1.2	Programming Flowchart	5
1.3	MotionView / MotionView Studio	6
1.3.1	Main Toolbar	6
1.3.2	Program Toolbar	7
1.3.3	MotionView Studio - Indexer Program	9
1.4	Programming Basics	10
1.5	Using Advanced Debugging Features	17
1.6	Inputs and Outputs	17
1.7	Events	22
1.8	User Variables and the Define Statement	23
1.9	IF/ELSE Statements	24
1.10	Motion	25
1.10.1	Drive Operating Modes	26
1.10.2	Point To Point Moves	26
1.10.3	Segment Moves	27
1.10.4	Registration	28
1.10.5	S-Curve Acceleration/Deceleration	29
1.10.6	Motion Queue	29
1.11	Subroutines and Loops	30
1.11.1	Subroutines	30
1.11.2	Loops	31
2.	Programming	32
2.1	Program Structure	32
2.2	Variables	34
2.3	Arithmetic Expressions	36
2.4	Logical Expressions and Operators	36
2.4.1	Bitwise Operators	36
2.4.2	Boolean Operators	37
2.5	Comparison Operators	37
2.6	System Variables and Flags	37
2.7	System Variables Storage Organization	38
2.7.1	RAM File for User's Data Storage	38
2.7.2	Memory Access Through Special System Variables	39
2.7.3	Memory Access Through MEMSET, MEMGET Statements	40
2.7.4	Store and Retrieve Variables from the EPM	41
2.8	System Variables and Flags Summary	42
2.8.1	System Variables	42
2.8.2	System Flags	43
2.9	Control Structures	44
2.9.1	IF Structure	44
2.9.2	DO/UNTIL Structure	45
2.9.3	WHILE Structure	45
2.9.4	WAIT Statement	45
2.9.5	GOTO Statement and Labels	46
2.9.6	Subroutines	46
2.10	Scanned Event Statements	47

Contents

2.11	Motion.....	48
2.11.1	How Moves Work.....	48
2.11.2	Incremental (MOVED) and Absolute (MOVEP) Motion	48
2.11.3	Incremental (MOVED) Motion.....	49
2.11.4	Absolute (MOVEP) Move.....	49
2.11.5	Registration (MOVEDR MOVEPR) Moves	50
2.11.6	Segment Moves.....	50
2.11.7	MDV Segments.....	50
2.11.8	S-curve Acceleration/Deceleration	52
2.11.9	Motion SUSPEND/RESUME	52
2.11.10	Conditional Moves (MOVE WHILE/UNTIL)	52
2.11.11	Motion Queue and Statement Execution while in Motion	53
2.12	System Status Register (DSTATUS register).....	55
2.13	Fault Codes (DFAULTS register)	56
2.14	Limitations and Restrictions.....	57
2.15	Homing	58
2.15.1	What is Homing?	58
2.15.2	The Homing Function	58
2.15.3	Home Offset.....	58
2.15.4	Homing Velocity.....	59
2.15.5	Homing Acceleration.....	59
2.15.6	Homing Switch.....	59
2.15.7	Homing Start.....	59
2.15.8	Homing Method	60
2.15.9	Homing Methods.....	61
2.15.9.1	Homing Method 1: Homing on the Negative Limit Switch & Index Pulse	62
2.15.9.2	Homing Method 2: Homing on the Positive Limit Switch & Index Pulse	62
2.15.9.3	Homing Method 3: Homing on the Positive Home Switch & Index Pulse	63
2.15.9.4	Homing Method 4: Homing on the Positive Home Switch & Index Pulse	63
2.15.9.5	Homing Method 5: Homing on the Negative Home Switch & Index Pulse	64
2.15.9.6	Homing Method 6: Homing on the Negative Home Switch & Index Pulse	64
2.15.9.7	Homing Method 7: Homing on the Home Switch & Index Pulse.....	65
2.15.9.8	Homing Method 8: Homing on the Home Switch & Index Pulse.....	66
2.15.9.9	Homing Method 9: Homing on the Home Switch & Index Pulse.....	67
2.15.9.10	Homing Method 10: Homing on the Home Switch & Index Pulse.....	68
2.15.9.11	Homing Method 11: Homing on the Home Switch & Index Pulse.....	69
2.15.9.12	Homing Method 12: Homing on the Home Switch & Index Pulse.....	70
2.15.9.13	Homing Method 13: Homing on the Home Switch & Index Pulse.....	71
2.15.9.14	Homing Method 14: Homing on the Home Switch & Index Pulse.....	72
2.15.9.15	Homing Method 17: Homing to Negative Limit Switch (without index pulse).....	73
2.15.9.16	Homing Method 18: Homing to Positive Limit Switch (without index pulse)	74
2.15.9.17	Homing Method 19: Homing to Homing Switch (without index pulse)	75
2.15.9.18	Homing Method 21: Homing to Homing Switch (without index pulse)	76
2.15.9.19	Homing Method 23: Homing to Homing Switch (without index pulse)	77
2.15.9.20	Homing Method 25: Homing to Homing Switch (without index pulse)	78
2.15.9.21	Homing Method 27: Homing to Homing Switch (without index pulse)	79
2.15.9.22	Homing Method 29: Homing to Homing Switch (without index pulse)	80
2.15.9.23	Homing Method 33: Homing to an Index Pulse	81
2.15.9.24	Homing Method 34: Homing to an Index Pulse	81
2.15.9.25	Homing Method 35: Using Current Position as Home	81
2.15.10	Homing Mode Operation Example.....	82
3.	Reference	83
3.1	Program Statement Glossary	83
3.2	Variable List.....	103
3.3	Quick Start Examples	122
3.3.1	Quick Start - External Torque/Velocity.....	122
3.3.2	Quick Start - External Positioning	124
3.3.3	Quick Start - Internal Torque/Velocity.....	126
3.3.4	Quick Start - Internal Positioning	128
3.4	PositionServo Reference Diagrams	130

About These Instructions

This documentation applies to the programming of the PositionServo drive with model numbers ending in S or M. This documentation should be used in conjunction with the PositionServo User Manual (Document S94H201) that shipped with the drive. These documents should be read in their entirety as they contain important technical data and describe the installation and operation of the drive.

Safety Warnings

Take note of these safety warnings and those in the PositionServo User Manual and related documentation.



WARNING! Hazard of unexpected motor starting!

When using MotionView, or otherwise remotely operating the PositionServo drive, the motor may start unexpectedly, which may result in damage to equipment and/or injury to personnel. Make sure the equipment is free to operate and that all guards and covers are in place to protect personnel.

All safety information contained in these Programming Instructions is formatted with this layout including an icon, signal word and description:



Signal Word! (Characterizes the severity of the danger)

Safety Information (describes the danger and informs on how to proceed)

Table 1: Pictographs used in these Instructions

Icon		Signal Words	
	Warning of hazardous electrical voltage	DANGER!	Warns of impending danger . Consequences if disregarded: Death or severe injuries.
	Warning of a general danger	WARNING!	Warns of potential, very hazardous situations . Consequences if disregarded: Death or severe injuries.
	Warning of damage to equipment	STOP!	Warns of potential damage to material and equipment . Consequences if disregarded: Damage to the controller/drive or its environment.
	Information	NOTE	Designates a general, useful note. If the note is observed then handling the controller/drive system is made easier.

Related Documents

The documentation listed in Table 2 contains information relevant to the operation and programming of the PositionServo drive. To obtain the latest documentation, visit the Technical Library at <http://www.lenzeamericas.com>.

Table 2: Reference Documentation

Document #	Description
S94H201	PositionServo (with MVOB) User Manual
PM94H201	PositionServo (with MVOB) Programming Manual
P94MOD01	Position Servo ModBus RTU and ModBus TCP/IP
P94CAN01	PositionServo CANopen Communications Reference Guide
P94DVN01	PositionServo DeviceNet Communications Reference Guide
P94ETH01	PositionServo EtherNet/IP Communications Reference Guide
P94PFB01	PositionServo PROFIBUS DP Communications Reference Guide

1. Introduction

1.1 Definitions

Included herein are definitions of several terms used throughout this programming manual and the PositionServo user manual.

PositionServo: The PositionServo is a programmable digital drive/motion controller, that can be configured as a stand alone programmable motion controller, or as a high performance torque, velocity or position amplifier for centralized control systems. The PositionServo family of drives includes the 940 Encoder-based drive and the 941 Resolver-based drive.

MotionView: MotionView is a universal communication and configuration software that is utilized by the PositionServo drive family. Starting with revision 4.xx, drives will have MotionView OnBoard (MVOB) built into the drive. MotionView has an automatic self-configuration mechanism that recognizes what drive it is connected to and configures the tool set accordingly. The MotionView platform is divided up into three sections or windows, the “Parameter Tree Window”, the “Parameter View Window” and the “Message Window”. Refer to Section 1.3 for more detail.

MotionView OnBoard (MVOB): MotionView OnBoard is the embedded version of MotionView software in PositionServo drives with a part number ending in ES, RS, EM or RM.

SimpleMotion Language (SML): SML is the programming language utilized by MotionView. The SML interface within the MotionView software provides a very flexible development environment for creating solutions to motion applications. The SML programming statements allow the programmer to create complex and intelligent motion, process I/O, perform complex logic decision making, execute program branching, utilize timed event processes, as well as a number of other functions common to the majority of motion control and servo applications.

User Program (or Indexer Program): This is the SML program, developed by the user to describe the programmatic behavior of the PositionServo drive. The User Program can be stored in a text file on your PC as well as in the PositionServo’s EPM memory. The User Program needs to be compiled (translated) into binary form with the aid of the MotionView Studio tools before the PositionServo can execute it.

MotionView Studio: MotionView Studio is the front end programming interface of the MotionView platform. It is a tool suite containing all the software tools needed to program and debug the PositionServo. These tools include a full-screen text editor, a program compiler, status and monitoring utilities, an online oscilloscope and a debug function that allows the user to step through the program during program development.



WARNING!

- Hazard of unexpected motor starting! When using the MotionView software, or otherwise remotely operating the PositionServo drive, the motor may start unexpectedly, which may result in damage to equipment and/or injury to personnel. Make sure the equipment is free to operate safely, and that all guards and covers are in place to protect personnel.
- Hazard of electrical shock! Circuit potentials are up to 480 VAC above earth ground. Avoid direct contact with the internal printed circuit boards or with circuit elements to prevent the risk of serious injury or fatality. Disconnect incoming power and wait 60 seconds before servicing drive. Capacitors retain charge after power is removed.



NOTE

To run MotionView OnBoard (MVOB) on a Mac OS, run the PC emulation tool first.

1.2 Programming Flowchart

MotionView utilizes a BASIC-like programming structure referred to as SimpleMotion Programming Language (SML). SML is a quick and easy way to create powerful motion applications.

With SML the programmer describes his system's motion, I/O processing and process flow using the SML structured code. The programming language includes a full set of arithmetic and logical statements that allow the user to perform mathematical calculations and comparisons of variables and apply the results within their application.

Before the PositionServo drive can execute the user's program, the program must first be compiled (translated) into binary machine code, and downloaded to the drive. Compiling the program is done by selecting the [Compile] button from the toolbar located within the indexer program folder. The user can also compile and download the program at the same time by selecting the [Load W Source] button from the toolbar. Once downloaded, the compiled program is stored in both the PositionServo's EPM memory and the internal flash memory. Figure 1 illustrates the flow of the program preparation process.

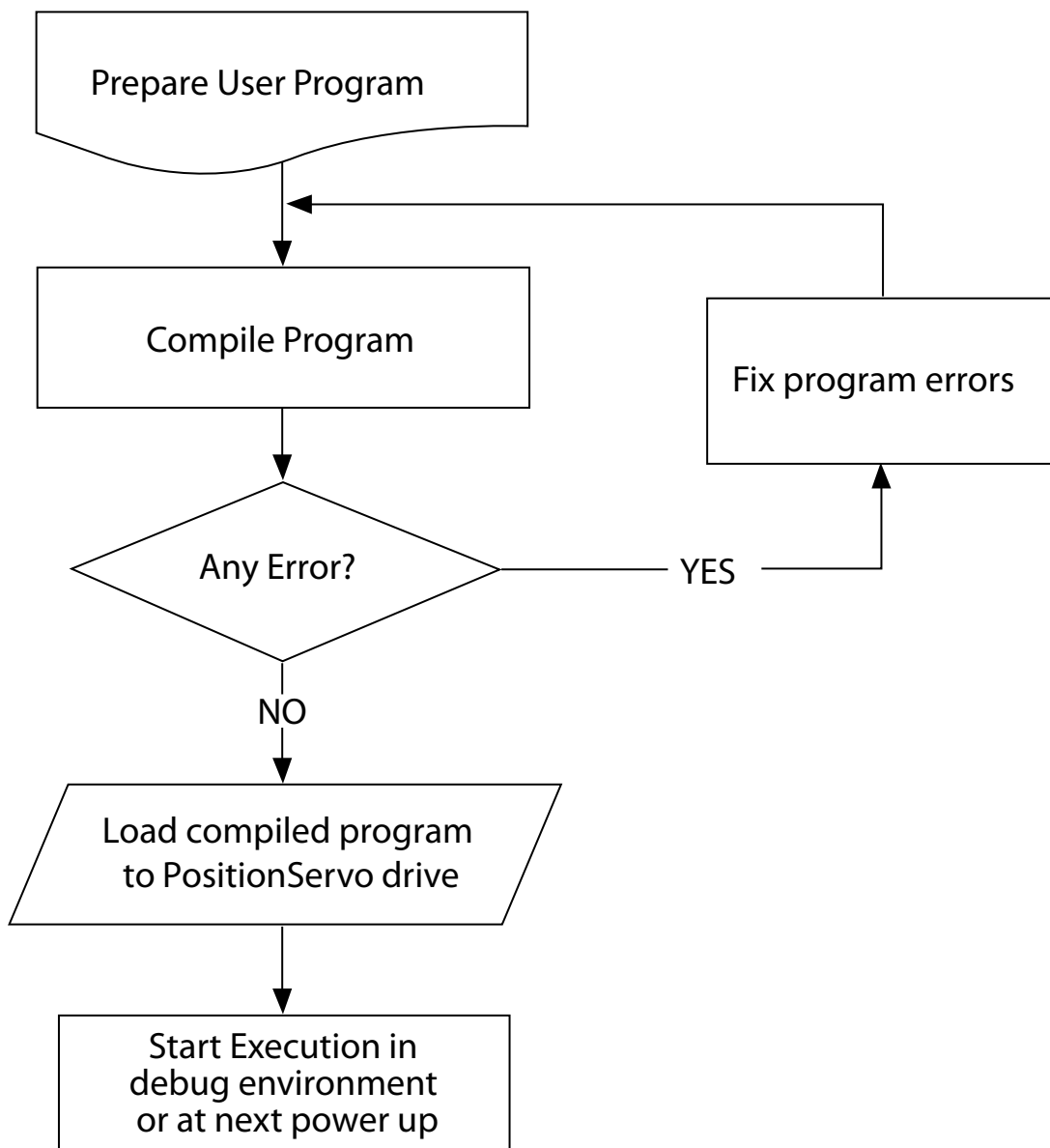


Figure 1: Program Preparation

Introduction

1.3 MotionView / MotionView Studio

There are two versions of MotionView Software. The current version of MotionView resides inside the drive's memory and is referred to as "MotionView on Board" or MVOB. Previous versions were supplied as a PC-installed software package and were referred to simply as MotionView. This manual refers only to the MotionView OnBoard software. MVOB drives are identified by the model number ending in either an 'S' or an 'M'.

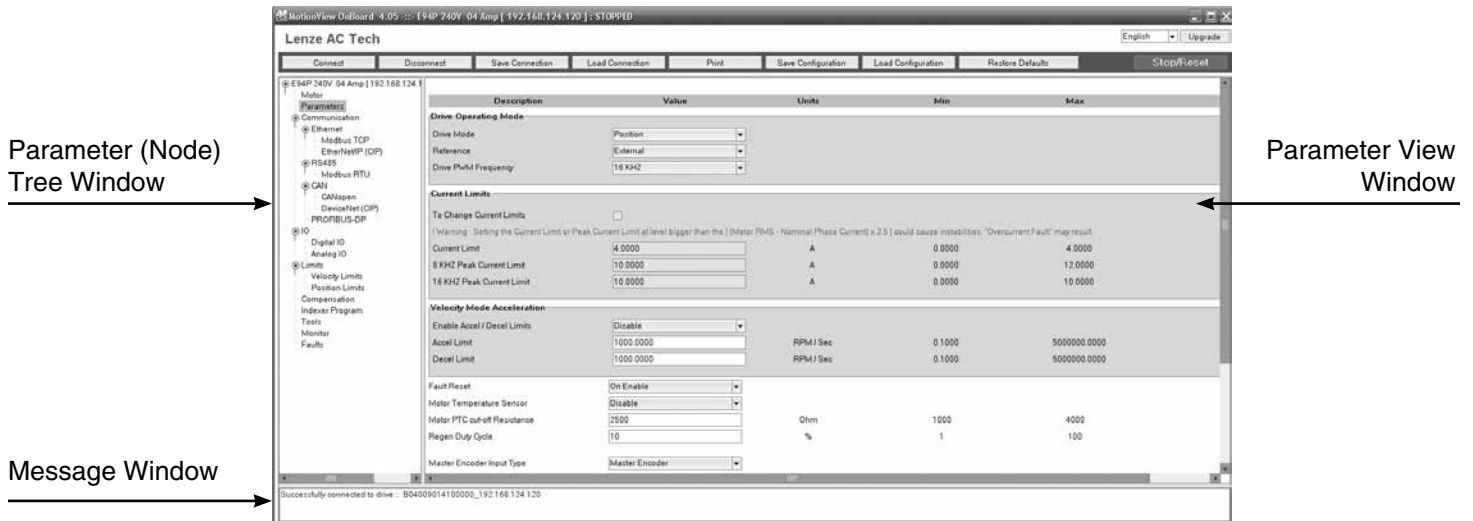


Figure 2: MotionView OnBoard Parameters Display

MotionView is the universal programming software used to communicate with and configure the PositionServo drive. The MotionView platform is segmented into three windows. The first window is the "Parameter Tree Window". This window is used much like Windows Explorer. The various parameter groups for the drive are represented here as folders or files. Once the desired parameter group file is selected, all of the corresponding parameters within that parameter group will appear in the second window, the "Parameter View Window". The user can then enable, disable or edit drive features or parameters from the "Parameter View Window". The third window is the "Message Window". This window is located at the bottom of the screen and will display communication status and errors.



NOTE

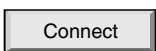
To run MotionView OnBoard (MVOB) on a Mac OS, run the PC emulation tool first.

1.3.1 Main Toolbar

The most commonly used functions of MotionView are accessible via the Main Toolbar as illustrated in Figure 3. If a function icon is greyed out that denotes the function is presently unavailable. A function may be unavailable because a drive is not physically connected to the network or the present set-up and operation of the drive prohibits access to that function. Use the pull-down menu in the top right-hand corner to select the language. [English] is the default language.



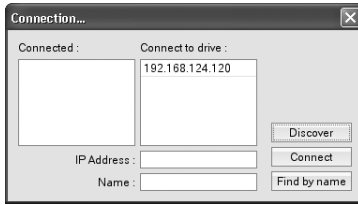
Figure 3: Main Toolbar



Build a connection list of the drive(s) to communicate with on the network. Build the connection list by using any one of these three methods:

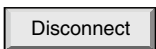




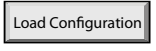
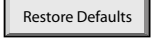


Introduction

[Discover] button automatically discovers all drives on the network that are available for connection. Once drives have been discovered they are listed in the 'Connect to drive' list box. To connect one or more drives highlight their IP address in this window and press the [Connect] button. The [Ctrl] key on the keyboard can be used to select multiple drives for connection.



If the IP address of the drive to be connected is known, enter it in the IP Address dialog box and then select [Connect] to access the drive.

If a drive has previously been assigned a name (or text label) within its "Drive Name" parameter then this name can be used to subsequently connect to that drive. Enter the drive name into the "Name" dialog box and select [Find by name]. The IP address for that drive will then appear in the "Connect To Drive" list. The drive can now be connected by highlighting the IP address and pressing the [Connect] button.

-  Terminate connection to the drive selected (highlighted) in the Parameter (Node) Tree.
-  Save the connection parameters for all drives currently listed in the Parameter (Node) Tree window. This function saves MVOB communications setup for the project only (for quick reconnect of all project drives at a later date), it does not save the individual parameter and programming configuration of each drive.
-  Connect (Reconnect) to project. Opens a previously saved connection file and automatically connects to all drives listed within that file (provided they are available).
-  Print a configuration report for the currently selected drive, containing all parameter set-up and programming information.
-  Saves the configuration file of the selected drive. All parameters, indexing program, I/O configuration and compensation gains will be saved within this file.
-  Load a saved configuration to the drive.
-  Set drive parameters back to factory default values. Note: has no effect on motor data or drive IP address.
-  Stops the drive execution and resets the drive.
-  Launches firmware upgrade utility.

1.3.2 Program Toolbar

To view the Program Toolbar, click on the [Indexer Program] folder in the Parameter (Node) Tree. This section contains a brief description of the programming tools: Compile, Load with Source, Load Without Source, Reload, Export, Import, Run, Reset, Pause, Step, Step Over and Clear. For detailed descriptions of the program toolbar functions refer to paragraph 1.4.

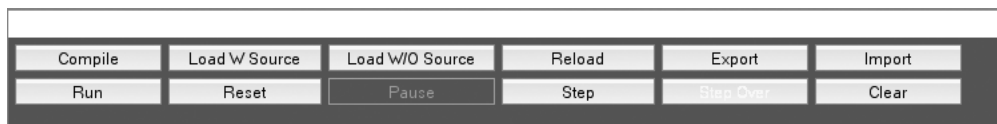
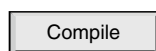
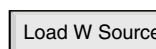
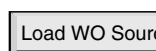


Figure 4: Program Toolbar

-  Perform compilation and check for syntax errors for the indexer program currently selected in the List View window.
-  Compile and Load Binary program and text source file to the PositionServo drive listed in the Parameter (Node) Tree.
-  Compile and Load Binary program only (excluding text source file) to the PositionServo drive listed in the Parameter (Node) Tree.

Introduction

- Reload

Reload the text source file presently stored in the selected drive back into the MotionView Indexer program folder.
- Export

Export text source file (User program). Saves a copy of the program from the Indexer Program folder as a text file on the PC.
- Import

Import text source file (User program). Loads a program from a text file stored on the PC to the Indexer Program folder.
- Run

Start/Continue Program execution. Refer to section 1.4 for full description and prior to operation.
- Reset

Reset Drive. Disable drive, stop program execution, and return program processing to the beginning. Program will **not** restart program execution automatically.
- Pause

Stop program execution on completion of the current statement being executed. **WARNING:** Pause button does not place the drive in a disable state or prevent execution of motion commands waiting on the motion stack.
- Step

Execute each line of code in the program sequentially following on each press of the [Step] button. Include step to instructions contained within subroutines.
- Step Over

Reserved for future use.
- Clear

Clears the Indexer code.



WARNING

“Load W/O Source” will delete the text source file from both the indexer screen and the drive memory. The user must ensure they save a copy of the text source file to their PC before proceeding with this operation.

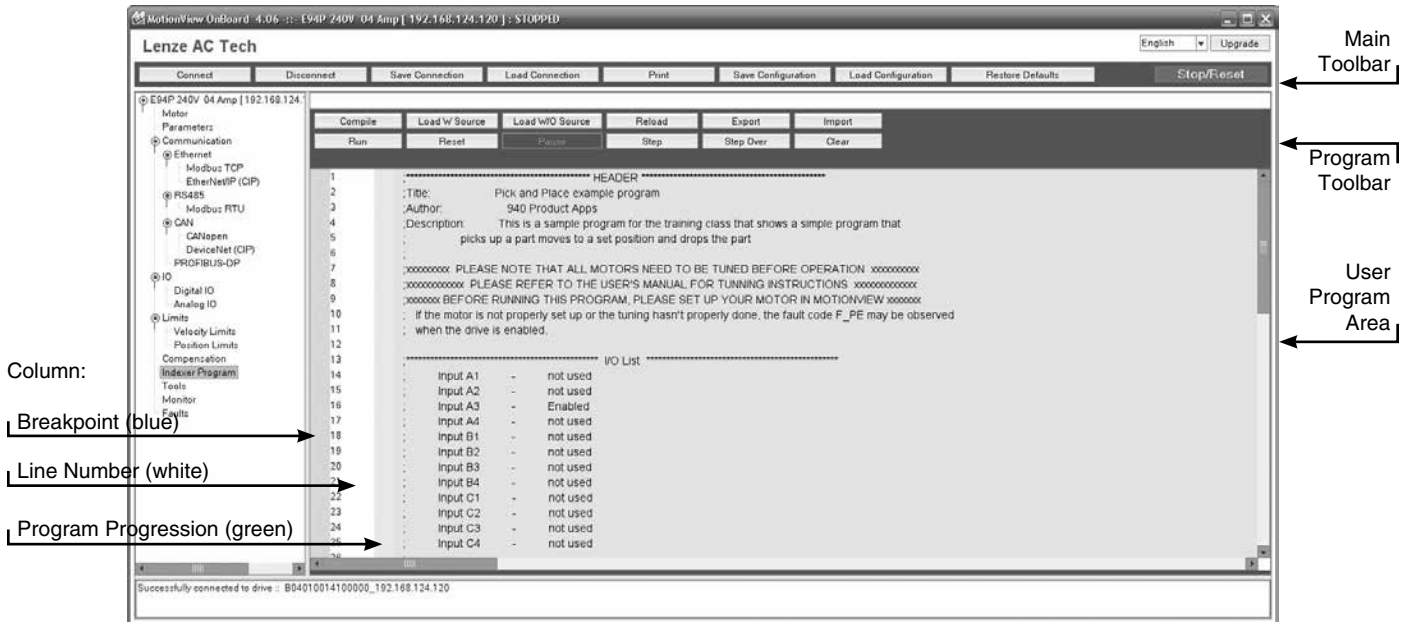


Figure 5: MotionView OnBoard Studio - Indexer Program Display

1.3.3 MotionView Studio - Indexer Program

The MotionView Studio provides a tool suite used by MotionView OnBoard to enter, compile, load and debug the user program. To view and develop the user program, select the [Indexer Program] folder in the Parameter (Node) Tree window. Once selected the program text editor screen and program toolbar are displayed. The program displayed in the text editor window is uploaded from the drive when the indexer folder is selected, any data not compiled to the drive or saved to PC file will be lost once this window is exited. Click anywhere in the Parameter View Window to edit the Indexer program.

Common Programming Actions

Load User program from the PC to the MotionView Indexer Program folder text editor window.

- Select [**Indexer Program**] in the Parameter (Node) Tree.
- Select [**Import**] on the program toolbar.

Select the program to import from the PC folder where it is located. This procedure loads the program from the file to the editor window. It doesn't load the program to the drive's memory.

Compile program and **Load** to the drive

- Select [**Indexer Program**] in the Parameter (Node) Tree.
- Select [**Load WO Source**] on the program toolbar to compile the program and load the compiled binary code to the PositionServo drive. A copy of the original source code is not stored to the drive's memory and therefore cannot be obtained from the drive subsequently. This feature can be used to protect the program from copy but the programmer must ensure that a copy of the program is safely stored to his PC.
- Select [**Load W Source**] on the program toolbar to compile the program and load the source code and the compiled binary file to the PositionServo drive. The original source code contained in the drive can be viewed whenever the drive is accessed through MotionView and the Indexer Program folder is opened.
- Select [**Compile**] to check syntax errors without loading the program to the drive. If the compiler finds any syntax error, further compilation is halted. Errors are reported in the message window at the bottom of the screen.

Save User program from MotionView to PC.

- Select [**Indexer Program**] in the Parameter (Node) Tree.
- Select [**Export**] on the program toolbar.

Provide a name and folder location for the source file to be stored under. The program will be saved to the Windows "My Documents" folder by default.

Run User program in drive.

- Select [**Indexer Program**] in the Parameter (Node) Tree.
- Select [**Run**] on the program toolbar. Note all warnings contained within product manuals prior to running the user program.

Step Through the User program.

- Select [**Indexer Program**] in the Parameter (Node) Tree.
- Select [**Step**] on the program toolbar.

If [Step] is selected, the drive will execute the program one step at a time including subroutines. For the Step function to be used the drive must be in a 'Indexer program Stopped' condition. If Indexer program is running then Step functions are disabled. If the user program displayed in the Indexer program window does not match the program currently residing within the drive (last compiled and downloaded) then Step functions are also disabled.

Statement execution is tracked by a pointer located in the progression column of the program editor. The pointer indicates the next line of code to be executed. At each Step the pointer will disappear until the statement has been fully executed and will then reappear at the next statement.

Introduction

Set **Breakpoint(s)** in the program

- Select **[Indexer Program]** in the Parameter (Node) Tree.
- Place the cursor in the 'Breakpoint' Column next to the line number on which a breakpoint is to be added.
- Right-click and select Add Breakpoint (or Clear Breakpoint).

A convenient way to debug a user program is to insert breakpoints at critical junctions throughout the program. These breakpoints are marked by a red plus sign (+) and stop the drive from executing further program statements once a breakpoint is reached, but do not disable the drive and the position variables. Once the program has stopped, the user can continue to run the program, step through the program or reset the program.

Pause program execution

- Select **[Indexer Program]** in the Parameter (Node) Tree.
- Select **[Pause]** on the program toolbar.

The program will stop after completing the current statement. Select **[Run]** or use Step functions to resume the program from the same point.



IMPORTANT!

The [Pause] button only stops the execution of the program code. It does **not** stop motion or disable the drive.

Reset Program execution

- Select **[Indexer Program]** in the Parameter (Node) Tree.
- Select **[Reset]** on the program toolbar.

The program will be reset and the drive will be disabled. Variables within the drive are not cleared (reset) when program execution is reset. It is important that any variables used by the programmer are set to safe values at the start of the user program.

1.4 Programming Basics

The user program consists of statements which when executed will not only initiate motion but will also process the drives I/O and make decisions based on drive variables, calculations, and comparisons. Before motion can be initiated, certain drive and I/O parameters must be configured. When first getting started with PositionServo programming it is recommended that the following parameters be set within MotionView parameter folders to aid initial program creation.

Parameter setup

Select **[Parameter]** folder in the Parameter (Node) Tree window and set the following parameters.

Set the Drive Operating Mode:

- Select **[Drive mode]** from the Parameter View Window.
- Select **[Position]**, **[Velocity]**, or **[Torque]** from the drop down menu depending on the mode the drive is to be operated in. In order to execute the examples contained in this section of the manual the drive will need to be in **[Position]** mode.

Set the [Reference] to [Internal]:

- Select **[Reference]** from the Parameter View Window.
- Select **[Internal]** from the pull down menu to select the user program as the source of the Torque, Velocity, or Position Reference.

Select **[Digital IO]** folder in the Parameter (Node) Tree window and set the following parameter.

Set the [Enable switch function] to [Inhibit]:

- Select **[Enable switch function]** from the Parameter View Window.
- Select **[Inhibit]** from the menu to allow the user program control of the enable / disable status of the drive. Input A3 will now act as a hardware inhibit.

I/O Configuration

Input A3 is the Inhibit/Enable special purpose input. Refer to the PS User Manual (S94H201) for more information. Before executing any motion related statements, the drive must be enabled by executing "ENABLE" statement. "ENABLE" statement can only be accepted if input A3 is made. If at any time while drive is enabled A3 deactivates then the fault "F36" ("Drive Disabled") will result. This is a hardware safety feature.

Introduction

Basic Motion Program

Select **[Indexer program]** from the Parameter (Node) Tree. The Parameter View window will display the current User Program stored in the drive. Note that if there is no valid program in the drive's memory the program editing window will be empty.



WARNING!

This program will cause motion. The motor should be disconnected from the application (free to rotate) or if a motor is connected, the shaft must be free to spin 10 revs forward and reverse from the location of the shaft at power up. Also, the machine must be capable of 10 RPS and an accel / decel of 5 RPSS.

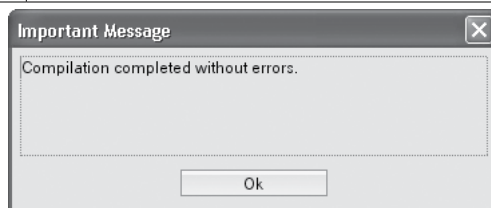
In the program area, clear any existing program (save if required) and replace it with the following program:

Program	Compile	Resultant MotionView OnBoard Messages
<pre> UNITS=1 ACCEL = 5 DECEL = 5 MAXV = 10 ENABLE MOVED 10 MOVEDISTANCE -10 END </pre>	<p><input type="button" value="Compile"/></p> <p>Enter the program, then select [Compile] on the toolbar. After compilation is done, a "Compilation Error" message will appear.</p>	

Click **[OK]** to dismiss the "Compilation error" dialog box. The cause of the compilation error will be displayed in the Message window, located at the bottom of the MotionView OnBoard screen. MotionView will also highlight the program line where the error occurred. In the example program above, in the green 'Program Progression' column there is a red box next to the "MOVEDISTANCE -10" statement.

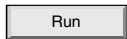
The problem in this example is that **"MOVEDISTANCE"** is not a valid command. Change the text **"MOVEDISTANCE"** to **"MOVED"**.

Program	Compile	Resultant MotionView OnBoard Messages
<pre> UNITS=1 ACCEL = 5 DECEL = 5 ENABLE MOVED 10 MOVED -10 END </pre>	<p><input type="button" value="Compile"/></p> <p>After editing the program, select [Compile] on the program toolbar. After compilation is done, the "Compilation Complete" message box should appear.</p>	



Introduction

The program has now been compiled without errors. Select [Load W Source] to load the program to the drive's memory. Click [OK] to dismiss the dialog box.



To **Run** the program, input A3 must be active to remove the hardware inhibit. Select the [**Run**] icon on the program toolbar. The drive will start to execute the User Program. The motor will spin 10 revolutions in the CCW direction and then 10 revolutions in the CW direction. After all the code has been executed, the program will stop and the drive will stay enabled.



To **Restart** the program, select the [**Reset**] icon on the program toolbar. This will disable the drive and reset the program to execute from the start. The program does not run itself automatically. To run the program again, select the [**Run**] icon on the toolbar.

Program Layout

When developing a program, structure is very important. It is recommended that the program be divided up into the following 7 segments:

- Header:** The header defines the title of the program, who wrote the program and description of what the program does. It may also include a date and revision number.
- I/O List:** The I/O list describes what the inputs and outputs of the drive are used for. For example input A1 might be used as a Start Switch.
- Init & Set Var:** Initialize and Set Variables defines the drives settings and system variables. For example here is where acceleration, deceleration and max speed might be set.
- Events:** An Event is a small program that runs independently of the main program. This section is used to define the Events.
- Main Program:** The Main Program is the area where the main process of the drive is defined.
- Sub-Routines:** This is the area where all sub-routines should reside. These routines will be called out from the Main Program with a GOSUB command.
- Fault Handler:** This is the area where the Fault Handler code resides. If the Fault handler is utilized, then this code will be executed when the drive detects a fault condition.

The following is an example of a Pick and Place program divided up into the above segments.

```
***** HEADER *****
;Title:          Pick and Place example program
;Author:         Lenze - AC Technology
;Description:    This is a sample program showing a simple sequence that
;               picks up a part, moves to a set position and places the part
;***** I/O List *****
;   Input A1    -   not used
;   Input A2    -   not used
;   Input A3    -   Enable Input
;   Input A4    -   not used
;   Input B1    -   not used
;   Input B2    -   not used
;   Input B3    -   not used
;   Input B4    -   not used
;   Input C1    -   not used
;   Input C2    -   not used
;   Input C3    -   not used
;   Input C4    -   not used
;   Output 1    -   Pick Arm
;   Output 2    -   Gripper
;   Output 3    -   not used
;   Output 4    -   not used
```

Introduction

```
;***** Initialize and Set Variables *****
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
;V1 =
;V2 =
;***** Events *****
;Set Events handling here
;No events are currently defined in this program
;***** Main Program *****
RESET_DRIVE:           ;Place holder for Fault Handler Routine
WAIT UNTIL IN_A3:      ;Make sure that the Enable input is made before continuing
ENABLE                 ;Enable output from drive to motor
PROGRAM_START:        ;Place holder for main program loop
MOVEP 0                ;Move to Pick position
OUT1 = 1               ;Turn on output 1 to extend Pick arm
WAIT TIME 1000         ;Delay 1 sec to extend arm
OUT2 = 1               ;Turn on output 2 to Engage gripper
WAIT TIME 1000         ;Delay 1 sec to Pick part
OUT1 = 0               ;Turn off output 1 to Retract Pick arm
MOVED -10              ;Move 10 REVs to Place position
OUT1 = 1               ;Turn on output 1 to extend Pick arm
WAIT TIME 1000         ;Delay 1 sec to extend arm
OUT2 = 0               ;Turn off output 2 to Disengage gripper
WAIT TIME 1000         ;Delay 1 sec to Place part
OUT1 = 0               ;Retract Pick arm
GOTO PROGRAM_START    ;Loop back and continuously execute main program loop
END

;***** Sub-Routines *****
Enter Sub-Routine code here

;***** Fault Handler Routine *****
;      Enter Fault Handler code here
ON FAULT                ;No Fault Handler is currently defined in this program
ENDFAULT
```

Saving Configuration File to PC

The “Configuration File” consists of all the parameter settings for the drive, as well as the User Program. Once you are done setting up the drive’s parameters and have written your User Program, you can save these setting to your computer. To save the settings, select **[Save All]** from the **Main** toolbar. Then simply assign your configuration file a name, (e.g. Basic Motion), and click [Save] in the dialog box. The configuration file has a “dcf.xml” extension and by default will be saved to the “My Documents” folder.

Loading Configuration File to the Drive

There are times when it is helpful to import a a complete set-up or drive configuration to another drive. To load the configuration file from the PC to the drive, select **[Load All]** from the **Main** toolbar. Select the configuration file to load and click [Open] in the dialog box. MotionView will open the selected configuration file, set all parameters within the drive to the values contained within that file, and then extract, compile and download the saved user program. When the process is complete the [Compilation Complete] dialog box will appear.

Introduction

Click [OK] to dismiss this dialog box. MotionView will then load the selected file to the drive. When complete, a second dialog box will appear indicating 'indexer program compiled and downloaded successfully'. Click [OK] too clear this message. Load of the configuration file is now complete.

Motion source (Reference)

The PositionServo can be set up to operate in one of three modes: Torque, Velocity, or Position. The drive must be given a command relative to its mode of operation before it can initiate any motion. The source for commanding this motion is referred to as the "Reference". With the PositionServo you have two methods of commanding motion, or two types of References. When the drive's reference signal is from an external source, for example a PLC or Motion Controller, it is referred to as an External Reference. When the drive is being given its reference from the User program or through one of the system variables it is referred to as an Internal Reference.

Table 3: Setting the Reference

Mode	"Reference" Parameter Setting	
	External	Internal
Torque	Analog input AIN1	System variable "IREF"
Velocity	Analog input AIN1	System variable "IREF"
Position	Step/Direction Inputs Master Encoder Pulse Train Inputs	User Program/Interface (Trajectory generator)

Units

All motion statements in the drive work with User units. The statement on the first line of the test program, UNITS=1, sets the relationship between programming units and motor revolutions. For example, if UNITS=0.5 the motor will turn 1/2 of a revolution when commanded to move 1 Unit. When the UNITS variable is set to zero, programming units for motion will be in motor feedback pulses (User units set to 1 divided into motor feedback pulses).

Time base

Time base for motion is always in seconds i.e. all time-related values are set in USER UNITS/SEC. Time Base for programming statements (such as wait statements) are always in milliseconds.

Enable/Disable/Inhibit drive

Set "Enable switch function" to "Run".

When the "Enable switch function" parameter is set to Run, and the Input A3 is made, the drive will be enabled. Likewise, toggling input A3 to the off state will disable the drive.

- Select [IO] then [Digital IO] from the Parameter Tree Window.
- Select "Enable switch function" from the Parameter View Window.
- Select "Run" from the drop down menu. This setting is primarily used when operating without any user program in torque or velocity mode or as position follower with Step&Direction/Master Encoder reference.

Set "Enable switch function" to "Inhibit".

In the example of the Enable switch function being set to Run the decision on when to enable and disable the drive is determined by the logic status of input A3 (typically controlled by an external device, PLC or Motion controller). The PositionServo's User Program allows the programmer to define (control) within their program the enable and disable of the drive through execution of program statements. The drive will execute the User Program whether the drive is enabled or disabled, however if a motion statement is executed while the drive is disabled, an F27 fault will occur. If the user program commands the drive to enable and Input A3 (hardware enable) is not present or Input A3 is removed and the drive is enabled through programming then the drive will trip on Fault 36.

Introduction

When the “**Enable switch function**” parameter is set to **Inhibit**, and Input A3 is on, the drive will be disabled and remain disabled until the ENABLE statement is executed by the User Program.

- Select **[IO] then [Digital IO]** from the Parameter Tree Window.
- Select “**Enable switch function**” from the Parameter View Window.
- Select “**Inhibit**” from the popup menu.

Faults

When a fault condition has been detected by the drive, the following actions will occur:

- Drive will Immediately be placed in a Disabled Condition.
- Motion Stack will be flushed of any Motion Commands
- Execution of the user program will be terminated and program control will be handed over to the Fault Handler section. If no Fault handler is described then program execution will terminate. See fault handler section.
- A fault code defining the nature of the drive trip will be written to the DFAULTS system variable and can be accessed by the fault handler. Refer to section 2.13 for a list of fault codes.
- The fault code will be displayed on the drive display.
- Dedicated Ready/Enabled output will turn off, provided drive was in enable state prior to fault detection.
- Any Output with assigned special function “Fault” will turn on.
- Any Output with assigned special function “ready/enabled” will turn off, provided drive was in enable state prior to fault detection
- The “enable” status indicator on the drive display will turn off indicating drive in disabled state.

Clearing a fault condition can be done in one of the following ways:



- Select the **[Reset]** button from the toolbar.
- Execute the **RESUME** statement at the end of the Fault Handler routine (see Fault Handler example). This permits the continuation of program execution at the discretion of the programmer and when the fault does not present an issue to the safety or integrity of the system.
- Send “Reset” command over the Host Interface.
- Cycle power (hard reset).

Fault Handler

The Fault Handler is a code segment that will be executed immediately on the drive detecting a fault condition. The fault handler allows the programmer to analyze the type of fault and (when necessary) define a recovery process for the drive Full stop. While the drive is executing the Fault Handler Routine the drive is disabled and therefore will not be able to detect any additional faults that might occur. Fault handler code is the drive’s first reaction to a fault condition. While it executes, the drive will not respond to any I/O, interface commands or program events. Therefore the user should use the fault handler to manipulate time critical and safety related I/O and variables and then exit the Fault Handler Routine either by executing a “**RESUME**” statement or by executing the EndFault statement and ending program execution. The Resume statement permits program execution to leave the fault handler and resume back in the main program section of the user code. Use the Resume statement to jump back to a section of the main program that designates the recovery process for the fault. Wait statements within the fault handler for I/O state change or for interface command is not allowed. If a wait statement is required (for example from a fault reset input) then this must be done subsequent to the Resume command when program execution is handed back to the main program.

Without Fault Handler

To simulate a fault, restart the Pick and Place example program. While the program is running, switch the ENABLE input IN_A3 to the off state. This will cause the drive to generate an F_36 fault (Hardware disable while drive enabled in inhibit mode) and put the drive into Fault Mode. While the drive is in Fault Mode, any digital output currently active will remain active and any output deactivated will remain deactivated, excluding the dedicated ready output and any output that has been assigned pre-defined functionality. The program execution will stop and any motion commands will be terminated.

Introduction

With Fault Handler

Add the following code to the end of your sample program. When the program is running, switch the ENABLE input IN_A3, to the off state. This will cause the drive to generate an F_36 fault ((Hardware disable while drive enabled in inhibit mode) and put the drive into a Fault Mode. From this point the Fault Handler Routine will take over.

```
F_PROCESS:
WAIT UNTIL IN_A4==1 ;Wait until reset switch is made
WAIT UNTIL IN_A4==0 ;and then released before
GOTO RESET_DRIVE ;returning to the beginning of the program
END
;***** Sub-Routines *****
Enter Sub-Routines here;
;***** Fault Handler Routine *****
ON FAULT ;Statement starts fault handler routine
;Motion stopped, drive disabled, and events no longer
;scanned while executing the fault handler routine.
OUT2 = 0 ;Output 1 off to Disengage gripper.
;This will drop the part in the gripper
OUT1 = 0 ;Retract Pick arm to make sure it is up and out of the way
RESUME F_PROCESS ;program restarts from label F_PROCESS
ENDFAULT ;fault handler MUST end with this statement
```



NOTE


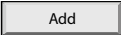
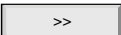
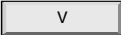
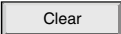
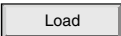
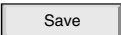
The following statements can not be used inside the Fault Handler Routine:

- ENABLE
- WAIT
- MOVE
- MOVED
- MOVEP
- MOVEDR
- MOVEPR
- MDV
- MOTION SUSPEND
- MOTION RESUME
- GOTO, GOSUB
- JUMP
- VELOCITY ON/OFF
- WHILE / ENDWHILE
- DO / UNTIL
- EVENT (ON, OFF)
- EVENTS (ON, OFF)
- HOME
- HALT
- STOP MOTION (QUICK)

Refer to section 2.1 for additional details and the Language Reference section for the statement "ON FAULT/ENDFAULT".

1.5 Using Advanced Debugging Features

To debug a program or view the I/O, open the Diagnostic panel by clicking on the [Tools] in the Parmeter (Node) Tree list then click on the [Parameter & I/O View] button. The Diagnostic panel will open. This panel allows the programmer to monitor and set variables, and to view status of drive digital inputs and outputs.

-  Use the up [^] button to move the blue highlighted bar up through the variable list and select a parameter
-  Use the [Add] button to open the Parameters dialog box. Select the variable(s) to add by clicking on the box adjacent to the variable #. When finished selecting variables, click [Add] in the Parameter dialog box to add these variables to the watch window.
-  Use the right arrow button to remove highlighted variable from the watch window.
-  Use the down [V] button to move the blue highlighted bar down through the variable list and select a parameter
-  Use the [Clear] button to clear all the parameters listed in the watch window.
-  Use the [Load] button to load a set of previously saved variables to the watch window.
-  Use the [Save] button to save the configuration of variables listed in the watch window to a file on the PC. Configuration can then easily be restored using the [Load] button.

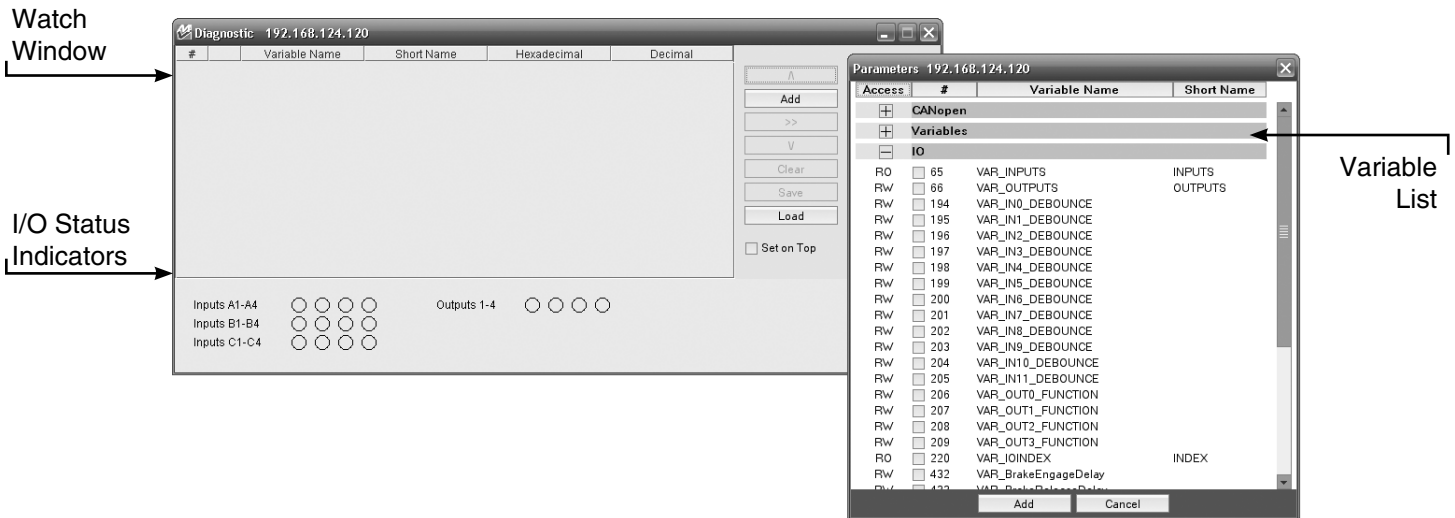


Figure 6: Variable Diagnostic Display



NOTE

Write-only variables cannot be read. Attempts to either display a write-only variable in the diagnostic panel or to read a write-only variable via network communications can show erroneous data.

1.6 Inputs and Outputs

Analog Input and Output

- The PositionServo has two analog inputs. These analog inputs are utilized by the drive as System Variables and are labeled “AIN1” and “AIN2”. Their values can be directly read by the User Program or via a Host Interface. Their value can range from -10 to +10 and correlates to ±10 volts analog input.
- The PositionServo has one analog output. This analog output is utilized by the drive as a System Variable and is labeled “AOUT”. It can be directly written by the User Program or via a Host Interface. Its value can range from -10 to +10 which correlates to ± 10 volts analog input.



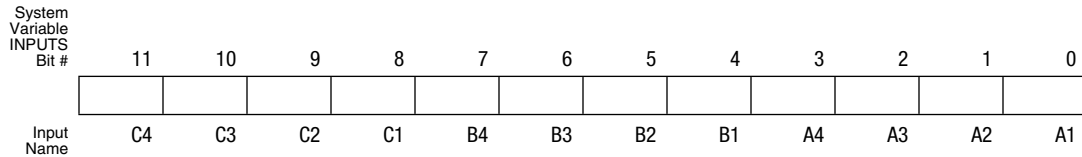
NOTE

If an analog output is assigned to any special function from MotionView, writing to AOUT from the User Program will have no effect. If an analog output is set to “Not assigned” then it can be controlled by writing to the AOUT variable.

Introduction

Digital Inputs

- The PositionServo has twelve digital inputs that are utilized by the drive for decision making in the User Program. Example uses: travel limit switches, proximity sensors, push buttons and hand shaking with other devices.
- Each input can be assigned an individual debounce time via MotionView. From the **Parameter Tree**, select [IO]. Then select the [**Digital Input**] folder. The debounce times will be displayed in the **Parameter View Window**. Debounce times can be set between 0 and 1000 ms (1ms = 0.001 sec). Debounce times can also be set via variables in the user program.
- The twelve inputs are separated into three groups: A, B and C. Each group has four inputs and share one common: Acom, Bcom and Ccom respectfully. The inputs are labeled individually as **IN_A1 - IN_A4, IN_B1 - IN_B4 and IN_C1 - IN_C4**.
- In addition to monitoring each input individually, the status of all twelve inputs can be represented as one binary number. Each input corresponds to 1 bit in the INPUTS system variable. Use the following format:



- Some inputs can be configured for additional predefined functionality such as Travel Limit switch, Enable input, and Registration input. Configuration of these inputs is done from MotionView or through variables in the user program. Input special functionality is summarized in the table below and in the following sections. Table 4 summarizes the special functions for the inputs.

Table 4: Input Functions

Input Name	Special Function
Input A1	Negative limit switch
Input A2	Positive limit switch
Input A3	Inhibit/Enable input
Input A4	N/A
Input B1	N/A
Input B2	N/A
Input B3	N/A
Input B4	N/A
Input C1	N/A
Input C2	N/A
Input C3	Registration sensor input
Input C4	N/A

The current status of the drive's inputs is available to the programmer through dedicated System Flags or as bits of the System Variable INPUTS.

Read Digital Inputs

The Pick and Place example program has been modified below to utilize the “WAIT UNTIL” statement in place of the “WAIT TIME” statement. **IN_A1** and **IN_A4** will be used as proximity sensors to detect when the pick and place arm is extended and when it is retracted. When the arm is extended, **IN_A1** will be in an ON state and will equal “1”. When the arm is retracted, **IN_A4** will be in an ON state and will equal “1”.

```
***** Main Program *****
RESET_DRIVE:           ;Place holder for Fault Handler Routine
WAIT UNTIL IN_A3       ;Make sure that the Enable input is made before continuing
ENABLE
OUT1 = 0               ;Initialize Pick Arm - Place in Retracted Position
WAIT UNTIL IN_A4==1    ;Check Pick Arm is in Retracted Position
PROGRAM_START:
MOVEP 0                ;Move to Pick position
OUT1 = 1               ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1    ;Arm extends
OUT2 = 1               ;Turn on output 2 to Engage gripper
WAIT TIME 1000         ;Delay 1 sec to Pick part
OUT1 = 0               ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1    ;Make sure Arm is retracted
MOVED -10              ;Move 10 REVs to Place position
OUT1 = 1               ;Turn on output 1 on to extend Pick arm
WAIT UNTIL IN_A1==1    ;Arm is extended
OUT2 = 0               ;Turn off output 2 to Disengage gripper
WAIT TIME 1000         ;Delay 1 sec to Place part
OUT1 = 0               ;Retract Pick arm
WAIT UNTIL IN_A4==1    ;Arm is retracted
GOTO PROGRAM_START
END
```

Once the above modifications have been made, export the program to file and save it as “Pick and Place with I/O”, then compile, download and test the program.

ASSIGN & INDEX - Using inputs to generate predefined indexes

“INDEX” is a variable on the drive that can be configured to represent a specified group of inputs as a binary number. “ASSIGN” is the command that designates which inputs are utilized and how they are configured.

Below the Pick and Place program has been modified to utilize this “INDEX” function. The previous example program simply picked up a part and moved it to a place location. For demonstration purposes we will add seven different place locations. These locations will be referred to as Bins. What Bin the part is placed in will be determined by the state of three inputs, B1, B2 and B3.

Bin 1	-	Input B1 is made
Bin 2	-	Input B2 is made
Bin 3	-	Inputs B1 and B2 are made
Bin 4	-	Input B3 is made
Bin 5	-	Inputs B1 and B3 are made
Bin 6	-	Inputs B2 and B3 are made
Bin 7	-	Inputs B1, B2 and B3 are made

The “ASSIGN” command is used to assign the individual input to a bit in the “INDEX” variable. ASSIGN INPUT <input name> AS BIT <bit #>

```
***** Initialize and Set Variables *****
ASSIGN INPUT IN_B1 AS BIT 0 ;Assign the Variable INDEX to equal 1 when IN_B1 is made
ASSIGN INPUT IN_B2 AS BIT 1 ;Assign the Variable INDEX to equal 2 when IN_B2 is made
ASSIGN INPUT IN_B3 AS BIT 2 ;Assign the Variable INDEX to equal 4 when IN_B4 is made
```


Introduction

Table 5: Bin Location, Inputs & Index Values

Bin Location	Input state	INDEX Value
Bin 1	Input B1 is made	1
Bin 2	Input B2 is made	2
Bin 3	Inputs B1 and B2 are made	3
Bin 4	Input B3 is made	4
Bin 5	Inputs B1 and B3 are made	5
Bin 6	Inputs B2 and B3 are made	6
Bin 7	Inputs B1, B2 and B3 are made	7

The Main program has been modified to change the end place position based on the value of the “INDEX” variable.

```

;***** Main Program *****
ENABLE
OUT1 = 0 ;Initialize Pick Arm - Place in Retracted Position
WAIT UNTIL IN_A4==1 ;Check Pick Arm is in Retracted Position
PROGRAM_START:
MOVEP 0 ;Move to (ABS) to Pick position
OUT1 = 1 ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1 ;Arm extends
OUT2 = 1 ;Turn on output 2 to Engage gripper
WAIT TIME 1000 ;Delay 1 sec to Pick part
OUT1 = 0 ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==0 ;Make sure Arm is retracted

IF INDEX==1 ;In this area we use the If statement to
GOTO BIN_1 ;check and see what state inputs B1, B2 & B3
ENDIF ;are in.
IF INDEX==2 ; INDEX = 1 when input B1 is made
GOTO BIN_2 ; INDEX = 2 when input B2 is made
ENDIF ; INDEX = 3 when input B1 & B2 are made.
. ; INDEX = 4 when input B3 is made
. ; INDEX = 5 when input B1 & B3 are made.
. ; INDEX = 6 when input B2 & B3 are made.
IF INDEX==7 ; INDEX = 7 when input B1, B2 & B3 are made
GOTO BIN_7 ;We can now direct the program to one of seven
ENDIF ;locations based on three inputs.

BIN_1: ;Set up for Bin 1
MOVEP 10 ;Move to Bin 1 location
GOTO PLACE_PART ;Jump to place part routine
BIN_2: ;Set up for Bin 2
MOVEP 20 ;Move to Bin 2 location
GOTO PLACE_PART ;Jump to place part routine
BIN_7: ;Set up for Bin 7
MOVEP 70 ;Move to Bin 7 location
GOTO PLACE_PART ;Jump to place part routine
PLACE_PART:
OUT1 = 1 ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A4 == 1 ;Arm extends
OUT2 = 0 ;Turn off output 2 to Disengage gripper
WAIT TIME 1000 ;Delay 1 sec to Place part
OUT1 = 0 ;Retract Pick arm
WAIT UNTIL IN_A4 == 0 ;Arm is retracted
GOTO PROGRAM_START
END

```

NOTE: with all digital inputs (B1-B3) off, none of the ‘If’ statements to detect place position are true and the program defaults to placing the part it has picked into bin location 1.



NOTE

Any one of the 12 inputs can be assigned as a bit position within the INDEX variable. Only bits 0 through 7 can be used with the INDEX variable. Bits 8-31 are not used and are always set to 0. Unassigned bits in the INDEX variable are set to 0.

BITS 8-31 (not used)	A1	0	A2	A4	0	0	0	0
----------------------	----	---	----	----	---	---	---	---

Limit Switch Input Functions

Inputs A1 and A2 can be configured as special purpose inputs from the **[Digital IO]** folder in MotionView. They can be set to one of three settings:

- The **“Not assigned”** setting designates the inputs as general purpose inputs which can be utilized by the User Program.
- The **“Fault”** setting will configure A1 and A2 as Hard Limit Switches. When either input is made the drive will be disabled, the motor will come to an uncontrolled stop, and the drive will generate a fault. If the negative limit switch is activated, the drive will display an F-33 fault. If the positive limit switch is activated the drive will display an F32 fault.
- The **“Stop and fault”** setting will configure A1 and A2 as End of Travel limit switches. When either input is made the drive will initiate a rapid stop before disabling the drive and generating an F34 or F35 fault (refer to section 2.15 for details). The speed of the deceleration will be set by the value stored in the **“QDECEL”** System Variable.



NOTE

The “Stop and Fault” function is available in position mode only, (“Drive mode” is set to “Position”). In all other cases, the Stop and Fault function will act the same as the Fault function.

To set this parameter, select the **[IO]** folder from the Parameter Tree. Then select the **[Digital IO]** folder. From the Parameter View Window, use the pull-down menu next to **[Hard Limit Switches Action]** to select the status: Not Assigned, Fault or Stop and Fault.

Digital Outputs Control

- The PositionServo has 5 digital outputs. The **“RDY”** or READY output is dedicated and will only come on when the drive is enabled, i.e. in **RUN** mode. The other outputs are labeled **OUT1 - OUT4**.
- Outputs can be configured as Special Purpose Outputs. If an output is configured as a **Special Purpose Output** it will activate when the state assigned to it becomes true. For example, if an output is assigned the function **“Zero speed”**, the assigned output will come on when the motor is not in motion. To configure an output as a Special Purpose Output, select the **[IO]** folder from the Parameter Tree. Then select the **[Digital IO]** folder. From the Parameter View Window, select the **“Output function”** parameter you wish to set (1, 2, 3 or 4).
- Outputs that are configured as “Not assigned” can be activated either via the User Program or from a host interface. If an output is assigned as a Special Purpose Output, neither the user program nor the host interface can overwrite its status.
- The Systems Variable **“OUTPUTS”** is a read/write variable that allows the User Program, or host interface, to monitor and set the status of all four outputs. Each output allocates 1 bit in the OUTPUTS variable. For example, if you set this variable equal to 15 in the User Program, i.e. 1111 in binary format, then all 4 outputs will be turned on.
- The example below summarizes the output functions and corresponding System Flags. To set the output, write any non-0 value (TRUE) to its flag. To clear the output, write a 0 value (FALSE) to its flag. You can also use flags in an expression. If an expression is evaluated as TRUE then the output will be turned ON. Otherwise, it will be turned OFF.

```

OUT1 = 1           ;turn OUT1 ON
OUT2 = 10          ;any value but 0 turns output ON
OUT3 = 0           ;turn OUT3 OFF
OUT2 = APOS>3 && APOS<10 ;ON when position within window, otherwise OFF
    
```

Introduction

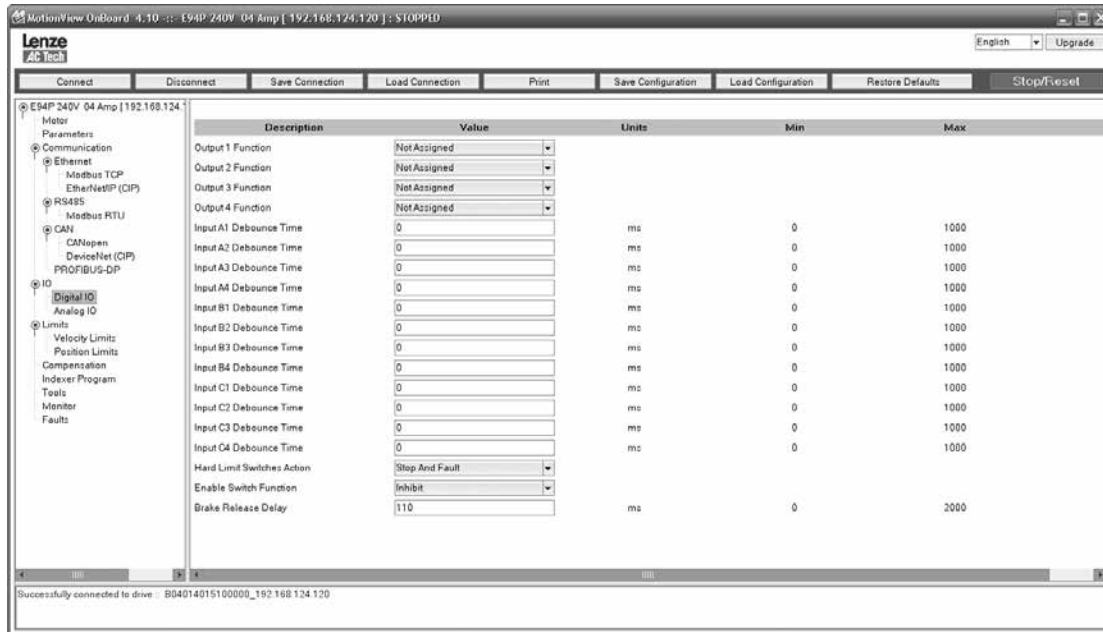


Figure 7: Digital IO Folder

1.7 Events

A Scanned Event is a small program that runs independently of the main program. An event statement establishes a condition that is scanned on a regular basis. Once established, the scanned event can be enabled and disabled in the main program. If condition becomes true and EVENT is enabled, the code placed between EVENT and ENDEVENT executes. Scanned events are used to trigger the actions independently of the main program.

In the following example the Event “**SPRAY_GUNS_ON**” will be setup to turn Output 3 on when the drive’s position becomes greater than 25. Note: the event will be triggered only at the instant when the drive position becomes greater than 25. It will not continue to execute while the position remains greater than 25. (i.e the event is triggered by the transition in logic from false to true). Note also that the main program does not need to be interrupted to perform this action.

```
; ***** EVENT SETUP *****
EVENT SPRAY_GUNS_ON      APOS>25
OUT3=1
ENDEVENT
```

Enter the Event code in the EVENT SETUP section of the program. To Setup an Event, the “**EVENT**” command must be entered. This is followed by the Event Name “**SPRAY_GUNS_ON**” and the triggering mechanism, “**APOS>25**”. After that a sequence of programming statements can be entered once the event is triggered. In our case, we will turn on output 3. To end the Event, the “**ENDEVENT**” command must be used. Events can be activated (turned on) and deactivated (turned off) throughout the program. To turn on an Event, the “**EVENT**” command is entered, followed by the Event Name “**SPRAY_GUNS_ON**”. This is completed by the desired state of the Event, “**ON**” or “**OFF**”. Refer to Section 2.10 for more on Scanned Events.

```
; *****
EVENT SPRAY_GUNS_ON      ON
; *****
```

Two Scanned Events have been added to the Pick and Place program below to trigger a spray gun on and off. The Event will be triggered after the part has been picked up and is passing in front of the spray guns (position greater than 25). Once the part is in position, output 3 is turned on to activate the spray guns. When the part has passed by the spray guns, (position greater than 75), output 3 is turned off, deactivating the spray guns.

Introduction

```
;***** Events *****
EVENT  SPRAY_GUNS_ON      APOS>25    ;Event will trigger as position passes 25 in pos dir.
OUT3=1                                ;Turn on the spray guns (out 3 on)
ENDEVENT                              ;End event
EVENT  SPRAY_GUNS_OFF     APOS>75    ;Event will trigger as position passes 75 in pos dir.
OUT3=0                                ;Turn off the spray guns (out 3 off)
ENDEVENT                              ;End event
;***** Main Program *****
WAIT UNTIL IN_A3                    ;Make sure the Enable input is made before continuing
ENABLE
OUT1 = 0                            ;Initialize Pick Arm - Place in Retracted Position
WAIT UNTIL IN_A4==1                 ;Check Pick Arm is in Retracted Position
EVENT  SPRAY_GUNS_ON      ON
EVENT  SPRAY_GUNS_OFF     ON
PROGRAM_START:
MOVEP 0                             ;Move to Pick position
OUT1 = 1                             ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1                 ;Arm extends
OUT2 = 1                             ;Turn on output 2 to Engage gripper
WAIT TIME 1000                      ;Delay 1 sec to Pick part
OUT1 = 0                             ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1                 ;Make sure Arm is retracted
MOVEP 100                           ;Move to Place position
OUT1 = 1                             ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1                 ;Arm extends
OUT2 = 0                             ;Turn off output 2 to Disengage gripper
WAIT TIME 1000                      ;Delay 1 sec to Place part
OUT1 = 0                             ;Retract Pick arm
WAIT UNTIL IN_A4==1                 ;Arm is retracted
GOTO PROGRAM_START
END
```

1.8 User Variables and the Define Statement

In the previous program for the pick and place machine constant values were used for position limits to trigger the events and turn the spray gun ON and OFF. If limits must be calculated based on some parameters unknown before the program runs (like home origin, material width, etc.), then this system data can be stored in user variables. The PositionServo provides 32 User Variables V0-V31 and 32 User Network Variables NV0-NV31. Network variables have an additional function associated to them (refer to 'Send' Command) but can, for most purposes, be considered as user variables in the same way as the standard user variables (V0-31). Hence 64 user variables or data storage locations are available to the programmer. In the program following the example DEFINE statements, the limit APOS (actual position) is compared to V1 for an ON event and V2 for an OFF event. The necessary limit values could be calculated earlier in the program or supplied by an HMI or host PC. The DEFINE statement can be used to assign a name to a constant, variable, or drive Input/Output. In the program below, constants 1 and 0 are defined as Output_On and Output_Off. DEFINE is a pseudo statement, i.e it is not executed by the program interpreter, but rather substitutes expressions in the subsequent program at the time of compilation. Examples of the DEFINE statement:

```
; Definition of Constant Values
DEFINE Move_1 100
DEFINE BallScrewPitch 0.357

; Definition of Inputs/Outputs
DEFINE System_Run_IP In_B1
DEFINE Process_Run_OP Out1

; Definition User Variables
DEFINE Distance_Travelled V2
DEFINE Network_Healthy NV10
```

Programming the following statement: Distance_Travelled = Move_1 * BallScrewPitch
Is now the equivalent of writing: V2 = 100 * 0.357

Introduction

```
;***** Initialize and Set Variables *****
UNITS = 1 ;Define units for program, 1=revolution of motor shaft
ACCEL = 5 ;Set Acceleration rate for Motion command
DECEL = 5 ;Set Deceleration rate for Motion command
MAXV = 10 ;Maximum Velocity for Motion commands
V1 = 25 ;Set Variable V1 equal to 25
V2 = 75 ;Set Variable V2 equal to 75
DEFINE Output_On 1 ;Define Name for output On
DEFINE Output_Off 0 ;Define Name for output Off
;***** EVENTS *****
EVENT SPRAY_GUNS_ON APOS > V1 ;Event will trigger as position passes 25 in pos dir.
OUT3= Output_On ;Turn on the spray guns (out 3 on)
ENDEVENT ;End event

EVENT SPRAY_GUNS_OFF APOS > V2 ;Event will trigger as position passes 75 in pos dir.
OUT3= Output_Off ;Turn off the spray guns (out 3 off)
ENDEVENT ;End event
;***** Main Program *****
WAIT UNTIL IN_A3 ;Make sure the Enable input is made before continuing
ENABLE
OUT1 = 0 ;Initialize Pick Arm - Place in Retracted Position
WAIT UNTIL IN_A4==1 ;Check Pick Arm is in Retracted Position
EVENT SPRAY_GUNS_ON ON ;Enable the Event
EVENT SPRAY_GUNS_OFF ON ;Enable the Event
PROGRAM_START:
MOVEP 0 ;Move to position 0 to pick part
OUT1 = Output_On ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1 ;Check input to make sure Arm is extended
OUT2 = Output_On ;Turn on output 2 to Engage gripper
WAIT TIME 1000 ;Delay 1 sec to Pick part
OUT1 = Output_Off ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1 ;Check input to make sure Arm is retracted
MOVED 100 ;Move to Place position
OUT1 = Output_On ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1 ;Check input to make sure Arm is extended
OUT2 = Output_Off ;Turn off output 2 to Disengage gripper
WAIT TIME 1000 ;Delay 1 sec to Place part
OUT1 = Output_Off ;Retract Pick arm
WAIT UNTIL IN_A4==1 ;Check input to make sure Arm is retracted
GOTO PROGRAM_START
END
```

1.9 IF/ELSE Statements

An IF/ELSE statement allows the user to execute one or more statements conditionally. The programmer can use an IF or IF/ELSE construct:

Single IF example:

This example increments a counter, Variable "V1", until the Variable, "V1", is greater than 10.

Again:

```
V1=V1+1
IF V1>10
    V1=0
ENDIF
GOTO Again
END
```

IF/ELSE example:

This example checks the value of Variable V1. If V1 is greater than 3, then V2 is set to 1. If V1 is not greater than 3, then V2 is set to 0.

```

IF V1>3
    V2=1
ELSE
    V2=0
ENDIF
    
```

Whether you are using an IF or IF/ELSE statement the construct must end with ENDIF keyword.

1.10 Motion

Figure 8 illustrates the Position and Velocity regulator of the PositionServo drive.

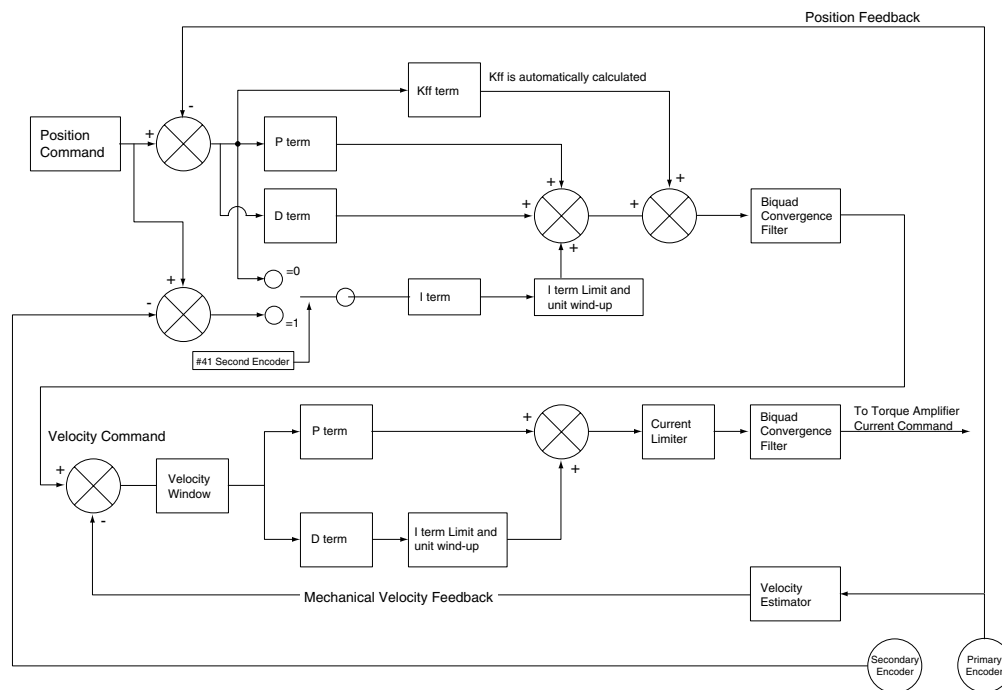


Figure 8: PositionServo Position and Velocity Regulator's Diagram

The “**Position Command**”, as shown in the regulator’s diagram (Figure 9), is produced by a **Trajectory Generator**. The Trajectory Generator processes the motion commands produced by the User’s program to calculate the position increment or decrement, also referred to as the “index” value, for every servo loop. This calculated target (or theoretical) position is then supplied to the **Regulator** input.

The main purpose of the **Regulator** is to set the motors position to match the target position created by the Trajectory Generator. This is done by comparing the input from the Trajectory Generator with the position feedback from the primary motor feedback (resolver or encoder) to control the torque and velocity of the motor. There will always be some error in the position following. Such error is referred to as “Position Error” and is expressed as follows:

$$\text{Position Error} = \text{Target Position} - \text{Actual Position}$$

When the actual Position Error exceeds a certain threshold value for greater than the predefined time limit a “Position Error limit”, fault (F_{PE}) will be generated. The Position Error limit and Position Error time can be set under the Parameter (Node) Tree “Limits”/ “Position Limits” in MotionView. The Position Error time specifies how long the actual position error can exceed the Position Error limit before the fault is generated.

Introduction

1.10.1 Drive Operating Modes

There are three modes of operation for the PositionServo: Torque, Velocity and Position. Torque and Velocity modes are generally used when the command reference is from an external device (via analog input 1), however mechanisms also exist for operation in these modes from within the internal user program. Position mode is used when the command comes from the drives User Program, or from an external device (drive fed from encoder or step/direction signal). Setting the drive's mode is done from the [Parameter] folder in MotionView. To command motion from the user program the drive must be configured to internal reference mode. When the drive is in position mode, it can be placed into a simulated velocity mode without the need to change operating mode to 'Velocity'. Velocity profiling from Positioning mode can be turned on and off from the User Program. Executing the VELOCITY ON statement is used to activate this mode while VELOCITY OFF will deactivate this mode. This mode is used for special case indexing moves. When in Velocity simulation mode the target position is constantly advanced with a rate set in the VEL system variable. The Reference arrangements for the different modes of operation are illustrated in Figure 9.

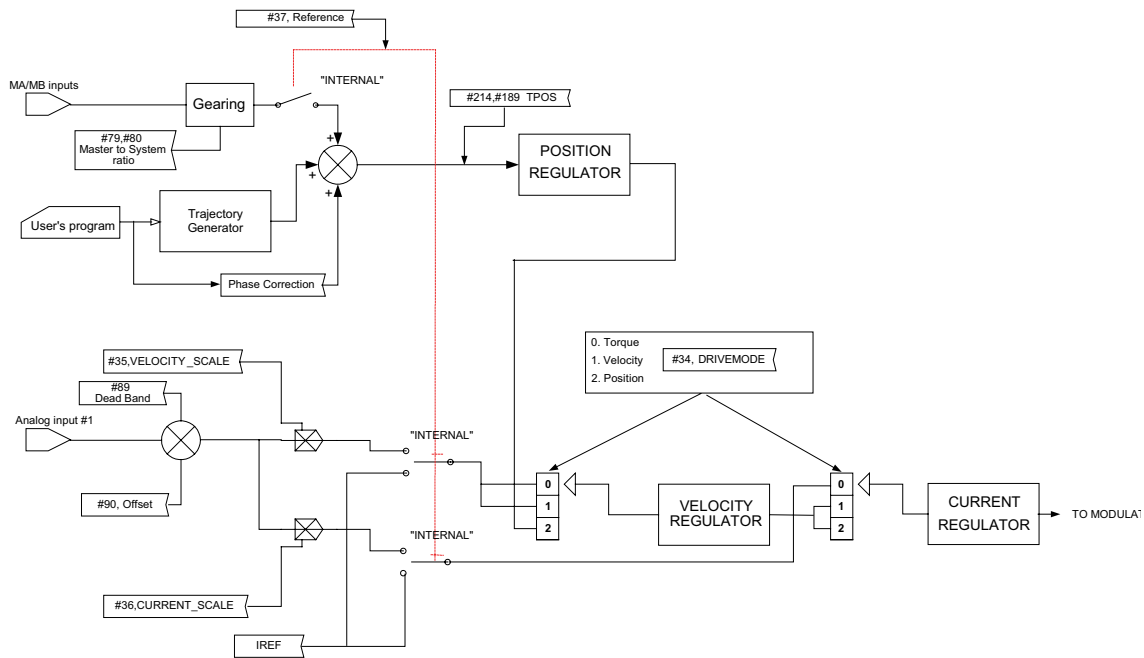


Figure 9: Reference Arrangement Diagram

1.10.2 Point To Point Moves

The PositionServo supports two types of moves: absolute and incremental. The statement MOVEP (Move to Position) is used to make an absolute move. When executing an absolute move, the motor is instructed to move to a known position. The move to this known position is always referenced from the motor's "home" or "zero" location. For example, the statement (MOVEP 0) will cause the motor to move to its zero or home position, regardless of where the motor is located at the beginning of the move. The statement MOVED (Move Distance) makes incremental, (or relative), moves from its current position. For example, MOVED 10, will cause the motor to move forward 10 user units from its current location.

MOVEP and MOVED statements generate what is called a trapezoidal point to point motion profile. A trapezoidal move is when the motor accelerates, using the current acceleration setting, (ACCEL), to a pre-defined top speed, (MAXV), it then maintains that speed for a period of time before decelerating to the end position using the deceleration setting, (DECEL). If the distance to be moved is fairly small, a triangular move profile will be used. A triangular move is a move that starts to accelerate toward the Max Velocity setting but has to decelerate before ever achieving the max velocity in order to reach the desired end point.

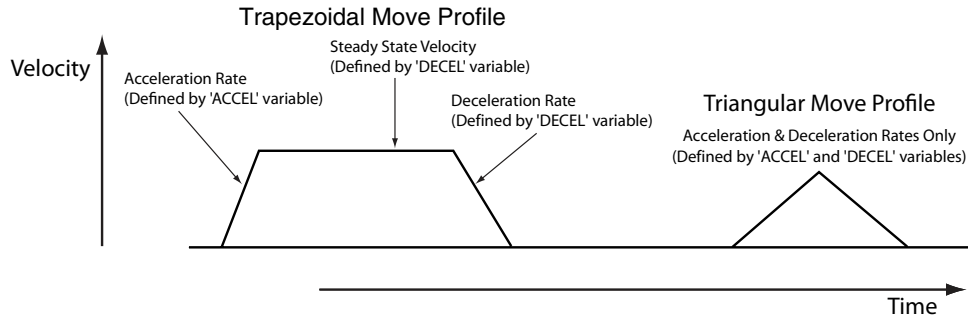


Figure 10: Trapezoidal Move

1.10.3 Segment Moves

MOVED and MOVEP commands facilitate simple motion to be commanded, but if the required move profile is more complex than a simple trapezoidal will allow, then the segment MDV move can be used.

The profile shown in Figure 11 is divided into 8 segments or 8 MDV moves. An MDV move (Move Distance Velocity) has two arguments. The first argument is the distance moved in that segment. This distance is referenced from the motor's current position in User Units. The second argument is the desired target velocity for the end of the segment move. That is the velocity at which the motor will run at the moment when the specified distance in this segment is completed.

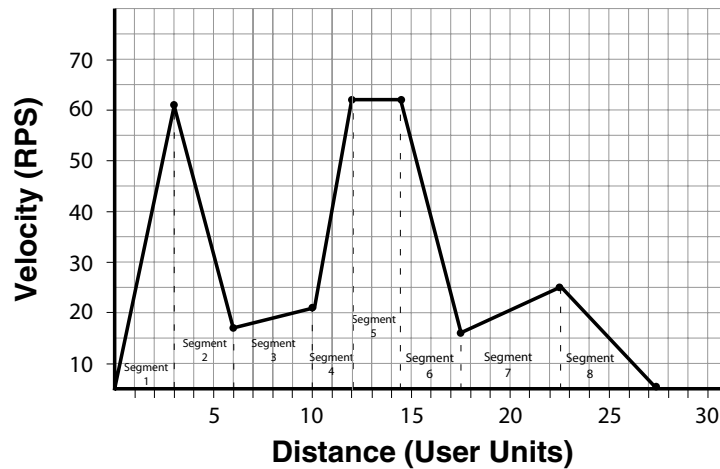


Figure 11: Segment Move

Table 6: Segment Move

Segment Number	Distance moved during segment	Velocity at the end of segment
1	3	56
2	3	12
3	4	16
4	2	57
5	2.5	57
6	3	11
7	5	20
8	5	0
-	-	-

Introduction

Here is the user program for the segment move example. The last segment move must have a "0" for the end velocity, (MDV 5 , 0). Otherwise, fault F_24 (Motion Queue Underflow), will occur.

```
;Segment moves
LOOP:
WAIT UNTIL IN_A4==0 ;Wait until input A4 is off before starting the move
MDV 3 , 56          ;Move 3 units accelerating to 56 User Units per sec
MDV 3 , 12         ;Move 3 units decelerating to 12 User Units per sec
MDV 4 , 16         ;Move 4 units accelerating to 16 User Units per sec
MDV 2 , 57         ;Move 2 units accelerating to 57 User Units per sec
MDV 2.5 , 57       ;Move 2.5 units maintaining 57 User Units per sec
MDV 3 , 11         ;Move 3 units decelerating to 11 User Units per sec
MDV 5 , 20         ;Move 5 units accelerating to 20 User Units per sec
MDV 5 , 0          ;Move 5 units decelerating to 0 User Units per sec
WAIT UNTIL IN_A4==1 ;Wait until input A4 is on before looping
GOTO LOOP
END
```



NOTE

When an MDV move is executed, the segment moves are stored to a Motion Queue. A maximum of 32 moves (MDV segments) can be held on the Motion Queue at any one time. When a move or segment is completed it is cleared from the Motion Queue. If the program attempts to place more than 32 moves in the Motion Queue (because motion is complex or the program continuously loops on itself) then a fault 23 (F_23) will occur to indicate motion queue overflow.

Since a series of MDV segments need to be loaded quickly to the Motion Queue, the [Step] debugging feature can not be used.

1.10.4 Registration

Both absolute and incremental motion can be used for registration moves. The statements associated with these moves are MOVEPR and MOVEDR. These statements have two arguments. The first argument specifies the commanded move distance or position. The second argument specifies the move made after the registration input is detected. If the registration move is an absolute move, for MovePR, the first argument is absolute (referenced to the 0 position), the second argument is relative to the registration position. For MoveDR, both arguments are relative. The first is relative to the shaft position when motion is started and the second is relative to the registration position.

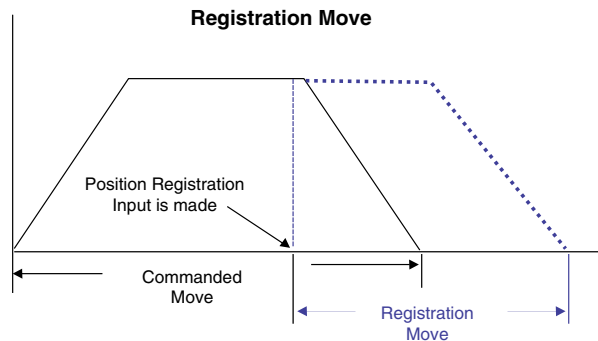


Figure 12: Registration Move

1.10.5 S-Curve Acceleration/Deceleration

It is often necessary, particularly for very dynamic applications, to smooth transition between periods of acceleration / deceleration and steady state velocity. A smoothing of this transition could improve the results of tuning and hence improve overall performance of the system. Additionally smoothing the ramp rates can have the effect of minimizing wear and tear on the system's mechanical components.

With normal straight line ramp rates, the axis is accelerated or decelerated to the target velocity in a linear fashion. With S-curve acceleration/deceleration, the motor ramp rate changes slowly at the first and then slowly stops accelerating/decelerating as it reaches the target velocity. In order for the overall or average ramp rate to remain the same (as specified in the ACCEL/DECEL variables) the slow rates of change at the beginning and the end of the S-curve are compensated by a faster ramp rate in the middle section of the ramp. Maximum ramp rate (occurring in the mid-point of the S-curve) is twice that of using straight line ramps and of the values entered in the ramp rate variables. With straight line ramp rates, the acceleration/deceleration changes can be abrupt at the beginning of the ramp period and again once the motor reaches the target velocity. With S-curve ramp rates, the ramp rate gradually builds to the peak value then gradually decreases to no acceleration/deceleration. The disadvantage with S-curve acceleration/deceleration is that for the same accel/decel distance the peak acceleration/deceleration is twice that of straight line acceleration/deceleration, which often requires twice the peak torque. Note that the axis will arrive at the target position at the same time regardless of which acceleration/deceleration method is used.

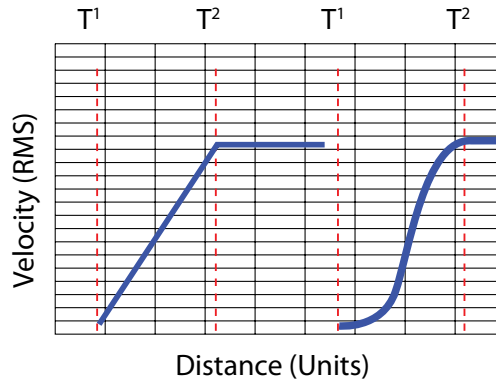


Figure 13: Sequential Move

To use S-curve acceleration/deceleration in a MOVED, MOVEP or MDV statement requires only the additional “S” at the end of the statement.

Examples:

```

MOVED 10 , S
MOVEP 10 , S
MDV 10,20,S
MDV 10,0,S
    
```

1.10.6 Motion Queue

The PositionServo drive executes the User Program one statement at a time. When a move statement (MOVED or MOVEP) is executed, the move profile is stored to the Motion Queue. The program will, by default, wait on that statement until the Motion Queue has executed the move. Once the move is completed, the next statement in the program will be executed. By default motion commands (other than MDV statements) effectively suspend the program until the motion is complete.

In order for subsequent program statements to be executed during a motion command (Move, MoveD, MoveP) an additional line argument can be used. ‘C’ placed on the end of the move statement, for example MoveP 0,C or MoveD 100,C will continue user program execution while those motion commands are executed.

Continuous program execution during a move allows for additional move commands or motion profiles to be stored to the Motion Queue. The Motion Queue has a limit of 32 profiles and exceeding this will result in a ‘Motion Stack Overflow’. The Continue “C” argument is used when it is necessary to trigger an action (e.g. handle I/O) while the motor is in motion. The following Pick and Place Example Program has been modified to utilize the Continue, “C”, argument.

Introduction

```
;***** Main Program *****
WAIT UNTIL IN_A3          ;Make sure the Enable input is made before continuing
ENABLE
OUT1 = 0                  ;Initialize Pick Arm - Place in Retracted Position
WAIT UNTIL IN_A4==1      ;Check Pick Arm is in Retracted Position
PROGRAM_START:
MOVEP 0                   ;Move to position 0 to pick part
OUT1 = 1                  ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1      ;Check input to make sure Arm is extended
OUT2 = 1                  ;Turn on output 2 to Engage gripper
WAIT TIME 1000           ;Delay 1 sec to Pick part
OUT1 = 0                  ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1      ;Check input to make sure Arm is retracted
MOVED 100,C              ;Move to Place position and continue code execution
WAIT UNTIL APOS >25      ;Wait until pos is greater than 25
OUT3 = 1                  ;Turn on output 3 to spray part
WAIT UNTIL APOS >=75     ;Wait until pos is greater than or equal to 75
OUT3 = 0                  ;Turn off output 3 to shut off spray guns
WAIT UNTIL APOS >=95     ;Wait until move is almost done before extending arm
OUT1 = 1                  ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1      ;Check input to make sure Arm is extended
OUT2 =0                   ;Turn off output 2 to Disengage gripper
WAIT TIME 1000           ;Delay 1 sec to Place part
OUT1 = 0                  ;Retract Pick arm
WAIT UNTIL IN_A4==1      ;Check input to make sure Arm is retracted
GOTO PROGRAM_START
END
```

When the “C” argument is added to the standard MOVED and MOVEP statements, program execution is not interrupted by the execution of the motion command. Note: with an MDV move the execution of the program is never suspended.

Generated motion profiles are stored directly to the Motion Queue and are then executed in sequence. If the MOVED and MOVEP statements don't have the “C” modifier, then the motion profiles generated by these statements go to the motion stack and the program is suspended until each profile has been executed.

1.11 Subroutines and Loops

1.11.1 Subroutines

Often it is necessary to repeat a series of program statements in several places in a program. Subroutines are typically used where code is used multiple times and within various sections of the main program. Subroutines are placed after the main program, i.e. after the END statement, and must start with the subname: label (where subname is the name of subroutine), and must end with a statement RETURN.

Note that there can be more than one RETURN statement in a subroutine. Subroutines are called using the GOSUB statement.

1.11.2 Loops

SML language supports WHILE/ENDWHILE block statement which can be used to create conditional loops. Note that IF-GOTO and DO/UNTIL statements can also be used to create loops.

The following example illustrates calling subroutines as well as how to implement looping by utilizing WHILE / ENDWHILE statements.

```
;***** Initialize and Set Variables *****
UNITS = 1 ;Units in Revolutions (R)
ACCEL = 15 ;15 Rev per second per second (RPSS)
DECEL = 15 ;15 Rev per second per second (RPSS)
MAXV = 100 ;100 Rev per second (RPS)/6000RPM
APOS = 0 ;Set current position to 0 (absolute zero position)
DEFINE LOOPCOUNT V1
DEFINE LOOPS 10
DEFINE DIST V2
DEFINE REPETITIONS V3
REPETITIONS = 0

;***** Main Program *****
WAIT UNTIL IN_A3 ;Make sure the Enable input is made before continuing
ENABLE
PROGRAM_START:
MAINLOOP:
    LOOPCOUNT=LOOPS ;Set up the loopcount to loop 10 times
    DIST=10 ;Set distance to 10
    WHILE LOOPCOUNT ;Loop while loopcount is greater than zero
        GOSUB MDS ;Call to subroutine
        WAIT TIME 100 ;Delay executes after returned from the subroutine
        LOOPCOUNT=LOOPCOUNT-1 ;decrement loop counter
    ENDWHILE
    REPETITIONS=REPETITIONS+1 ;outer loop
    IF REPETITIONS < 5
GOTO MAINLOOP
Wait Motioncomplete ;Wait for MDV segments to be completed
ENDIF
END

;***** Sub-Routines *****
MDS:
    V4=dist/3
    MDV V4,10
    MDV V4,10
    MDV V4,0
RETURN
```

Note: Execution of this code will most likely result in F_23. There are 3 MDV statements that are executed 10 times totaling 30 moves. Then the condition set on the repetitions variable makes the program execute the above another 4 times. $4 \times 30 = 120$. The 120 moves, with no waits anywhere in the program will most likely produce an F_23 fault (Motion Queue overflow). Where the possibility exists to overflow the Motion Queue additional code should be used to detect 'Motion Queue Full' condition and to wait for space on the Motion Queue to become available.

2. Programming

2.1 Program Structure

One of the most important aspects of programming is developing the program's structure. Before writing a program, first develop a plan for that program. What tasks must be performed? And in what order? What things can be done to make the program easy to understand and allow it to be maintained by others? Are there any repetitive procedures?

Most programs are not a simple linear list of instructions where every instruction is executed in exactly the same order each time the program runs. Programs need to perform different functions in response to external events and operator input. SML contains program control structures and scanned event functions that may be used to control the flow of execution in an application program. Control structure statements are the instructions that cause the program to change the path of execution. Scanned events are instructions that execute at the same time as the main body of the application program.

Header - Enter in program description and title information

```
***** HEADER *****
;Title:           Pick and Place example program
;Author:          Lenze
;Description:     This is a sample program showing a simple sequence that
;                picks up a part, moves to a set position and drops the part
```

I/O List - Define what I/O will be used

```
***** I/O List *****
;   Input A1   -   not used
;   Input A2   -   not used
;   Input A3   -   Enable Input
;   Input A4   -   not used
;   Input B1   -   not used
;   Input B2   -   not used
;   Input B3   -   not used
;   Input B4   -   not used
;   Input C1   -   not used
;   Input C2   -   not used
;   Input C3   -   not used
;   Input C4   -   not used
;
;   Output 1   -   Pick Arm
;   Output 2   -   Gripper
;   Output 3   -   not used
;   Output 4   -   not used
```

Initialize and Set Variables - Define and assign Variables values

```
***** Initialize and Set Variables *****
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
;V1 =
;V2 =
DEFINE Output_on 1
DEFINE Output_off 0
```

Programming

Events - Define Event name, Trigger and Program Statements

```
;***** Events *****
EVENT SPRAY_GUNS_ON  APOS > 25 ;Event will trigger as position passes 25 in pos dir.
    OUT3= Output_On          ;Turn on the spray guns (out 3 on)
ENDEVENT              ;End event

EVENT SPRAY_GUNS_OFF APOS > 75 ;Event will trigger as position passes 75 in pos dir.
    OUT3= Output_Off         ;Turn off the spray guns (out 3 off)
ENDEVENT              ;End even
```

Main Program - Define the motion and I/O handling of the machine

```
;***** Main Program *****
RESET_DRIVE:          ;Place holder for Fault Handler Routine
WAIT UNTIL IN_A3      ;Make sure the ENABLE input is made before continuing
ENABLE
OUT1 = 0              ;Initialize Pick Arm - Place in Retracted Position
WAIT UNTIL IN_A4==1   ;Check Pick Arm is in Retracted Position
EVENT SPRAY_GUNS_ON ON ;Enable the Event
EVENT SPRAY_GUNS_OFF ON ;Enable the Event
PROGRAM_START:
MOVEP 0               ;Move to position 0 to pick part
OUT1 = Output_On      ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1   ;Check input to make sure Arm is extended
OUT2 = Output_On      ;Turn on output 2 to Engage gripper
WAIT TIME 1000        ;Delay 1 sec to Pick part
OUT1 = Output_Off     ;Turn off output 1 to Retract Pick arm
WAIT UNTIL IN_A4==1   ;Check input to make sure Arm is retracted
MOVED 100             ;Move to Place position
OUT1 = Output_On      ;Turn on output 1 to extend Pick arm
WAIT UNTIL IN_A1==1   ;Check input to make sure Arm is extended
OUT2 = Output_Off     ;Turn off output 2 to Disengage gripper
WAIT TIME 1000        ;Delay 1 sec to Place part
OUT1 = Output_Off     ;Retract Pick arm
WAIT UNTIL IN_A4==1   ;Check input to make sure Arm is retracted
GOTO PROGRAM_START
END
```

Sub-Routine - All Sub-Routine code should reside here

```
;***** Sub-Routines *****
; Enter Sub-Routine code here
```

Fault Handler - Define what the program should do when a fault is detected

```
;***** Fault Handler Routine *****
; Enter Fault Handler code here
ON FAULT
ENDFAULT
```

The **header section** of the program contains description information, program name, version number, description of process and programmers name. The **I/O List section** of the program contains a listing of all the I/O used within the application. The **Initialize and Set Variables section** of the program defines the names for the user variables and constants used in the program and provides initial setting of these and other variables.

Programming

The **Events section** contains all scanned events. Remember to execute the **EVENT <eventname> ON** statement in the main program to enable the events. Please note that not all of the SML statements are executable from within the EVENT body. For more detail, reference “EVENT” and “ENDEVENT” in Section 3 of the manual. The GOTO statement can not be executed from within the Event body. However, the JUMP statement can be used to jump to code in the main program body. This technique allows the program flow to change based on the execution of an event. For more detail, reference “JUMP”, in Section 3.1 (Program Statement Glossary) of this manual.

The **main program** body of the program contains the main part of the program, which can include all motion and math statements, labels, I/O commands and subroutine calls. The main body should be finished with an END statement, however, if the program loops indefinitely then the END statement can be omitted.

Subroutines are routines that are called from the main body of the program. When a subroutine is called, (GOSUB), the program’s execution is transferred from the main program to the called subroutine. It will then process the subroutine until a RETURN statement occurs. Once a RETURN statement is executed, the program’s execution will return back to the main program at the line of code immediately following the GOSUB statement.

Fault handler is the section of code that is executed when the drive detects a fault. This section of code begins with the “ON FAULT” statement and ends with an “ENDFAULT” statement. When a fault occurs, the normal program flow is interrupted, motion is stopped, the drive is disabled, Event scanning is stopped and the statements in the Fault Handler are executed. The Fault handler can be exited in two ways:

- The “RESUME” statement will cause the program to end the Fault Handler routine and return the execution to the main program. The location (label) called out in the “RESUME” command will determine where the program will commence.
- The “ENDFAULT” statement will cause the user program to be terminated.



While the Fault Handler is being executed, Events are not being processed and detection of additional faults is not possible. Because of this, the Fault Handler code should be kept as short as possible.

If extensive code must be written to process the fault, then this code should be placed in the main program and the “RESUME” statement should be utilized. Not all SML statements can be utilized by the Fault Handler. For more details reference “ON FAULT/ENDFAULT”, in Section 3.1 (Program Statement Glossary) of this manual.

Comments are allowed in any section of the program and are preceded by a semicolon. They may occur on the same line as an instruction or on a line by themselves. Any text following a semicolon in a line will be ignored by the compiler.

2.2 Variables

Variables can be System or User. User variables do not have a predefined meaning and are available to the programmer to store any valid numeric value. System variables have a predefined meaning and are used to configure, control or monitor the operations of the PositionServo. (Refer to paragraph 2.6 for more information on System Variables).

All variables can be used in valid arithmetic expressions. All variables have their own corresponding index or identification number. Any variable can be accessed by their identification number from the User’s program or from a Host Interface. In addition to identification numbers all of the variables have predefined names and can be accessed by that name from the user program.

The following syntax is used when accessing variables by their identification number:

```
@102 = 20 ; set variable #102 to 20
@88=@100 ; copy value of variable #100 to variable #88
```

Variable @102 has the variable name ‘V2’; Variable @88 has the variable name ‘VAR_AOUT’ and Variable @100 has the variable name ‘V0’. Hence the program statements above could be written as:

```
V2 = 20
VAR_AOUT = V0
```

Using variable names rather than identification numbers creates code that is more easily read and understood.

There are two types of variables in the PositionServo drive - **User Variables** and **System Variables**.

User Variables are a fixed set of variables that the programmer can use to store data and perform arithmetic calculations. All variables are of a single type. Single type variables, i.e. typeless variables, relieve the programmer of the task of remembering to apply conversion rules between types, thus greatly simplifying programming.

User Variables

- V0-V31** User defined variables. Variables can hold any numeric value including logic (Boolean 0 - FALSE and non 0 - TRUE) values. They can be used in any valid arithmetic or logical expressions.
- NV0-NV31** User defined network variables. Variables can hold any numeric value including logic (Boolean 0 - FALSE and non 0 - TRUE) values. They can be used in any valid arithmetic or logical expressions. Variables can be shared across Ethernet network with use of statements SEND and SENDTO.

Since SML is a typeless language, there is no special type for Boolean type variables (variables that can be only 0 or 1). Regular variables are used to facilitate Boolean variables. Assigning a variable a "FALSE" state is done by setting it equal to "0". Assigning a variable a "TRUE" state is done by assigning it any value other than "0".

Scope

SML variables are accessible from several sources. Each of the variables can be read and set from the user program or Host communications interface at any time. There is no provision to protect a variable from change. This is referred to as global scope.

Volatility

User variables are volatile i.e. they don't maintain their values after the drive is powered down. After power up the values of the user variables are set to 0. Loading or resetting the user program doesn't reset variables values. Two programming statements are provided should the programmer wish to implement some non-volatile memory storage within their application (the LoadVars and StoreVars Statements - refer to section 3.1).

In addition to the user variables, system variables are also provided. System variables are dedicated variables that contain specific information relative to the set-up and operation of the drive. For example, **APOS** variable holds actual position of the motor shaft. For more details refer to Section 2.9.

Resolution and Accuracy

Any variable can be used as a condition in a conditional expression. Variables are often used to indicate that some event has occurred, logic state of an input has changed or that the program has executed to a particular point. Variables with non '0' values are evaluated as "TRUE" and variables with a "0" value are evaluated as "FALSE".

Variables are stored internally as 4 bytes (double word) for integer portion and 4 bytes (double word) for fractional portion. Every variable in the system is stored as 64 bit in 32.32 fixed point format. Maximum number can be represented by this format is +/- 2,147,483,648. Variable resolution in this format is 2.3E-10.

2.3 Arithmetic Expressions

Table 7 lists the four arithmetic functions supported by the Indexer program. Constants as well as User and System variables can be part of the arithmetic expressions.

Examples.

```
V1 = V1+V2           ;Add two user variables
V1 = V1-1           ;Subtract constant from variable
V2 = V1+APOS        ;Add User and System (actual position) variables
APOS = 20           ;Set System variable
V5 = V1*(V2+V3*5/3) ;Complicated expression
```

Table 7: Supported Arithmetic Expressions

Operator	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/

Register (variable) overflow for “*” and “/” operations will cause arithmetic overflow fault F_19. Register (variable) overflow/underflow for “+” and “-” operations does not cause an arithmetic fault.

2.4 Logical Expressions and Operators

Bitwise, Boolean, and comparison operators are referred to as Logical Operators. Bitwise operators are used to change individual bits within an operand (variable). Bitwise operation works at the binary level of the variables, changing specified bits or bit patterns within those variables.

Boolean operators are used to combine simple or complex expressions within a single logic statement. They are used to define a condition that ultimately equates to either True or False.

Comparison operators are used to perform a test between two values and to return a result indicating whether or not the test (Comparison) evaluates to true or false.

2.4.1 Bitwise Operators

Table 8 lists the bitwise operators supported by the Indexer program.

Table 8: Supported Bitwise Operators

Operator	Symbol
AND	&
OR	
XOR	^
NOT	!

Both User or System variables can be used with these operators. In order to perform a bitwise (Boolean) operation, the value often easier in entered in hexadecimal format. To enter a number in hexadecimal use the characters ‘0x’ immediately prior to the hexadecimal number. Example: bit 22 alone would be referenced as 0x40000.

Examples:

```
V1 = V2 & 0xF           ;clear all bits but lowest 4
IF (INPUTS & 0x3)      ;check inputs 0 and 1
V1 = V1 | 0xFF         ;set lowest 8 bits
V1 = INPUTS ^ 0xF      ;invert inputs 0-3
V1 = !IN_A1            ;invert input A1
```

2.4.2 Boolean Operators

Table 9 lists the boolean operators supported by the Indexer program. Boolean operators are used in logical expressions.

Table 9: Supported Boolean Operators

Operator	Symbol
AND	&&
OR	
NOT	!

Examples:

```
IF (APOS >2 && APOS <6) || (APOS >10 && APOS <20)
    {statements if true}
ENDIF
```

The above example checks if APOS (actual position) is within one of two windows; 2 to 6 units or 10 to 20 units. In other words:

```
If (APOS is more than 2 AND less than 6)
OR
If (APOS is more than 10 AND less than 20)
THEN the logical expression is evaluated to TRUE. Otherwise it is FALSE
```

2.5 Comparison Operators

Table 10 lists the comparison operators supported by the Indexer program.

Table 10: Supported Comparison Operators

Operator	Symbol
More	>
Less	<
Equal or more	>=
Equal or less	=<
Not Equal	<>
Equal	==

Examples:

```
IF APOS <=10          ;If Actual Position equal or less than 10
IF APOS > 20         ;If Actual Position greater than 20
IF V0==5            ;If V0 equal to 5
IF V1<2 && V2 <>4   ;If V1 less than 2 And V2 doesn't equal 4
```

2.6 System Variables and Flags

System variables are variables that have a predefined meaning. They give the programmer/user access to drive parameters and functions. Some of these variables can also be set via the parameters in MotionView. In most cases the value of these variables can be read and set in the user program or via a Host Interface. Variables are either read only, write only or read and write. Read only variables can only be read and can't be set. For example, INPUTS = 5, is an illegal action because you can not set an input. Conversely, write-only variables cannot be read. Reading a write-only variable by either the variable watch window or network communications can result in erroneous data.

System Flags are predefined bits that are used by a program either to remember something or to signal some condition. Flags are binary values so contain only values 1 or 0 (True or False). For example, IN_A1 is the system flag that reflects the state of digital input A1. Since inputs can only be ON or OFF, then the value of IN_A1 can only be 0 or 1.

2.7 System Variables Storage Organization

The PositionServo drive contains dual variable storage locations, the operational memory (RAM), that is the volatile operating memory, and the EPM memory, that is the non-volatile configuration memory. When the PositionServo is turned on it copies the retained settings from the EPM non-volatile memory into the RAM memory for use during program execution.

When a system variable is changed during normal program execution its value is changed only in the RAM memory and subsequently these values are lost following power down. System variables that are changed through the MotionView parameter set are stored in both EPM and RAM memory so changes have both immediate effect and are retained after power down. The StoreVars command (Refer to section 3.1) can be used to store the user variables (V0-V31) from RAM memory into the EPM memory during program execution so the programmer has the opportunity to retain these after power down.

Host Interfaces have the capability of changing all of the system variable values through any one of the adopted communications protocols available for PositionServo. Communications protocols contain mechanisms to write to RAM memory only, or to RAM memory and EPM memory.



NOTE:

EPM memory is specified for a limited number of write cycles (approximately 1 million). Care must be taken not to excessively write to the EPM memory or not to exceed the maximum write cycle count.

2.7.1 RAM File for User's Data Storage

In addition to the standard user variables (V0-V31 & NV0-NV31) MotionView OnBoard drives have a section of RAM memory (256k) allocated as data storage space and available to the programmer for storage of program data.

The RAM file data storage is often required in systems where it is desirable to store large amounts of data prepared by a host controller (PLC, HMI, PC, etc). This data might represent more complex Pick and Place coordinates, complicated trajectory coordinates, or sets of gains/limits specific for given motion segments.

RAM memory is also utilized in applications that require data collection during system operation. At the end of a period of time the collected data can be acquired by the host controller for analysis. For example, position errors and phase currents collected during the move are then analyzed by the host PLC/PC to qualify system tolerance to error free operation.

Implementation

There are 256K (262,144) bytes provided as RAM file for data storage. Since the basic data type in the drive is 64 bit (8 bytes) 32,768 data elements can be stored in the RAM file. The file is accessible from within the User's program or through any external communications interface (Ethernet, ModBus, CAN etc.). Two statements and three system variables are provided for accessing the RAM file memory. The RAM file is volatile storage and is intended for "per session" usage. The data saved in the RAM file will be lost when the drive is powered off.

The three system variables provided to support file access are:

VAR_MEM_VALUE	(PID = 4)
VAR_MEM_INDEX	(PID = 5)
VAR_MEM_INDEX_INCREMENT	(PID = 6)

In addition, two statements are provided to allow access and storage to the RAM file direct from the user program. The statements MEMSET, MEMGET are described in paragraph 2.7.3 and Tables 44 & 45.

2.7.2 Memory Access Through Special System Variables

VAR_MEM_VALUE holds the value that will be read or written to the RAM file. VAR_MEM_INDEX points to the position in the RAM file (0 to 32767) that data will be read from or written to, and VAR_MEM_INDEX_INCREMENT holds the value that will be modified after the read or write operation is completed.

The RAM memory access is illustrated with the example program herein.

```
-----
;User's program to read/write to RAM file.
;Advance index after writing/reading by 1
;Record position error to RAM file every 100 ms for 10 seconds. 10/0.1 = 100
;locations are needed
-----

DEFINE      IndexStart      0
DEFINE      MemIncrement    1
DEFINE      RecordLength    100
DEFINE      PELimit         0.1                ;0.1 user unit

VAR_MEM_INDEX = IndexStart                ;set start position
VAR_MEM_INDEX_INCREMENT=MemIncrement      ;set increment

-----
EVENT  StorePE TIME 100

          VAR_MEM_VALUE = VAR_POSEERROR    ;store in RAM file.

ENDEVENT

PROGRAMSTART:

          EVENT StorePE ON

          {
              Start some motion activity...
          }

;wait until data collection is over

WHILE    VAR_MEM_INDEX <    (IndexStart+RecordLength)
ENDWHILE
EVENT StorePE Off                ;turn off storage

;Analyze data collected. If PE > PELimit then signal system has low performance...
VAR_MEM_INDEX= IndexStart
WHILE    VAR_MEM_INDEX <    (IndexStart+RecordLength)
    IF (VAR_MEM_VALUE > PELimit)
        GOTO Label_SignalBad
    ENDIF
ENDWHILE

LabelSignalBad:

          {
              Signal that PE out of limits
              ...
          }

END
```

Programming

In the RAM memory access program example, the values of PE (position error) are stored sequentially in the RAM file every 100ms for 10 seconds. (100 samples). After collection is done the data is read from the file one by one and compared with limit value set.

Variable VAR_MEM_INDEX is incremented every read or write by the value stored in VAR_MEM_INDEX_INCREMENT. This could be any value from -32767 to 32767. This allows for decrement through storage locations in the RAM file in addition to Increment. If the value is 0 (zero) no increment/decrement is produced. Var_Mem_Index is a modular variable (it wraps around it maximum or minimum values). I.e. if the next increment or decrement of Var_Mem_Index results in a value beyond the modulus (32767 or -32767) then the variable will wrap around to the opposite end of the variable range. This allows for the creation of circular arrays. This feature can be used for diagnostics when certain parameter(s) are stored in the memory continuously and then, if the system fails, the data array can be examined to simplify diagnostics.

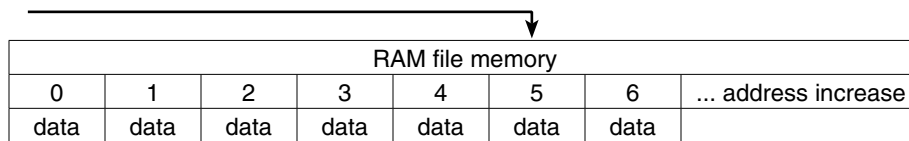
2.7.3 Memory Access Through MEMSET, MEMGET Statements

The memory access statements MEMSET and MEMGET are provided for simplified transfer of data between the RAM memory and the user variables V0-V31. Using these statements, any combinations of variables V0-V31 can be stored/retrieved with a single statement. This allows for efficient access to the RAM memory area. For example, reading 10 values from RAM memory and storing them in 10 user variables using the system variables would normally require 10 separate program statements ($V_x = \text{Var_Mem_Value}$). With the MEMGET statement all 10 user variables can be read in one program statement. The format of MEMSET/MEMGET is as follows:

```
MEMSET    <offset> [ <varlist>]
MEMGET    <offset> [ <varlist>]
<offset>  any valid expression that evaluates to a number between -32767 to 32767
           This specifies the offset in the RAM file where data will be stored or retrieved.
<varlist> any combinations of variables V0-V31
```

Examples for <offset> expression

```
5          constant
10+23+1   constant expression
V0         variable           Must hold values in -32767 to 32767 range
V0+V1+3   expression        Must evaluate to -32767 to 32767 range
Example: <offset> =5
```



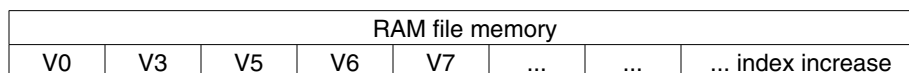
Examples for <varlist> instruction

```
[V0]      single variable will be stored/retrieved
[V0,V3,V2] variables V0,V3,V2 will be stored/retrieved
[V3-V7]   variables V3 to V7 inclusively will be stored/retrieved
[V0,V2,V4-V8] variables V0,V2, V4 through V8 will be stored/retrieved
```

Storage/Retrieval order with MEMSET/MEMGET

Variables in the list are always stored in order: the variable with lowest index first and the variables with highest index last regardless of the order they appear in the <varlist> argument.

Example: [V0,V3, V5-V7] will be stored in memory in the order of increasing memory index as follows:



For comparison: [V5-V7, V0, V3] will have the same storage order as the above list regardless of the order in which the variables are listed.

When retrieving data with MEMGET statements memory locations will be sequentially copied to variables starting from the one with lowest index in the list to the last with highest index. Consider the list for the MEMGET statement:

[V2, V5-V7, V3]

RAM file memory							
Data1	Data2	Data3	Data4	Data5	Data6 index increase

Here is how the data will be assigned to variables:

V2 <- Data1

V3 <- Data2

V5 <- Data3

V6 <- Data4

V7 <- Data5

2.7.4 Store and Retrieve Variables from the EPM

The EPM access statements LOADVARS and STOREVARS are provided to store/retrieve the values of the user variables, V0-V31, to/from the EPM. The LOADVARS statement loads the stored values of the users variables V0-V31 from the EPM. Variable values V0-V31 can be previously stored via the interface or the STOREVARS statement. The STOREVARS statement stores the values of the user variables V0-V31 to the EPM. Variable values V0-V31 can be later retrieved via the interface or the LOADVARS statement. Refer to the Program Statement Glossary in section 3.1 for syntax and example details.



NOTE

At Bootup, variables V0-V31 are automatically retrieved from the EPM.



NOTE

EPM memory is specified for a limited number of write cycles (approximately 1 million). Care must be taken not to excessively write to the EPM memory or not to exceed the maximum write cycle count.

Programming

2.8 System Variables and Flags Summary

2.8.1 System Variables

Section 3.2 provides a complete list of the system variables. Every aspect of the PositionServo can be controlled by the manipulation of the values stored in the System Variables. All System Variables start with a "VAR_" followed by the variable name. Alternatively, System Variables can be addressed as an @NUMBER where the number is the variable Index. The most frequently used variables also have alternate names as listed in Table 11.

Table 11: System Variables

Index	Variable	Access	Variable Description	Units
181	ACCEL	R/W	Acceleration for motion commands	User Units/Sec ²
71	AIN1	R	Analog input. Scaled in volts. Range from -10 to +10 volts	V(olt)
72	AIN2	R	Analog input 2. Scaled in Volts. Range from -10 to +10 volts	V(olt)
88	AOUT	R/W	Analog output. Value in Volts. Valid range from -10 to +10 (V) ⁽²⁾	V(olt)
215	APOS	R/W	Actual motor position	User Units
190	APOS_PLS	R/W	Actual Motor Position	Encoder Counts
182	DECEL	R/W	Deceleration for motion commands	User Units/Sec ²
83	DEXSTATUS	R	Drive Extended Status Word	-
54	DSTATUS	R	Status flags register	-
	DFAULTS	R	Fault code register	-
245	HOME	W	Start Homing (pre-defined homing)	-
	INDEX	R	Lower 8 bits are used. See ASSIGN statement for details.	-
184	INPOSLIM	R/W	Maximum deviation of position for INPOSITION Flag to remain set	User Units
65	INPUTS	R	Digital Inputs states. The first 12 bits correspond to the 12 drive inputs	-
139	IREF	W	Internal Reference: Velocity / Torque	RPS/A
187	MECOUNTER	R	Master Encoder Counts (Master Encoder Input)	Encoder Counts
180	MAXV	R/W	Maximum velocity for motion commands	User Units/Sec
140-171	NV0 - NV31	R/W	User Network Variables	-
66	OUTPUTS	R/W	Digital outputs. Bits #0 to #4 represent outputs 1 through 5	-
216	PERROR	R	Position Error	Feedback Pls
191	PERROR_PLS	R	Position Error	User Units
48	PGAIN_D	R/W	Position loop D-gain	-
47	PGAIN_I	R/W	Position loop I-gain	-
49	PGAIN_ILIM	R/W	Position loop I gain limit	-
46	PGAIN_P	R/W	Position loop P-gain	-
188	PHCUR	R	Motor phase current	A(mpere)
183	QDECEL	R/W	Quick Deceleration for STOP MOTION QUICK statement	User Units/Sec ²
213	RPOS	R	Registration position. Valid when system flag F_REGISTRATION set	User Units
212	RPOS_PLS	R	Registration position	Feedback Pls
218	TA	R	Commanded acceleration	User units/Sec ²
214	TPOS	R/W	Theoretical/commanded position	User Units
219	TPOS_ADV	W	Theoretical/commanded position advance	Feedback Pls
189	TPOS_PLS	R/W	Theoretical/commanded position	Feedback Pls
217	TV	R	Commanded velocity in	User Units/Sec
186	UNITS	R/W	User Units scale. ⁽¹⁾	UserUnits/Rev
185	VEL	R/W	Set Velocity when in velocity mode	User Units/Sec
44	VGAIN_P	R/W	Velocity loop P-gain	-
45	VGAIN_I	R/W	Velocity loop I-gain	-
100-131	V0 - V31	R/W	User Variables	

(1) When a "0", (zero), value is assigned to the variable "UNITS", then "USER UNITS" is set to QUAD ENCODER COUNTS.

(2) Any value outside +/- 10 range assigned to AOUT will be automatically trimmed to that range.

Example:

```
AOUT=100 , AOUT will be assigned value of 10.
V0=236
VOUT=V0, VOUT will be assigned 10 and V0 will be unchanged.
```

2.8.2 System Flags

Flags don't have an Index number assigned to them. They are the product of a BIT mask applied to a particular system variable within the drive and are available to the programmer only from the User's program. Table 12 lists the System Flags with access rights and description.

Table 12: System Flags

Name	Access	Description
IN_A1-4, IN_B1-4, IN_C1-4	R	Digital inputs . TRUE if input active, FALSE otherwise
OUT1, OUT2, OUT3, OUT4, OUT5	W	Digital outputs OUTPUT1- OUTPUT5
F_ICONTROLOFF	R	Interface Control Status (ON/OFF) #27 in DSTATUS register
F_IN_POSITION	R	TRUE when Actual Position (APOS) is within limits set by INPOSLIM variable and motion completed
F_ENABLED	R	Set when drive is enabled
F_EVENTSOFF	R	Events Disabled Status (ON/OFF) #30 in DSTATUS register
F_MCOMPLETE	R	Set when motion is completed and there are no motion commands waiting in the Motion Queue
F_MQUEUE_FULL	R	Motion Queue full
F_MQUEUE_EMPTY	R	Motion Queue empty
F_FAULT	R	Set if any fault detected
F_ARITHMETIC_FLT	R	Arithmetic fault
F_REGISTRATION	R	Set when registration mark is detected. Contents of the RPOS variable valid when this flag is active. Flag reset by any registration moves MOVEPR, MOVEDR or by command REGISTRATION ON
F_MSUSPENDED	R	Set if motion suspended by statement MOTION SUSPEND

Flag logic is shown herein.

```
IF (TPOS-INPOSLIM < APOS) && (APOS < TPOS+INPOSLIM) && F_MCOMPLETE && F_MQUEUE_EMPTY
    Out1 = 1
ELSE
    Out1 = 0
ENDIF
```

For VELOCITY mode F_MCOMPLETE and F_MQUEUE_EMPTY flags are ignored and assumed TRUE.

2.9 Control Structures

Control structures allow the user to control the flow of the program's execution. Most of the control and flexibility of any programming language comes from its ability to change statement order with structure and loops.

2.9.1 IF Structure

The flowchart and code segment in Figure 17 illustrate the use of the IF statement. The "IF" statement is used to execute an instruction or block of instructions one time if a condition is true. The simplified syntax for the IF statement is:

```
IF condition
    ...statement(s)
ENDIF
```

```
...statements
IF IN_A2
    OUT2 = 1
    MOVED 3
ENDIF
..statements
```

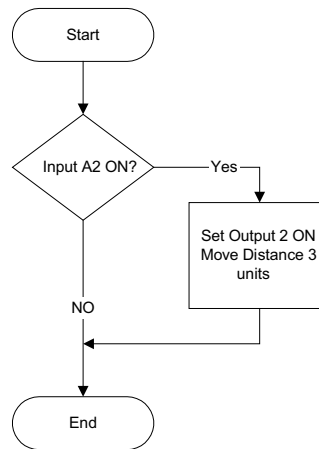


Figure 17: IF Code and Flowchart

IF/ELSE

The flowchart and code segment in Figure 18 illustrate the use of the IF/ELSE instruction. The IF/ELSE statement is used to execute a statement or a block of statements one time if a condition is true and a different statement or block of statements if condition is false. The simplified syntax for the IF/ELSE statement is:

```
IF <condition>
    ...statement(s)
ELSE
    ...statement(s)
ENDIF
```

```
...statements
IF IN_A2
    OUT2=1
    MOVED 3
ELSE
    OUT2=0
    MOVED 5
ENDIF
..statements
```

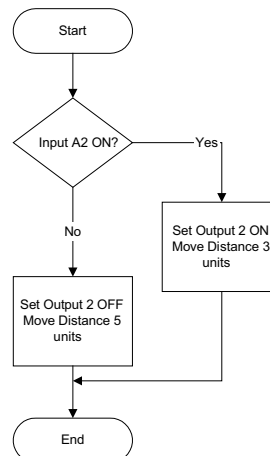


Figure 18: IF/ELSE Code and Flowchart

2.9.2 DO/UNTIL Structure

The flowchart and code segment in Figure 14 illustrate the use of the DO/UNTIL statement. This statement is used to execute a block of code one time and then continue executing that block until a condition becomes true (satisfied). The difference between DO/UNTIL and WHILE statements is that the DO/UNTIL instruction tests the condition after the block is executed so the conditional statements are always executed at least one time. The syntax for DO/UNTIL statement is:

```
DO
    ...statements
UNTIL <condition>
```

```
... statements
DO
    MOVED 3
    WAIT TIME 2000
UNTIL IN_A3
...statements
```

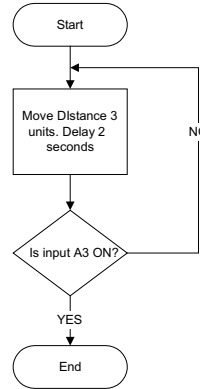


Figure 14: DO/UNTIL Code and Flowchart

2.9.3 WHILE Structure

The flowchart and code segment in Figure 15 illustrate the syntax for the WHILE instruction. This statement is used if you want a block of code to execute while a condition is true.

```
WHILE <condition>
```

```
    ...statements
```

```
ENDWHILE
```

```
...statements
WHILE IN_A3
    MOVED 3
    WAIT TIME 2000
ENDWHILE
...statements
```

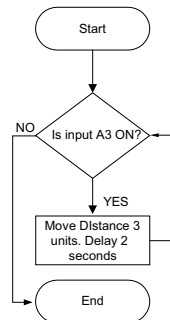


Figure 15: WHILE Code and Flowchart

2.9.4 WAIT Statement

The WAIT statement is used to suspend program execution until or while a condition is true, for a specified time period (delay) or until motion has been completed. The simplified syntax for this statement is:

```
WAIT UNTIL <condition>
WAIT WHILE <condition>
WAIT TIME <time>
WAIT MOTION COMPLETE
```

Programming

2.9.5 GOTO Statement and Labels

The GOTO statement can be used to transfer program execution to a section of the Main Program identified by a label. This statement is often executed conditionally based on the logical result of an If Statement. The destination label may be above or below the GOTO statement in the application program.

Labels must be an alphanumeric string of up to 64 characters in length, ending with a colon “:” and containing no spaces.

```
GOTO TestInputs
    ...statements
TestInputs:
    ...statements
IF (IN_A1) GOTO TestInputs
```

Table 13 provides a short description of the instructions used for program branching.

Table 13: Program Branching Instructions

Name	Description
GOTO	Transfer code execution to a new line marked by a label
DO/UNTIL	Do once and keep doing until conditions becomes true
IF and IF/ELSE	Execute once if condition is true
RETURN	Return from subroutine
WAIT	Wait fixed time or until condition is met
WHILE	Execute while a condition is true
GOSUB	Go to Subroutine

2.9.6 Subroutines

A subroutine is a group of SML statements that is located at the end of the main body of the program. It starts with a label which is used by the GOSUB statement to call the subroutine and ends with a RETURN statement. The subroutine is executed by using the GOSUB statement in the main body of the program. Subroutines can not be called from an EVENT or from the FAULT handler.

When a GOSUB statement is executed, program execution is transferred to the first line of the subroutine. The subroutine is then executed until a RETURN statement is met. When the RETURN statement is executed, the program’s execution returns to the program line (in the main program) following the GOSUB statement. A subroutine may have more than one RETURN statement in its body.

Subroutines may be nested up to 32 times. Only the main body of the program and subroutines may contain a GOSUB statement. Refer to Section 3.1 for more detailed information on the GOSUB and RETURN statements. The flowchart and code segment in Figure 16 illustrate the use of subroutines.

```
...statements
GOSUB CalcMotionParam
MOVED V1
OUT2=1
...statements
END
;
CalcMotionParam:
V1 = (V3*2)/V4
RETURN
```

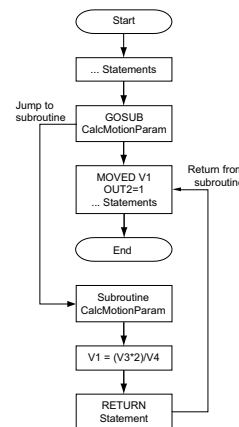


Figure 16: GOSUB Code and Flowchart

2.10 Scanned Event Statements

A Scanned Event is a small program that runs independently of the main program. SCANNED EVENTS are very useful when it is necessary to trigger an action (i.e. handle I/O) while the motor is in motion or other tasks within the Main Program are executing. When setting up Events, the first step is to define both the action that will trigger the event as well as the sequence of statements to be executed once the event has been triggered. Events are scanned every 512µs. Before an Event can be scanned however it must be enabled. Events can be enabled or disabled from the user program or from another event (see explanations below). Once the Event is defined and enabled, the Event will be constantly scanned until the trigger condition is met, this scan rate is independent of the main program's timing. Once the trigger condition is met, the Event statements will be executed independently of the user program.

Scanned events are used to record events and perform actions independent of the main body of the program. For example, if the programmer wants output 3 to come ON when the position is greater than 4 units, or if he needs to turn output 4 ON whenever inputs A4 and B1 are ON, he could use the following scanned event statements.

```

EVENT      PositionIndicator APOS > 4
           OUT3=1

ENDEVENT

EVENT      InputsLogic      IN_A4 & IN_B1
           OUT4=1

ENDEVENT
    
```

Scanned events may also be used with a timer to perform an action on a periodic time basis.

The program statements contained in the scanned event code cannot include any that are related to the command of Motion from the motor or that result in a delay to program execution. A full list of illegal event code statements is given in section 3.1. Syntax for defining Events is as follows.

```
EVENT <name> INPUT <inputname> RISE
```

This scanned event statement is used to execute a block of code each time a specified input <inputname> changes its state from low to high.

```
EVENT <name> INPUT <inputname> FALL
```

This scanned event statement is used to execute a block of code each time a specified input <inputname> changes its state from high to low.

```
EVENT <name> TIME <timeout>
```

This scanned event statement is used to execute a block of code with a repetition rate specified by the <timeout> argument. The range for "timeout" is 0 - 50,000ms (milliseconds). Specifying a timeout period of 0 ms will result in the event running every event cycle (512µs).

```
EVENT <name> expression
```

This scanned event statement is used to execute a block of code when the expression evaluates as true.

```
EVENT <name> ON/OFF
```

This statement is used to enable/disable a scanned event.

Table 14 contains a summary of instructions that relate to scanned events. Refer to Section 3 "Language Reference" for more detailed information.

Table 14: Scanned Events Instructions

Name	Description
EVENT <name> ON/OFF	enable / disable event
EVENT <name> INPUT <inputname> RISE	Scanned event when <input name> goes low to high
EVENT <name> INPUT <inputname> FALL	Scanned event when <input name> goes high to low
EVENT <name> TIME <value>	Periodic event with <value> repetition rate.
EVENT <name> expression	Scanned event on expression = true

2.11 Motion

2.11.1 How Moves Work

The position command that causes motion to be generated comes from the profile generator or profiler for short. The profile generator is used by the MOVE, MOVED, MOVEP, MOVEPR, MOVEDR and MDV statements. MOVE commands generate motion in a positive or negative direction, while or until certain conditions are met. For example you can specify a motion while a specific input remains ON (or OFF). MOVEP generates a move to specific absolute position. MOVED generates incremental distance moves, i.e. move some distance from its current position. MOVEPR and MOVEDR are registration moves. MDV commands are used to generate complicated profiles. Profiles generated by these commands are put into the motion stack which is 32 level. By default when one of these statements (except for MDV) is executed, the execution of the main User Program is suspended until the generated motion is completed. Motion requests generated by an MDV statement, or by MOVE statement with the "C" modifier do not suspend the program. All motion statements are put into the motion stack and executed by the profiler in the order in which they where loaded. The Motion Stack can hold up to 32 moves. The SML language allows the programmer to load moves into the stack and continue on with the program. It is the responsibility of the programmer to check the motion stack to make sure there is room available before loading new moves. This is done by checking the appropriate bits in the System status register or the appropriate system flag.

2.11.2 Incremental (MOVED) and Absolute (MOVEP) Motion

MOVED and MOVEP statements are used to create incremental and absolute moves respectively. The motion that results from these commands is by default a trapezoidal velocity move or an S-curved velocity move if the ",S" modifier is used within the statement.

For example:

```
MOVEP 10 ;will result in a trapezoidal move
```

But

```
MOVEP 10,S ;will result in an S-curved move
```

In the above example, (MOVEP 10), the length of the move is determined by the argument following the MOVEP command, (10). This argument can be a number, a variable or any valid arithmetic expression. The maximum velocity of the move is determined by setting the system variable MAXV. The acceleration and deceleration are determined by setting the system variables ACCEL and DECEL respectively.

If values for velocity, acceleration and deceleration, for a specified distance, are such that there is not enough time to accelerate to the specified velocity, the motion profile will result in triangular or double S profile Full Stop. The following code extract generates the motion profiles shown in Figure 19.

```
ACCEL = 200
DECEL = 200
MAXV = 20
MOVED 4 ;Move 1
MOVED 1.5 ;Move 2
MOVED 4 , S ;Move 3
MOVED 1.5 , S ;Move 4
```

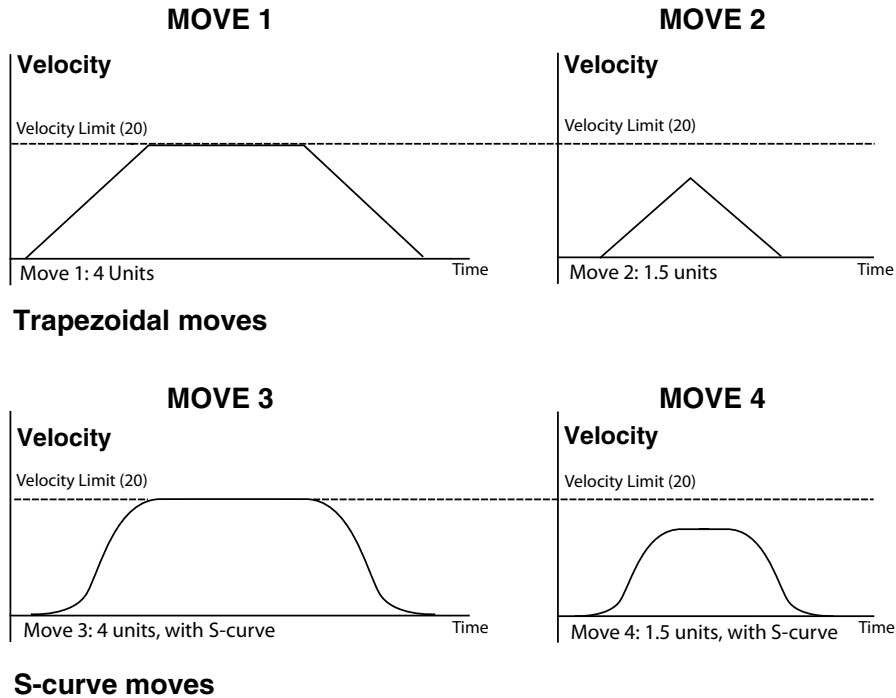


Figure 19: Move Illustration

All four of the moves shown in Figure 19 have the same Acceleration, Deceleration and Max Velocity values. Moves 1 and 3 have a larger value for the move distance than Moves 2 and 4. In Moves 1 and 3 the distance is long enough to allow the motor to accelerate to the profiled max velocity and maintain that velocity before decelerating down to a stop. In Moves 2 and 4 the commanded distance is so small that the calculated point of deceleration occurs before the motor has reached the profiled Maximum velocity. On reaching the calculated deceleration point the drive will start decelerating the motor in order to arrive at the commanded target position.

2.11.3 Incremental (MOVED) Motion

Incremental motion is defined as a move of some distance from the current position. 'Move four revolutions from the current position' is an example of an incremental move.

MOVED is the statement used to create incremental moves. The simplified syntax is:

MOVED <+/-distance>

+/- sign will tell the drive in which direction to move the motor shaft.

2.11.4 Absolute (MOVEP) Move

Absolute motion is defined as motion that is always specified relative to the same 'known' location. The location that each move is specified relative to is termed the zero (0) position. For example an absolute move of 20 will result in a move to a position that is 20 user units from the zero position regardless of whether the current shaft location is less than or greater than this commanded position (required motion is forward or reverse). The Zero position is normally established during a homing cycle performed after power up where the programmer specifies (using a switch or other device) a known point within the system mechanics from where they will reference all further motion.

If an incremental move is repeated (e.g. MoveD 10) then a subsequent move will result as motion is relative to the position of the shaft at the point the motion is initiated. If an absolute move is repeated (e.g. MoveP 10) then only one motion is executed as the subsequent target position commanded is already equal to the motor shaft's current position.

2.11.5 Registration (MOVEDR MOVEPR) Moves

MovePR and MoveDR are move commands subject to (modified by) the drive registration input (C3) activating. They are defined as registration moves as their function is to capture a position based on a sensor input and then move to a subsequent position determined by the captured position plus an offset. Registered move commands contain two motion arguments, the first defining the initial move to attempt detection of registration, and the second defining the modified motion to complete subject to registration being detected.

The difference between MoveDR and MovePR is that MoveDR is incremental and performs the initial move subject to its current position while checking for registration. MovePR is absolute so initial target position (motion) is referenced to the absolute zero position.

If registration is not detected during a MoveDR or MovePR command then the initial move commanded by the first motion argument will be completed and the registration flag will not be set. If registration is detected then both MoveDR or MovePR will modify target position to the captured registration position (stored in the RPOS variable) plus the second motion argument. If registration is detected then the registration flag will be set to true (1).

MOVEPR and MOVEDR are used to move to position or distance respectively just like MOVEP and MOVED. The difference is that while the statements are being executed they are looking for a registration signal or registration input (C3). If during the motion a registration signal is detected, then a new end position is generated. With both the MoveDR and MovePR statements the drive will increment the distance called out in the registration argument. This increment will be referenced from the position where the registration input has detected.

Example:

```
MOVEDR 5, 1 ;Statement moves a distance of 5 user units or registration position +  
           ;1 user units if registration input is activated during motion.
```

There are two exceptions to the behavior of registration moves.

Exception one:

The move will not be modified to "Registration position +displacement" if the registration was detected while system was decelerating to complete the initial motion command.

Exception two:

Once the registration input is detected, there must be enough distance set by the second argument to allow for the motor to decelerate to a stop using the profiled Decel Value. If the modified registration move is smaller than the distance necessary to come to a stop, then the motor will overshoot the programmed registration position. Over-shoot of the target position is not rectified automatically, either realistic arguments must be entered for the registered move command and deceleration rate or a comparison statement used to detect and rectify over-shoot.

2.11.6 Segment Moves

In addition to the simple moves that can be generated by MOVED and MOVEP statements, complex profiles can be generated using segment moves. A segment move represents one portion of a complete move. A complete move is constructed out of two or more segments, starting and ending at zero velocity.

2.11.7 MDV Segments

Profiles are created using a sequence of MDV statements. The simplified syntax for the **MDV (Move Distance with Velocity)** statement is:

```
MDV <distance>,<velocity>
```

The <distance> is the total distance completed during the segment move. The <velocity> is the target velocity for the end of the segment move. The starting velocity is either zero or the final velocity of the previous segment. The final segment in a complete profile must have a velocity of zero. If the final segment has a velocity other than zero, a motion stack under flow fault will occur (F_24).

Programming

The profile shown in Figure 20 can be broken up into 8 MDV moves. The first segment defines the distance between point 1 and point 2 and the velocity at point 2. So, if the distance between point 1 and 2 was 3 units and the velocity at point 2 was 56 Units/S, the command would be: MDV 3 , 56. The second segment gives the distance between point 2 and 3 and the velocity at point 3, and so on.

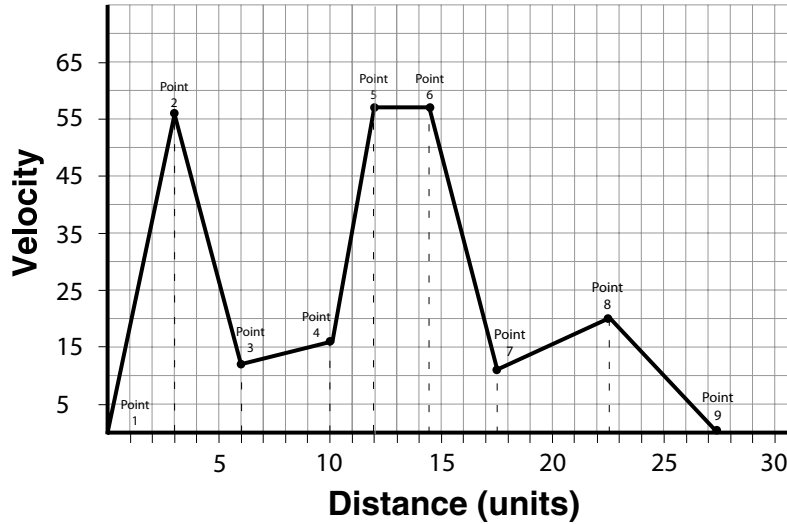


Figure 20: MDV Segment Example

Table 15 lists the supporting data for the graph in Figure 20.

Table 15: MDV Segment Example

Segment Number	Distance moved during segment	Velocity at the end of segment
1	3	56
2	3	12
3	4	16
4	2	57
5	2.5	57
6	3	11
7	5	20
8	5	0
-	-	-

;Segment moves

```
MDV 3 , 56
MDV 3 , 12
MDV 4 , 16
MDV 2 , 57
MDV 2.5 , 57
MDV 3 , 11
MDV 5 , 20
MDV 5 , 0
END
```

The following equation can be used to calculate the acceleration / deceleration that results from a segment move.

$$\text{Accel} = (V_f^2 - V_0^2) / [2 * D]$$

V_f = Final velocity
 V_0 = Starting velocity
 D = Distance

2.11.8 S-curve Acceleration/Deceleration

Instead of using a linear acceleration/deceleration, the motion created using segment moves (MDV statements) can use S-curve acceleration/deceleration. The syntax for MDV move with S-curve acceleration/deceleration is:

```
MDV <distance>,<velocity>,S
```

Segment moves using S-curve acceleration/deceleration will take the same amount of time as linear acceleration/deceleration segment moves. S-curve acceleration/deceleration is useful because it is much smoother at the beginning and end of the segment, however, the peak acceleration/deceleration of the segment will be twice as high as the acceleration/deceleration used in the linear acceleration/deceleration segment.

2.11.9 Motion SUSPEND/RESUME

At times it is necessary to control motion by preloading the motion stack with motion profiles and then executing them consecutively, based on the user program and/or some logical condition being detected. The statement "MOTION SUSPEND" will suspend motion until the statement "MOTION RESUME" is executed. While motion is suspended, any motion statement executed by the User Program will be loaded into the motion stack. When the "MOTION RESUME" statement is executed, the preloaded motion profiles will be executed in the order that they were loaded.

Example:

```
MOTION SUSPEND
MDV 10,2 ;placed in stack
MDV 20,2 ;placed in stack
MDV 2,0 ;placed in stack
MOVED 3,C ;must use ",C "modifier. Otherwise program will hang.
MOTION RESUME
```

Caution should be taken when using MOVED, MOVEP and MOVE statements. If any of the MOVE instructions are written without the "C" modifier, the program will hang or lock up. The "MOTION SUSPEND" command effectively halts all execution of motion. In the example, as the program executes the "MDV" and "MOVED" statements, those move profiles are loaded into the motion stack. If the final "MOVED" is missing the "C" modifier then the User Program will wait until that move profile is complete before continuing on. Because motion has been suspended, the move will never be complete and the program will hang on this instruction.

2.11.10 Conditional Moves (MOVE WHILE/UNTIL)

The statements "MOVE UNTIL <expression>" and "MOVE WHILE <expression>" will both start their motion profiles based on their acceleration and max velocity profile settings. The "MOVE UNTIL <expression>" statement will continue the move until the <expression> becomes true. The "MOVE WHILE <expression>" will also continue its move while it's <expression> is true. Expression can be any valid arithmetic or logical expressions or their combination.

Examples:

```
MOVE WHILE APOS<20 ;Move while the position is less then 20, then
;stop with current deceleration rate.
MOVE UNTIL APOS>V1 ;Move positive until the position is greater than
;the value in variable V1
MOVE BACK UNTIL APOS<V1 ;Move negative until the position is less than the
;value in variable V1
MOVE WHILE IN_A1 ;Move positive while input A1 is activated.
MOVE WHILE !IN_A1 ;Move positive while input A1 is not activated.
;The exclamation mark (!) in front of IN_A1 inverts
;(or negates) the value of IN_A1.
```

This last example is a convenient way to find a sensor or switch.

2.11.11 Motion Queue and Statement Execution while in Motion

By default when the program executes a MOVE, MOVED or MOVEP statement, it waits until the motion is complete before going on to the next statement. This effectively will suspend the program until the requested motion is complete. Note that "EVENTS" are not suspended however and continue executing in parallel with the User Program. The Continue "C" argument is very useful when it is necessary to trigger an action (handle I/O) while the motor is in motion. Below is an example of the Continue "C" argument.

```
;This program monitors I/O in parallel with motion:
START:
    MOVED 100,C                ;start moving max 100 revs
WHILE F_MCOMPLETE=0          ;while moving
    IF IN_A2 == 1             ;if sensor detected
        OUT1=1                ;turn ON output
        WAIT TIME 500         ;500 mS
        OUT1=0                ;turn output OFF
        WAIT TIME 500         ;wait 500 ms
    ENDIF
ENDWHILE
MOVED -100                    ;Return back
WAIT TIME 1000                ;wait time
GOTO START                    ;and start all over
END
```

This program starts a motion of 100 revolutions. While the motor is in motion, input A2 is monitored. If Input A2 is made during the move, then output 1 is turned on for 500ms and then turned off. The program will continue to loop in the WHILE statement, monitoring input A2, until the move is completed. If input 2 remains ON, or made, during the move, then Output 1 will continue to toggle On and Off every 500ms until the move is complete. If input A2 is only made while the motion passes by a sensor wired to the input, then output 1 will stay on for 500ms only. By adding the "Continue" argument "C" to the MOVE statement, the program is able to monitor the input while executing the motion profile. Without this modifier the program would be suspended until all motion is complete. After the motor has traveled the full distance it then returns back to its initial position and the process repeats.

Figure 21 illustrates the structure and operation of the Motion Queue. All moves are loaded into the Motion Queue before they are executed. If the move is a standard move, "MOVEP 10" or "MOVED 10", then the move will be loaded into the queue and the execution of the User Program will be suspended until the move is completed. If the move has the continue argument, e.g. "MOVEP 10,C" or "MOVED 10,C", or if it is an "MDV" move, then the moves will be loaded into Motion Queue and executed simultaneously with the User Program.

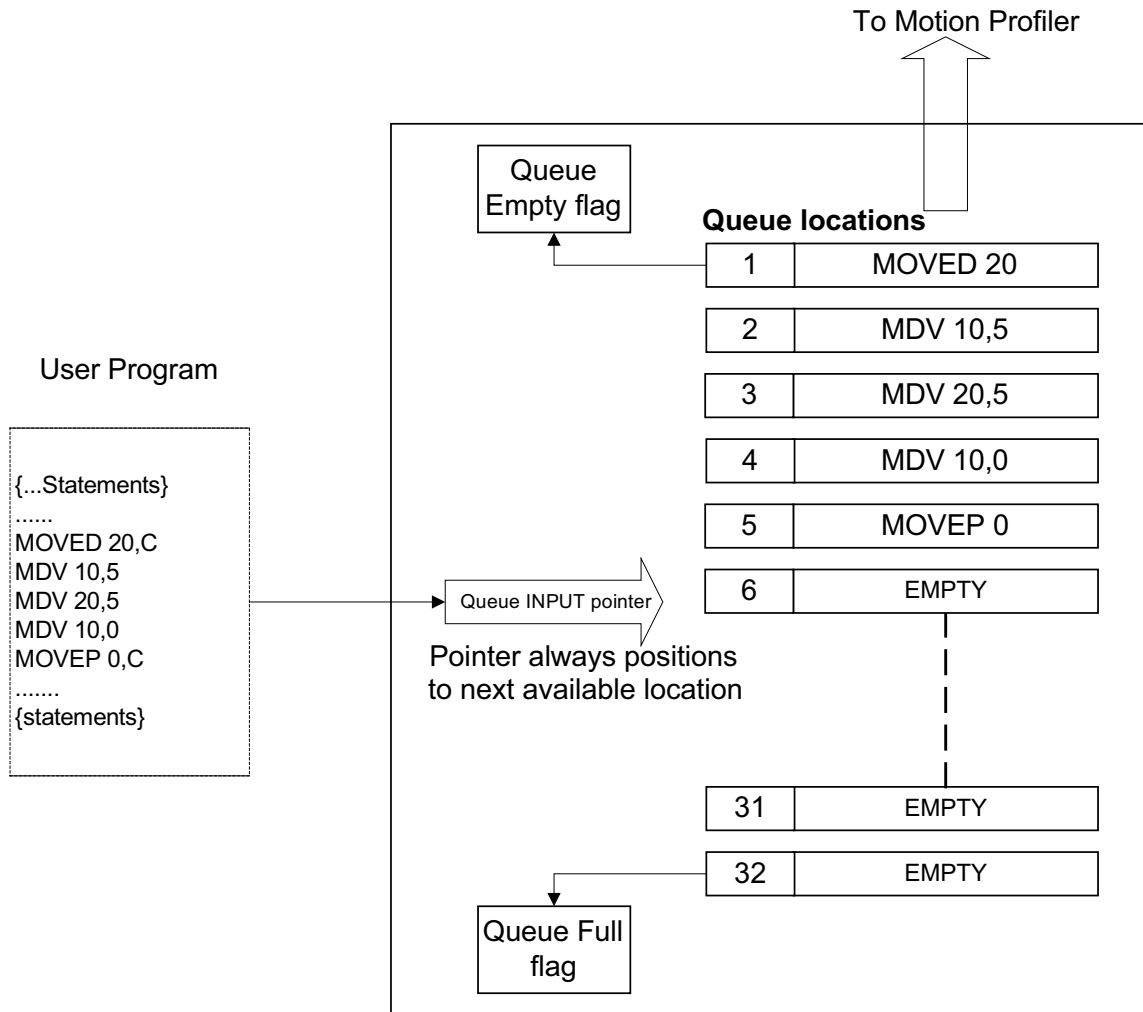


Figure 21: Motion Queue

The Motion Queue can hold a maximum of 32 motion statements. The System Status Register contains bit values that indicate the state of the Motion Queue. Additionally, system flags (representing individual bits of the status register) are available for ease of programming. If the possibility of motion queue overflow exists, the programmer should check the Motion Queue full flag before executing any MOVE statements, especially in programs where MOVE statements are executed in a continuous cycle. Attempts to execute a motion statement while the Motion Queue is full will result in fault #23. MDV statements don't have the "C" option because the program is never suspended by these statements. If the last MDV statement in the Queue doesn't specify a return to 0 velocity then a Stack Underflow (Fault #24) will occur.

The "MOTION SUSPEND" and "MOTION RESUME" statements can be utilized to help manage the User Program and the Motion Queue. If the motion profiles loaded into the queue are not managed correctly, the Motion Queue can become overloaded which will cause the drive to fault.

2.12 System Status Register (DSTATUS register)

System Status Register, (DSTATUS), is a Read Only register. Its bits indicate the various states of the PositionServo's subsystems. Some of the bits are available as System Flag Variables and previously summarized in Table 12.

Table 16: DSTATUS Register

Bit in register	Description
0	Set when drive enabled
1	Set if DSP subsystem at any fault
2	Set if drive has a valid program
3	Set if byte-code or system or DSP at any fault
4	Set if drive has a valid source code
5	Set if motion completed and target position is within specified limits
6	Set when scope is triggered and data collected
7	Set if motion stack is full
8	Set if motion stack is empty
9	Set if byte-code halted
10	Set if byte-code is running
11	Set if byte-code is set to run in step mode
12	Set if byte-code has reached the end of program
13	Set if current limit is reached
14	Set if byte-code at fault
15	Set if no valid motor selected
16	Set if byte-code at arithmetic fault
17	Set if byte-code at user fault
18	Set if DSP initialization completed
19	Set if registration has been triggered
20	Set if registration variable was updated from DSP after last trigger
21	Set if motion module at fault
22	Set if motion suspended
23	Set if program requested to suspend motion
24	Set if system waits completion of motion
25	Set if motion command completed and motion Queue is empty
26	Set if byte-code task requested reset
27	If set interface control is disabled. This flag is set/clear by ICONTROL ON/OFF statement.
28	Set if positive limit switch active
29	Set if negative limit switch active
30	Events disabled. All events disabled when this flag is set. After executing EVENTS ON all events previously enabled by EVENT EventName ON statements become enabled again

PositionServo variable #83 provides Extended Status Bits, the encoding of which is listed in Table 17.

Programming

Table 17: Extended Status Bits (Variable #83 EXSTATUS)

Bit #	Function	Comment
0	Reserved	
1	Velocity in specified window	Velocity in limits as per parameter #59: VAR_VLIMIT_SPEEDWND
2-4	Reserved	
5	Velocity at 0 (zero)	Velocity 0: Zero defined by parameter #58: VAR_VLIMIT_ZEROSPEED
6,7	Reserved	
8	Bus voltage below under-voltage limit	Utilized to indicate drive is operating from +24V keep alive and a valid DC bus voltage level is not present.
9,10	Reserved	
11	Regen circuit is on	Drive regeneration circuit is active. Drive will be dissipating power through the braking resistor (if fitted).
12-20	Reserved	
21	Set if homing operation in progress	Drive executing Pre-defined homing function (see section 2.15).
22	Set if system homed	Drive completed Pre-defined homing function (see section 2.15).
23	If set then last fault will remain on the display until re-enabled.	User can set this bit to retain fault code on the display until re-enabled. It is useful if there is a fault handler routine. When the fault handler is exited, the fault number on the display will be replaced by current status (usually DiS if bit #23 is not set). Setting bit #23 retains diagnostics on the display.
24	Set if EIP IO exclusive owner connection is established. Cleared if closed.	Checks if drive is controlled by EthernetIP master. Use bit #24 and bit #25 to process "loss of connection" condition (if needed) in the user's program
25	Set if EIP IO exclusive owner connection times out. Cleared if exc. owner conn exists.	Checks if connection with Ethernet/IP master is lost. Use bit #24 and bit #25 to process "loss of connection" condition (if needed) in the user's program
26-31	Reserved	

2.13 Fault Codes (DFAULTS register)

Whenever a fault occurs in the drive, a record of that fault is recorded in the Fault Register (DFAULTS). In addition, specific flags in the System Status Register will be set helping to indicate what class of fault the current fault belongs to. Table 18 summarizes the fault codes. Codes from 1 to 16 are used for DSP subsystem errors. Codes above that range are generated by various subsystems of the PositionServo.

Table 18: DFAULTS Register

Fault ID	Associated flags in status register	Description
1	1, 3	Over voltage
2	1, 3	Invalid Hall sensors code
3	1, 3	Over current
4	1, 3	Over temperature
5	1, 3	The drive is disabled by the ISO 13849-1 Safety Function
6	1, 3	Over speed. (Over speed limit set by motor capability in motor file)
7	1, 3	Position error excess.
8	1, 3	Attempt to enable while motor data array invalid or motor was not selected.
9	1,3	Motor over temperature switch activated
10	1,3	Sub processor error
11-13	-	Reserved
14	1,3	Under voltage (hardware revision 1)
15	1,3	Hardware current trip protection
16	-	Reserved
18	16	Division by zero
19	16	Arithmetic overflow
20	3	Subroutine stack overflow. Exceeded 32 levels subroutines stack depth.

Programming

Fault ID	Associated flags in status register	Description
21	3	Subroutine stack underflow. Executing RETURN statement without preceding call to subroutine.
22	3	Variable evaluation stack overflow. Expression too complicated for compiler to process.
23	21	Motion Queue overflow. 32 levels depth exceeded
24	21	Motion Queue underflow. Last queued MDV statement has non 0 target velocity
25	3	Unknown opcode. Byte code interpreter error; Occurs when program is missing END statement
26	3	Unknown byte code. Byte code interpreter error; Occurs when RETURN statement missing from subroutine; or when EPM data is corrupted at run-time
27	21	Drive disabled. Attempt to execute motion while drive is disabled.
28	16, 21	Accel/Decel too high. Motion statement parameters calculate Accel /Decel value above system capability
29	16, 21	Accel/Decel too low. Motion statement parameters calculate Accel/Decel value below system capability.
30	16, 21	Velocity too high. Motion statement parameters calculate a velocity above the system capability.
31	16, 21	Velocity too low. Motion statement parameters calculate a velocity below the system capability.
32	3,21	Positive limit switch engaged
33	3,21	Negative limit switch engaged
34	3,21	Attempt at positive motion with engaged positive limit switch
35	3,21	Attempt at negative motion with engaged negative limit switch
36	3	Hardware disable (enable input not active when attempting to enable drive from program or interface)
37	3	Under voltage (hardware revision 2)
38	3	EPM loss
39	3,21	Positive soft limit reached
40	3,21	Negative soft limit reached
41	3	Attempt to use variable with unknown ID from user program
45	1,3	Second encoder position error excess
49	1,3	Illegal manipulation of APOS variable

2.14 Limitations and Restrictions

Communication Interfaces Usage Restrictions

Simultaneous connection to the RS485 port is allowed for retransmitting (conversion) between interfaces.



WARNING!

Usage of the RS485 simultaneously with Ethernet may lead to unpredictable behavior since the drive will attempt to perform commands from both interfaces concurrently.

Motion Parameters Limitation

Due to a finite precision in the calculations there are some restrictions for acceleration/deceleration and max velocity for a move. If the programmer receives arithmetic faults during his program's execution, it is likely due to these limitations. Min/Max values are expressed in counts or counts/sample, where the sample is a position loop sample interval (512μsec).

Table 19: Motion Parameter Limits

Parameter	MIN	MAX	Units
Accel / Decel	65/(2 ³²)	512	counts/sample ²
MaxV (maximum velocity)	0	2048	counts/sample
Max move distance	0	+/- 2 ³¹	counts

Stacks and Queues Depth Limitations

Table 20: Stack Depth Limit

Stack/Queue	Motion Queue	Subroutines Stack	Number of Events
Depth	32	32	32

2.15 Homing

2.15.1 What is Homing?

Homing is the method by which a drive seeks the home position (also called the datum, reference point, or zero point). There are various methods of achieving this using:

- limit switches at the ends of travel, or
- a dedicated home switch, or
- an Index Pulse or zero reference from the motor feedback device, or
- a combination of the above.

Predefined (firmware based) homing functionality is available on PositionServo drives with firmware 3.03 or later. In addition custom homing functionality can be created by the programmer within the user program by utilizing the programming command set available.

Examples of custom homing routine creation as well as user program code to replicate each of the predefined homing routines is available from technical support.

2.15.2 The Homing Function

The homing function provides a set of trajectory parameters to the position loop, as shown in Figure 22. They are calculated based on user supplied variable values as listed below:

VAR_HOME_OFFSET
 VAR_HOME_METHOD
 VAR_HOME_SWITCH_INPUT
 VAR_HOME_FAST_VEL
 VAR_HOME_SLOW_VEL
 VAR_HOME_ACCEL
 VAR_START_HOMING

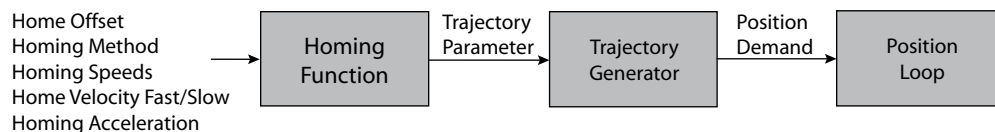


Figure: 22: Homing Function

Homing Function Monitoring:

The extended drive status variable (#83 EXSTATUS variable) contains bit values for monitoring the homing function over a communications interface.

Bit 21 of EXSTATUS indicates homing procedure in progress and is set to logic 1 while homing is being executed.

Bit 22 of EXSTATUS indicates homing complete. It is set to 1 upon the successful completion of the homing routine.

2.15.3 Home Offset

The home offset is the difference between the zero position for the application and the machine home position (found during homing). During homing the home position is found and once the homing is completed the zero position is offset from the home position by adding the home offset to the home position. All subsequent absolute moves are made relative to this new zero position. This is illustrated in Figure 23. Offset can either be set in User Units (UU) by writing to variable #240, or in encoder counts by writing to variable #241. Setting a value for either variable #240 or #241 will result in a value automatically being calculated and stored in the respective variable.

VAR_HOME_OFFSET (#240)
 VAR_HOME_OFFSET_PULSES (#241)

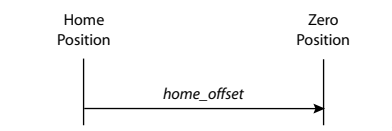


Figure 23: Home Offset

2.15.4 Homing Velocity

There are two homing velocities: fast and slow. These velocity variables are used to find the home switch and to find the index pulse. How the two velocities are implemented within the homing routines depends on the homing routine selected. Refer to section 2.5.9.

VAR_HOME_FAST_VEL (#242)

VAR_HOME_SLOW_VEL (#243)

2.15.5 Homing Acceleration

Homing acceleration establishes the velocity ramp rate to be used for all accelerations and decelerations within the standard homing modes. Note that in the pre-defined homing methods, it is not possible to program a separate deceleration rate.

VAR_HOME_ACCEL (#239)

2.15.6 Homing Switch

The homing switch variable enables the user to select the PositionServo input used for the Home Switch connection. The Homing Switch Input Assignment range is 0 - 11. Inputs A1-A4 are assigned 0 to 3, respectively; inputs B1-B4 are assigned 4 to 7, respectively; and inputs C1-C4 are assigned 8 to 11, respectively.

VAR_HOME_SWITCH_INPUT (#246)



WARNING!

- Setting inputs A1 and A2 as the home switch, even in methods that do NOT use limit switches can cause the drive to behave in an unexpected manner.
- Input A3 is a dedicated hardware enable input and should never be assigned as the homing switch input.
- Input C3 can be used as the homing switch input only in methods that do not home to an index pulse.

2.15.7 Homing Start

There are two methods of starting pre-defined homing operation, the 'HOME' command and the Var_Start_Homing variable. When Homing is initiated from the user program the 'HOME' command should always be used. The HOME command is a blocking instruction that prevents further execution of the Main Program until homing operation is completed. Any events that are enabled whilst homing is carried out will continue to process.



WARNING!

If using firmware prior to 4.50 then execution of homing functionality does not prevent simultaneous execution of subsequent programming statements and it is required to immediately follow the HOME command with the following code line:

```
WAIT UNTIL VAR_EXSTATUS & 0x400000 == 0x400000.
```

Doing this ensures no further lines of code will be executed until homing is complete.

The home start variable (Var_Start_Homing) is used to initiate pre-defined homing functionality from a host interface. It should not be used if the drive contains or is executing a user program. Var_Start_Homing range is: 0 or 1. When set to 0, no action occurs. When set to 1, the homing operation is started.

VAR_START_HOMING (#245)

Programming

2.15.8 Homing Method

VAR_HOME_METHOD (#244)

The Home Method variable establishes the method that will be used for homing. All supported methods are summarized in Table 21 and described in sections 2.15.9.1 through 2.15.9.25. These homing methods define the required operation of the drive in location of the home position. The zero position is always the home position adjusted by the homing offset.

Table 21: Homing Methods

Method	Home Position
0	No operation/reserved. An attempt to execute 0 will result in execution of method 1.
1	Location of first index pulse is on the positive side of the negative limit switch.
2	Location of first index pulse is on the negative side of the positive limit switch.
3	Location of first index pulse is on the negative side of a positive home switch. ¹
4	Location of first index pulse is on the positive side of a positive home switch. ¹
5	Location of first index pulse is on the positive side of a negative home switch. ²
6	Location of first index pulse is on the negative side of a negative home switch. ²
7	Location of first index pulse is on the negative side of the negative edge of an intermittent home switch. ³
8	Location of first index pulse is on the positive side of the negative edge of an intermittent home switch. ³
9	Location of first index pulse is on the negative side of the positive edge of an intermittent home switch. ³
10	Location of first index pulse is on the positive side of the positive edge of an intermittent home switch. ³
11	Location of first index pulse is on the positive side of the positive edge of an intermittent home switch. ³
12	Location of first index pulse is on the negative side of the positive edge of an intermittent home switch. ³
13	Location of first index pulse is on the positive side of the negative edge of an intermittent home switch. ³
14	Location of first index pulse is on the negative side of the negative edge of an intermittent home switch. ³
15	Reserved for future use.
16	Reserved for future use
17	The edge of a negative limit switch.
18	The edge of a positive limit switch.
19	The edge of a positive home switch.
20	Reserved for future use.
21	The edge of a negative home switch.
22	Reserved for future use.
23	Positive edge of an intermittent home switch.
24	Reserved for future use.
25	The negative edge of an intermittent home switch.
26	Reserved for future use.
27	Negative edge of an intermittent home switch.
28	Reserved for future use.
29	The positive edge of an intermittent home switch.
30	Reserved for future use.
31	Reserved for future use.
32	Reserved for future use.
33	The first index pulse on the negative side of the current position.
34	The first index pulse on the positive side of the current position.
35	Current position becomes home position. Home offset is also active and will be added to current position to set the zero position.

1 - A positive home switch is one that goes active at a set position, and remains active for all positions greater than the set position.

2 - A negative home switch is one that goes active at a set position, and remains active for all positions less than the set position.

3 - An intermittent home switch is one that is only active for a limited range of travel.

2.15.9 Homing Methods

There are several types of homing methods but each method establishes the:

- Homing signal (positive limit switch, negative limit switch, home switch, or index pulse)
- Direction of actuation and, where appropriate, the direction of the index pulse.

The homing method descriptions and diagrams in this manual are based on those in the CANopen Profile for Drives and Motion Control (DSP 402). As illustrated in Figure 24, each homing method diagram shows the motor in the starting position on a mechanical stage. The arrow line indicates direction of motion and the circled number indicates the homing method (the mode selected by the Homing Method variable).

The location of the circled method number indicates the home position reached with that method. The text designators (A, B) indicate the logical transition required for the homing function to complete its current phase of motion. Dashed lines overlay these transitions and reference them to the relevant transitions of limit switches, homing sensors, or index pulses.

Definitions

Positive home switch: goes active at a set position, and remains active for all positions greater than the set position.

Negative home switch: goes active at a set position, and remains active for all positions less than the set position.

Intermittent home switch: is one that is only active for a limited range of travel.

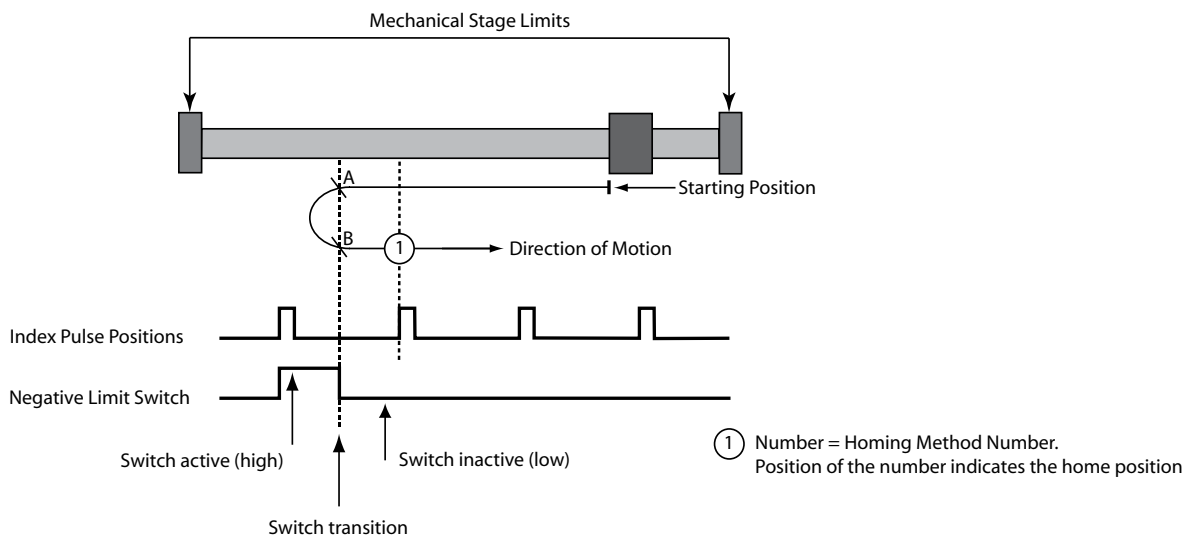


Figure 24: Homing Terms



NOTE

In the homing method descriptions, negative motion is leftward and positive motion is rightward

BLUE lines indicate fast velocity moves

GREEN lines indicate slow velocity moves

RED lines indicate slow velocity/100 moves

2.15.9.1 Homing Method 1: Homing on the Negative Limit Switch & Index Pulse

Using this method, the initial direction of movement is negative if the negative limit switch is inactive (here shown as low). The home position is at the first index pulse to the positive side of the position where the negative limit switch becomes active.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Negative Limit Switch (A1) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity. If the negative limit switch is already active when the homing routine commences then this initial move is not executed. Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until first the falling edge of the negative limit switch is detected (position B) and then the rising edge of the first index pulse (position 1) is detected.

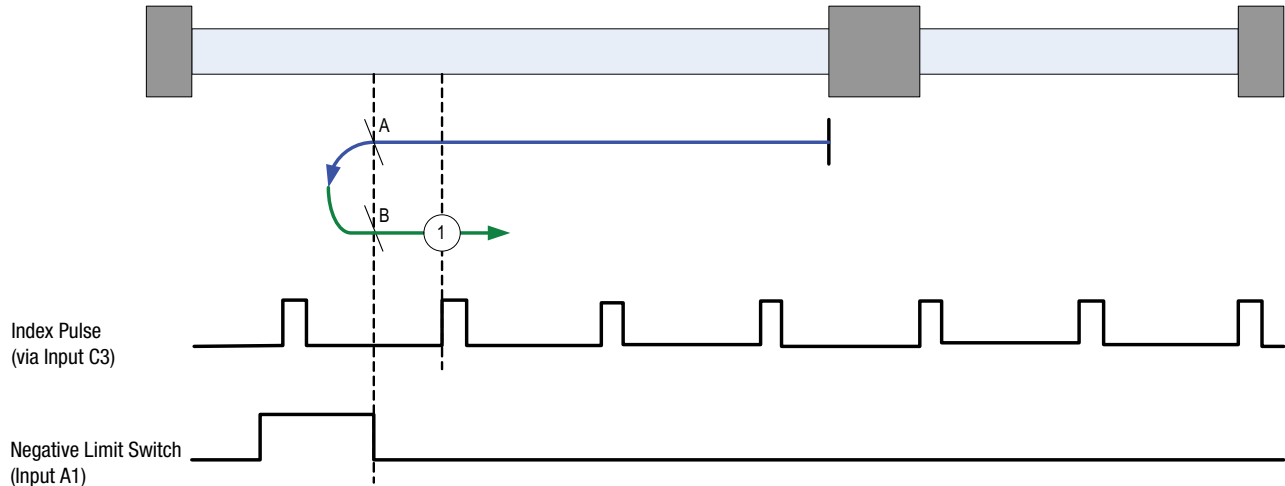


Figure 25: Homing Method 1

2.15.9.2 Homing Method 2: Homing on the Positive Limit Switch & Index Pulse

Using this method the initial direction of movement is positive if the positive limit switch is inactive (here shown as low). The position of home is at the first index pulse to the negative side of the position where the positive limit switch becomes active.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Positive Limit Switch (A2) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity. If the positive limit switch is already active when the homing routine commences then this initial move is not executed. Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until first the falling edge of the positive limit switch is detected (position B) and then the rising edge of the first index pulse (position 2) is detected.

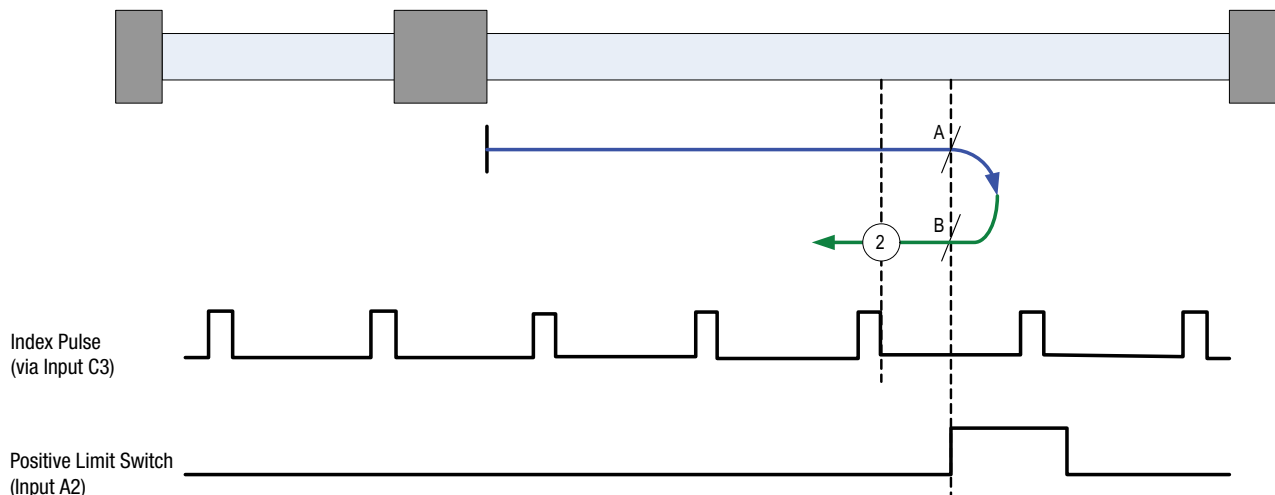


Figure 26: Homing Method 2

2.15.9.3 Homing Method 3: Homing on the Positive Home Switch & Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is inactive). The home position is the first index pulse to the negative side of the position where the homing switch becomes active.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity. If the homing switch is already active when the homing routine commences then this initial move is not executed. Axis will then accelerate to **fast** homing velocity in negative direction. Motion will continue until first the falling edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 3) is detected.

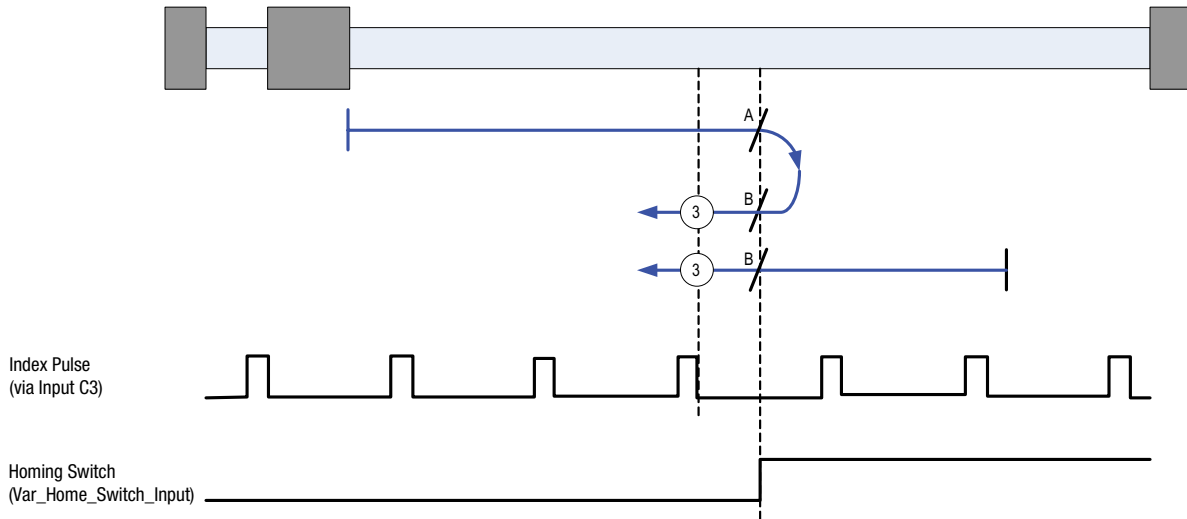


Figure 27: Homing Method 3

2.15.9.4 Homing Method 4: Homing on the Positive Home Switch & Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is active). The home position is the first index pulse to the positive side of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity. If the homing switch is already inactive when the homing routine commences then this initial move is not executed. Axis will then accelerate to **fast** homing velocity in positive direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 4) is detected.

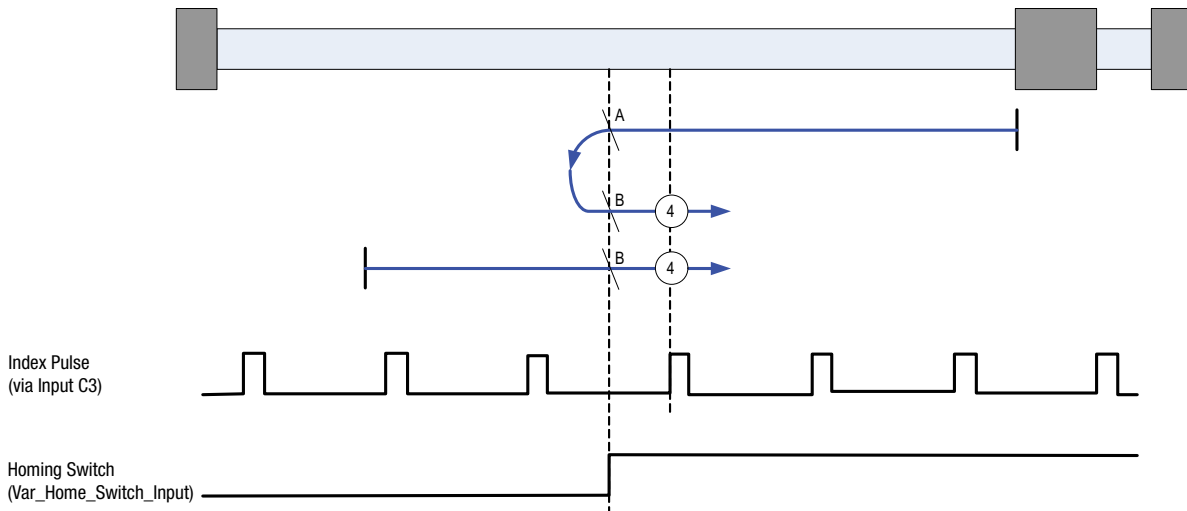


Figure 28: Homing Method 4

2.15.9.5 Homing Method 5: Homing on the Negative Home Switch & Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is inactive). The home position is the first index pulse to the positive side of the position where the homing switch becomes active.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity. If the homing switch is already active when the homing routine commences then this initial move is not executed. Axis will then accelerate to **fast** homing velocity in positive direction. Motion will continue until first the falling edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 5) is detected.

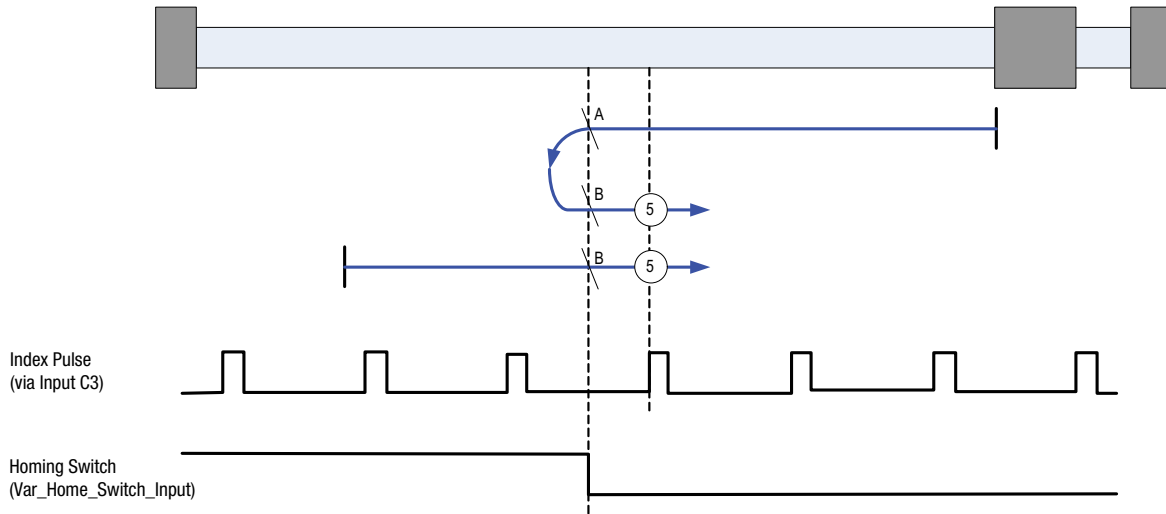


Figure 29: Homing Method 5

2.15.9.6 Homing Method 6: Homing on the Negative Home Switch & Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is active). The home position is the first index pulse to the negative side of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity. If the homing switch is already inactive when the homing routine commences then this initial move is not executed. Axis will then accelerate to **fast** homing velocity in negative direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 6) is detected.

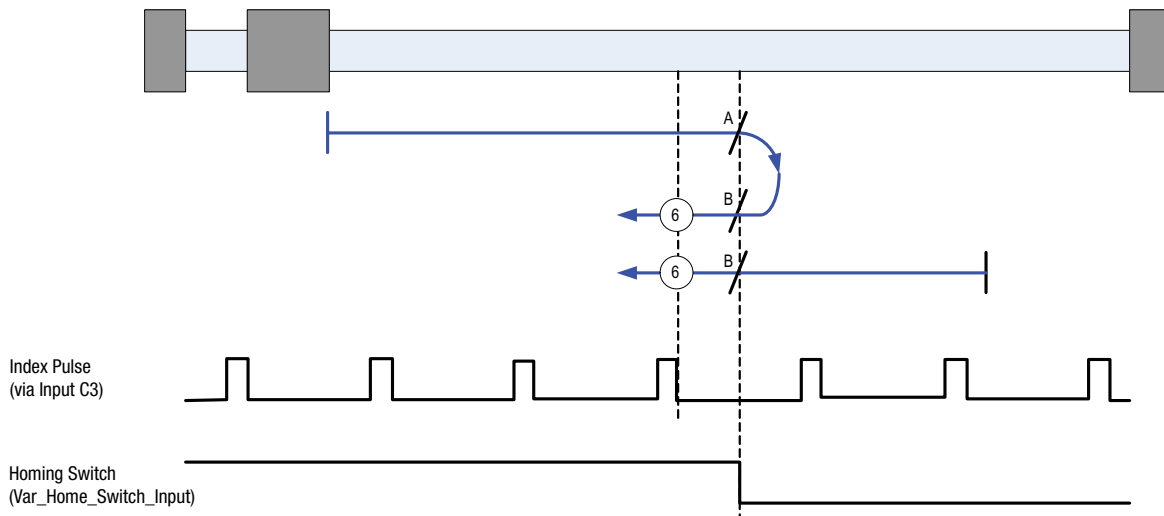


Figure 30: Homing Method 6

2.15.9.7 Homing Method 7: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is inactive). The home position is the first index pulse to the negative side of the position where the homing switch becomes active.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until first the falling edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 7) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) following the procedure as detailed above, but moving negative instead of positive and without stopping on detection of the homing switch rising edge.

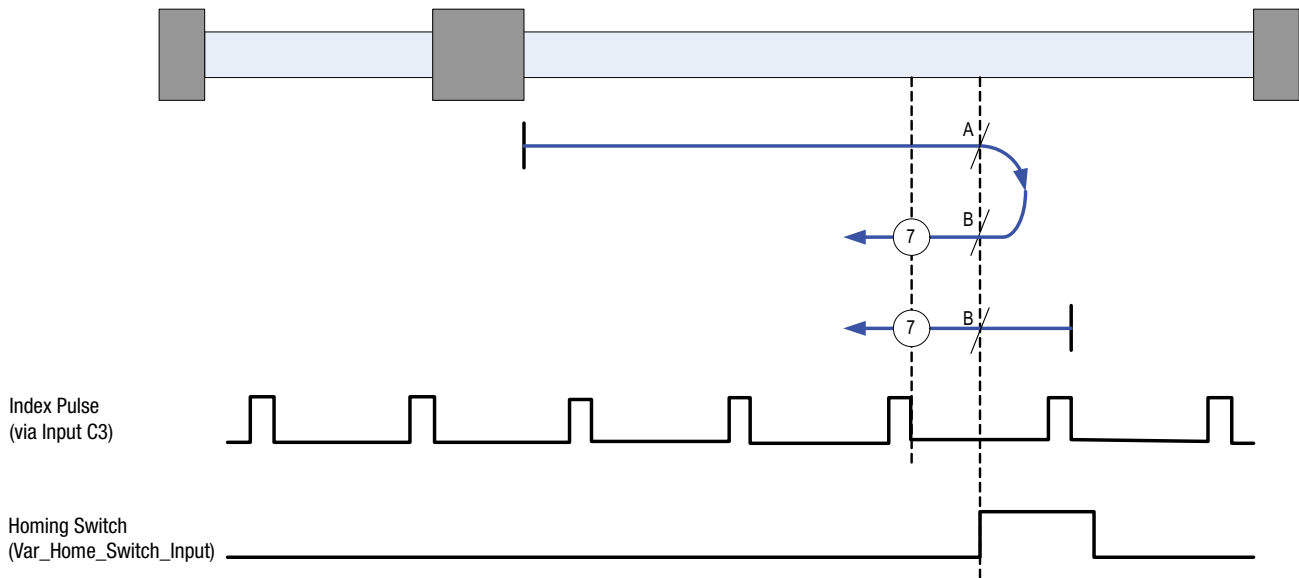


Figure 31: Homing Method 7

2.15.9.8 Homing Method 8: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is active). The home position is the first index pulse to the positive side of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already inactive when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 8) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) following the procedure as detailed above.

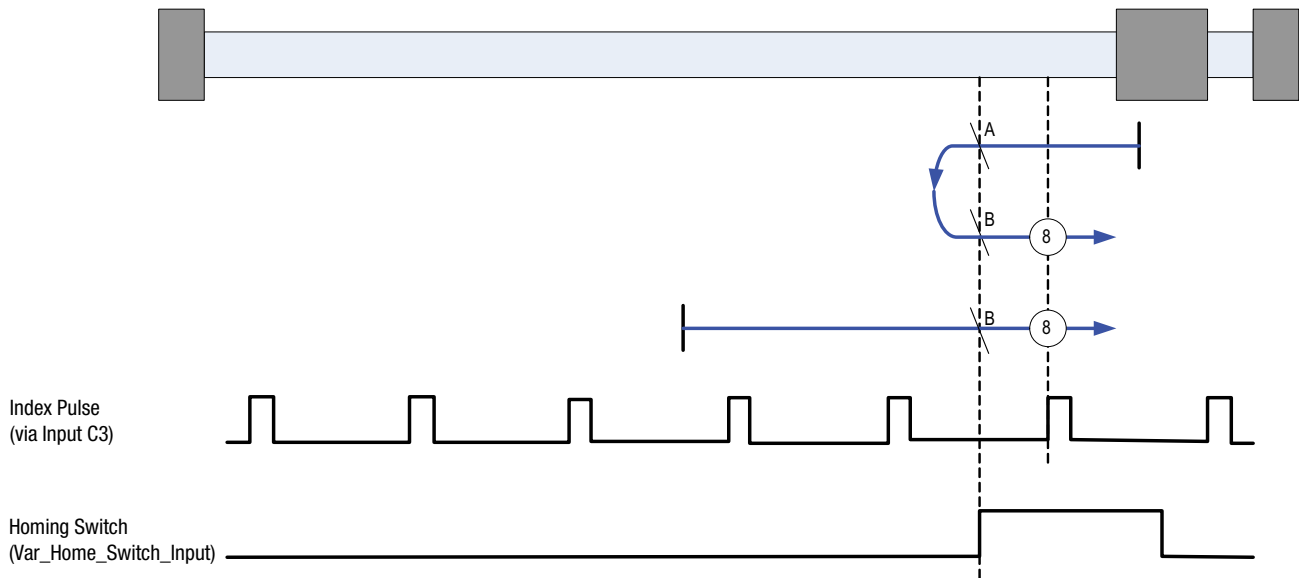


Figure 32: Homing Method 8

2.15.9.9 Homing Method 9: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is positive. The home position is the first index pulse to the negative side of the position where the homing switch becomes inactive on its negative edge.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 9) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) following the procedure as detailed above but ignoring the initial move in the positive direction.

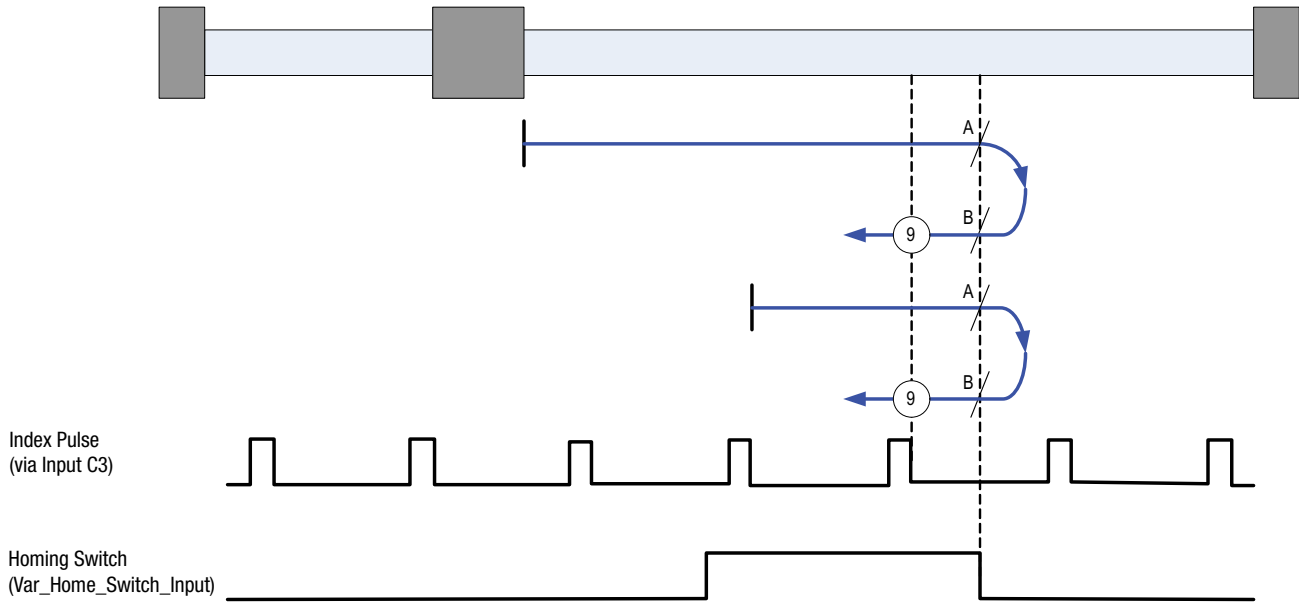


Figure 33: Homing Method 9

2.15.9.10 Homing Method 10: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is positive. The home position is the first index pulse to the positive side of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will continue running at **fast** homing velocity in the positive direction until the rising edge of the first index pulse (position 10) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) continuing motion until it sees the rising edge of the homing switch. The axis will then stop and follow the procedure as detailed above.

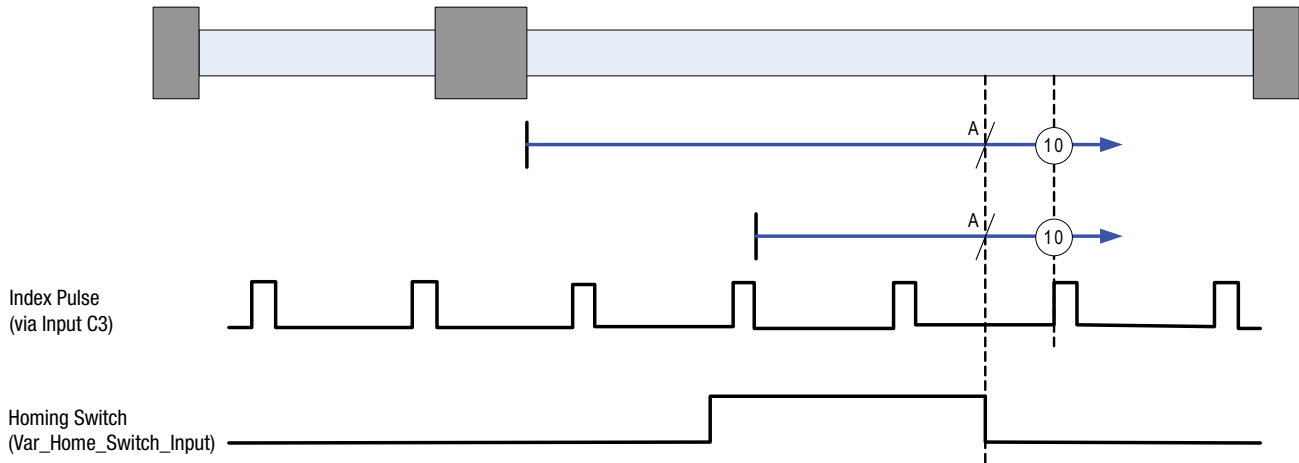


Figure 34: Homing Method 10

2.15.9.11 Homing Method 11: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is negative (if the homing switch is inactive). The home position is the first index pulse to the positive side of the position where the homing switch becomes active.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until first the falling edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 11) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) following the procedure as detailed above, but moving positive instead of negative and without stopping on detection of the homing switch rising edge.

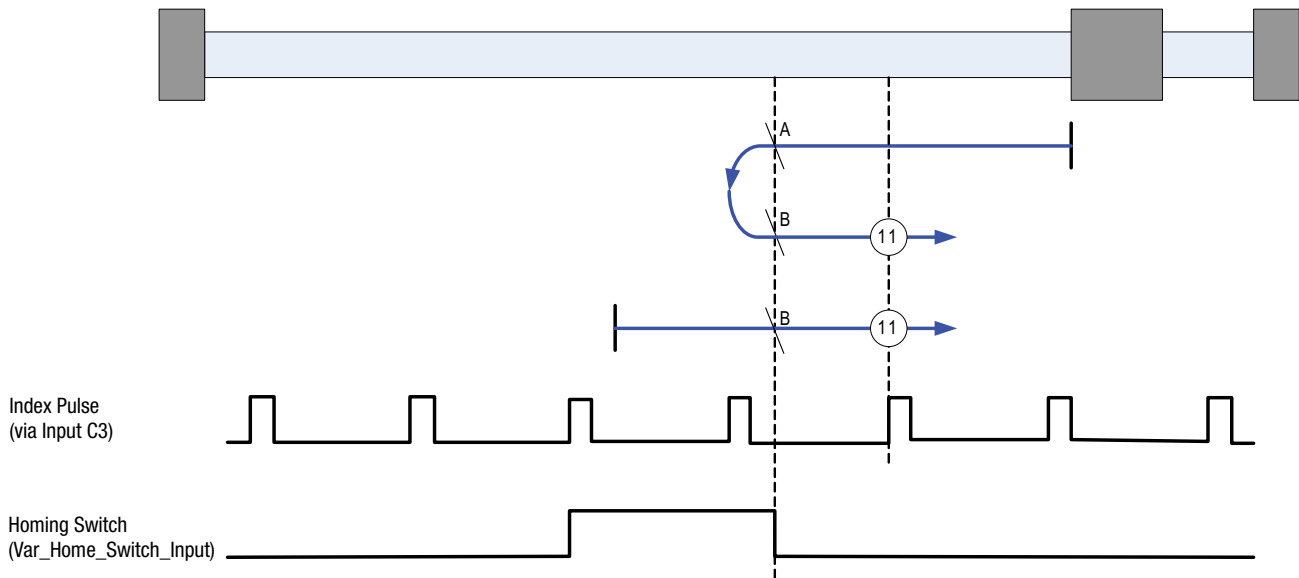


Figure 35: Homing Method 11

2.15.9.12 Homing Method 12: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is positive (if the homing switch is active). The home position is the first index pulse to the negative side of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already inactive when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 12) is detected.

NOTE: if it the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) following the procedure as detailed above.

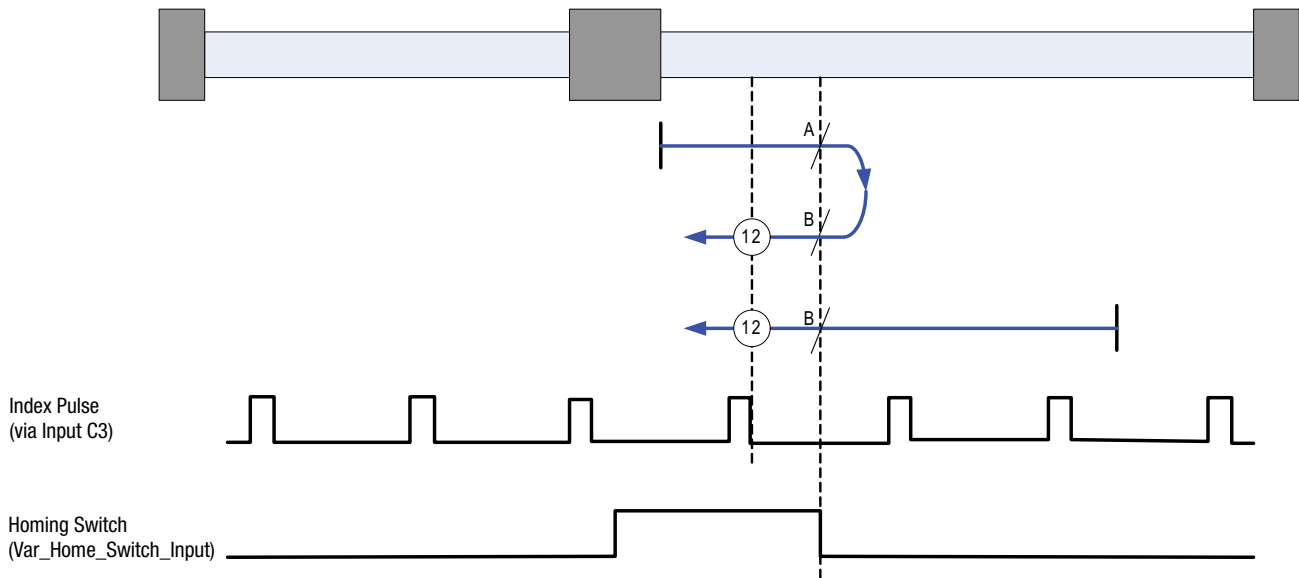


Figure 36: Homing Method 12

2.15.9.13 Homing Method 13: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is negative. The home position is the first index pulse to the positive side of the position where the homing switch becomes inactive on its positive edge.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until first the rising edge of the Homing switch is detected (position B) and then the rising edge of the first index pulse (position 13) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) following the procedure as detailed above but ignoring the initial move in the negative direction.

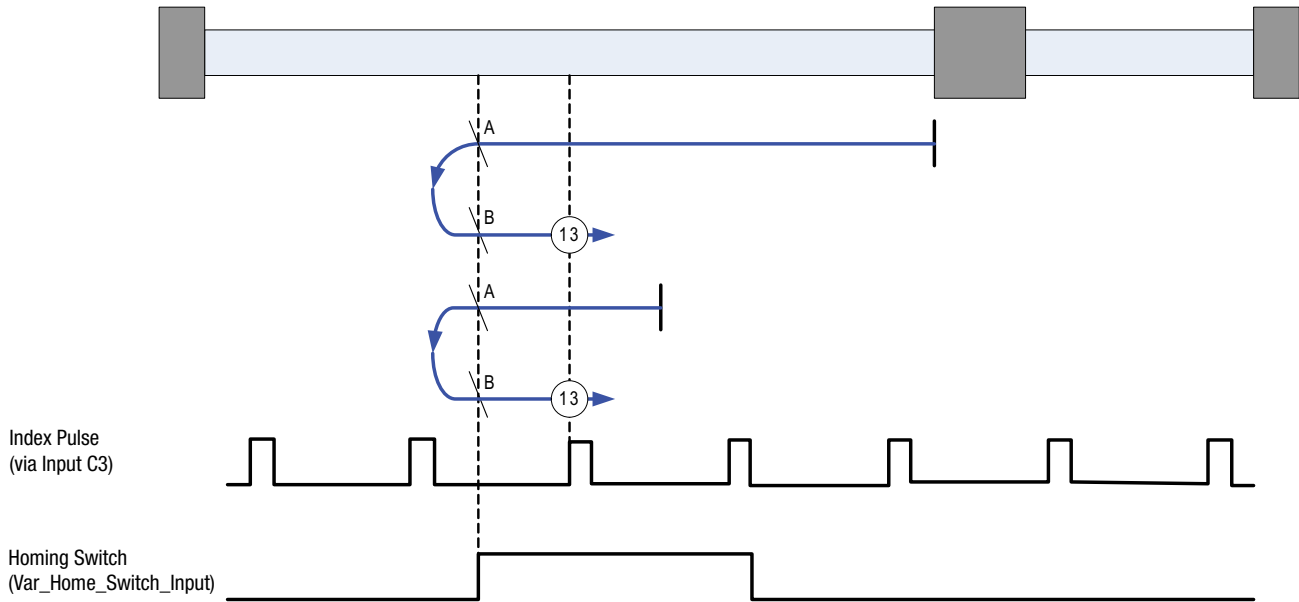


Figure 37: Homing Method 13

2.15.9.14 Homing Method 14: Homing on the Home Switch & Index Pulse

Using this method the initial direction of movement is negative. The home position is the first index pulse to the negative side of the position where the homing switch becomes inactive.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will continue running at **fast** homing velocity in the negative direction until the rising edge of the first index pulse (position 14) is detected.

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) continuing motion until it sees the rising edge of the homing switch. The axis will then stop and follow the procedure as detailed above.

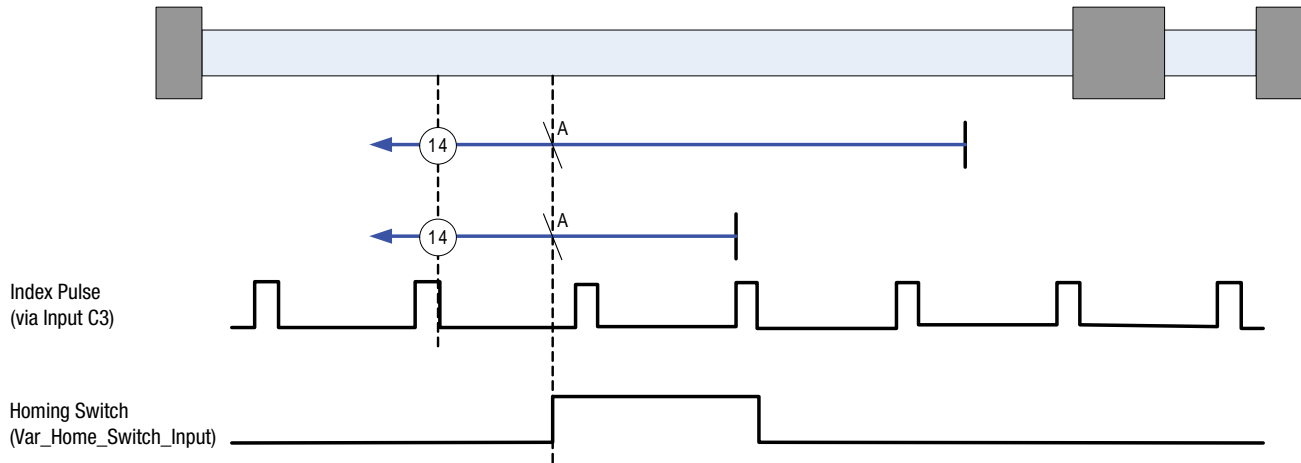


Figure 38: Homing Method 14

2.15.9.15 Homing Method 17: Homing to Negative Limit Switch (without index pulse)

Method 17 is similar to method 1, except that the home position is not dependent on the index pulse but only on the negative limit switch translation.

Using this method the initial direction of movement is negative. The home position is the leading edge of the Negative limit switch.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Negative Limit Switch (A1) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the negative limit switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until the falling edge of the negative limit switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the rising edge of the negative limit switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity **divided by 100** in the positive direction. Motion will continue until the falling edge of the negative limit switch is detected (position 17). This is the home position (excluding offset).

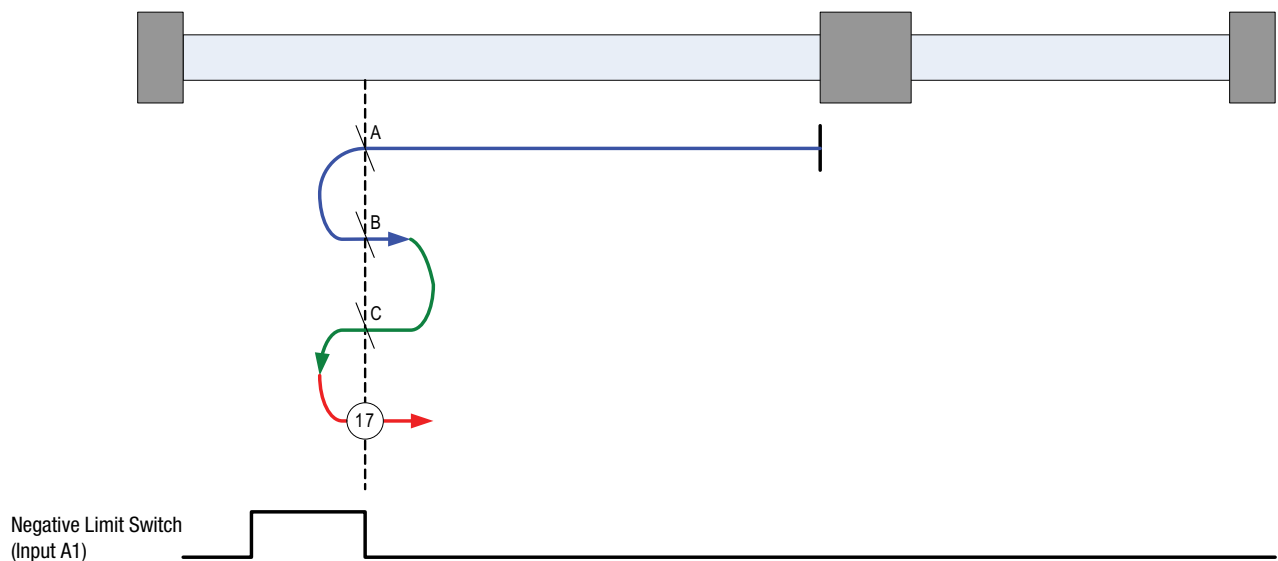


Figure 39: Homing Method 17

2.15.9.16 Homing Method 18: Homing to Positive Limit Switch (without index pulse)

Method 18 is similar to method 2, except that the home position is not dependent on the index pulse but only on the Positive limit switch translation.

Using this method the initial direction of movement is positive. The home position is the leading edge of the Positive limit switch.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Positive Limit Switch (A2) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the positive limit switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until the falling edge of the positive limit switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the rising edge of the positive limit switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity **divided by 100** in the negative direction. Motion will continue until the falling edge of the positive limit switch is detected (position 18). This is the home position (excluding offset).

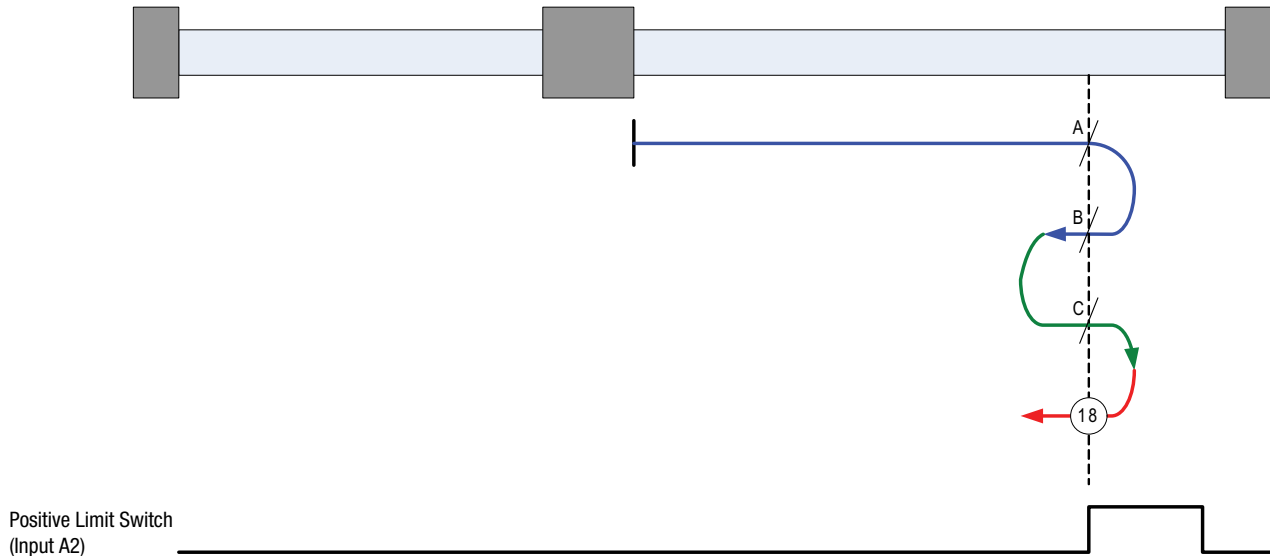


Figure 40: Homing Method 18

2.15.9.17 Homing Method 19: Homing to Homing Switch (without index pulse)

Using this method the initial direction of movement is positive (if the homing switch is inactive). The home position is the leading edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until the homing switch is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the rising edge of the homing switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position 19). This is the home position (excluding offset).

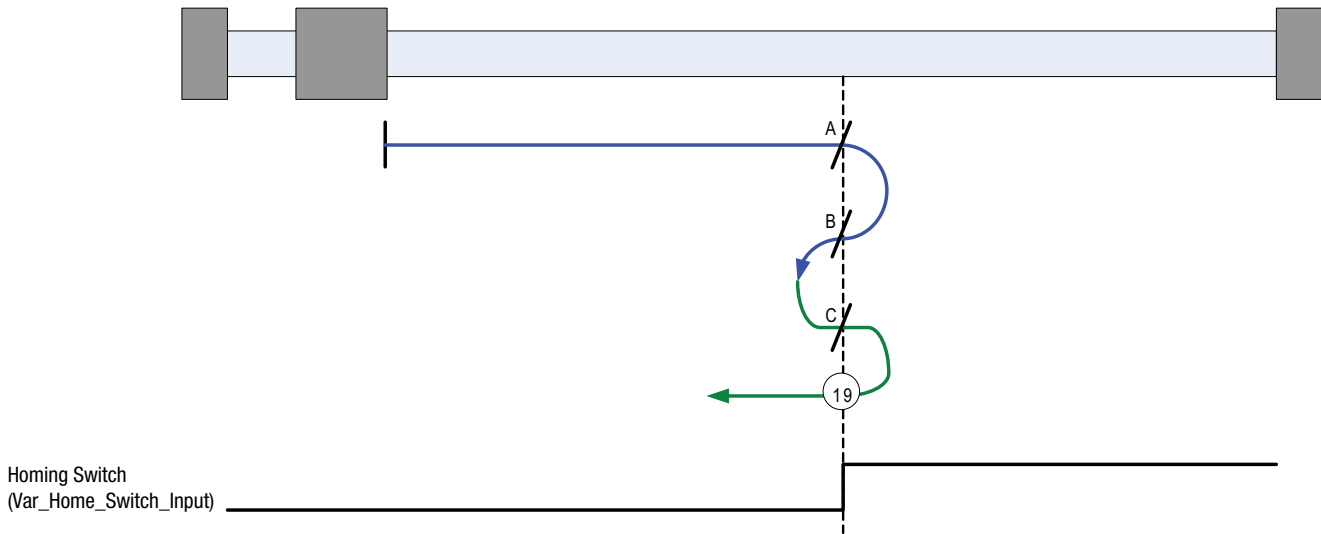


Figure 41: Homing Method 19

2.15.9.18 Homing Method 21: Homing to Homing Switch (without index pulse)

Using this method the initial direction of movement is negative (if the homing switch is inactive). The home position is the leading edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until the homing switch is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the rising edge of the homing switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position 21). This is the home position (excluding offset).

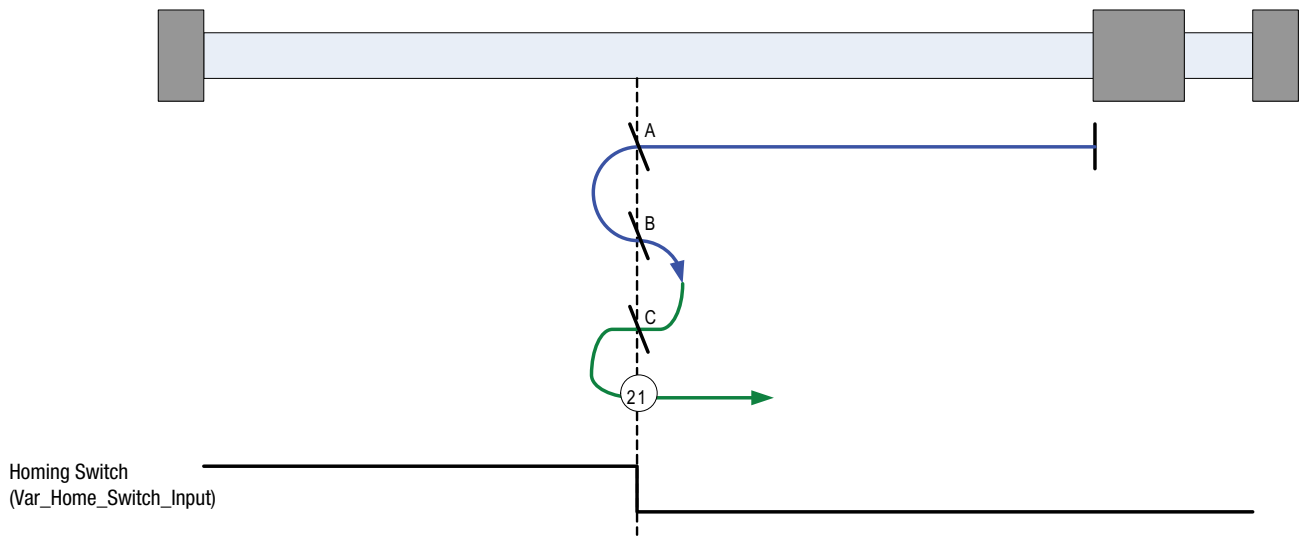


Figure 42: Homing Method 21

2.15.9.19 Homing Method 23: Homing to Homing Switch (without index pulse)

Using this method the initial direction of movement is positive (if the homing switch is inactive). The home position is the leading edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until the homing switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the rising edge of the homing switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position 23). This is the home position (excluding offset).

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) following the procedure as detailed above but ignoring the initial move in the positive direction.

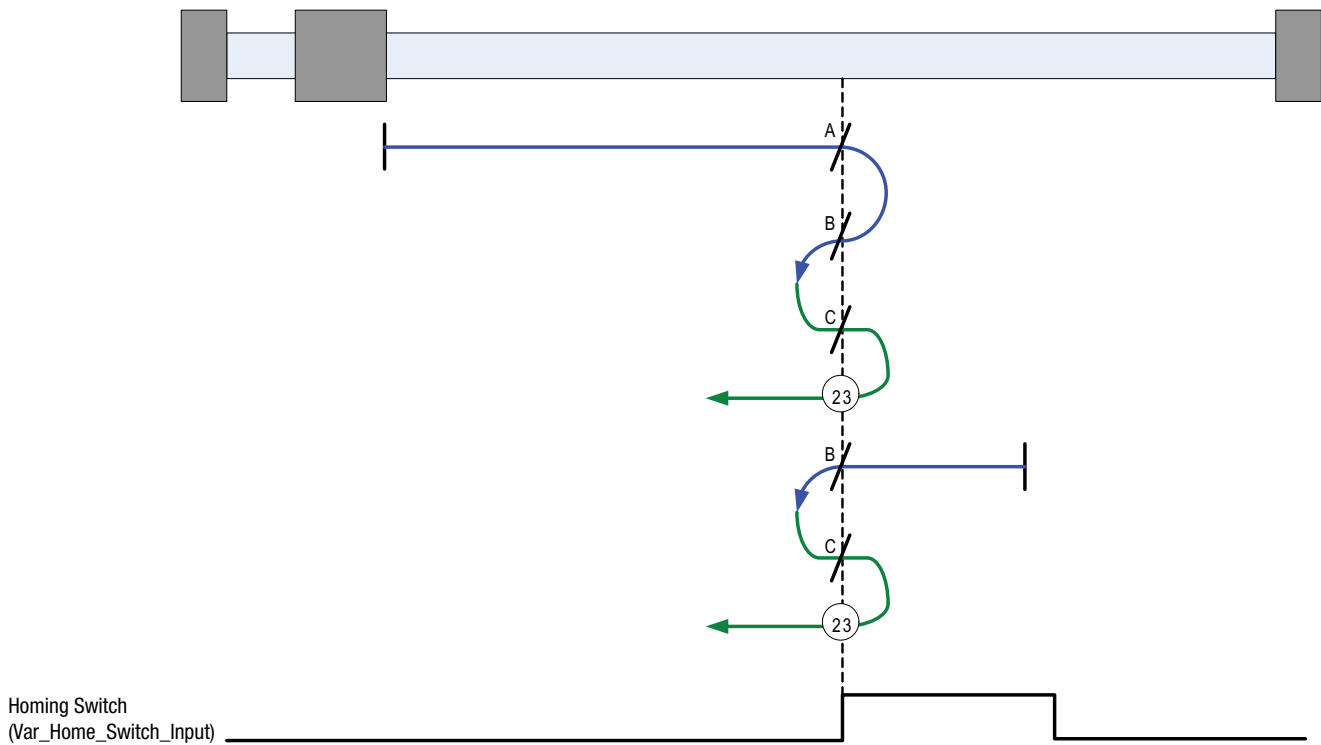


Figure 43: Homing Method 23

2.15.9.20 Homing Method 25: Homing to Homing Switch (without index pulse)

Using this method the initial direction of movement is positive. The home position is the negative edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the positive direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the rising edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position 25). This is the home position (excluding offset).

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move positive until it contacts the positive limit switch (A2). Upon activating the positive limit switch the axis will change direction (negative) continuing motion until it sees the rising edge of the homing switch. The axis will then stop and follow the procedure as detailed above.

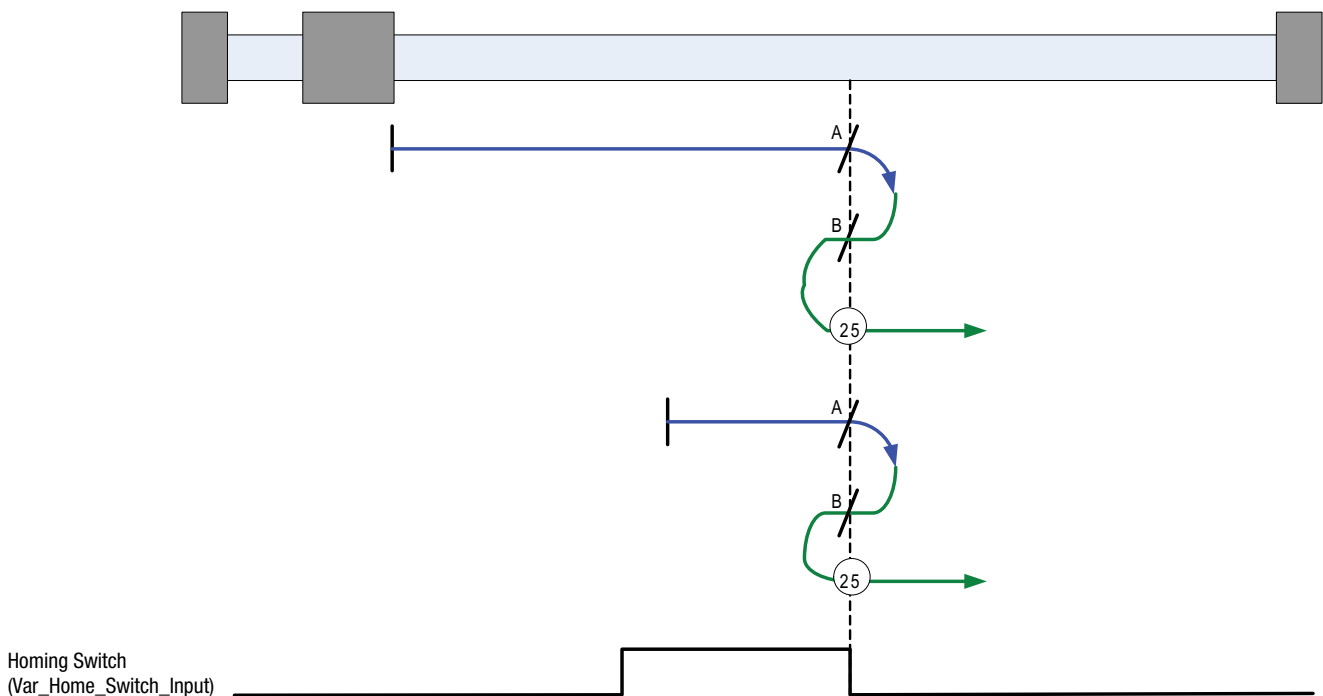


Figure 44: Homing Method 25

2.15.9.21 Homing Method 27: Homing to Homing Switch (without index pulse)

Using this method the initial direction of movement is negative. The home position is the negative edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until Homing Switch (selectable via Var_Home_Switch_Input Variable) is deactivated (falling edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this does not effect this mode of homing as the procedure is searching for falling edge of homing switch in both cases.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the rising edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the falling edge of the homing switch is detected (position 27). This is the home position (excluding offset).

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) continuing motion until it sees the rising edge of the homing switch. The axis will then stop and follow the procedure as detailed above.

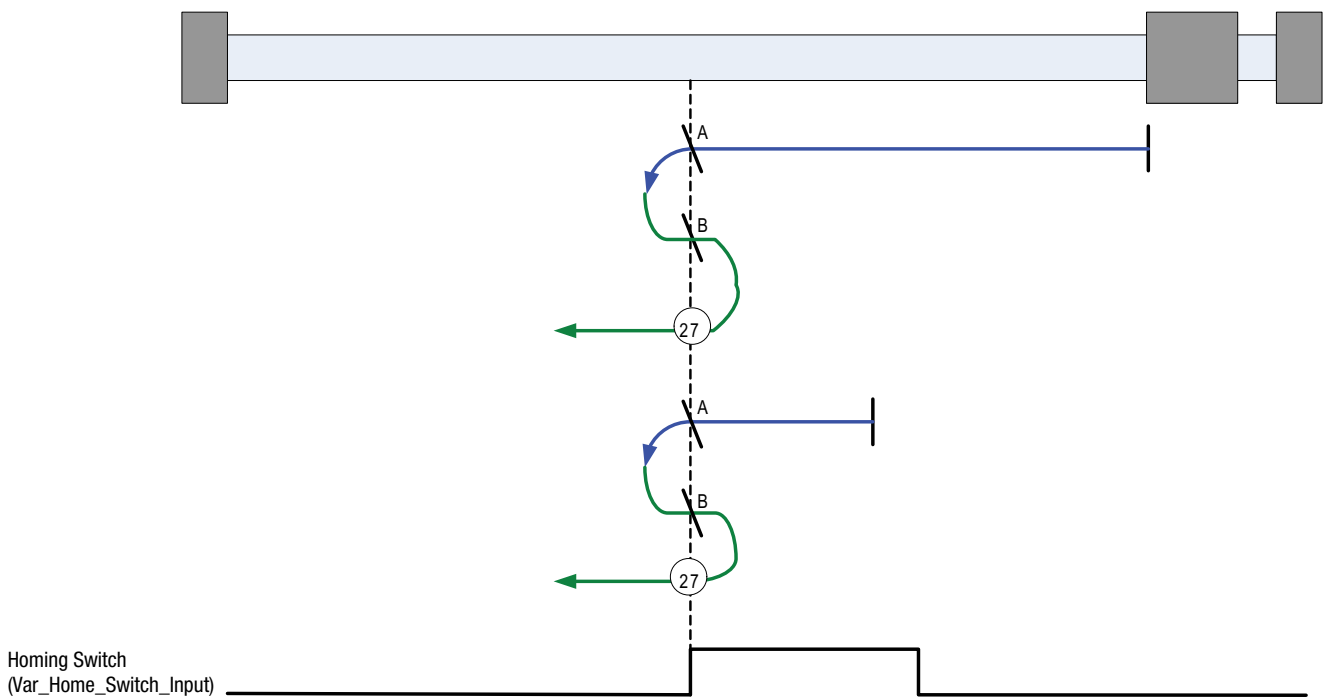


Figure 45: Homing Method 27

Programming

2.15.9.22 Homing Method 29: Homing to Homing Switch (without index pulse)

Using this method the initial direction of movement is negative (if the homing switch is inactive). The home position is the leading edge of the homing switch.

Axis will accelerate to **fast** homing velocity in the negative direction and continue until the homing switch (selectable via Var_Home_Switch_Input Variable) is activated (rising edge) shown at position A. Axis then decelerates to zero velocity.

If the homing switch is already active when the homing routine commences then this initial move is not executed.

Axis will then accelerate to **fast** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position B), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the negative direction. Motion will continue until the rising edge of the homing switch is detected (position C), where the axis will decelerate to 0 velocity.

Axis will then accelerate to **slow** homing velocity in the positive direction. Motion will continue until the falling edge of the homing switch is detected (position 29). This is the home position (excluding offset).

NOTE: if the axis is on the wrong side of the homing switch when homing is started then the axis will move negative until it contacts the negative limit switch (A1). Upon activating the negative limit switch the axis will change direction (positive) following the procedure as detailed above but ignoring the initial move in the negative direction.

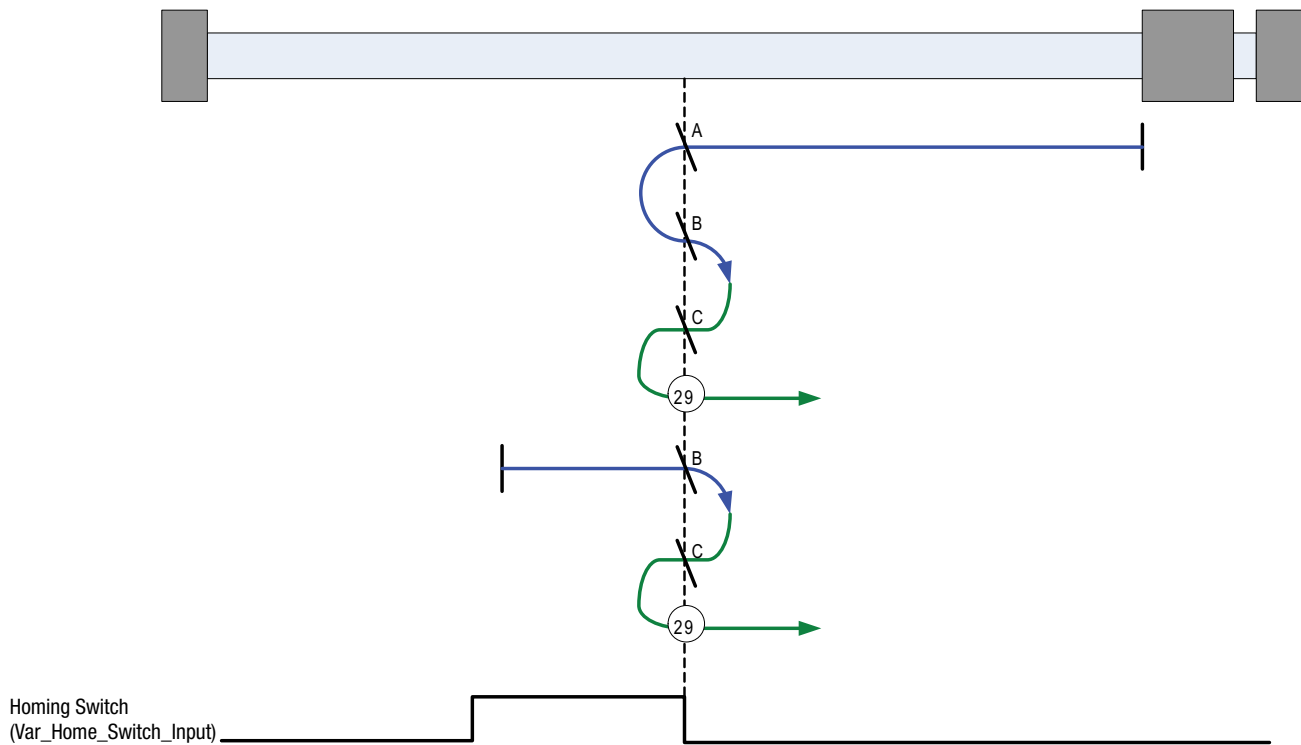


Figure 46: Homing Method 29

2.15.9.23 Homing Method 33: Homing to an Index Pulse

Using this method the initial direction of movement is negative. The home position is the first index pulse to the negative side of the shaft starting Position. Axis will accelerate to **fast** homing velocity in the negative direction and continue until the rising edge of the first index pulse (position 33) is detected.

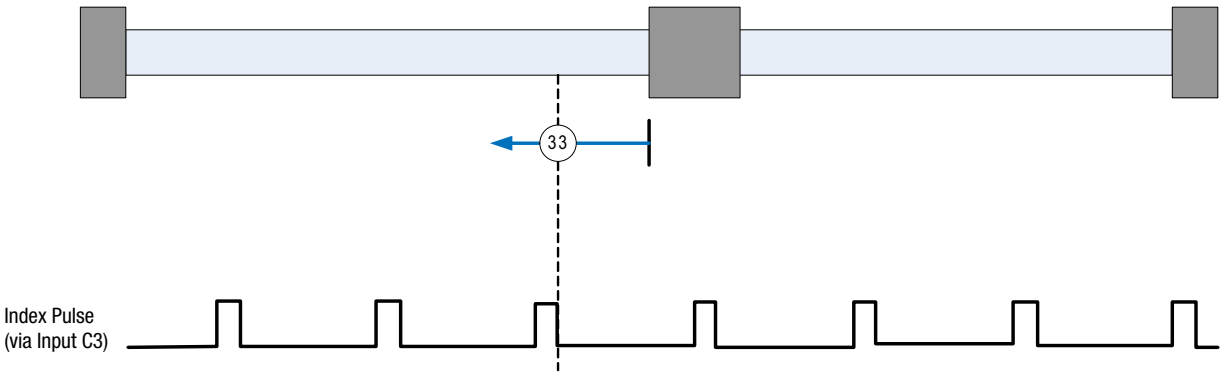


Figure 47: Homing Method 33

2.15.9.24 Homing Method 34: Homing to an Index Pulse

Using this method the initial direction of movement is positive. The home position is the first index pulse to the positive side of the shaft starting Position. Axis will accelerate to **fast** homing velocity in the positive direction and continue until the rising edge of the first index pulse (position 34) is detected.

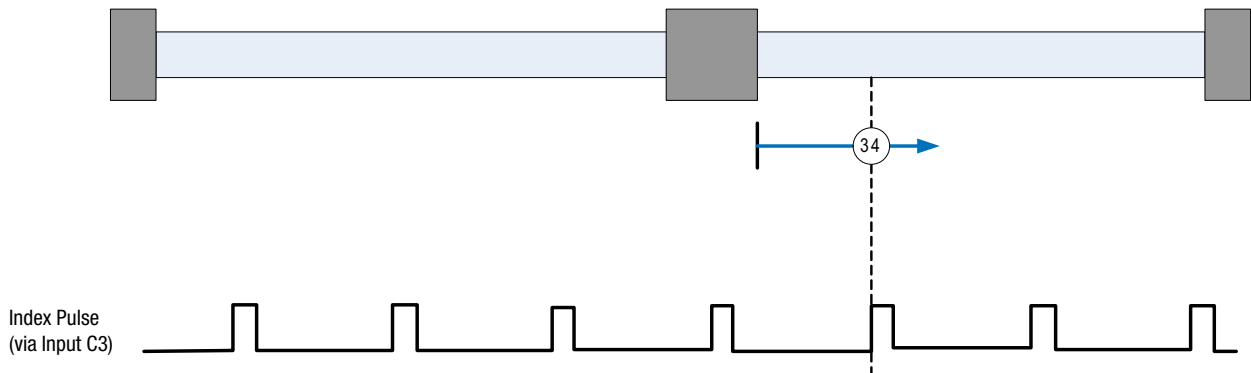


Figure 48: Homing Method 34

2.15.9.25 Homing Method 35: Using Current Position as Home

Using this method the current position of the axis is taken as the home position. There is no motion of the motor shaft during this procedure. Any offset specified (via the Var_Home_Offset Variable) will be added to the shaft's present position to create the home/zero position.

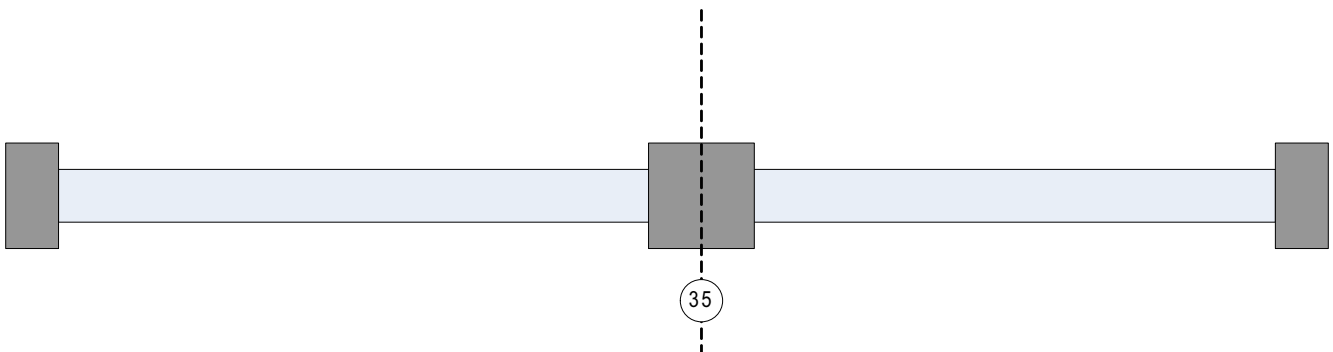


Figure 49: Homing Method 35

Programming

2.15.10 Homing Mode Operation Example

The following steps are needed to execute the homing operation from the user program or under interface control.

1. Set Fast homing speed: Variable #242
2. Set Slow homing speed: Variable #243
3. Set Homing accel/decel: Variable #239
4. Set home offset:
 - a. In User Units Variable #240
 - b. In encoder pulses Variable #241
5. Set Home Switch Input Variable #246
6. Select Home Method Variable #244

To start, execute the HOME command. Refer to the example herein.

There are two methods of starting pre-defined homing operation, the 'HOME' command and the Var_Start_Homing variable. When Homing is initiated from the user program the 'HOME' command should always be used. The HOME command is a blocking instruction that prevents further execution of the Main Program until homing operation is completed. Any events that are enabled whilst homing is carried out will continue to process.



WARNING!

If using firmware prior to 4.50 then execution of homing functionality does not prevent simultaneous execution of subsequent programming statements and it is required to immediately follow the HOME command with the following code line:

```
WAIT UNTIL VAR_EXSTATUS & 0x400000 == 0x400000.
```

Doing this ensures no further lines of code will be executed until homing is complete.

The home start variable (Var_Start_Homing) is used to initiate pre-defined homing functionality from a host interface. It should not be used if the drive contains or is executing a user program. Var_Start_Homing range is: 0 or 1. When set to 0, no action occurs. When set to 1, the homing operation is started.

```
;Program start-----  
;  
;  
    UNITS=1                ; rps  
  
    Accel=1000  
    Decel=1000  
    MaxV =20  
  
;some program statements..  
;  
;  
;Homing specific set up  
    VAR_HOME_FAST_VEL=    10        ; rps  
    VAR_HOME_SLOW_VEL=    1         ; rps  
    VAR_HOME_ACCEL=       100       ; rps/sec^2  
    VAR_HOME_OFFSET=      0         ; no offset from sensor  
    VAR_HOME_SWITCH_INPUT= 4         ; input B1 (0-A1, 1-A2...3-A4, 4-B1,...11-C4)  
    VAR_HOME_METHOD=      4         ; see table 21  
    ENABLE  
    HOME                  ; start homing sequence  
;The statement below MUST be included immediately after the Home command on drives containing  
;firmware releases prior to version 4.50  
    WAIT UNTIL VAR_EXSTATUS & 0x400000 == 0x400000 ; wait for homing complete  
;Drive homed  
  
;Program statements..  
END
```

3. Reference

3.1 Program Statement Glossary

Each programming statement is documented using the tabular format shown in Tables 22 and 23. The individual program statements are listed in this section in alphabetical order with detailed descriptions in Tables 24 through 62.

Table 22: Language Format

KEYWORD	Description	Type
Purpose		
Syntax	KEYWORD <ARGUMENTS> ,[MODIFIERS]	
Remarks		
See Also		
Example		

Table 23: Field Descriptions

Field	Descriptions
KEYWORD:	The KEYWORD is the name of the programming statement as it would appear in a program.
Description:	The description is an interpretation of the keyword. For example: MOVEP is the keyword and Move to Position would be a description. The description is provided only as an aid to the reader and may not be used in a program.
Type:	The type field will identify the Keyword as either a Statement or a Pseudo statement. Statements are actual instructions converted to machine code by the compiler and form executable commands within the drive programming. Pseudo statements add convenience to the programmer but do not form instructions in their own right. They are therefore not executable code and are effectively removed when the program is compiled to it's native state by the compiler.
Purpose:	Purpose or Function of the Keyword (Programming Statement).
Syntax:	This field shows proper usage of the keyword. Arguments will be written in < > brackets. Optional arguments will be contained within [] brackets.
Arguments:	The data that is supplied with a statement that modifies the behavior of the statement. For example, MOVED=100. MOVED is the statement and 100 is the argument.
Remarks:	The remark field contains additional information about the use of the statement.
See Also:	This field contains a list of statements that are related to the purpose of the keyword.
Example:	The example field contains a code segment that illustrates the usage of the keyword

Reference

Table 24: ASSIGN

ASSIGN	Assign Input As Index Bit	Statement
Purpose	Assign keyword causes a specified input to be assigned to a particular bit of system variable INDEX. Up to 8 digital inputs can be assigned to the first eight bits (bits 0 - 7) of the INDEX system variable in any order or combination. The purpose of the Assign Keyword and INDEX system Variable is to allow the creation of a custom input word for inclusion in the user program. Good examples of it's use are for implementing easy selection of preset torque, velocity or position values within the user program.	
Syntax	ASSIGN INPUT <input name> AS BIT <bit #> Input name (IN_A1..IN_A2 etc.) Bit# INDEX variable bit number from 0 to 7	
Remarks	Assign statements typically appear at the start of the program (Initialize and set Variables section) but can be included in other code sections with the exception of Events and the Fault Handler.	
See Also	VAR_IOINDEX Variable (#220)	

Example:

```
ASSIGN INPUT IN_B1 AS BIT 0 ;index bit 0 state matches state of input B1
ASSIGN INPUT IN_B2 AS BIT 1 ;index bit 1 state matches state of input B2
```

Program Start:

```
; <statements>
```

```
If Index == 0 ; If neither IN_B1 or IN_B2 is on
    MoveP 0 ; Move to Absolute Position 0
```

```
Endif
```

```
If Index == 1 ; If IN_B1 is on and IN_B2 is off
    MoveP 10 ; Move to Absolute Position 10
```

```
Endif
```

```
; If Index == 2 .....

```

Table 25: DEFINE

DEFINE	Define name	Pseudo-statement
Purpose	DEFINE is used to define symbolic names for User Variables, constants, and Digital I/O for programming convenience. Define statements greatly enhance program understanding by allowing the user to program using symbolic strings (names) relevant to their application. DEFINE can be used also to substitute a symbolic string.	
Syntax	DEFINE <name> <synonym> name any symbolic string synonym User Variable, constant, or Digital I/O Flag that symbolic string will represent	
Remarks:	DEFINE statements can be located anywhere within the user program (with the exception of events and the fault handler). Normally practice however is to place definitions at the start of the program prior to any executable code.	
See Also		
Example:	<pre> Define Start_Button IN_B1 ; Define a Digital Input Define System_Stop Out2 ; Define a Digital Output Define Loop_Counter V5 ; Define a User Variable Define Loop_Increment 1 ; Define a Constant Value Program_Start: ; Label Program Start If Start_Button == 0 ; If input B1 is off Disable ; Disable Servo System_Stop = 1 ; Turn on Output 2 Else ; Otherwise System_Stop = 0 ; Turn off Output 2 Enable ; Enable Servo MoveD 10 ; Move (increment) Distance 10 Loop_Counter = Loop_Counter + Loop_Increment ; Increment Variable V5 by 1 Endif Goto Program_Start ; Goto Label Program_Start </pre>	

Table 26: DISABLE

DISABLE	Disables the drive	Statement
Purpose	DISABLE turns OFF the drive output to the motor. Drive shows 'Dis' on display when in a disabled state.	
Syntax	DISABLE	
Remarks	Once the DISABLE statement is executed, the power to the motor is turned off and the motor can move freely. When disabled the drive will continue to monitor feedback and the actual position variable (APOS) will continue to update with the current position of the motor. The target position variable (TPOS) will be updated with the value of the actual position variable (APOS) on Enable to prevent unexpected motion from the motor shaft.	
See Also	ENABLE	
	<div style="border: 1px solid black; padding: 5px;"> WARNING! Work should not be carried out on the drive/system without the drive first being isolated from its mains supply. The disabled condition is not an indication that the motor or system is safe to work on as an Enable/run condition could result from execution of the programmers programming code, from a host interface, or controller. </div>	

Example:

```

If Start_Button == 0           ; If input B1 is off
  Disable                       ; Disable Servo
Else                             ; Otherwise
  Enable                         ; Enable Servo
  MoveD 10                       ; Move (increment) Distance 10
Endif
          
```

Reference

Table 27: DO UNTIL

DO UNTIL	Do/Until	Statement
Purpose	The DO / UNTIL statement is used to execute a statement or set of statements repeatedly until a logical condition becomes true. The Do / Until statements enclose the program code to be repeatedly executed with the UNTIL statement containing the logical statement for exit of the loop.	
Syntax	DO {statement(s)}... UNTIL <condition> {statement(s)} any valid statement(s) <condition> The condition to be tested.	
Remarks	The statement or statements contained within a DO / UNTIL loop will always be executed at least once because the logical condition to be tested is contained within the UNTIL statement in the last statement of the loop.	
See Also	WHILE, IF	
Example:	<pre> V0 = 0 ; Set V0 to Value 0 ; Create Loop to perform Move command 12 times DO V0 = V0 + 1 ; Add 1 to Variable V0 Moved 5 ; Move (incremental) distance 5 Until V0 == 12 ; Loop back to DO Statement, Repeat Until Logic True </pre>	

Table 28: ENABLE

ENABLE	Enables the drive	Statement
Purpose	Enable turns on drive output to the motor. Drive shows 'Run' on display when in the enabled state.	
Syntax	ENABLE	
Remarks	Once a drive is enabled motion can be commanded from the user program. Commanding motion while the drive is disabled will result in fault trip (F_27).	
See Also	DISABLE	
Example:	<pre> If Start_Button == 0 ; If input B1 is off Disable ; Disable Servo Else ; Otherwise Enable ; Enable Servo MoveD 10 ; Move (increment) Distance 10 Endif </pre>	

Table 29: END

END	END program	Statement
Purpose	This statement is used to terminate (finish) user program and its events.	
Syntax	END	
Remarks	END can be used anywhere in program	
See Also	DISABLE	
Example:	<pre> END ;end user program </pre>	

Table 30: EVENT

EVENT	Starts Event handler	Statement																																
Purpose	EVENT keyword is used to create scanned events within the user program. Statement also sets one of 4 possible types of events.																																	
Syntax	<p>Any one of the 4 syntax examples herein may be used:</p> <ol style="list-style-type: none"> 1. EVENT <name> INPUT <inputname> RISE 2. EVENT <name> INPUT <inputname> FALL 3. EVENT <name> TIME <period> 4. EVENT <name> <expression> <table style="margin-left: 20px; border: none;"> <tr> <td style="padding-right: 20px;">name</td> <td>any valid alphanumeric string</td> </tr> <tr> <td>inputname</td> <td>any valid input "IN_A1 - IN_C4"</td> </tr> <tr> <td>period</td> <td>any integer number. Expressed in ms</td> </tr> <tr> <td>expression</td> <td>any arithmetic or logical expression</td> </tr> </table> <p>The following statements can not be used within event's handler:</p> <table style="margin-left: 20px; border: none;"> <tr> <td>MOVE</td> <td>MOVED</td> <td>MOVEP</td> <td>MOVEDR</td> </tr> <tr> <td>MOVEPR</td> <td>MDV</td> <td>MOTION SUSPEND</td> <td>MOTION RESUME</td> </tr> <tr> <td>STOP MOTION</td> <td>DO/UNTIL</td> <td>GOTO</td> <td>GOSUB</td> </tr> <tr> <td>HALT</td> <td>VELOCITY ON/OFF</td> <td>WAIT</td> <td>WHILE/ENDWHILE</td> </tr> <tr> <td>ASSIGN</td> <td>END</td> <td>ON FAULT/END FAULT</td> <td>RESUME</td> </tr> <tr> <td>RETURN</td> <td></td> <td></td> <td></td> </tr> </table> <p>While GOTO or GOSUB are restricted, a special JUMP statement can be used for program flow change from within event handler. See JUMP statement description in Language Reference section.</p> <p>Program labels are also not permitted within event code.</p>		name	any valid alphanumeric string	inputname	any valid input "IN_A1 - IN_C4"	period	any integer number. Expressed in ms	expression	any arithmetic or logical expression	MOVE	MOVED	MOVEP	MOVEDR	MOVEPR	MDV	MOTION SUSPEND	MOTION RESUME	STOP MOTION	DO/UNTIL	GOTO	GOSUB	HALT	VELOCITY ON/OFF	WAIT	WHILE/ENDWHILE	ASSIGN	END	ON FAULT/END FAULT	RESUME	RETURN			
name	any valid alphanumeric string																																	
inputname	any valid input "IN_A1 - IN_C4"																																	
period	any integer number. Expressed in ms																																	
expression	any arithmetic or logical expression																																	
MOVE	MOVED	MOVEP	MOVEDR																															
MOVEPR	MDV	MOTION SUSPEND	MOTION RESUME																															
STOP MOTION	DO/UNTIL	GOTO	GOSUB																															
HALT	VELOCITY ON/OFF	WAIT	WHILE/ENDWHILE																															
ASSIGN	END	ON FAULT/END FAULT	RESUME																															
RETURN																																		

Remarks

For syntax 1 and 2:

The Event will occur when the input defined by the <name> transition from low to high, for syntax 1 (RISE) and from high to low for syntax 2 (FALL).

For syntax 3:

The Event will occur when the specified , <period>, period of time has expired. This event can be used as periodic event to check for some conditions.

For syntax 4

The Event will occur when the expression, <expression>, evaluates to be true. The expression can be any valid arithmetic or logical expression or combination of the two. This event can be used when implementing soft limit switches or when changing the program flow based on some conditions. Any variable, (user and system), or constants can be used in the expression. The event will only trigger when the logic transitions from False to True. Further occurrence of the event will not occur while the condition remains true.

See Also ENDEVENT, EVENT ON/OFF, EVENTS ON/OFF

Example:

```

EVENT InEvent IN_A1 RISE
    V0 = V0+1           ;V0 increments by 1 each time IN_A1 transitions from low to high
ENDEVENT
EVENT period TIME 1000 ;1000 ms = 1Sec
    V3=V0-V1           ;Event subtracts V1 from V0 and stores result in V3 every second
ENDEVENT
;-----
    EVENT InEvent ON   ;Statements in main program to turn individual events on
    EVENT period  ON
    {program statements}
END
    
```

Reference

Table 31: ENDEVENT

ENDEVENT	END of Event handler	Statement
Purpose	Indicates end of the scanned event code	
Syntax	ENDEVENT	
Remarks		
See Also	EVENT, EVENT ON/OFF, EVENTS ON/OFF	
Example:	<pre> EVENT InputRise IN_B4 RISE V0=V0+1 ENDEVENT </pre>	

Table 32: EVENT ON/OFF

EVENT ON/OFF	Turn events on or off	Statement
Purpose	Turns ON or OFF events created by an EVENT statement	
Syntax	<pre> EVENT <name> ON EVENT <name> OFF <name> Event name </pre>	
Remarks		
See Also	EVENT	
Example:	<pre> ; Events Section EVENT InputRise IN_B4 RISE V0=V0+1 ENDEVENT ; Main Program EVENT InputRise ON ; statements... EVENT InputRise OFF </pre>	

Table 33: EVENTS ON/OFF

EVENTS OFF/ON	Globally Disables/re-enables events	Statement
Purpose	EVENTS OFF command when executed will disable any events currently enabled (running). EVENTS ON Command re-enables any events previously disabled through the EVENTS OFF command. EVENTS ON is not a global enable of all declared events. Events status is indicated through bit #30 of the DSTATUS register or by system flag 'F_EVENTSOFF'. EVENTS OFF/ON allows for easy disable and re-activation of events in sections of the main program or subroutines that the programmer doesn't want interrupted by event code.	
Syntax	EVENTS OFF Disables execution of all events EVENTS ON Restores execution of previously enabled events.	
Remarks	Events are globally disabled after a program reset is made or a fault occurs. Events are re-enabled by executing the individual EVENT <name> ON statement following either a Reset or a Fault Condition.	
See Also	EVENT	

Example:

```

*****
EVENT SKIPOUT IN_B4 RISE      ;check for rising edge of input B4
  JUMP TOGGLE                 ;redirect code execution to TOGGLE
ENDEVENT                     ;end the event
EVENT OVERSHOOT IN_B3 RISE   ;check for rising edge of input B3
  JUMP SHUTDOWN              ;redirect code execution to SHUTDOWN
ENDEVENT                     ;end the event
*****
EVENT SKIPOUT ON
EVENT OVERSHOOT ON
*****
.....User code.....
EVENTS OFF                   ;turns off all events
.....User code.....
EVENTS ON                    ;turns on any event previously activated
  
```

Table 34: FAULT

FAULT	User generated fault	Statement
Purpose	Allows the user program to set a custom system fault. This is useful when the programmer needs to define a fault code and fault process for custom conditions like data supplied by interface out of range etc. Custom fault numbers must be in region of 128 to 240 (decimal)	
Syntax	FAULT <FaultNumber>	Sets system fault. <FaultNumber> constant in range 128-240
Remarks	Custom fault will be processed in the same way as a system fault. There will be a record in the fault log. Variables are not allowed in this statement.	
See Also	ON FAULT	
Example:	FAULT 200 ;Sets fault #200	

Reference

Table 35: GOSUB

GOSUB	Go To subroutine	Statement
Purpose	GOSUB transfers control to subroutine.	
Syntax	GOSUB <subname>	
	<subname>	a valid subroutine name
Remarks	After return from subroutine program resumes from next statement after GOSUB	
See Also	GOTO, JUMP, RETURN	
Example:		
<pre> DO GOSUB CALCMOVE ;Go to CALCMOVE Subroutine MOVED V1 ;Move distance calculated in Subroutine UNTIL INA1 END SUB CALCMOVE: V1=(V2+V3)/2 ;Subroutine statement, Calculates value for V1 RETURN ;Return to main program execution </pre>		

Table 36: GOTO

GOTO	Go To	Statement
Purpose	Transfer program execution to label following the GOTO instruction.	
Syntax	GOTO <label>	
Remarks	<Label> must be a valid program reference label (alphanumeric string, 64 characters in length, and ending with a colon “:”) contained within the user program. The GOTO statement can be located either above or below the program label in the user code.	
See Also	GOSUB, JUMP	
Example:		
<pre> GOTO Label2 {Statements...} Label2: {Statements...} </pre>		

Table 37: HALT

HALT	Halt the program execution	Statement
Purpose	Used to halt main program execution. Events are not halted by the HALT statement. Event code can restart main program execution by issuing the RESET statement or by executing a JUMP to a Main Program Label from the EVENT handler. With the RESET statement, Main Program execution will recommence on the code line immediately following the HALT Statement. With a Jump command program execution is forced to the program label defined within the argument of the JUMP command.	
Syntax	HALT	
Remarks	This statement is convenient when writing event driven programs.	
See Also	RESET, JUMP, EVENT	
Example:		
<pre> {Statements...} HALT ;halt main program execution and wait for event </pre>		

Table 38: HOME


HOME	Execute homing routine	Statement
Purpose	Used to initiate homing.	
Syntax	HOME	
Remarks	Homing is initiated from the user program using the 'HOME' command. The HOME command is a blocking instruction that prevents further execution of the Main Program until homing operation is completed. Any events that are enabled while homing is carried out will continue to process.	
	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px; text-align: center;">  </div> <div> <p>WARNING!</p> <p>If using firmware prior to 4.50 then execution of homing functionality does not prevent simultaneous execution of subsequent programming statements and it is required to immediately follow the HOME command with the following code line:</p> <pre>WAIT UNTIL VAR_EXSTATUS & 0x400000 == 0x400000.</pre> </div> </div>	
See Also		
Example:		
	<pre>{Statements...} HOME ;initiate homing routine</pre>	

Table 39: ICONTROL ON/OFF

ICONTROL ON/OFF	Enables interface control	Statement
Purpose	Enables/Disables interface control. Effects bit #27 in DSTATUS register and system flag F_ICONTROLOFF. All interface motion commands and commands changing any outputs will be disabled. See Host interface commands manual for details. This command is useful when the program is processing critical states (example limit switches) and can't be disturbed by the interface.	
Syntax	ICONTROL ON ICONTROL OFF	Enables Interface control Disables interface control
Remarks	After reset interface control is enabled by default.	
See Also		
Example:		
	<pre>EVENT LimitSwitch IN_A1 RISE ;limit switch event Jump LimitSwitchHandler ;jump to process limit switch ENDEVENT V0=0 ;V0 will be used to indicate fault condition EVENT LimitSwitch ON ;Turn on event to detect limit switch activation Again: HALT ;system controlled by interface and Events LimitSwitchHandler: EVENTS OFF ;turn off all events ICONTROL OFF ;disable interface control STOP MOTION QUICK DISABLE ;DISABLE V0=1 ;indicate fault condition to the interface ICONTROL ON ;Enable Interface Control EVENTS ON ;turn on events turned off by 'EVENTS OFF' GOTO AGAIN</pre>	

Reference

Table 40: IF

IF	IF/ENDIF	Statement
Purpose	The IF statement tests for a condition and then executes the specific action(s) between the IF and ENDIF statements if the condition is found to be true. If the condition is false, no action is taken and the instructions following the ENDIF statement are executed. Optionally, using the ELSE statement, a second series of statements may be specified to be executed if the condition is false.	
Syntax	<pre>IF <condition> {statements 1} ELSE {statements 2} ENDIF</pre>	
Remarks		
See Also	WHILE, DO	
Example:		
<pre>IF APOS > 4 ;If actual position is greater than 4 units V0=2 ELSE ;otherwise... (actual position equal or less than 4) V0=0 ENDIF ;----- If V1 <> V2 && V3>V4 ;If V1 doesn't equal V2 AND V3 is greater than V4 V2=9 ENDIF</pre>		

Table 41: JUMP

JUMP	Jump to label from Event handler	Statement
Purpose	This is a special purpose statement to be used only in the Event Handler code. When the EVENT is triggered and this statement is processed, execution of the main program is transferred to the <label> argument called out in the "JUMP" statement. The Jump statement is useful when there is a need for the program's flow to change based on some event(s). Transfer of program execution is to the instruction following the label. When a Jump statement is executed within an event, processing of subsequent events is suspended until the next event time cycle.	
Syntax	<pre>JUMP <label> <label> is any valid program label</pre>	
Remarks	Can be used in EVENT handler only.	
See Also	EVENT	
Example:		
<pre>EVENT ExternalFault INPUT IN_A4 RISE ;activate Event when IN_A4 goes high JUMP ExecuteStop ;redirect program to <ExecuteStop> ENDEVENT StartMotion: EVENT ExternalFault ON ENABLE MOVED 20 MOVED -100 {statements} END ExecuteStop: STOP MOTION ;Motion stopped here DISABLE ;drive disabled Wait Until !IN_A4 ;Wait Until Input A4 goes low GOTO StartMotion</pre>		

Table 42: LOADVARS

LOADVARS	EPM access statements	LOADVARS	Statement
Purpose	LOADVARS is the command to retrieve the values of the user variables (V0-V31) from the drive's EPM. Using this statement any combinations of variables V0-V31 can be retrieved from the EPM with a single statement. Loads the values of the user's variables (V0-V31) from EPM to the drive operating memory.		
Syntax	LOADVARS	[Va, Vx-Vy]	a,x,y any number from 0 - 31
Remarks	Values that are stored in EPM memory for the User Variables V0-V31 (using host interface or StoreVars command) are automatically transferred into operational memory at power up (a LoadVar statement is not required). Should a User Variable be altered by the user program it is altered only in the operational memory of the drive and can be restored back to its EPM value using the LoadVars statement.		
See Also	STOREVARS		

Example:

```

...{statements}...
V1=12
STOREVARS [V1]                                 ;Store V1 variable to EPM memory
...{statements}...
LOADVARS [V1]                                 ;Retrieve V1 variable
...{statements}...
END                                             ;End main program
-----
;Example to specify multiple variables list in a single LoadVar Statement

LOADVARS [V0,V1,V5-V20]   ;load values of V0, V1, V5-V20
-----

```

Table 43: MDV

MDV	Segment Move	Statement
Purpose	MDV (Move-Distance-Velocity) defines individual motion segment by specifying distance and final velocity (for each segment) in User Units. Acceleration (or deceleration) is calculated automatically based on these two parameters. This technique allows complicated moves to be created that consist of many segments. Each MDV sequence (series of MDV segments) must start and end with a velocity of 0, hence an MDV sequence must have at least two segments. The MDV statement doesn't suspend execution of the main program. Each segment is loaded into the Motion Queue and the sequence executed sequentially. If the last segment in the Motion Queue doesn't have a final velocity of 0, the drive will generate a "Motion Stack Underflow" fault #24. If the "S" modifier is used in the statement, then the velocity acceleration/deceleration will be S-curved as opposed to linear.	
Syntax	MDV	<[-]segment distance>,<segment final velocity> [,S]
	[,S]	optional modifier specifies S-curve acceleration / deceleration.
See Also	MOVE, MOVEP, MOVEPR, MOVED, MOVEDR, MOTION SUSPEND, MOTION RESUME	

Example:

```

{Statements...}
MDV 5, 10                                     ;Move 5 user units and accelerate to a velocity of 10
MDV 10,10                                   ;Move 10 user units and maintain a velocity of 10
MDV 10,5                                    ;Move 10 user units and decelerate to velocity of 5
MDV 5,0                                     ;Move 5 user units and decelerate to velocity 0.
                                           ;The last MDV must have a final velocity of 0.
{Statements...}

```

Reference

Table 44: MEMGET

MEMGET	Memory access statements MEMGET	Statement
Purpose	MEMGET provides command for simplified retrieval of data from the drives RAM memory file through transfer of data to the variables V0-V31. Using this statement any combinations of variables V0-V31 can be retrieved from the RAM file with a single statement.	
Syntax	MEMGET	<offset> [<varlist>]
	<offset>	specifies offset in RAM file where data will be retrieved. Range: -32767 to 32767
	<varlist>	any combinations of variables V0-V31
See Also	MEMSET	
Example:		
	MEMGET 5 [V0]	;single variable will be retrieved from location 5
	MEMGET V1 [V0,V3,V2]	;variables V0,V3,V2 will be retrieved from ;memory location starting at value held in V1
	MEMGET 10 [V3-V7]	;variables V3 to V7 inclusively will be retrieved
	MEMGET V1 [V0,V2,V4-V8]	;variables V0,V2, V4 through V8 will be retrieved

Table 45: MEMSET

MEMSET	Memory access statements MEMSET	Statement
Purpose	MEMSET provides command for simplified storage of data to the drives RAM memory file through transfer of data from variables V0-V31. Using this statement any combinations of variables V0-V31 can be stored in the RAM file with a single statement.	
Syntax	MEMSET	<offset> [<varlist>]
	<offset>	specifies offset in RAM file where data will be stored. Range: -32767 to 32767
	<varlist>	any combinations of variables V0-V31
See Also	MEMGET	
Example:		
	MEMSET 5 [V0]	;single variable will be stored in location 5
	MEMSET V1 [V0,V3,V2]	;variables V0,V3,V2 will be stored in memory ;location starting at value held in V1
	MEMSET 10 [V3-V7]	;variables V3 to V7 inclusively will be stored
	MEMSET V1 [V0,V2,V4-V8]	;variables V0,V2, V4 through V8 will be stored.

Table 46: MOTION RESUME

MOTION RESUME	Resume Motion	Statement
Purpose	Statement resumes motion previously suspended by MOTION SUSPEND. If motion was not previously suspended, this has no effect on operation.	
Syntax	MOTION RESUME	
Remarks	Any motion command executed in the user program while motion is suspended will be placed in the motion queue but not executed. Motion commands accumulated on the motion stack will be performed in the order they were loaded to the motion queue on execution of the Motion Resume statement.	
See Also	MOVE, MOVEP, MOVEDR, MOVED, MOVEPR, MDV, MOTION SUSPEND	
Example:		
	...{statements}	
	MOTION RESUME ;Motion is resumed from first command in motion Queue (if any)	
	...{statements}	

Table 47: MOTION SUSPEND

MOTION SUSPEND	Suspend	Statement
Purpose	This statement is used to temporarily suspend execution of motion. The Motion Suspend command does not flush the Motion Queue of any accumulated motion commands but will suspended further motion until the Motion Resume command is processed. If this statement is executed while a motion profile is already being processed, then the motion will not be suspended until after the completion of of the current motion profile. If executing a series of segment (MDV) moves, motion will not be suspended until after all the MDV segments have been processed. Any subsequent motion statement will be loaded into the queue and will remain there until the “Motion Resume” statement is executed. Any motion statements executed without the “C” modifier (except MDV statements) while motion is suspended will lock-up the User Program.	
Syntax	MOTION SUSPEND	
Remarks	Performing any MOVEx commands without “C” modifier will lock-up the user program. The programmer will be able to unlock program execution only by performing a Reset or by issuing a Motion Resume command from a host interface.	
See Also	MOVE, MOVEP, MOVEDR, MOVED, MOVEPR ,MDV, MOTION RESUME	
Example:		
<pre> ...{statements} MOTION SUSPEND ;Motion will be suspended after current motion ;command is finished. ...{statements} </pre>		

Table 48: MOVE

MOVE	Move	Statement
Purpose	There are two variations to the Move command, Move Until and Move While. MOVE UNTIL performs motion until a logical condition becomes TRUE. MOVE WHILE performs motion while a logical condition stays TRUE. The statement suspends the programs execution until the motion is completed, unless the statement is used with C modifier.	
Syntax	MOVE [BACK] UNTIL <condition> [,C] MOVE [BACK] WHILE <condition> [,C] BACK Changes direction of the move to negative. C (optional) C[ontinue] - modifier allows the program to continue while motion is being performed. If a second motion profile is executed while the first profile is still in motion, the second profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. If the queue becomes full, or overflows, then the drive will generate a fault. <condition> The condition to be tested.	
See Also	MOVEP, MOVED, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME	
Example:		
<pre> {Statements...} MOVE UNTIL V0<3 ;Move until V0 is less than 3 MOVE BACK UNTIL V0>4 ;Move back until V0 is greater than 4 MOVE WHILE V0<3 ;Move While V0 is less than 3 MOVE BACK WHILE V0>4 ;Move While V0 is greater than 4 MOVE WHILE V0<3,C ;Move While V0 < 3, continue program execution </pre>		

Reference

Table 49: MOVED

MOVED	Move Distance	Statement
Purpose	MOVED performs incremental motion (distance) specified in User Units. This statement will suspend the program's execution until the motion is completed, unless the statement is used with the "C" modifier. If the "S" modifier is used then S-curve acceleration/deceleration is performed during the move.	
Syntax	MOVED <distance>[,S] [,C] C[ontinue] The "C" argument is an optional modifier which allows the program to continue executing while the motion profile is being executed. If the drive is in the process of executing a previous motion profile the new motion profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. If the queue becomes full, or overflows, then the drive will generate a fault. S[-curve] optional modifier specifies S-curve acceleration/deceleration.	
Remarks	Maximum variable size is $2^{32} * \text{Units/QPPR}$. This is the max value for Var_APOS_Pulses. Maximum distance is then this maximum value that can be held in a variable divided by the feedback pulses. So assume a 4096 ppr encoder. Post quad = 16384. Max distance before register overflow = 131072. For resolver = 32768. For Moved, absolute position is not a concern. If overloaded, the register will simply roll over.	
See Also	MOVE, MOVEP, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME	
Example:		
<pre> {Statements...} MOVED 3 ;moves 3 user units forward MOVED BACK 3 ;moves 3 user units backward MOVED -3 ;moves 3 user units backward MOVED V5 ;moves distance / direction determined by value in v5 {Statements...} </pre>		

Table 50: MOVEDR

MOVEDR	Registered Distance Move	Statement
Purpose	MOVEDR performs incremental motion, specified in User Units, in search of the registration input. If during the move the registration input becomes activated (goes high) then the current position is recorded, and the displacement value (the second argument in the MOVEDR statement) is added to the captured registration position to form a new target position. The end of the move is then altered to this new target position. This statement suspends execution of the program until the move is completed, unless the statement is used with the "C" modifier. If the "S" modifier is used then S-curve acceleration/deceleration is performed during the move.	
Syntax	MOVEDR <distance>,<displacement> [,S] [,C] C[ontinue] The "C" argument is an optional modifier which allows the program to continue executing the User Program while a motion profile is being processed. If a new motion profile is requested while the drive is processing a move the new motion profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. If the queue becomes full, or overflows, then the drive will generate a fault. S[-curve] optional modifier specifies S-curve acceleration/deceleration.	
See Also	MOVE, MOVEP, MOVEPR, MOVED, MDV, MOTION SUSPEND, MOTION RESUME	
Example:		
This example moves the motor 3 user units while checking for the registration input. If registration isn't detected then the move is completed. If registration is detected, the registration position is recorded and the displacement value of 2 is added to the recorded registration position to calculate the new end position.		
<pre> {Statements...} MOVEDR 3, 2 {Statements...} </pre>		

Table 51: MOVEP

MOVEP	Move to Position	Statement
Purpose	MOVEP performs motion to a specified absolute position in User Units. This statement will suspend the program's execution until the motion is completed unless the statement is used with the "C" modifier. If the "S" modifier is used then an S-curve acceleration/deceleration is performed during the move.	
Syntax	MOVEP <absolute position>[,S] [,C] C[ontinue] The "C" argument is an optional modifier which allows the program to continue executing while the motion profile is being executed. If the drive is in the process of executing a previous motion profile the new motion profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. If the queue becomes full, or overflows, then the drive will generate a fault. S[-curve] optional modifier specifies S-curve acceleration/deceleration.	
Remarks	Maximum variable size is $2^{32} * \text{Units/QPPR}$. This is the max value for Var_APOS_Pulses. Maximum distance is then this maximum value that can be held in a variable divided by the feedback pulses. So assume a 4096 ppr encoder. Post quad = 16384. Max distance before register overflow = 131072. For resolver = 32768. This matters because if the register overflows, then the absolute position is flipped up-side down.	
See Also	MOVE, MOVED, MOVEPR, MOVEDR, MDV, MOTION SUSPEND, MOTION RESUME	
Example:		
<pre>{Statements...} MOVEP 3 ;moves to 3 user units absolute position MOVEP -3 ;moves to -3 user units absolute position MOVEP V5 ;moves to absolute position determined by value in v5 {Statements...}</pre>		

Table 52: MOVEPR

MOVEPR	Registered Distance Move	Statement
Purpose	MOVEPR performs absolute position moves, specified in User Units, in search of the registration input. If during a move the registration input becomes activated (goes high), then the current position is recorded, and the displacement value (the second argument in the MOVEPR statement) is added to the captured registration position to form a new target position. The end of the move is then altered to this new target position. This statement suspends the execution of the program until the move is completed, unless the statement is used with the C modifier. If the "S" modifier is used then S-curve acceleration/deceleration is performed during the move.	
Syntax	MOVEPR <distance>,<displacement> [,S] [,C] C[ontinue] The "C" argument is an optional modifier which allows the program to continue executing the User Program while a motion profile is being processed. If a new motion profile is requested while the drive is processing a move the new motion profile will be loaded into the Motion Stack. The Motion Stack is 32 entries deep. If the queue becomes full, or overflows, then the drive will generate a fault. S[-curve] optional modifier specifies S-curve acceleration/deceleration.	
See Also	MOVE, MOVEP, MOVEDR, MOVED, MDV, MOTION SUSPEND, MOTION RESUME	
Example:	This example moves the motor to the absolute position of 3 user units while checking for the registration input. If registration isn't detected, then the move is completed. If registration is detected, the registration position is recorded and the displacement value of 2 is added to the recorded registration position to calculate the new end position.	
<pre>{Statements...} MOVEPR 3, 2 {Statements...}</pre>		

Reference

Table 53: ON FAULT/ENDFAULT

ON FAULT/ ENDFAULT	Defines Fault Handler	Statement
Purpose	<p>This statement defines the Fault Handler section of the User Program. The Fault Handler is a section of code which is executed when a fault occurs in the drive. The Fault Handler program must begin with the "ON FAULT" statement and end with the "ENDFAULT" statement. If a Fault Handler routine is not defined, then the User Program will be terminated and the drive disabled upon the drive detecting a fault. Subsequently, if a Fault Handler is defined and a fault is detected, the drive will be disabled, all scanned events will be disabled, and the Fault Handler routine will be executed. The RESUME statement can be used to redirect the program execution from the Fault Handler back to the main program. If this statement is not utilized then the program will terminate once the ENDFAULT statement is executed.</p> <p>The following statements <i>can't</i> be used in fault handler: DO / UNTIL, ENABLE, EVENT (ON, OFF), EVENTS (ON, OFF), GOTO, GOSUB, HALT, HOME, JUMP, MDV, MOTION RESUME, MOTION SUSPEND, MOVE, MOVED, MOVEP, MOVEDR, MOVEPR, STOP MOTION (QUICK), VELOCITY ON/OFF, WAIT and WHILE / ENDWHILE</p>	
Syntax	<pre>ON FAULT {...statements} ENDFAULT</pre>	
See Also	RESUME	
Example:		
<pre>...{statements} ;User program FaultRecovery: ;Recovery procedure ...{statements} END ON FAULT ;Once fault occurs program is directed here ...{statements} ;code to deal with fault RESUME FaultRecovery ;Execution of RESUME ends Fault Handler and directs ;execution back to User Program. ENDFAULT ;If RESUME is omitted the program will terminate here Fault routine must end with an ENDFAULT statement</pre>		

Table 54: REGISTRATION ON

REGISTRATION ON	Registration On	Statement
Purpose	<p>This statement arms the registration input (input IN_C3). When the registration is armed and the registration input activated the Flag "F_REGISTRATION" is set and the current position is captured and stored to the "RPOS" System Variable. The "REGISTRATION ON" statement, when executed will reset the "F_REGISTRATION" flag ready for detection of the next registration input.</p>	
Syntax	<pre>REGISTRATION ON Flag "F_REGISTRATION" is reset and registration input is armed</pre>	
See Also	MOVEDR, MOVEPR	
Example:		
<pre>; Moves until registration input is activated and then returns to the sensor position. ...{statements} REGISTRATION ON ;Arm registration input MOVE UNTIL F_REGISTRATION ;Move until registration flag is activated (triggered by ;registration input to C3), (sensor hit) ;Drive will decelerate to stop beyond Sensor position MOVEP RPOS ;Absolute move back to the position of the sensor ...{statements}</pre>		

Table 55: RESUME

RESUME	Resume	Statement
Purpose	This statement redirects the code execution from the Fault Handler routine back to in the User Program. The specific line in the User Program to be directed to is called out in the argument <label> of the "RESUME" statement. This statement is only allowed in the fault handler routine.	
Syntax	RESUME <label> <label> Label in User Program to recommence program execution	
See Also	ON FAULT	
Example:		
<pre> ;Main Program Section ...{statements} FaultRecovery: ...{statements} END ;Fault Handler Section ON FAULT ;Once fault occurs program is directed here ...{statements} ;code to deal with fault RESUME FaultRecovery ;Execution of RESUME ends Fault Handler and directs ;execution back the "FaultRecovery" label in the User ;Program. ENDFAULT ;If RESUME is omitted the program will terminate here. ;Fault routine must end with a ENDFault statement </pre>		

Table 56: RETURN

RETURN	Return from subroutine	Statement
Purpose	This statement will return the code execution back from a subroutine to the point in the main program from where the subroutine was called. If this statement is executed without a previous call to subroutine, (GOSUB), fault #21 "Subroutine stack underflow" will result.	
Syntax	RETURN	
See Also	GOSUB	
Example:		
<pre> ;Main Program Section ...{statements}... GOSUB MySub ;Program jumps to Subroutine "MySub" MOVED 10 ;Move to be performed once the Subroutine has executed ...{statements} END ;main program end ;Subroutine Section MySub: ;Subroutine called out from User Program ...{statements} ;Code to be executed in subroutine RETURN ;Returns execution to the line of code under the "GOSUB" ;command, (MOVED 10 statement). </pre>		

Reference

Table 57: SEND / SEND TO

SEND/SEND TO	Send network variable(s)	Statement
Purpose	This statement is used to share the value of Network Variables between drives on an Ethernet network. Network Variables are variables NV0 through NV31. The variables to be sent out or synchronized between drives, are called out in the "SEND" statement. For example, "SEND [NV5]" will take the current value of variable NV5 in the drive executing the command and load it into the NV5 variable of every other drive on the network. The SENDTO statement only updates network variables of the drives set with the same group ID given in the command.	
Syntax	SEND [NVa,NVb, NVx-NVy], SENDTO GroupID [NVa,NVb, NVx-NVy] a,b,x,y Any number from 0 to 31 GroupID GroupID of the drives whose variables will be affected (synchronized)	
See Also		
Example:	<pre> ...{statements}... NV1=12 ;Set NV1 equal to 12 SEND [NV1] ;Set the NV1 variable to 12 in every drive in the Network. SEND [NV5-NV10] ;Sets the NV5 through NV10 variables in all drives on the Network. NV20=25 ;Set NV20 equal to 25 SENDTO 2 [NV20] ;Set the NV20 variable to 25 only in drives with GroupID=2 ...{statements} END ;End main program </pre>	

Table 58: STOP MOTION

STOP MOTION [Quick]	Stop Motion	Statement
Purpose	This statement is used to stop all motion. When the "STOP MOTION" statement is executed all motion profiles stored in the Motion Queue are cleared, and motion will immediately be stopped via the deceleration parameter set in the "DECEL" variable. If the "QUICK" modifier is used, then the deceleration value will come from the "QDECEL" variable. The main use for this command is to control an emergency stop or when the End Of Travel sensor is detected. Note that the current position will not be lost after this statement is executed.	
Syntax	STOP MOTION Stops using DECEL deceleration rate STOP MOTION QUICK Stops using QDECEL deceleration rate	
Remarks	Drive output is not disabled following a STOP MOTION [QUICK] command. Also Motion is not suspended after a STOP MOTION [QUICK] so any motion command processed subsequently will be loaded to the Motion Queue and will be executed.	
See Also	MOTION SUSPEND	
Example:	<pre> ...{statements}... DECEL = 100 QDECEL = 10000 ...{statements} STOP MOTION QUICK </pre>	

Table 59: STOREVARS

STOREVARS	EPM access statements STOREVARS	Statement
Purpose	STOREVARS is the command to store the values of the user variables (V0-V31) to the drive's EPM. Using this statement any combinations of variables V0-V31 can be stored to the EPM with a single statement. Stores the values of the user's variables (V0-V31) to the EPM. The purpose of the STOREVARS command is to store user variables from the drives operational memory to the EPM memory so they are retained on power down, or so they can be restored back to the operational memory should their values be altered during the execution of the user program (or by host interface).	
Syntax	STOREVARS [Va, Vx-Vy] a,x,y any number from 0 - 31	
Remarks	Values that are stored in EPM memory for the User Variables V0-V31 (using StoreVars command) are automatically transferred into operational memory at power up (a LoadVar statement is not required). Should a User Variable be altered by the user program, it is altered only in the operational memory of the drive and can be restored back to its EPM value using the LoadVars statement. Care must be taken with the STOREVAR statement not to exceed EPM write capacity of 1 million cycles.	
See Also	LOADVARS	
Example:		
<pre> ...{statements}... V1=12 ;Set V1=12 in drives operational memory (volatile) ...{statements}... STOREVARS [V1] ;store V1 variable to EPM Memory (non-volatile) ...{statements}... LOADVARS [V1] ;Restore value of V1 from EPM memory END ;End main program ;----- ;Example to specify multiple variables list in a single STOREVARS statement STOREVARS [V0,V1,V5-V20] ;store values of V0, V1, V5-V20 </pre>		

Table 60: VELOCITY ON/OFF

VELOCITY ON/OFF	Velocity Mode	Statement
Purpose	The VELOCITY ON statement enables the drive to simulate velocity mode operation while remaining in internal position mode. This allows the drive to transition between internal velocity and position mode while the drive is still enabled. The VELOCITY OFF statement disables velocity mode and returns drive to position mode. The velocity value for this mode is set by writing to the System Variable "VEL". All position related variables are valid in this mode.	
Syntax	VELOCITY ON VELOCITY OFF	
Remarks	The "VELOCITY ON" statement has to be implemented when the drive is enabled. If the "VELOCITY ON" statement is executed while the drive is disabled, fault # 27 - "Motion Attempted While Drive Disabled" will occur. Execution of any motion related profiles while the drive is in Velocity mode will be loaded into the Motion Queue. When the "VELOCITY OFF" statement is executed the drive will default back to Position mode and any motion commands contained in the Motion Queue will execute in sequence. Please note that the "VEL" variable can be set on the fly, allowing dynamic control of the velocity.	
See Also		
Example:		
<pre> VEL=0 ;Set velocity to 0 VELOCITY ON ;Turn on Velocity Mode VEL = 10 ;Set velocity to 10 ...{statements} VELOCITY OFF ;Turn off Velocity Mode </pre>		

Reference

Table 61: WAIT

WAIT	Wait	Statement
Purpose	This statement suspends the execution of the program until logical condition or conditions are met. Conditions can include logical expressions, time expiration, or completion of motion commands.	
Syntax	WAIT UNTIL <expression>	wait until expression becomes TRUE
	WAIT WHILE <expression>	wait while expression is TRUE
	WAIT TIME <time delay>	wait until <time delay> in mS is expired
	WAIT MOTION COMPLETE	wait until last motion in Motion Queue completes
	<expression>	Logical expression evaluating to True or False
	<time delay>	time delay expressed in milliseconds
Remarks	Events that have been declared and enabled will continue to process while the main program executes a WAIT statement. Events containing a JUMP statement could interrupt a WAIT statement and cause an immediate jump to another point in the main program.	
See Also		
Example:		
	WAIT UNTIL (APOS>2 && APOS<3)	;Wait until Apos is > 2 and APOS < 3
	WAIT WHILE (APOS <2 && APOS>1)	;Wait while Apos is <2 and >1
	WAIT TIME 1000	;Wait 1 Sec (1 Sec=1000mS)
	WAIT MOTION COMPLETE	;Wait until motion is done

Table 62: WHILE / ENDWHILE

WHILE/ ENDWHILE	While	Statement
Purpose	The WHILE <expression> executes statement(s) between keywords WHILE and ENDWHILE repeatedly while the expression contained in the WHILE statement evaluates to TRUE.	
Syntax	WHILE <expression>	
	{statement(s)}...	
	ENDWHILE	
Remarks	WHILE block of statements has to end with ENDWHILE keyword.	
See Also	DO/UNTIL	
Example:		
	WHILE APOS<3	;Execute the statements while Apos is <3
	{statement(s)}..	
	ENDWHILE	

3.2 Variable List

Table 63 provides a complete list of the accessible PositionServo variables. These variables can be accessed from the user's program or any supported communications interface protocol. From the user program, any variable can be accessed by either its variable name or by its index value (using the syntax: @<VARINDEX> , where <VARINDEX> is the variable index from Table 63). From the communications interface any variable can be accessed by its index value.

The column "**Type**" indicates the type of variable:

mtr	Motor: denotes a motor value
mtn	Motion: writing to an "mtn" variable could cause the start of motion ⚠
vel	Velocity: denotes a velocity or velocity scaling value

The column "**Format**" provides the native format of the variable:

W	32 bit integer
F	float (real)

When setting a variable via an external device the value can be addressed as floating or integer. The value will automatically adjusted to fit it's given form.

The column "**EPM**" shows if a variable has a non-volatile storage space in the EPM memory:

Y	Variable has non-volatile storage Space in EPM
N	Variable does not exist in EPM memory

The user's program uses a RAM (volatile) 'copy' of the variables stored on the EPM. At power up all RAM copies of the variables are initialized with the EPM values. The EPM's values are not affected by changing the variables in the user's program. When the user's program reads a variable it always reads from the RAM (volatile) copy of the variable. Communications Interface functions can change both the volatile and non-volatile copy of the variable. If the host interface requests a change to the EPM (non-volatile) value, this change is done both in the user program's RAM memory as well as in the EPM. Interface functions have the choice of reading from the RAM (volatile) or from the EPM (non-volatile) copy of the variable. LOADVARDS AND STOREVARD commands can be used to move user variables (V0-V31) between RAM and EPM memory.

The column "**Access**" lists the user's access rights to a variable:

R	read only
W	write only
R/W	read/write

Writing to an R (read-only) variable or reading from a W (write-only) variable is not permitted and many result in erroneous data.

The column "**Units**" shows units of the variable. Units unique to this manual that are used for motion are:

UU	user units
EC	encoder counts
S	seconds
PPS	pulses per sample. Sample time is 512µs - servo loop rate
PPSS	pulses per sample per sample. Sample time is 512µs - servo loop rate

Reference

Table 63: PositionServo Variable List

Index	Name	Type	Format	EPM	Access	Description	Units
1	VAR_IDSTRING			N	R	Drive's identification string	
2	VAR_NAME			Y	R/W	Drive's symbolic name	
3	VAR_SERIAL_NUMBER				R	Drive's serial number	
4	VAR_MEM_INDEX				R/W	Position pointer in RAM file (0 - 32767)	
5	VAR_MEM_VALUE				R/W	Value to be read or written to the RAM file	
6	VAR_MEM_INDEX_INCREMENT				R/W	Holds value the MEM_INDEX will increment once the R/W operation is complete	
7	VAR_VELOCITY_ACTUAL		F	N	R	Actual measured motor velocity	UU/sec
8	VAR_RSVD_2						
9	VAR_DFAULT Short Name: DFAULTS				R	Drive faults register. Holds current trip / fault code	
10	VAR_M_ID	mtr		Y	R/W*	Motor ID	
11	VAR_M_MODEL	mtr		Y	R/W*	Motor model	
12	VAR_M_VENDOR	mtr		Y	R/W*	Motor vendor	
13	VAR_M_ESET	mtr		Y	R/W*	Motor Feedback Resolver: 'Positive for CW' 1 - Positive for CW 0 - negative for clockwise	
14	VAR_M_HALLCODE	mtr		Y	R/W*	Hallcode index Range: 0 - 5	
15	VAR_M_HOFFSET	mtr		Y	R/W*	Reserved	
16	VAR_M_ZOFFSET	mtr		Y	R/W*	Resolver Offset Range: 0 - 360	
17	VAR_M_ICTRL	mtr		Y	R/W*	Reserved	
18	VAR_M_JM	mtr		Y	R/W*	Motor moment of inertia, Jm Range: 0 - 0.1	Kgm2
19	VAR_M_KE	mtr		Y	R/W*	Motor voltage or back EMF constant, Ke Range: 1 - 500	V/Krpm
20	VAR_M_KT	mtr		Y	R/W*	Motor torque or force constant, Kt Range: 0.01 - 10	Nm/A
21	VAR_M_LS	mtr		Y	R/W*	Motor phase-to-phase inductance, Lm Range: 0.1 - 500	mH
22	VAR_M_RS	mtr		Y	R/W*	Motor phase-to-phase resistance, Rm Range: 0.01 - 500	[Ohm]
23	VAR_M_MAXCURRENT	mtr		Y	R/W*	Motor's max current(RMS) Range: 0.5 - 50	[A]mp
24	VAR_M_MAXVELOCITY	mtr		Y	R/W*	Motor's max velocity Range: 500 - 20,000	RPM
25	VAR_M_NPOLES	mtr		Y	R/W*	Motor's poles number Rnpage: 2 - 200	
26	VAR_M_ENCODER	mtr		Y	R/W*	Encoder resolution Range: 256 - 65536 * 12/Npoles	PPR
27	VAR_M_TERMVOLTAGE	mtr		Y	R/W*	Nominal Motor's terminal voltage Range: 50 - 800	[V]olt
28	VAR_M_FEEDBACK	mtr		Y	R/W*	Feedback type 1 - Encoder 2 - Resolver	

* These are all R/W variables that only become active after variable 247 is set.


Reference

Index	Name	Type	Format	EPM	Access	Description	Units
29	VAR_ENABLE_SWITCH_TYPE		W	Y	R/W	Enable switch function 0 - inhibit only 1 - Run	Bit
30	VAR_CURRENTLIMIT		F	Y	R/W	Current limit	[A]mp
31	VAR_PEAKCURRENTLIMIT16		F	Y	R/W	Peak current limit for 16kHz operation	[A]mp
32	VAR_PEAKCURRENTLIMIT		F	Y	R/W	Peak current limit for 8kHz operation	[A]mp
33	VAR_PWMFREQUENCY		W	Y	R/W	PWM frequency selection 0 - 16kHz 1 - 8kHz	
34	VAR_DRIVEMODE		W	Y	R/W	Drive mode 0 - torque 1 - velocity 2 - position  WARNING! You can change operating modes when required during program execution but do not change modes on the fly (with drive enabled), as this may cause unexpected behavior of the motor.	
35	VAR_CURRENT_SCALE		F	Y	R/W	Analog input #1 current reference scale Range: model dependent	A/V
36	VAR_VELOCITY_SCALE	vel	F	Y	R/W	Analog input #1 velocity reference scale Range: -10,000 to +10,000	RPM/V
37	VAR_REFERENCE		W	Y	R/W	Reference selection: 1 - internal source 0 - external	
38	VAR_STEPINPUTTYPE		W	Y	R/W	External Position Mode - Input configuration 0 - Quadrature inputs (A/B) 1 - Step & Direction	
39	VAR_MOTORTHERMALPROTECT		W	Y	R/W	Motor thermal protection function: 0 - disabled 1 - enabled	
40	VAR_MOTORPTCRESISTANCE		F	Y	R/W	Motor thermal protection PTC cut-off resistance in Ohms	[Ohm]
41	VAR_SECONDENCODER		W	Y	R/W	Second encoder: 0 - Disabled 1 - Enabled	
42	VAR_REGENDUTY		W	Y	R/W	Regen circuit PWM duty cycle in % Range: 1-100%	%
43	VAR_ENCODERREPEATSRC		W	Y	R/W	Selects source for repeat buffers: 0 - Model 940 - Encoder Port P4 0 - Model 941 - 2nd Encoder Option Bay 1 - Model 940 - 2nd Encoder Option Bay 1 - Model 941 - Resolver Port P4	
44	VAR_VP_GAIN Short Name: VGAIN_P	vel	W	Y	R/W	Velocity loop Proportional gain Range: 0 - 32767	
45	VAR_VI_GAIN Short Name: VGAIN_I	vel	W	Y	R/W	Velocity loop Integral gain Range: 0 - 32767	
46	VAR_PP_GAIN Short Name: PGAIN_P		W	Y	R/W	Position loop Proportional gain Range: 0 - 32767	
47	VAR_PI_GAIN Short Name: PGAIN_I		W	Y	R/W	Position loop Integral gain Range: 0 - 16383	
48	VAR_PD_GAIN Short Name: PGAIN_D		W	Y	R/W	Position loop Differential gain Range: 0 - 32767	
49	VAR_PI_LIMIT Short Name: PGAIN_ILIM		W	Y	R/W	Position loop integral gain limit Range: 0 - 20000	






Reference

Index	Name	Type	Format	EPM	Access	Description	Units
50	VAR_SEI_GAIN					Not used	
51	VAR_VREG_WINDOW	vel	W	Y	R/W	Gains scaling coefficient Range: -16 to +4	
52	VAR_ENABLE		W	N	W	Software Enable/Disable 0 - disable 1 - enable	
53	VAR_RESET		W	N	W	Drive reset. Disables drive, halts program execution, reset active fault 0 - no action 1 - reset drive	
54	VAR_STATUS Short Name: DSTATUS		W	N	R	Drive's status register	
55	VAR_BCF_SIZE		W	Y	R	User's program Byte-code size	Bytes
56	VAR_AUTOBOOT		W	Y	R/W	User's program autostart flag. 0 - program has to be started manually (MotionView or interface) 1 - program started automatically after drive power up	
57	VAR_GROUPID		W	Y	R/W	Network group ID Range: 1 - 32767	
58	VAR_VLIMIT_ZEROSPEED		F	Y	R/W	Zero Speed window Range: 0 - 100	Rpm
59	VAR_VLIMIT_SPEEDWND		F	Y	R/W	At Speed window Range: 10 - 10000	Rpm
60	VAR_VLIMIT_ATSPEED		F	Y	R/W	Target Velocity for At Speed window Range: -10000 - +10000	Rpm
61	VAR_PLIMIT_POSEERROR		W	Y	R/W	Position error Range: 1 - 32767	EC
62	VAR_PLIMIT_ERRORTIME		F	Y	R/W	Position error time (time which position error has to remain to set-off position error fault) Range: 0.25 - 8000	mS
63	VAR_PLIMIT_SEPOSEERROR		W	Y	R/W	Second encoder Position error Range: 1 - 32767	EC
64	VAR_PLIMIT_SEERRORTIME		F	Y	R/W	Second encoder Position error time (time which position error has to remain to set-off position error fault) Range: 0.25 - 8000	mS
65	VAR_INPUTS Short Name: INPUTS		W	N	R	Digital inputs status variable. A1 occupies Bit 0, A2- Bit 1 ... C4 - bit 11.	
66	VAR_OUTPUT Short Name: OUTPUTS		W	N	R/W	Digital outputs status variable. Writing to this variable sets/resets digital outputs, except outputs which have been assigned special function. Output 1 Bit0 Output 2 Bit 1 Output 3 Bit 2 Output 4 Bit 3	
67	VAR_IP_ADDRESS		W	Y	R/W	Ethernet IP address. IP address changes at next boot up. 32 bit value	
68	VAR_IP_MASK		W	Y	R/W	Ethernet IP NetMask. Mask changes at next boot up. 32 bit value	
69	VAR_IP_GATEWAY		W	Y	R/W	Ethernet Gateway IP address. Address changes at next boot up. 32 bit value	



Reference

Index	Name	Type	Format	EPM	Access	Description	Units
70	VAR_IP_DHCP		W	Y	R/W	Use DHCP 0-manual 1- use DHCP service	
71	VAR_AIN1 Short Name: AIN1		F	N	R	Analog Input AIN1 current value	[V]olt
72	VAR_AIN2 Short Name: AIN2		F	N	R	Analog Input AIN2 current value	[V]olt
73	VAR_BUSVOLTAGE		F	N	R	Bus voltage current value	[V]olt
74	VAR_HTEMP		F	N	R	Heatsink temperature Returns: 0 - for temperatures < 40C and actual heat sink temperature for temperatures >40 C	[c]
75	VAR_ENABLE_ACCELDECEL	vel		Y	R/W	Enable Accel/Decel function for velocity mode 0 - disable 1 - enable	
76	VAR_ACCEL_LIMIT System variable for ramp parameters in MotionView	vel	F	Y	R/W	Accel value for velocity mode Range: 0.1 - 5000000	Rpm*Sec
77	VAR_DECEL_LIMIT System variable for ramp parameters in MotionView	vel	F	Y	R/W	Decel value for velocity mode Range: 0.1 - 5000000	Rpm*Sec
78	VAR_FAULT_RESET		W	Y	R/W	Fault Reset configuration: 1 - on deactivation of Enable/Inhibit input A3 0 - on activation of Enable/Inhibit input (A3)	
79	VAR_M2SRATIO_MASTER		W	Y	R/W	Master to system ratio. Master counts range: -32767 - +32767 Value will be applied upon write to PID #80. Write to this PID followed by writing to PID#80 to apply new ratio pair	
80	VAR_M2SRATIO_SYSTEM		W	Y	R/W	Master to system ratio. System counts range: 1 - 32767 Writing to this PID also applies value currently held in PID #79. If you need to change both values Set #79 first then write to this PID desired value to apply new ratio.	
81	VAR_S2PRATIO_SECOND		W	Y	R/W	Secondary encoder to prime encoder ratio. Second counts range: -32767 - +32767 Value will be applied upon write to PID #82. Write to this PID followed by writing to PID#82 to apply new ratio pair	
82	VAR_S2PRATIO_PRIME		W	Y	R/W	Secondary encoder to prime encoder ratio. Prime counts range: 1 - 32767 Writing to this PID also applies value currently held in PID #81. If you need to change both values Set #81 first then write to this PID desired value to apply new ratio.	
83	VAR_EXSTATUS Short Name: DEXSTATUS		W	N	R	Extended status. Lower word copy of DSP status flags.	
84	VAR_HLS_MODE		W	Y	R/W	Hardware limit switches. 0 - not used 1 - stop and fault 2 - fault  NOTE: When the Hard Limit Switches are activated, the drive will remember this state until the drive is disabled or a fault occurs.	



Reference

Index	Name	Type	Format	EPM	Access	Description	Units
85	VAR_AOUT_FUNCTION		W	Y	R/W	Analog output function range: 0 - 8 0 - Not assigned 1 - Phase Current (RMS) 2 - Phase Current (Peak Value) 3 - Motor Velocity 4 - Phase Current R 5 - Phase Current S 6 - Phase Current T 7 - Iq current 8 - Id current	
86	VAR_AOUT_VELSCALE		F	Y	R/W	Analog output scale for velocity quantities. Range: 0 - 10	mV/Rpm
87	VAR_AOUT_CURSCALE		F	Y	R/W	Analog output scale for current related quantities. Range: 0 - 10	V/A
88	VAR_AOUT Short Name: AOUT		F	N	W	Analog output value.(Used if VAR #85 is set to 0 - no function) Range: 0 - 10	V
89	VAR_AIN1_DEADBAND		F	Y	R/W	Analog input #1 dead-band. Applied when used as current or velocity reference. Range: 0 - 100	mV
90	VAR_AIN1_OFFSET			Y	R/W	Analog input #1 offset. Applied when used as current/velocity reference Range: -10,000 to +10,000	mV
91	VAR_SUSPEND_MOTION		W	N	R/W	Suspend motion. Suspends motion produced by trajectory generator. Current move will be completed before motion is suspended. 0 - motion suspended 1 - motion resumed	
92	VAR_MOVEP	 mtn	W	N	W	Target position for absolute move. Writing value executes Move to position as per MOVEP statement using current values of acceleration, deceleration and max velocity.	UU
93	VAR_MOVED	 mtn	W	N	W	Incremental position. Writing value executes Incremental move as per MOVED statement using current values of acceleration, deceleration and max velocity.	UU
94	VAR_MDV_DISTANCE		F	N	W	Distance for MDV move	UU
95	VAR_MDV_VELOCITY	 mtn	F	N	W	Velocity for MDV move. Writing to this variable executes MDV move with Distance value last written to variable #94	UU
96	VAR_MOVE_PWI1	 mtn	W	N	W	Writing value executes Move in positive direction while input true (active). Value specifies input # 0 - 3 : A1 - A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	
97	VAR_MOVE_PWI0	 mtn	W	N	W	Writing value executes Move in positive direction while input false (not active). Value specifies input # 0 - 3 : A1 - A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
98	VAR_MOVE_NWI1	 mtn	F	N	W	Writing value executes Move negative direction while input true (active). Value specifies input # 0 - 3 : A1 - A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	
99	VAR_MOVE_NWI0	 mtn	F	N	W	Writing value executes Move negative direction while input false (not active). Value specifies input # 0 - 3 : A1 - A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	
100	VAR_V0 Short Name: V0		F	Y	R/W	User variable General purpose user defined variable	
101	VAR_V1 Short Name: V1		F	Y	R/W	User variable General purpose user defined variable	
102	VAR_V2 Short Name: V2		F	Y	R/W	User variable General purpose user defined variable	
103	VAR_V3 Short Name: V3		F	Y	R/W	User variable General purpose user defined variable	
104	VAR_V4 Short Name: V4		F	Y	R/W	User variable General purpose user defined variable	
105	VAR_V5 Short Name: V5		F	Y	R/W	User variable General purpose user defined variable	
106	VAR_V6 Short Name: V6		F	Y	R/W	User variable General purpose user defined variable	
107	VAR_V7 Short Name: V7		F	Y	R/W	User variable General purpose user defined variable	
108	VAR_V8 Short Name: V8		F	Y	R/W	User variable General purpose user defined variable	
109	VAR_V9 Short Name: V9		F	Y	R/W	User variable General purpose user defined variable	
110	VAR_V10 Short Name: V10		F	Y	R/W	User variable General purpose user defined variable	
111	VAR_V11 Short Name: V11		F	Y	R/W	User variable General purpose user defined variable	
112	VAR_V12 Short Name: V12		F	Y	R/W	User variable General purpose user defined variable	
113	VAR_V13 Short Name: V13		F	Y	R/W	User variable General purpose user defined variable	
114	VAR_V14 Short Name: V14		F	Y	R/W	User variable General purpose user defined variable	
115	VAR_V15 Short Name: V15		F	Y	R/W	User variable General purpose user defined variable	
116	VAR_V16 Short Name: V16		F	Y	R/W	User variable General purpose user defined variable	
117	VAR_V17 Short Name: V17		F	Y	R/W	User variable General purpose user defined variable	
118	VAR_V18 Short Name: V18		F	Y	R/W	User variable General purpose user defined variable	
119	VAR_V19 Short Name: V19		F	Y	R/W	User variable General purpose user defined variable	
120	VAR_V20 Short Name: V20		F	Y	R/W	User variable General purpose user defined variable	




Reference

Index	Name	Type	Format	EPM	Access	Description	Units
121	VAR_V21 Short Name: V21		F	Y	R/W	User variable General purpose user defined variable	
122	VAR_V22 Short Name: V22		F	Y	R/W	User variable General purpose user defined variable	
123	VAR_V23 Short Name: V23		F	Y	R/W	User variable General purpose user defined variable	
124	VAR_V24 Short Name: V24		F	Y	R/W	User variable General purpose user defined variable	
125	VAR_V25 Short Name: V25		F	Y	R/W	User variable General purpose user defined variable	
126	VAR_V26 Short Name: V26		F	Y	R/W	User variable General purpose user defined variable	
127	VAR_V27 Short Name: V27		F	Y	R/W	User variable General purpose user defined variable	
128	VAR_V28 Short Name: V28		F	Y	R/W	User variable General purpose user defined variable	
129	VAR_V29 Short Name: V29		F	Y	R/W	User variable General purpose user defined variable	
130	VAR_V30 Short Name: V30		F	Y	R/W	User variable General purpose user defined variable	
131	VAR_V31 Short Name: V31		F	Y	R/W	User variable General purpose user defined variable	
132	VAR_MOVEDR_DISTANCE		F	N	W	Registered move distance. Incremental motion as per MOVEDR statement	UU
133	VAR_MOVEDR_DISPLACEMENT	 mtn	F	N	W	Registered move displacement Writing to this variable executes the move MOVEDR using value set by #132	UU
134	VAR_MOVEPR_DISTANCE		F	N	W	Registered move distance. Absolute motion as per MOVEPR statement	UU
135	VAR_MOVEPR_DISPLACEMENT	 mtn	F	N	W	Registered move displacement Writing to this variable makes the move MOVEPR using value set by #134	UU
136	VAR_STOP_MOTION		W	N	W	Stops motion: 1 - stops motion 0 - no action	
137	VAR_START_PROGRAM		W	N	W	Starts user program 1 - starts program 0 - no action	
138	VAR_VEL_MODE_ON		W	N	W	Turns on Profile Velocity for Internal Position Mode. (Acts as statement VELOCITY ON) 0 - normal operation 1 - velocity mode on	
139	VAR_IREF Short Name: IREF		F	N	W	Internal Reference: Torque or Velocity mode Set value in Amps for Torque mode Set Value in RPM for Velocity Mode	RPS Amps
140	VAR_NV0 Short Name: NV0		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
141	VAR_NV1 Short Name: NV1		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
142	VAR_NV2 Short Name: NV2		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
143	VAR_NV3 Short Name: NV3		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
144	VAR_NV4 Short Name: NV4		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
145	VAR_NV5 Short Name: NV5		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
146	VAR_NV6 Short Name: NV6		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
147	VAR_NV7 Short Name: NV7		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
148	VAR_NV8 Short Name: NV8		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
149	VAR_NV9 Short Name: NV9		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
150	VAR_NV10 Short Name: NV10		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
151	VAR_NV11 Short Name: NV11		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
152	VAR_NV12 Short Name: NV12		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
153	VAR_NV13 Short Name: NV13		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
154	VAR_NV14 Short Name: NV14		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
155	VAR_NV15 Short Name: NV15		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
156	VAR_NV16 Short Name: NV16		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
157	VAR_NV17 Short Name: NV17		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
158	VAR_NV18 Short Name: NV18		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
159	VAR_NV19 Short Name: NV19		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
160	VAR_NV20 Short Name: NV20		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
161	VAR_NV21 Short Name: NV21		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
162	VAR_NV22 Short Name: NV22		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
163	VAR_NV23 Short Name: NV23		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
164	VAR_NV24 Short Name: NV24		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
165	VAR_NV25 Short Name: NV25		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
166	VAR_NV26 Short Name: NV26		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
167	VAR_NV27 Short Name: NV27		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
168	VAR_NV28 Short Name: NV28		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
169	VAR_NV29 Short Name: NV29		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
170	VAR_NV30 Short Name: NV30		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
171	VAR_NV31 Short Name: NV31		F	N	R/W	User defined Network variable. Variable can be shared across Ethernet network.	
172	VAR_SERIAL_ADDRESS		W	Y	R/W	RS485 drive ID. Range: 0 - 254	
173	VAR_MODBUS_BAUDRATE		W	Y	R/W	Baud rate for ModBus operations: 2 - 9600 3 - 19200 4 - 38400 5 - 57600 6 - 115200	
174	VAR_MODBUS_DELAY		W	Y	R/W	ModBus reply delay in mS Range: 0 - 1000	mS
175	VAR_RS485_CONFIG		W	Y	R/W	Rs485 configuration: 0 - normal IP over PPP 1 - ModBus	
176	VAR_PPP_BAUDRATE NOTE: Does NOT apply to MVOB.		W	Y	R/W	RS232/485 (normal mode) baud rate: 2 - 9600 3 - 19200 4 - 38400 5 - 57600 6 - 115200	
177	VAR_MOVEPS	 mtn	F	N	W	Same as variable #92 but using S-curve acceleration/deceleration	
178	VAR_MOVEDS	 mtn	F	N	W	Same as variable #93 but using S-curve acceleration/deceleration	
179	VAR_MDVS_VELOCITY	 mtn		N	W	Velocity for MDV move using S-curve accel/deceleration. Writing to this variable executes MDV move with Distance value last written to variable #94 (unless motion is suspended by #91).	UU
180	VAR_MAXVEL Short Name: MAXV		F	N	R/W	Max velocity for motion profile	UU/S
181	VAR_ACCEL Short Name: ACCEL		F	N	R/W	Accel value for indexing	UU/S ²
182	VAR_DECEL Short Name: DECEL		F	N	R/W	Decel value for indexing	UU/S ²
183	VAR_QDECEL Short Name: QDECEL		F	N	R/W	Quick decel value	UU/S ²
184	VAR_INPOSLIM Short Name: INPOSLIM		W	N	R/W	Sets window for "In Position" Limits	UU
185	VAR_VEL Short Name: VEL		F	N	R/W	Velocity reference for "Profiled" velocity	UU/S
186	VAR_UNITS Short Name: UNITS		F	Y	R/W	User units	
187	VAR_MECOUNTER Short Name: MECOUNTER		W	N	R/W	A/B inputs reference counter value	Count
188	VAR_PHCUR Short Name: PHCUR		F	N	R	Phase current	A
189	VAR_POS_PULSES Short Name: TPOS PLS		W	N	R/W	Target position in encoder pulses	EC
190	VAR_APOS_PULSES Short Name: APOS PLS		W	N	R/W	Actual position in encoder pulses	EC
191	VAR_POSEERROR_PULSES Short Name: PERROR PLS		W	N	R	Position error in encoder pulses	EC

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
192	VAR_CURRENT_VEL_PPS		F	N	R	Set-point (target) velocity in PPS	PPS
193	VAR_CURRENT_ACCEL_PPSS		F	N	R	Set-point (target) acceleration (demanded value) value	PPSS
194	VAR_IN0_DEBOUNCE		W	Y	R/W	Input A1 de-bounce time in mS Range: 0 - 1000	mS
195	VAR_IN1_DEBOUNCE		W	Y	R/W	Input A2 de-bounce time in mS Range: 0 - 1000	mS
196	VAR_IN2_DEBOUNCE		W	Y	R/W	Input A3 de-bounce time in mS Range: 0 - 1000	mS
197	VAR_IN3_DEBOUNCE		W	Y	R/W	Input A4 de-bounce time in mS Range: 0 - 1000	mS
198	VAR_IN4_DEBOUNCE		W	Y	R/W	Input B1 de-bounce time in mS Range: 0 - 1000	mS
199	VAR_IN5_DEBOUNCE		W	Y	R/W	Input B2 de-bounce time in mS Range: 0 - 1000	mS
200	VAR_IN6_DEBOUNCE		W	Y	R/W	Input B3 de-bounce time in mS Range: 0 - 1000	mS
201	VAR_IN7_DEBOUNCE		W	Y	R/W	Input B4 de-bounce time in mS Range: 0 - 1000	mS
202	VAR_IN8_DEBOUNCE		W	Y	R/W	Input C1 de-bounce time in mS Range: 0 - 1000	mS
203	VAR_IN9_DEBOUNCE		W	Y	R/W	Input C2 de-bounce time in mS Range: 0 - 1000	mS
204	VAR_IN10_DEBOUNCE		W	Y	R/W	Input C3 de-bounce time in mS Range: 0 - 1000	mS
205	VAR_IN11_DEBOUNCE		W	Y	R/W	Input C4 de-bounce time in mS Range: 0 - 1000	mS
206	VAR_OUT1_FUNCTION		W	Y	R/W	Programmable Output function 0 - Not Assigned 1 - Zero Speed 2 - In Speed Window 3 - Current Limit 4 - Run time fault 5 - Ready 6 - Brake 7 - In position	
207	VAR_OUT2_FUNCTION		W	Y	R/W	Programmable Output Function. See range (settings) for Variable #206	
208	VAR_OUT3_FUNCTION		W	Y	R/W	Programmable Output Function. See range (settings) for Variable #206	
209	VAR_OUT4_FUNCTION		W	Y	R/W	Programmable Output Function. See range (settings) for Variable #206	
210	VAR_HALLCODE		W	N	R	Current hall code Bit 0 - Hall 1 Bit 1 - Hall 2 Bit 2 - Hall 3	
211	VAR_ENCODER		W	N	R	Primary encoder current value	EC
212	VAR_RPOS_PULSES Short Name: RPOS_PLS		W	N	R	Registration position in encoder pulses	EC
213	VAR_RPOS Short Name: RPOS		F	N	R	Registration position	UU
214	VAR_POS Short Name: TPOS		F	N	R/W	Target position	UU

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
215	VAR_APOS Short Name: APOS		F	N	R/W	Actual position	UU
216	VAR_POSEERROR Short Name: PERROR		W	N	R	Position error	EC
217	VAR_CURRENT_VEL Short Name: TV		F	N	R	Set-point (target) velocity (demanded value)	UU/S
218	VAR_CURRENT_ACCEL Short Name: TA		F	N	R	Set-point (target) acceleration (demanded value)	UU/S ²
219	VAR_TPOS_ADVANCE Short Name: TPOS_ADV		W	N	W	Target position advance. Every write to this variable adds value to the Target position summing point. Value gets added once per write. This variable useful when loop is driven by Master encoder signals and trying to correct phase. Value is in encoder counts	EC
220	VAR_IOINDEX Short Name: INDEX		W	N	R/W	Same as INDEX variable in user's program. See "INDEX" in Language Reference section of this manual.	
221	VAR_PSLIMIT_PULSES		W	Y	R/W	Positive Software limit switch value in Encoder counts	EC
222	VAR_NSLIMIT_PULSES		W	Y	R/W	Negative Software limit switch value in Encoder counts	EC
223	VAR_ SLS_MODE		W	Y	R/W	Soft limit switch action code: 0 - no action 1- Fault. 2- Stop and fault (When loop is driven by trajectory generator only. With all the other sources same action as 1) --	
224	VAR_PSLIMIT		F	Y	R/W	Same as var 221 but value in User Units	UU
225	VAR_NSLIMIT		F	Y	R/W	Same as var 222 but value in User Units	UU
226	VAR_SE_APOS_PULSES		W	N	R	2nd encoder actual position in encoder counts	EC
227	VAR_SE_POSEERROR_PULSES		W	N	R	2nd encoder position error in encoder counts	EC
228	VAR_MODEBUS_PARITY		W	Y	R/W	Parity for Modbus Control: 0 - No Parity 1 - Odd Parity 2 - Even Parity	
229	VAR_MODEBUS_STOPBITS		W	Y	R/W	Number of Stopbits for Modbus Control: 0 - 1.0 1 - 1.5 2 - 2.0	
230	VAR_M_NOMINALVEL		F	Y	R/W	Induction Motor Nominal Velocity Range: 500 - 20000 RPM	RPM
231	VAR_M_COSPHI		F	Y	R/W	Induction Motor Cosine Phi Range: 0 - 1.0	
232	VAR_M_BASEFREQUENCY		F	Y	R/W	Induction Motor Base Frequency: Range: 0 - 400Hz	Hz
233	VAR_M_SERIES					Induction Motor Series	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
234	VAR_CAN_BAUD_EPM		W	Y	R/W	CAN Bus Parameter: Baud Rate: 1 - 8 1 - 10k 2 - 20k 3 - 50k 4 - 125k 5 - 250k 6 - 500k 7 - 800k 8 - 1000k	
235	VAR_CAN_ADDR_EPM		W	Y	R/W	CAN Bus Parameter: Address: 1-127	
236	VAR_CAN_OPERMODE_EPM		W	Y	R/W	CAN Bus Parameter: Boot-up Mode: 0 - 2 (Operational State Control) 0 - enters into pre-operational state 1 - enters into operational state 2 - pseudo NMT: sends NMT Start Node command after delay (set by variable 237)	
237	VAR_CAN_OPERDELAY_EPM		W	Y	R/W	CAN Bus Parameter: pseudo NMT mode delay time in seconds (refer to variable 236)	sec
238	VAR_CAN_ENABLE_EPM		W	Y	R/W	CAN Bus Parameter: Mode Control: 0, 1, 2 0 - Disable CAN interface 1 - Enable CAN interface in DS301 mode Concurrent user's program execution possible 2 - Enable CAN interface in DS402 mode Concurrent user's program execution possible 3 - Enable DeviceNet 4 - Enable PROFIBUS DP	
239	VAR_HOME_ACCEL		F	Y		Homing Mode: ACCEL rate: 0 - 10000000.0	UU/sec ²
240	VAR_HOME_OFFSET		F	Y	R/W	Homing Mode: Home Position Offset Range: -32767 to +32767	UU
241	VAR_HOME_OFFSET_PULSES		W	Y	R/W	Homing Mode: Home Position Offset in encoder counts Range: +/- 2,147,418,112	EC
242	VAR_HOME_FAST_VEL		F	Y	R/W	Homing Mode: Fast Velocity Range: -10,000 to +10,000	UU/sec
243	VAR_HOME_SLOW_VEL		F	Y	R/W	Homing Mode: Slow Velocity Range: -10,000 to +10,000	UU/sec
244	VAR_HOME_METHOD		W	Y	R/W	Homing Mode: Homing Method Range: 1 - 35	
245	VAR_START_HOMING Short Name: HOME		W	N	W	Homing Mode: Start Homing: 0, 1 0 - No action 1 - Start Homing	
246	VAR_HOME_SWITCH_INPUT		W	Y	R/W	Homing Mode: Switch Input Assignment: Range: 0 - 11 0 - 3: A1 - A4 4 - 7: B1 - B4 8 - 11: C1 - C4 Warning: If using A1, A2 or C3 refer to the homing section. Do not use input A3 as homing switch.	



Reference

Index	Name	Type	Format	EPM	Access	Description	Units
247	VAR_M_VALIDATE_MOTOR		W	N	W	Initiate / accept drive motor parameters entered in motor data PIDs. Motor parameters are variables whose identifier starts with VAR_M_xxxxxx 0 - No Action 1 - Validate Motor Data	
248	VAR_M_I2T		F	Y	R/W	Not used	
249	VAR_M_EABSOLUTE		F	Y	R/W	Indicates type of ABS encoder for models with ABS encoder support. Otherwise currently not active.	
250	VAR_M_ABSWAP		F	Y	R/W	Motor Encoder Feedback: B leads A 0 - No Action 1 - B leads A for forward checked (active)	
251	VAR_M_HALLS_INVERTED		F	Y	R/W	Motor Encoder Feedback: Halls 0 - No Action 1 - Inverted Halls Box checked (active)	
252	RESERVED					Do NOT use	
253	RESERVED					Do NOT use	
254	RESERVED					Do NOT use	
255	RESERVED					Do NOT use	
256	RESERVED					Do NOT use	
257	RESERVED					Do NOT use	
258	RESERVED					Do NOT use	
259	RESOLVER_EMU_TRK		W	Y	R/W	Resolver Emulation Track Number Range: 0 - 15 0 - 1024 1 - 256 2 - 360 3 - 400 4 - 500 5 - 512 6 - 720 7 - 800 8 - 1000 9 - 1024 10 - 2000 11 - 2048 12 - 2500 13 - 2880 14 - 250 15 - 4096	
260	RESERVED						
261	VAR_CIP_LINK_A_IN_CTRL		W	Y	R/W	Datalink "A" for input assembly (Refer to Ethernet/IP manual for details)	
262	VAR_CIP_LINK_B_IN_CTRL		W	Y	R/W	Datalink "B" for input assembly (Refer to Ethernet/IP manual for details)	
263	VAR_CIP_LINK_C_IN_CTRL		W	Y	R/W	Datalink "C" for input assembly (Refer to Ethernet/IP manual for details)	
264	VAR_CIP_LINK_D_IN_CTRL		W	Y	R/W	Datalink "D" for input assembly (Refer to Ethernet/IP manual for details)	
265	VAR_CIP_LINK_A_OUT_CTRL		W	Y	R/W	Datalink "A" for output assembly (Refer to Ethernet/IP manual for details)	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
266	VAR_CIP_LINK_B_OUT_CTRL		W	Y	R/W	Datalink "B" for output assembly (Refer to Ethernet/IP manual for details)	
267	VAR_CIP_LINK_C_OUT_CTRL		W	Y	R/W	Datalink "C" for output assembly (Refer to Ethernet/IP manual for details)	
268	VAR_CIP_LINK_D_OUT_CTRL		W	Y	R/W	Datalink "D" for output assembly (Refer to Ethernet/IP manual for details)	
269	VAR_CIP_DAT_REG_CTRL		W	Y	R/W	Data format control for Ethernet/IP assemblies (Refer to Ethernet/IP manual for details)	
270	VAR_CIP_CTRL_REG		W	Y	R/W	Control register for control via Ethernet/IP (Refer to Ethernet/IP manual for details)	
271	VAR_CIP_STATUS_REG		W	N	R	Status register 2 (Format for Ethernet/IP) (Refer to Ethernet/IP manual for details)	
272	VAR_CIP_HEART_BEAT		W	Y	R/W	CIP Heart beat timer (Ethernet/IP)	
273	VAR_EIP_MCACT_TTL		W	Y	R/W	Ethernet/IP multicast "time to leave" parameter	
274	VAR_EIP_MCAST_CTRL		W	Y	R/W	Multicast enable/disable control register (Ethernet/IP)	
275	EIP_MCAST_ADDRESS		W	Y	R/W	Multicast address (Default = 239,192,15,32)	
276	DNET_SCALE_POLL_IO		W	Y	R/W	DeviceNet polled IO data scale factor (Refer to DeviceNet manual for details)	
277	TCP_REPLY_DELAY		W	Y	R/W	TCP reply delay value	
278	RESERVED					Do NOT use	
279	RESERVED					Do NOT use	
280	RESERVED					Do NOT use	
281	RESERVED					Do NOT use	
282	RESERVED					Do NOT use	

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
	NOTE: PIDs 283 - 309 are for REFERENCE ONLY. These variables are set through MotionView. Do NOT use directly.						
283	PBUS_ADDR		W	Y	R/W	Profibus address	
284	PBUS_DOUT_SIZE		W	Y	R/W	Number of Profibus Data Out channels Range: 0 - 12	
285	PBUS_DIN_SIZE		W	Y	R/W	Number of Profibus Data In channels Range: 0 - 12	
286	PBUS_OUT_LINK1		W	Y	R/W	Profibus Data Out, Channel link 1 PID map	
287	PBUS_OUT_LINK2		W	Y	R/W	Profibus Data Out, Channel link 2 PID map	
288	PBUS_OUT_LINK3		W	Y	R/W	Profibus Data Out, Channel link 3 PID map	
289	PBUS_OUT_LINK4		W	Y	R/W	Profibus Data Out, Channel link 4 PID map	
290	PBUS_OUT_LINK5		W	Y	R/W	Profibus Data Out, Channel link 5 PID map	
291	PBUS_OUT_LINK6		W	Y	R/W	Profibus Data Out, Channel link 6 PID map	
292	PBUS_OUT_LINK7		W	Y	R/W	Profibus Data Out, Channel link 7 PID map	
293	PBUS_OUT_LINK8		W	Y	R/W	Profibus Data Out, Channel link 8 PID map	
294	PBUS_OUT_LINK9		W	Y	R/W	Profibus Data Out, Channel link 9 PID map	
295	PBUS_OUT_LINK10		W	Y	R/W	Profibus Data Out, Channel link 10 PID map	
296	PBUS_OUT_LINK11		W	Y	R/W	Profibus Data Out, Channel link 11 PID map	
297	PBUS_OUT_LINK12		W	Y	R/W	Profibus Data Out, Channel link 12 PID map	
298	PBUS_IN_LINK1		W	Y	R/W	Profibus Data In, Channel link 1 PID map	
299	PBUS_IN_LINK2		W	Y	R/W	Profibus Data In, Channel link 2 PID map	
300	PBUS_IN_LINK3		W	Y	R/W	Profibus Data In, Channel link 3 PID map	
301	PBUS_IN_LINK4		W	Y	R/W	Profibus Data In, Channel link 4 PID map	
302	PBUS_IN_LINK5		W	Y	R/W	Profibus Data In, Channel link 5 PID map	
303	PBUS_IN_LINK6		W	Y	R/W	Profibus Data In, Channel link 6 PID map	
304	PBUS_IN_LINK7		W	Y	R/W	Profibus Data In, Channel link 7 PID map	
305	PBUS_IN_LINK8		W	Y	R/W	Profibus Data In, Channel link 8 PID map	
306	PBUS_IN_LINK9		W	Y	R/W	Profibus Data In, Channel link 9 PID map	
307	PBUS_IN_LINK10		W	Y	R/W	Profibus Data In, Channel link 10 PID map	
308	PBUS_IN_LINK11		W	Y	R/W	Profibus Data In, Channel link 11 PID map	
309	PBUS_IN_LINK12		W	Y	R/W	Profibus Data In, Channel link 12 PID map	
310	PBUS_ACYC_MODE		W	Y	R/W	Profibus Acyclic Mode Type Refer to Profibus Manual (P94PFB01)	
	NOTE: PIDs 311 - 406 are for REFERENCE ONLY. These variables are set through MotionView. Do NOT use directly These variables are used by MotionView for non-volatile settings of CAN TPDO/RPDO.						
311	VAR_RPDO_1_COM					Receive PDO	
312	VAR_RPDO_2_COM						
313	VAR_RPDO_3_COM						
314	VAR_RPDO_4_COM						

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
315	VAR_RPDO_5_COM						
316	VAR_RPDO_6_COM						
317	VAR_RPDO_7_COM						
318	VAR_RPDO_8_COM						
319	VAR_RPDO_1_MAP1					RPDO Mapping	
320	VAR_RPDO_1_MAP2						
321	VAR_RPDO_1_MAP3						
322	VAR_RPDO_1_MAP4						
323	VAR_RPDO_2_MAP1						
324	VAR_RPDO_2_MAP2						
325	VAR_RPDO_2_MAP3						
326	VAR_RPDO_2_MAP4						
327	VAR_RPDO_3_MAP1						
328	VAR_RPDO_3_MAP2						
329	VAR_RPDO_3_MAP3						
330	VAR_RPDO_3_MAP4						
331	VAR_RPDO_4_MAP1						
332	VAR_RPDO_4_MAP2						
333	VAR_RPDO_4_MAP3						
334	VAR_RPDO_4_MAP4						
335	VAR_RPDO_5_MAP1						
336	VAR_RPDO_5_MAP2						
337	VAR_RPDO_5_MAP3						
338	VAR_RPDO_5_MAP4						
339	VAR_RPDO_6_MAP1						
340	VAR_RPDO_6_MAP2						
341	VAR_RPDO_6_MAP3						
342	VAR_RPDO_6_MAP4						
343	VAR_RPDO_7_MAP1						
344	VAR_RPDO_7_MAP2						
345	VAR_RPDO_7_MAP3						
346	VAR_RPDO_7_MAP4						
347	VAR_RPDO_8_MAP1						
348	VAR_RPDO_8_MAP2						
349	VAR_RPDO_8_MAP3						
350	VAR_RPDO_8_MAP4						
351	VAR_TPDO_1_COM					Transmit PDO	
352	VAR_TPDO_2_COM						
353	VAR_TPDO_3_COM						
354	VAR_TPDO_4_COM						
355	VAR_TPDO_5_COM						
356	VAR_TPDO_6_COM						

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
357	VAR_TPDO_7_COM						
358	VAR_TPDO_8_COM						
359	VAR_TPDO_1_MAP1					TPDO Mapping	
360	VAR_TPDO_1_MAP2						
361	VAR_TPDO_1_MAP3						
362	VAR_TPDO_1_MAP4						
363	VAR_TPDO_2_MAP1						
364	VAR_TPDO_2_MAP2						
365	VAR_TPDO_2_MAP3						
366	VAR_TPDO_2_MAP4						
367	VAR_TPDO_3_MAP1						
368	VAR_TPDO_3_MAP2						
369	VAR_TPDO_3_MAP3						
370	VAR_TPDO_3_MAP4						
371	VAR_TPDO_4_MAP1						
372	VAR_TPDO_4_MAP2						
373	VAR_TPDO_4_MAP3						
374	VAR_TPDO_4_MAP4						
375	VAR_TPDO_5_MAP1						
376	VAR_TPDO_5_MAP2						
377	VAR_TPDO_5_MAP3						
378	VAR_TPDO_5_MAP4						
379	VAR_TPDO_6_MAP1						
380	VAR_TPDO_6_MAP2						
381	VAR_TPDO_6_MAP3						
382	VAR_TPDO_6_MAP4						
383	VAR_TPDO_7_MAP1						
384	VAR_TPDO_7_MAP2						
385	VAR_TPDO_7_MAP3						
386	VAR_TPDO_7_MAP4						
387	VAR_TPDO_8_MAP1						
388	VAR_TPDO_8_MAP2						
389	VAR_TPDO_8_MAP3						
390	VAR_TPDO_8_MAP4						
391	VAR_TPDO_1_COM_ET						
392	VAR_TPDO_2_COM_ET						
393	VAR_TPDO_3_COM_ET						
394	VAR_TPDO_4_COM_ET						
395	VAR_TPDO_5_COM_ET						
396	VAR_TPDO_6_COM_ET						



NOTE: PIDs 311 - 406 are for REFERENCE ONLY. Do **NOT** use directly. These variables are used by MotionView for non-volatile settings of CAN TPDO/RPDO

Reference

Index	Name	Type	Format	EPM	Access	Description	Units
397	VAR_TPDO_7_COM_ET						
398	VAR_TPDO_8_COM_ET						
399	VAR_TPDO_1_COM_IT						
400	VAR_TPDO_2_COM_IT						
401	VAR_TPDO_3_COM_IT						
402	VAR_TPDO_4_COM_IT						
403	VAR_TPDO_5_COM_IT						
404	VAR_TPDO_6_COM_IT						
405	VAR_TPDO_7_COM_IT						
406	VAR_TPDO_8_COM_IT						
407	VAR_CAN_HEARTBEAT				R/W	CAN Heartbeat rate (0x1017) Range: 0 - 65335 milliseconds	
408	VAR_PBUS_STATUS				R	PROFIBUS Status	
409	VAR_PBUS_MASTER_TIMEOUT_VAL				R/W	Timeout Value for PROFIBUS master	
410	VAR_PBUS_DATA_EXCHANGE_TIMEOUT				R/W	Data Exchange Timeout for PROFIBUS Range: 0 - 327680 milliseconds	
411	VAR_PTC_RX				R	PTC resistance in ohms	
412	VAR_PBUS_FIRMWARE_REV					PROFIBUS firmware revision (hex number): 1 ST word = major; Least word = minor Example: 0x00010001 = rev 1.1	
413	VAR_PBUS_TIMEOUT_ACTION_CFG			Y		PROFIBUS timeout action. Bits encoded as: Data Exchange Timeout: Bit 0 = 1 Fault, Bit 0 = 0 No Action Master Monitor Timeout: Bit 1 = 1 Fault, Bit 1 = 0 No Action Module Timeout (card not present): Bit 2 = 1 Fault, Bit 2 = 0 No Action	
433	VAR_BRAKE_RELEASE_DELAY				R/W	Range: 0 - 1000 milliseconds; Default 0mS	mS



NOTE: PIDs 311 - 406 are for REFERENCE ONLY. Do **NOT** use directly. These variables are used by MotionView for non-volatile settings of CAN TPDO/RPDO

3.3 Quick Start Examples

Contained in the following four sections are the connections and parameter settings to quickly setup a PositionServo drive for External Torque/Velocity, External Positioning, Internal Torque/Velocity and Internal Positioning modes. These Quick Start reference tables are NOT a substitute for reading the PositionServo User Manual. Observe all safety notices in the PositionServo User and Programming Manuals.

3.3.1 Quick Start - External Torque/Velocity

Table 64: Connections for External Torque/Velocity Mode

I/O (P3)		
Pin	Name	Function
5	GND	Drive Logic Common
6	+5V	+5V Output (max 100mA)
7	BA+	Buffered Encoder Output: Channel A+
8	BA-	Buffered Encoder Output: Channel A-
9	BB+	Buffered Encoder Output: Channel B+
10	BB-	Buffered Encoder Output: Channel B-
11	BZ+	Buffered Encoder Output: Channel Z+
12	BZ-	Buffered Encoder Output: Channel Z-
22	ACOM	Analog common
23	A01	Analog output
24	AIN1+	Positive (+) of Analog signal input
25	AIN1 -	Negative (-) of Analog signal input
26	IN_A_COM	Digital input group A COM terminal
27	IN_A1	Digital input A1
28	IN_A2	Digital input A2
29	IN_A3	Digital input A3
41	RDY+	Ready output Collector
42	RDY-	Ready output Emitter
43	OUT1-C	Programmable output #1 Collector
44	OUT1-E	Programmable output #1 Emitter
45	OUT2-C	Programmable output #2 Collector
46	OUT2-E	Programmable output #2 Emitter
47	OUT3-C	Programmable output #3 Collector
48	OUT3-E	Programmable output #3 Emitter
49	OUT4-C	Programmable output #4 Collector
50	OUT4-E	Programmable output #4 Emitter

Note 1: Connections highlighted in BLUE are mandatory/necessary for operation in this mode.

Reference

Table 65: Parameter Settings for External Torque/Velocity Mode

MVOB Folder	Sub-Folder	Setting	
Parameters	--	Parameter Name	Description
		Drive Mode	Set to [Torque] for Torque Mode; [Velocity] for Velocity Mode
		Analog Input (Current Scale)	Torque Mode Only: Set to Required Amps per Volt
		Analog Input (Velocity Scale)	Velocity Mode Only: Set to Required RPM per Volt
		Enable Accel/Decel Limits	Velocity Mode Only: Set to [Enable] to switch on velocity ramp rates; Set to [Disable] to switch OFF (accelerate at current limit)
		Accel Limit	Velocity Mode Only: Set Acceleration Limit in RPM/Sec
		Decel Limit	Velocity Mode Only: Set Deceleration Limit in RPM/Sec
		Reference	Set to [External] for external Torque/Velocity Mode
		Enable Switch Input	Set to [Run] to allow Enable/Disable of the PositionServo to be controlled via Input A3 (Dedicated Enable)
IO	Digital IO	Parameter Name	Description
		Output 1 Function	Output # indicates Digital Output No. 1-4; Set value to select Output Functionality; Output Function Values: 1=Not Assigned; 2=Zero Speed; 3=In Speed Window; 4=Current Limit; 5=Run Time Fault; 6=Ready; 7=Brake; 8=In Position
		Output 2 Function	
		Output 3 Function	
		Output 4 Function	
IO	Analog IO	Parameter Name	Description
		Analog Input Dead Band	Set Zero Speed Dead Band in mV for Torque/Velocity Reference on Analog Input 1
		Analog Input Offset	Set Torque/Velocity Reference Input Offset on Analog Input 1 to match Controller Offset
		Adjust Analog Input Zero Offset	Tool to automatically learn the Analog Input Offset (of Analog Input 1)
Limits	Velocity Limits	Parameter Name	Description
		Zero Speed	Velocity Mode Only: Set a bandwidth (around ORPM) for activation of the Zero Speed Output/Flag. Set digital output function to '2'.
		At Speed	Velocity Mode Only: Set a Target Speed for activation of the At Speed Output/Flag
		Speed Window	Velocity Mode Only: Set a bandwidth (around At Speed parameter) for activation of the At Speed Output/Flag. Set digital output function to '3'.
Compensation	--	Parameter Name	Description
		Velocity P-Gain	Velocity Mode Only: Set P-Gain for Velocity Loop
		Velocity I-Gain	Velocity Mode Only: Set I-Gain for Velocity Loop
		Gain Scaling	Velocity Mode Only: Apply Scaling Factor to Velocity Gain Set

Note 1: Parameters highlighted in BLUE are mandatory/necessary for operation in this mode.

Reference

3.3.2 Quick Start - External Positioning

Table 66: Connections for External Positioning Mode

I/O (P3)		
Pin	Name	Function
1	MA+	Master Encoder A+ / Step+ input
2	MA-	Master Encoder A- / Step- input
3	MB+	Master Encoder B+ / Direction+ input
4	MB-	Master Encoder B- / Direction- input
5	GND	Drive Logic Common
6	+5V	+5V Output (max 100mA)
7	BA+	Buffered Encoder Output: Channel A+
8	BA-	Buffered Encoder Output: Channel A-
9	BB+	Buffered Encoder Output: Channel B+
10	BB-	Buffered Encoder Output: Channel B-
11	BZ+	Buffered Encoder Output: Channel Z+
12	BZ-	Buffered Encoder Output: Channel Z-
26	IN_A_COM	Digital input group A COM terminal
27	IN_A1	Digital input A1
28	IN_A2	Digital input A2
29	IN_A3	Digital input A3
41	RDY+	Ready output Collector
42	RDY-	Ready output Emitter
43	OUT1-C	Programmable output #1 Collector
44	OUT1-E	Programmable output #1 Emitter
45	OUT2-C	Programmable output #2 Collector
46	OUT2-E	Programmable output #2 Emitter
47	OUT3-C	Programmable output #3 Collector
48	OUT3-E	Programmable output #3 Emitter
49	OUT4-C	Programmable output #4 Collector
50	OUT4-E	Programmable output #4 Emitter

Note 1: Connections highlighted in BLUE are mandatory/necessary for operation in this mode.

Note 2: Connections highlighted in GREEN are frequently required in applications of this type.

Reference

Table 67: Parameter Settings for External Positioning Mode

MVOB Folder	Sub-Folder	Setting	
Parameters	--	Parameter Name	Description
		Drive Mode	Set to [Position] for Position Mode
		Reference	Set to [External] for external Position Mode
		Step Input Type	Set to either [Step and Direction] or [Master Encoder] to match the Position Controller
		System to Master Ratio	Set Electronic Gear Ratio for Reference Signal to the PositionServo Motor Output
		Enable Switch Input	Set to [Run] to allow Enable/Disable of the PositionServo to be controlled via Input A3 (Dedicated Enable)
		Resolver Track	If using Resolver Feedback, set value that represents the pulses per revolution required on the PositionServo simulated encoder. 0=1024ppr; 1=256ppr; 2=360ppr; 3=400ppr; 4=500ppr; 5=512ppr; 6=720ppr; 7=800ppr; 8=1000ppr; 9=1024ppr; 10=2000ppr; 11=2048ppr; 12=2500ppr; 13=2880ppr; 14=250ppr; 15=4096ppr
IO	Digital IO	Parameter Name	Description
		Output 1 Function	Output # indicates Digital Output No. 1-4; Set value to select Output Functionality; Output Function Values: 1=Not Assigned; 2=Zero Speed; 3=In Speed Window; 4=Current Limit; 5=Run Time Fault; 6=Ready; 7=Brake; 8=In Position
		Output 2 Function	
		Output 3 Function	
		Output 4 Function	
Hard Limit Switches Action	Set to Enable Inputs A1 and A2 to act as System Hard Limit Switches and define functionality in the event of an active input.		
Limits	Position Limits	Parameter Name	Description
		Position Error	Set Position Error Limit at which Position Error Timer starts counting
		Max Error Time	Set Maximum Error Time for Position Error Correction before position error trip occurs.
Compensation	--	Parameter Name	Description
		Velocity P-Gain	Set P-Gain for Velocity Loop
		Velocity I-Gain	Set I-Gain for Velocity Loop
		Position P-Gain	Set P-Gain for Position Loop
		Position I-Gain	Set I-Gain for Position Loop
		Position D-Gain	Set D-Gain for Position Loop
		Position I-Limit	The Position I-Limit will clamp the Position I-Gain compensator to prevent excessive torque overshoot caused by an over-accumulation of I-Gain.
Gain Scaling	Apply Scaling Factor to Velocity Gain Set		

Note 1: Parameters highlighted in BLUE are mandatory/necessary for operation in this mode.

Reference

3.3.3 Quick Start - Internal Torque/Velocity

Table 68: Internal Torque/Velocity Mode

Connections for Internal Torque/Velocity: I/O (P3)			Variable References for Internal Torque/Velocity				
Pin	Name	Function	Index	Name	EPM	R/W	Description
20	AIN2+	Positive (+) of Analog signal input	29	VAR_ENABLE_SWITCH_TYPE	Y	R/W	Enable switch function: 0-inhibit only, 1- Run
21	AIN2-	Negative (-) of Analog signal input	34	VAR_DRIVEMODE	Y	R/W	Drive mode selection: 0-torque 1-velocity, 2-position
22	ACOM	Analog common	37	VAR_REFERENCE	Y	R/W	Reference source: set to: 1 - internal (for 'internal torque' or 'internal velocity' mode)
23	A01	Analog output	44	VAR_VP_GAIN	Y	R/W	Velocity loop Proportional gain Range: 0 - 32767
24	AIN1+	Positive (+) of Analog signal input	45	VAR_VI_GAIN	Y	R/W	Velocity loop Integral gain Range: 0 - 16383
25	AIN1 -	Negative (-) of Analog signal input	51	VAR_VREG_WINDOW	Y	R/W	Gains scaling coefficient Range: -5 - +4
26	IN_A_COM	Digital input group A COM terminal	52	VAR_ENABLE	N	W	Software Enable/Disable: 0 – disable, 1 - enable
27	IN_A1	Digital input A1	58	VAR_VLIMIT_ZEROSPEED	Y	R/W	Zero Speed value Range: 0 - 100
28	IN_A2	Digital input A2	59	VAR_VLIMIT_SPEEDWND	Y	R/W	Speed window Range: 10 - 10000
29	IN_A3	Digital input A3	60	VAR_VLIMIT_ATSPEED	Y	R/W	Target speed for velocity window Range: -10000 - +10000
30	IN_A4	Digital input A4	71	VAR_AIN1	N	R	Analog Input AIN1 current value
31	IN_B_COM	Digital input group B COM terminal	72	VAR_AIN2	N	R	Analog Input AIN2 current value
32	IN_B1	Digital input B1	75	VAR_ENABLE_ACCELDECEL	Y	R/W	Enable Accel/Decel (velocity mode), 0 – disable, 1 - enable
33	IN_B2	Digital input B2	76	VAR_ACCEL_LIMIT	Y	R/W	Accel value for velocity mode Range: 0.1 - 5000000
34	IN_B3	Digital input B3	77	VAR_DECEL_LIMIT	Y	R/W	Decel value for velocity mode Range: 0.1 - 5000000
35	IN_B4	Digital input B4	139	VAR_IREF	N	R/W	Internal ref Current or Velocity mode
36	IN_C_COM	Digital input group C COM terminal	192	VAR_CURRENT_VEL_PPS	N	R	Current velocity in PPS (pulses per sample)
37	IN_C1	Digital input C1	193	VAR_CURRENT_ACCEL_PPSS	N	R	Current acceleration (demanded value) value
38	IN_C2	Digital input C2	217	VAR_CURRENT_VEL	N	R	Current velocity (demanded value)
39	IN_C3	Digital input C3	218	VAR_CURRENT_ACCEL	N	R	Current acceleration (demanded value)
40	IN_C4	Digital input C4					
41	RDY+	Ready output Collector					
42	RDY-	Ready output Emitter					
43	OUT1-C	Programmable output #1 Collector					
44	OUT1-E	Programmable output #1 Emitter					
45	OUT2-C	Programmable output #2 Collector					
46	OUT2-E	Programmable output #2 Emitter					
47	OUT3-C	Programmable output #3 Collector					
48	OUT3-E	Programmable output #3 Emitter					
49	OUT4-C	Programmable output #4 Collector					
50	OUT4-E	Programmable output #4 Emitter					

Positional Mode Language Reference - Enable/Disable		
Command	Syntax	Long Name
DISABLE	DISBALE	Turns OFF Servo output
ENABLE	ENABLE	Turns ON Servo output

Note 1: Connections **highlighted in BLUE** are mandatory/necessary for operation in this mode.

Example Internal Torque Program

```
;Program slowly increases Motor Torque until nominal motor current is reached
VAR_DriveMode = 0 ;Set Drive to Torque mode
VAR_Reference = 1 ;Set Reference to Internal control
Program Start:
IREF = 0 ;Reset Torque Reference to 0(Amps)
Wait While !In_A3 ;Wait while Enable input is OFF
Enable ;Enable Drive
Torque_Loop:
Wait Time 500 ;Set time between step increases in Torque
If REF < VAR_CurrentLimit ;If Set Torque < Motor Nominal Torque
IREF = IREF+0.1 ;Then increase by 0.1(Amps)
GOTO Torque_Loop ;Loop to next torque increase
Else
Goto Program_Start ;Else restart program
Endif
END
```

Example Internal Velocity Program

```
;Program slowly increases and decreases Motor Velocity between Maximum Velocity Forward direction and
;Maximum Velocity Reverse direction producing a saw-tooth velocity profile.
Define MaxVelocityRPS 60 ;Enter Maximum Velocity (RPS) value here
Define VelocityStepRPS 1 ;Define Velocity INC/DEC per Step/Program Loop (RPS)
Define VelocityStepTime 200 ;Define Time for Velocity Steps in mS
Define Velocity_Inc_Dec V0 ;Define a Variable to identify if Velocity is currently INC/DEcreasing
VAR_DriveMode = 1 ;Set Drive to Velocity mode
VAR_Reference = 1 ;Set Reference to Internal control
VAR_Enable_AccelDecel = 1 ;Enable Accel/Decel Ramps
VAR_Accel_Limit = 3000 ;Set Accel Rate required in RPS^2
VAR_Decel_Limit = 3000 ;Set Decel Rate required in RPS^2
Program Start:
IREF = 0 ;Reset Velocity Reference to 0(RPS)
Wait While !In_A3 ;Wait while Enable input is OFF
Enable ;Enable Drive
Velocity_Loop:
Wait Time VelocityStep Time ;Set Time between Step Increases/Decreases in Velocity (mS)
If REF <= MaxVelocityRPS ;If Current Motor Velocity < MaxVelocityRPS
IREF = IREF+VelocityStepRPS ;Then increase Velocity by VelocityStepRPS
Else
Velocity_Inc_Dec = 1 ;Set Variable to start decreasing velocity
Endif
Else ;If Speed Decreasing
If REF >= -1* MaxVelocityRPS ;If Current Motor Velocity > -MaxVelocityRPS
IREF = IREF-VelocityStepRPS ;Then decrease Velocity by VelocityStepRPS
Else
Velocity_Inc_Dec = 0 ;Set Variable to start increasing velocity
Endif
Endif
Goto Velocity_Loop ;Loop to next Velocity Increase/Decrease
END ;End Code - Never Reached
On Fault ;Fault Handler
Resume Program_Start ;Resume at Program Start
EndFault
```

Reference

3.3.4 Quick Start - Internal Positioning

Table 69: Internal Positioning

Connections: I/O (P3)		
Pin	Name	Function
26	IN_A_COM	Digital input group A COM terminal
27	IN_A1	Digital input A1
28	IN_A2	Digital input A2
29	IN_A3	Digital input A3
30	IN_A4	Digital input A4
31	IN_B_COM	Digital input group B COM terminal
32	IN_B1	Digital input B1
33	IN_B2	Digital input B2
34	IN_B3	Digital input B3
35	IN_B4	Digital input B4
36	IN_C_COM	Digital input group C COM terminal
37	IN_C1	Digital input C1
38	IN_C2	Digital input C2
39	IN_C3	Digital input C3
40	IN_C4	Digital input C4
41	RDY+	Ready output Collector
42	RDY-	Ready output Emitter
43	OUT1-C	Programmable output #1 Collector
44	OUT1-E	Programmable output #1 Emitter
45	OUT2-C	Programmable output #2 Collector
46	OUT2-E	Programmable output #2 Emitter
47	OUT3-C	Programmable output #3 Collector
48	OUT3-E	Programmable output #3 Emitter
49	OUT4-C	Programmable output #4 Collector
50	OUT4-E	Programmable output #4 Emitter

Language Reference		
Enable/Disable		
Command	Syntax	Long Name
DISABLE	DISBALE	Turns OFF Servo output
ENABLE	ENABLE	Turns ON Servo output

Program Structure		
Command	Syntax	Long Name
STOP MOTION	STOP MOTION	Stop AA Motion - Clear
STOP MOTION QUICK	STOP MOTION QUICK	Motion Slack
WAIT	WAIT MOTION COMPLETE	Wait

Move / Motion Commands		
Command	Syntax	Long Name
MOVE	MOVE [BACK] UNTIL <condition> [,C]	Move
MOVED	MOVED <distance> [,S] [,C]	Move Distance
MOVEP	MOVEP <absolute position> [,S] [,C]	Move to Position
MOVEDR	MOVEDR <distance> , <displacement> [,C]	Registered Distance Move
MOVEPR	MOVEPR <distance> , <displacement> [,C]	Registered Position Move
MDV	MDV <[-]segment distance> , <segment final velocity> [,S]	Segmented Move
MOTION SUSPEND	MOTION SUSPEND	Temporarily Suspend Motion
MOTION RESUME	MOTION RESUME	Statement Resumes Motion

Example Internal Positioning Program

```
;***** HEADER *****
;Title:                Pick and Place example program
;Author:               940 Product Management
;Description:          This is a sample program that shows a simple application that
;                      picks up a part moves to a set position and drops the part

;***** I/O List *****
;  Input A1           -      not used
;  Input A2           -      not used
;  Input A3           -      Enabled
;  Input A4           -      not used
;  Input B1           -      not used
;  Input B2           -      not used
;  Input B3           -      not used
;  Input B4           -      not used
;  Input C1           -      not used
;  Input C2           -      not used
;  Input C3           -      not used
;  Input C4           -      not used
;
;  Output 1           -      Pick Arm
;  Output 2           -      Gripper
;  Output 3           -      not used
;  Output 4           -      not used

;***** Initialize and Set Variables *****
UNITS = 1
ACCEL = 75
DECEL =75
MAXV = 10
APOS = 0

;***** Events *****
;Set Events handling here

;***** Main Program *****
RESET_DRIVE:
WAIT UNTIL IN_A3           ;Check the Enable / Inhibit switch is made before continuing
ENABLE                     ;Enable the Drive
PROGRAM_START:
MOVEP 0                    ;Move to Pick position
OUT1 = 1                   ;Turn on output 1 on to extend Pick arm
WAIT TIME 1000             ;Delay 1 sec to extend arm
OUT2 = 1                   ;Turn on output 2 to Engage gripper
WAIT TIME 1000             ;Delay 1 sec to Pick part
OUT1 = 0                   ;Turn off output 1 to Retract Pick arm
MOVEP 100                  ;Move to Place position
OUT1 = 1                   ;Turn on output 1 on to extend Pick arm
WAIT TIME 1000             ;Delay 1 sec to extend arm
OUT2 = 0                   ;Turn off output 1 to Disengage gripper
WAIT TIME 1000             ;Delay 1 sec to Place part
OUT1 = 0                   ;Retract Pick arm
GOTO PROGRAM_START
END

;***** Sub-Routines *****
;      Enter Sub-Routine code here

;***** Fault Handler Routine *****
;      Enter Fault Handler code here
ON FAULT

ENDFAULT
```

Reference

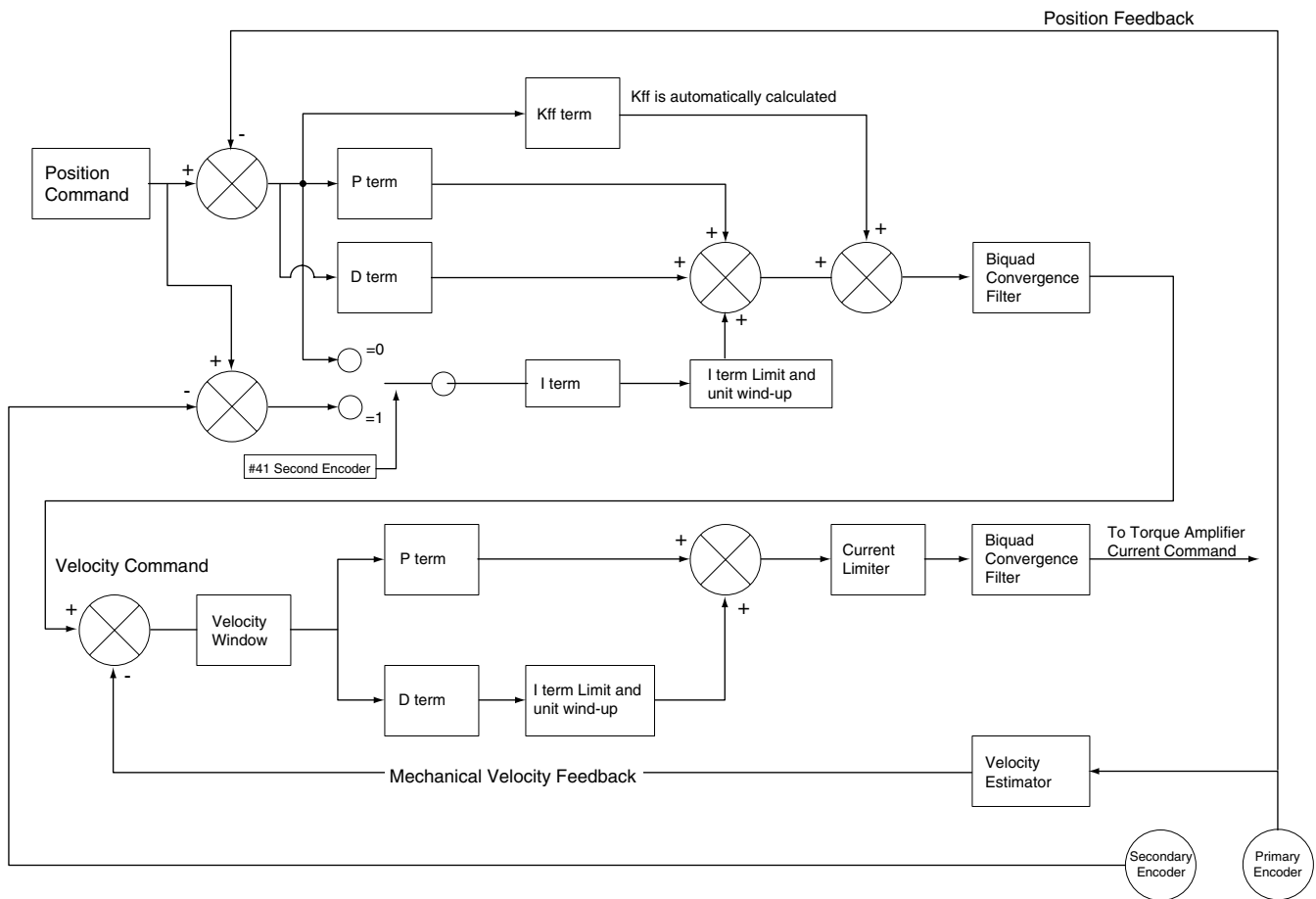
3.4 PositionServo Reference Diagrams

This section contains the process flow diagrams listed in Table 70. These diagrams are for reference only.

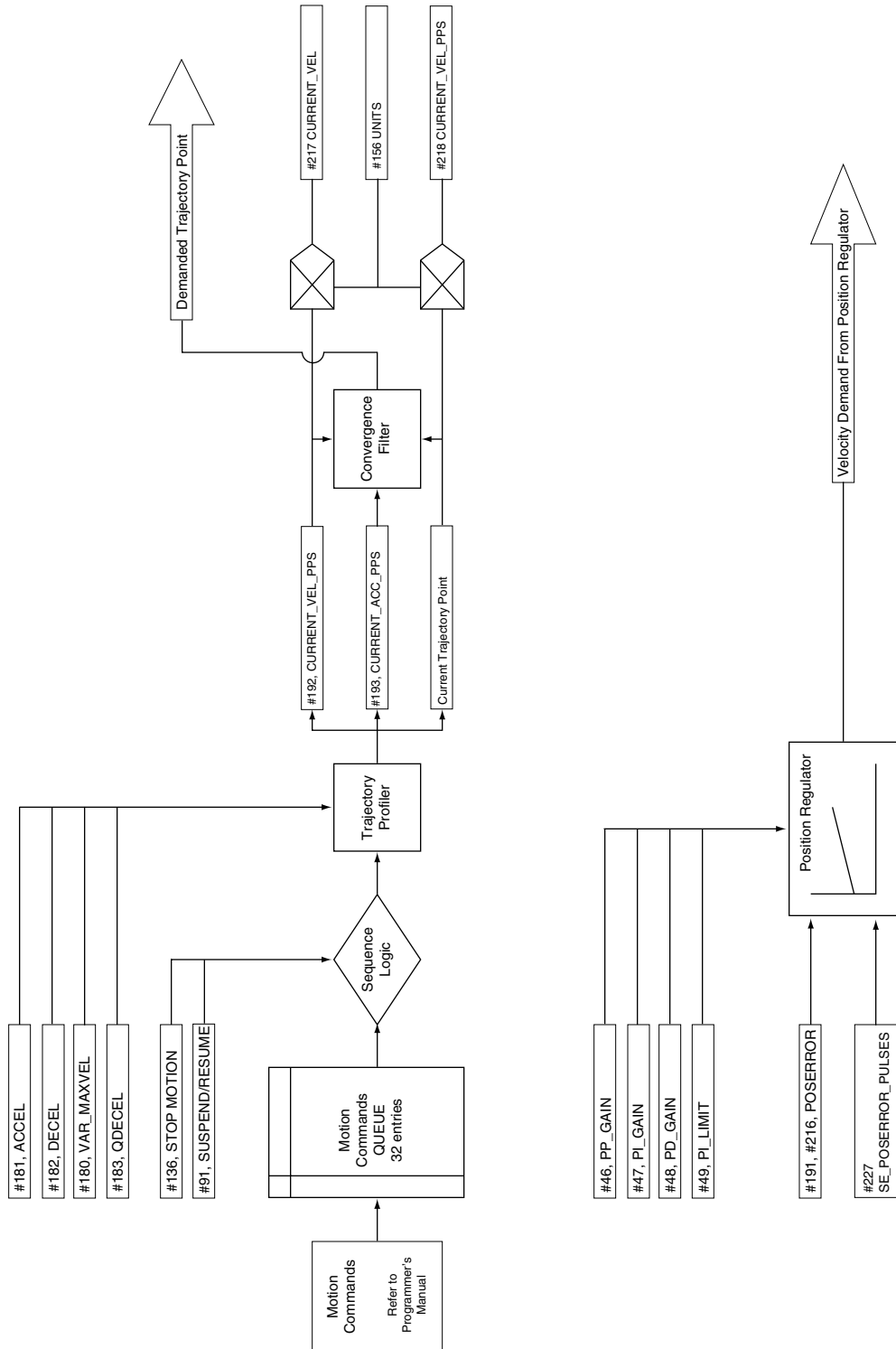
Table 70: PositionServo Process Flow Diagrams

Drawing #	Description
S999	Position and Velocity Regulator
S1000	Motion Commands -> Motion Queue -> Trajectory Generator
S1001	Current Command -> Motor
S1002	Encoder Inputs
S1003	Analog Inputs
S1004	Analog Outputs
S1005	Digital Inputs
S1006	Digital Outputs

Position and Velocity Regulators

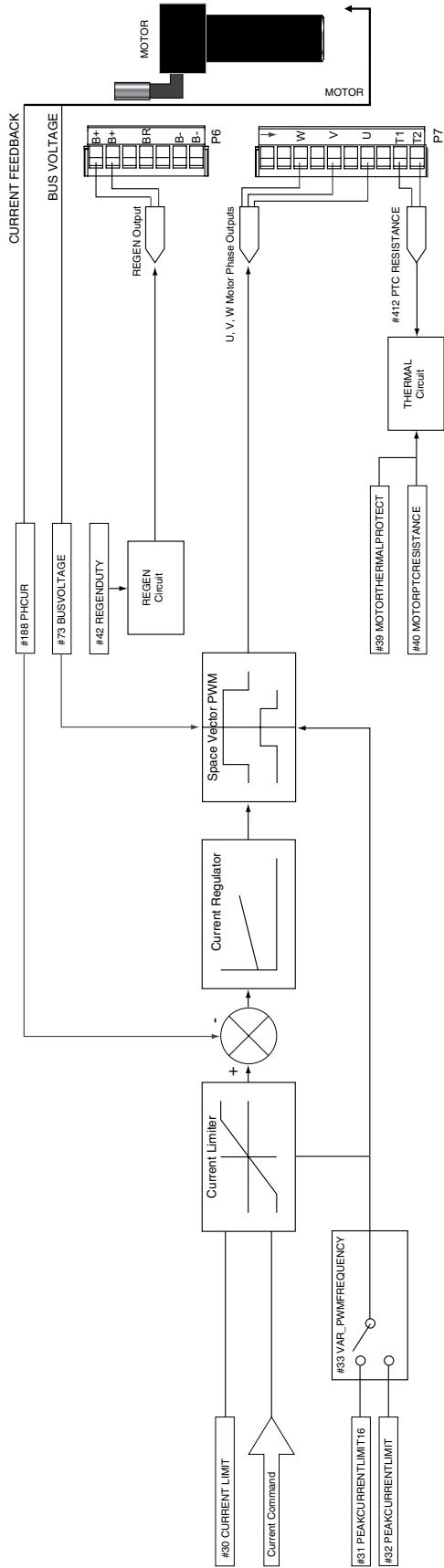


Motion Commands, Motion Queue & Trajectory Generator

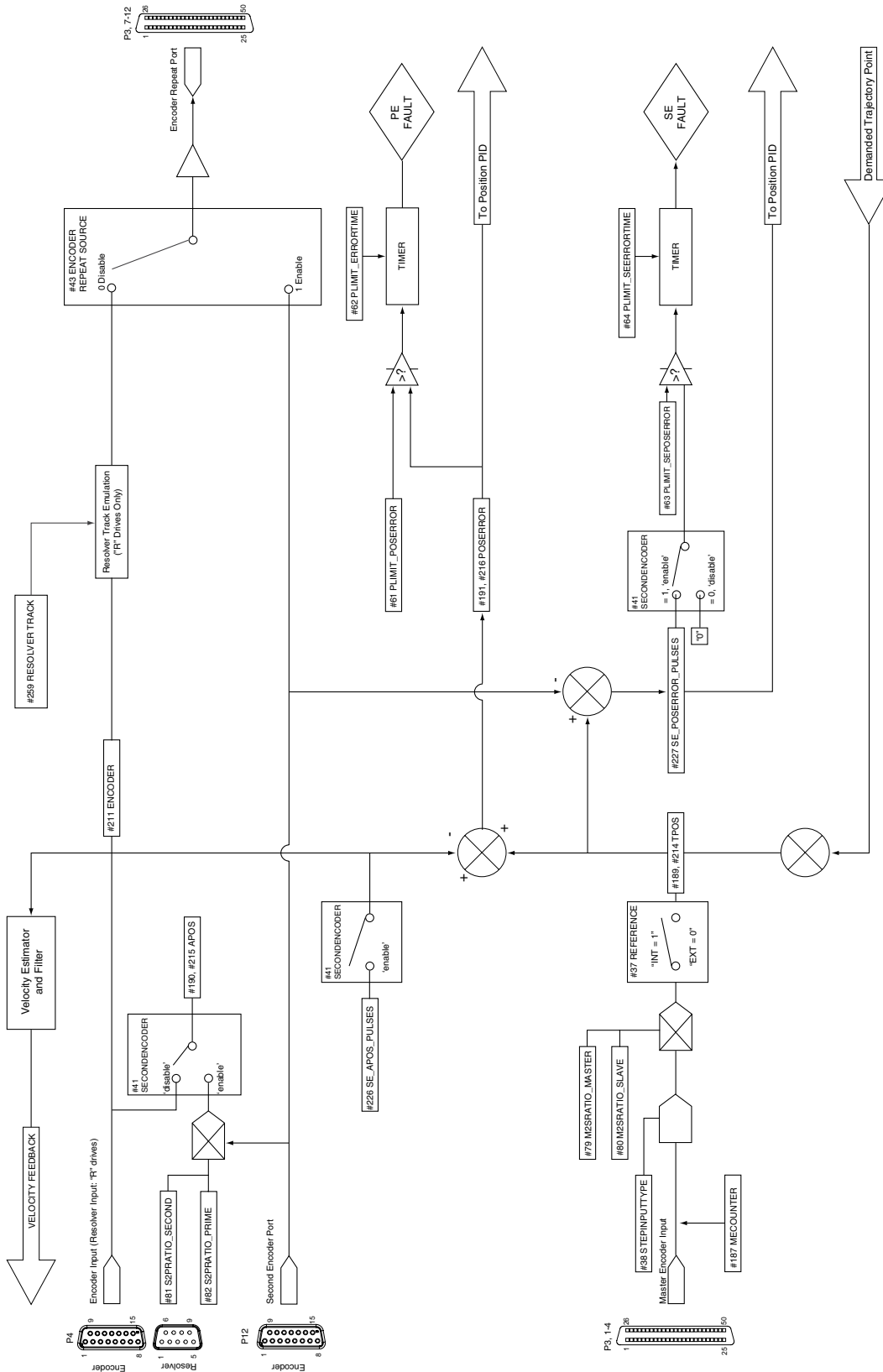


Reference

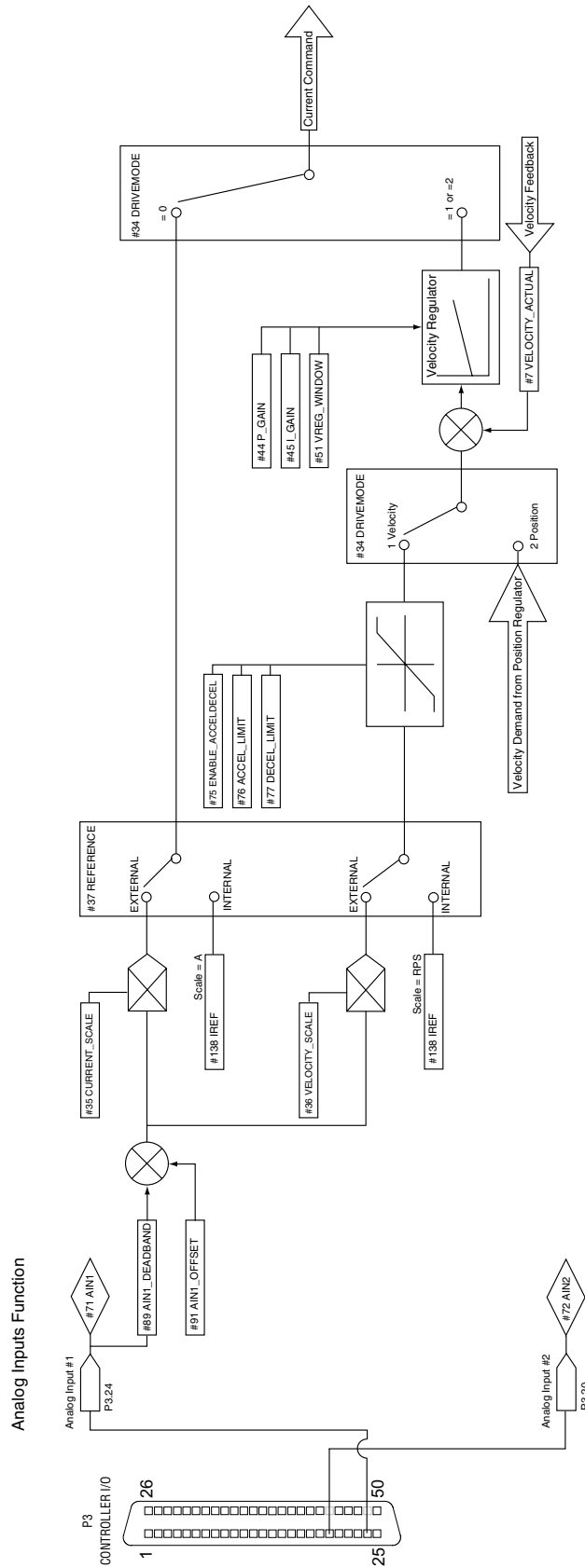
Current Command --> Motor



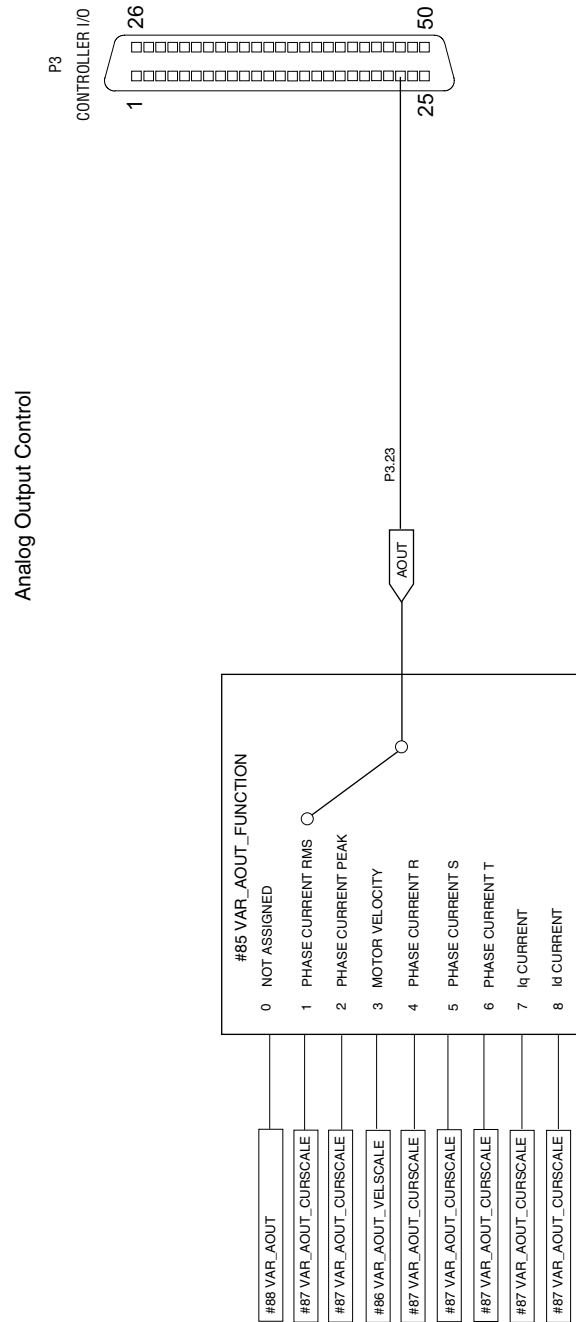
Encoder Inputs



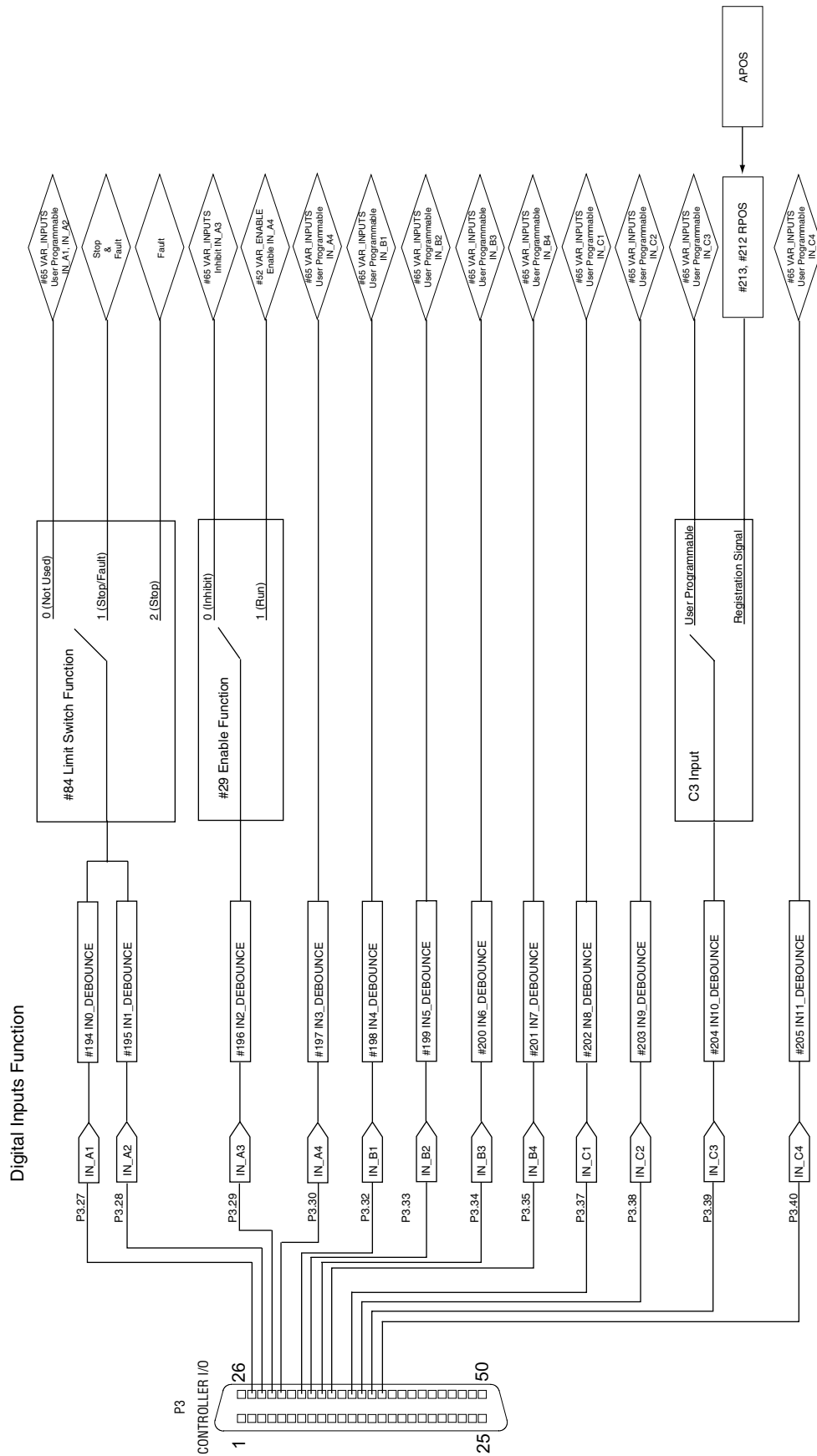
Analog Inputs



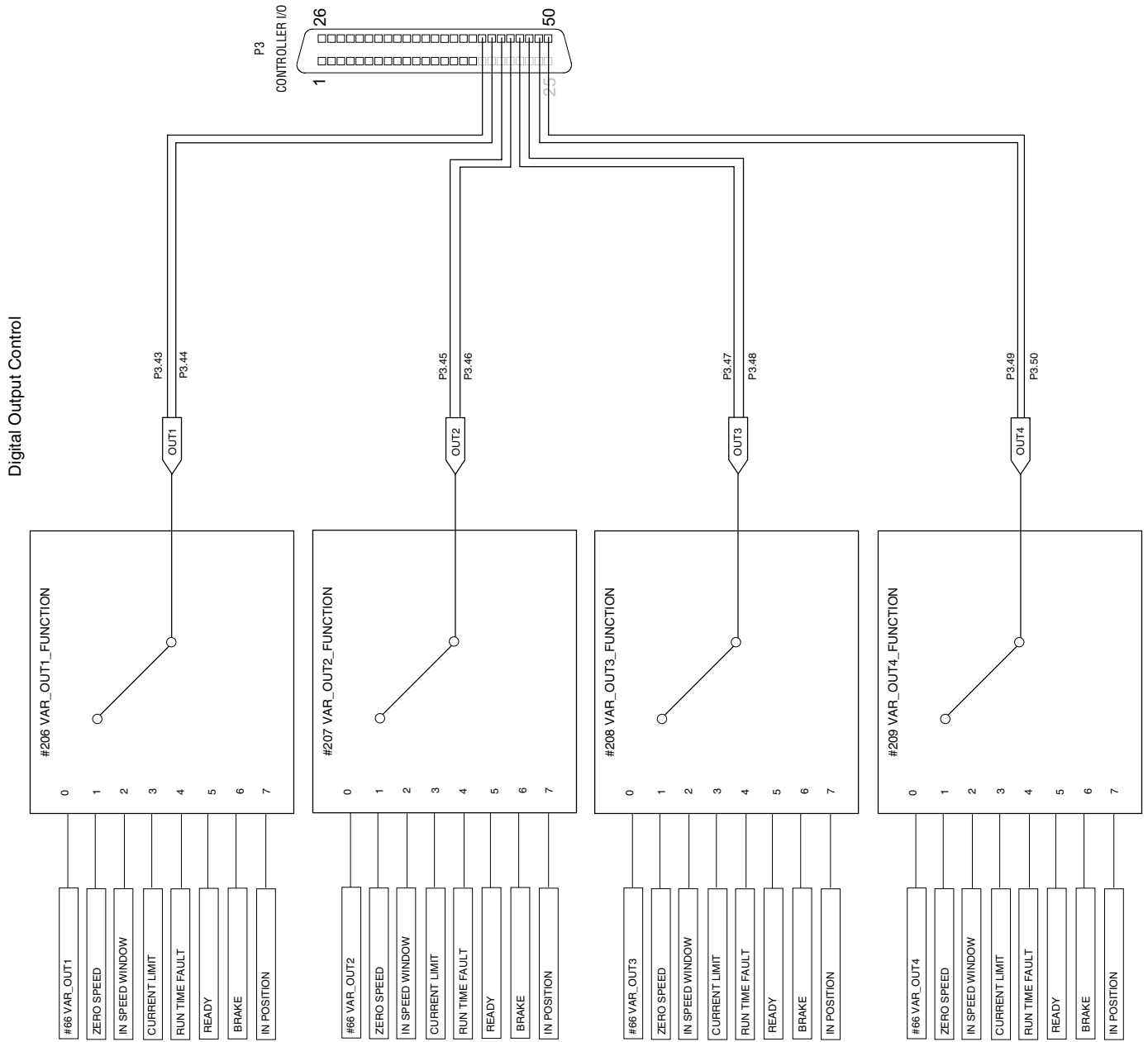
Analog Output



Digital Inputs



Digital Outputs



Lenze AC Tech Corporation

630 Douglas Street • Uxbridge, MA 01569 • USA
Sales 800 217 9100 • Service 508 278 9100
www.lenzeamericas.com

PM94201B

POSITIONServo



Dynamic Link Library (DLL) Communication Reference Guide

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.

Copyright ©2007 by AC Technology Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from AC Technology Corporation. The information and technical data in this manual are subject to change without notice. AC Tech makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. AC Tech assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView[®], PositionServo[®], and all related indicia are either registered trademarks or trademarks of Lenze AG in the United States and other countries.

Microsoft Windows[®], Visual Basic[®], Visual C++[®] and all related indicia are registered trademarks of the Microsoft Corporation in the United States and other countries.

This document printed in the United States of America

Table of Contents

1.	Safety Information	4
1.1	Warnings, Cautions & Notes	4
1.2	Reference Documents	5
2.	PositionServo DLL Overview	6
3	Files in the DLL Library.....	6
4	Communication Flowchart.....	7
5	DLL Functions Overview	8
6	Return Codes	8
7	DLL Functions Usage Examples.....	8
8	DLL Functions	9
8.1	Connection Services Functions.....	9
8.2	Data Manipulation Functions	10

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.

	<p>DANGER!</p> <ul style="list-style-type: none"> • 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.
	<p>WARNING!</p> <p>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.</p>

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)

	<p>NOTE:</p> <p>Note 1: SS940Control.dll must be kept in the project root directory or in the environment declared path.</p> <p>Note 2: SS940Control.dll and oncrpc.dll must also be in %SysPath%\SYSTEM directory.</p> <p>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.</p>
---	--

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.

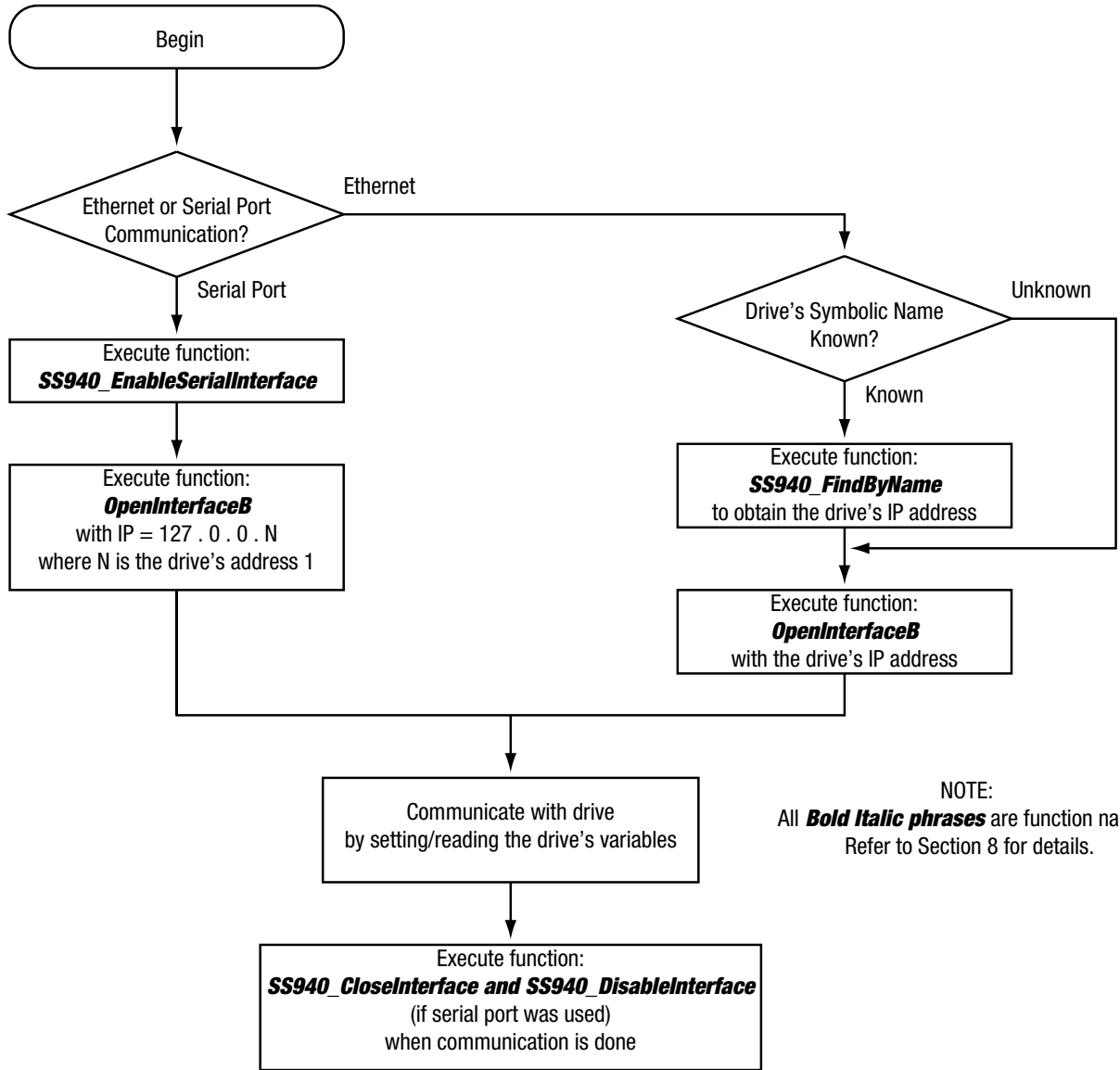


Figure 1 Communication Flowchart

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



CANopen Communication Module Communications Interface Reference Guide

Addendum to: PositionServo CANopen Communication Reference Guide, Document: P94CAN01B

Date: 06-05-2009

Note: Release of 5-pin CANopen Option Module (E94ZACAN1)

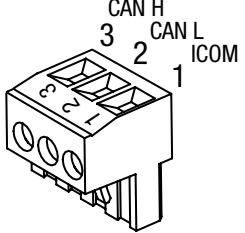
Page 10, Paragraph 2.2: Electrical Installation

Replace entire paragraph, Table 1 and Figure 2 with the following information:

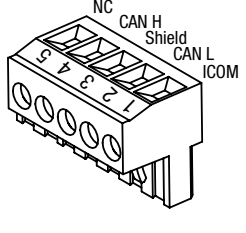
An optional CANopen communication module (E94ZACAN1) is available for the PositionServo drive. Installed in Option Bay 1 as P21, the CANopen module is optically isolated from the rest of the drive's circuitry. The 3-pin CANopen module is for HW/SW 1A10 and the 5-pin CANopen module is for HW/SW 1B10 or higher. Refer to the PS CANopen Reference Guide (P94CAN01) for more information.

CANopen Interface Pin Assignments

3-pin			5-pin		
Pin	Name	Function	Pin	Name	Function
1	ICOM	Isolated Common	1	ICOM	Isolated Common
2	CAN L	CAN Bus Low	2	CAN L	CAN Bus Low
3	CAN H	CAN Bus High	3	Shield	
			4	CAN H	CAN Bus High
			5	NC	No connection



CAN H
3
CAN L
2
ICOM
1



NC
5
CAN H
4
Shield
3
CAN L
2
ICOM
1

About These Instructions

This documentation applies to the optional CANopen Communication Module 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 manual describes the CANopen implementation developed by AC Tech Corporation for 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 CANopen before implementing a CANopen program.

Copyright ©2007 by AC Technology Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from AC Technology Corporation. The information and technical data in this manual are subject to change without notice. AC Technology makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. AC Technology assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView[®], PositionServo[®], and all related indicia are either registered trademarks or trademarks of Lenze AG in the United States and other countries.

CANopen[®] is a registered trademark of 'CAN in Automation (CiA)'.

This document is printed in the United States of America

Table of Contents

1.	Safety Information	5
1.1	Warnings, Cautions & Notes.....	5
1.2	Reference Documents.....	7
1.3	Conventions for Object Descriptions.....	7
1.4	Commonly Used Terms, Acronyms & Definitions	8
2	Installation	9
2.1	Mechanical Installation	9
2.2	Electrical Installation.....	10
3	Introduction.....	11
3.1	CAN Overview	11
3.2	PositionServo Drive Configuration	12
3.3	CAN Protocol	14
3.4	Accessing the Object Dictionary.....	16
3.4.1	SDOs and PDOs	16
3.4.2	SDOs: Description and Examples	17
3.4.3	PDOs: Description and Examples	18
3.4.4	SDO or PDO? Design Considerations.....	20
3.4.5	Mapping a PDO.....	20
3.5	Objects that Define SDO's and PDO's.....	21
4	Network Management.....	25
4.1	Network Management Overview	25
4.1.1	Network Management Services and Objects	25
4.1.2	General Device State Control.....	25
4.1.3	Device Monitoring.....	26
4.1.4	Time Stamp PDOs.....	27
4.1.5	Emergency Messages.....	27
4.2	Network Management Objects.....	28
5	Device Configuration and Control through Native Variables List	29
5.1	Native Control	29
5.2	Objects to Access the Drive's RAM Variables.....	29
6	Device Control, Configuration and Status.....	30
6.1	Device Control and Status Overview.....	30
6.1.1	Control Word, Status Word, and Device Control Function.....	30
6.1.2	State Changes Diagram	32
6.2	Device Control and Status Objects	34
6.3	Error Management Objects	37
6.4	Basic Amplifier Configuration Objects	40
6.5	Basic Motor Configuration Objects	46

7	Control Loops.....	51
7.1	Control Loop Configuration.....	51
7.1.1	Nested Control Loops.....	51
7.1.2	The Position Loop	52
7.1.3	The Velocity Loop	53
7.1.4	The Current Loop.....	54
7.2	Position Loop Configuration Objects.....	55
7.3	Velocity Loop Configuration Objects	59
7.4	Current Loop Configuration Objects.....	60
8	Non Profiled Operating Modes	62
8.1	Current Follower Mode.....	62
8.2	Velocity Follower Mode	62
9	Homing Mode	62
9.1	Homing Mode Operation	62
9.1.1	Homing Overview.....	63
9.1.2	Homing Methods	64
9.1.3	Homing Method 1: Homing on the Negative Limit Switch	64
9.1.4	Homing Method 2: Homing on the Positive Limit Switch.....	65
9.1.5	Homing Method 3 and 4: Homing on the Positive Home Switch and Index Pulse	65
9.1.6	Homing Methods 5 and 6: Homing on the Negative Home Switch and Index Pulse	66
9.1.7	Homing Methods 7-14: Homing on the Home Switch and Index Pulse.....	66
9.1.8	Homing Methods 15, 16, 20, 22, 24, 26, 28, and 30: Reserved	67
9.1.9	Homing Methods 17 and 18: Homing without an Index Pulse	67
9.1.10	Homing Methods 19, 21, 23, 25, 27, and 29: Homing without an Index Pulse	67
9.1.11	Homing Methods 31 and 32: Reserved	68
9.1.12	Homing Methods 33 and 34: Homing on the Index Pulse.....	68
9.1.13	Homing Method 35: Homing on the Current Position	68
9.2	Homing Mode Operation Objects.....	69
10	Profile Position and Profile Velocity Mode Operation	71
10.1	Profile Position Mode Operation Overview	71
10.1.1	Point-to-Point Motion Profiles	71
10.1.2	Handling a Series of Point-to-Point Moves	72
10.1.3	Point-to-Point Move Parameters and Related Data.....	73
10.1.4	Point-To-Point Move Sequence Examples.....	75
10.2	Profile Velocity Mode Operation	76
10.2.1	Position and Velocity Loops.....	76
10.3	Profile Position, Profile Velocity Mode Objects.....	77

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 drive user manual includes information on compliance with 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/EEC (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. Damaging or destroying electrical components may pose a danger to 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 standards 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.

	<p>DANGER!</p> <ul style="list-style-type: none"> • After the controller has been disconnected from the supply voltage, do not touch live components or the power connection until capacitors have had enough time to discharge. Wait at least 3 minutes before servicing the drive. 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.
	<p>WARNING!</p> <p>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.</p>

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

- CAN and CANopen Specifications: “CAN in Automation (CiA)”; visit: <http://www.can-cia.de/>
- PositionServo Programming Manual: PM94P01; Refer to: <http://www.actech.com>
- PositionServo User Manual: S94P01; Refer to: <http://www.actech.com>
- MotionView Software Manual: IM94MV01; Refer to: <http://www.actech.com>

1.3 Conventions for Object Descriptions

In this manual, object descriptions are as illustrated in the sample below. Each object description includes summary information including object type, access, units, range, map PDO and memory capability in tabular format. The table header includes the object title and index/sub-index number.

Object				Index	Sub-Index
SERVER SDO Parameters				0x1200	
Type	Access	Units	Range	Map PDO	Memory
Record	RO	--	--	NO	--
Description The Server SDO object holds the COB-ID values needed for access the drive's SDO. Sub-index 0 holds the number of sub-elements of the record. The COB-ID is the communication object ID or the CAN message ID.					

Object				Index	Sub-Index
SDO Receive COB-ID				0x1200	1
Type	Access	Units	Range	Map PDO	Memory
Unassigned 32	RO	--	0x600 – 0x671	NO	--
Description The SDO Receive COB-ID is the CANopen object ID used by the drive to receive an SDO packet. The SDO Receive COB-ID value is 0x600 plus the drive's CAN node ID.					

Relationships of Sub-Index Objects

This manual describes both objects and sub-index objects. Object descriptions and Sub-Index object descriptions are included in the bottom portion of the table. The Sub-index object 0 always contains the number of elements contained by the record.

Object Summary Description Fields

Field Name	Description
Type	The object type: Record, Array, Float, Visible String, Integer (8/16/32), Unassigned (8/16/32)
Access	The object's access type: RO, WO, RW
	RO: read only
	WO: write only
	RW: read and write
Units	The units used to express the object's value.
Range	The acceptable range of values if less than that specified by Type.
Map PDO	Object PDO map capability: YES, NO, EVENT
	YES: the object can be mapped to a PDO
	NO: the object cannot be mapped to a PDO
	EVENT: the object can be mapped and can be set to event triggering.
Memory	F: object can be held in the amplifier's flash memory
	R: object can be held in the amplifier's RAM
	RF: object can be held in the amplifier's flash memory and RAM
	-- (dash): object can not be stored or object contains sub-index objects

1.4 Commonly Used Terms, Acronyms & Definitions

CAN	Control Area Network
CANopen	Communication protocol to open and communicate with the Control Area Network
CMS	CAN-based Message Specification
COB-ID	Communication Object Identifier
CP	Communication Profile
CRC	Cyclic Redundancy Check
DCF	Device Configuration File: an ASCII file containing a description of the object configuration of an individual device
DP	Device Profile: defines the OD objects for a particular type of device
EDS	Electronic Data Sheet: an ASCII file containing a device's communication functionality and objects; plus its device-specific objects and their default values
EMCY	Emergency Object
Index	4-digit hexadecimal code used to identify an object: 16-bits Sub-Index: decimal code to further identify object's parameters: 8-bits
NMT	Network Management
OD	Object Dictionary
Object	Communication message - 4 types: Administrative: NMT Service Data Object: SDO Process Data Object: PDO Pre-Defined or Special Function Object: SYNC; TIME STAMP; EO
PDO	Process Data Object:
RPDO	Receive PDO
SDO	Service Data Object:
SYNC	Synchronization Special Function Object
TIME STAMP	Provides a common time reference, implemented as a CMS object
TPDO	Transmit PDO (also known as TxPDO)
Transferring Data:	SDO: for large low priority data transfer between devices (i.e. configuring devices on CANopen network) PDO: for fast data transfer of 8 bytes or less without protocol overhead (i.e. the data content has been previously defined)
Transmission Modes:	Synchronous: synchronization by receipt of a SYNC message Cyclic: transmission triggered periodically after every 'n' SYNC message Acylic: transmission 'pre-triggered' by remote transmission request from another device or by the occurrence of an object-specific event specified in the device profile Asynchronous: transmission is triggered by a remote transmission request from another device (by a CAN Frame)

2 Installation

2.1 Mechanical Installation

Install the CAN Communications Module as illustrated in Figure 1.

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 CAN COMM Module (E94ZACAN1) in Option Bay 1.

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

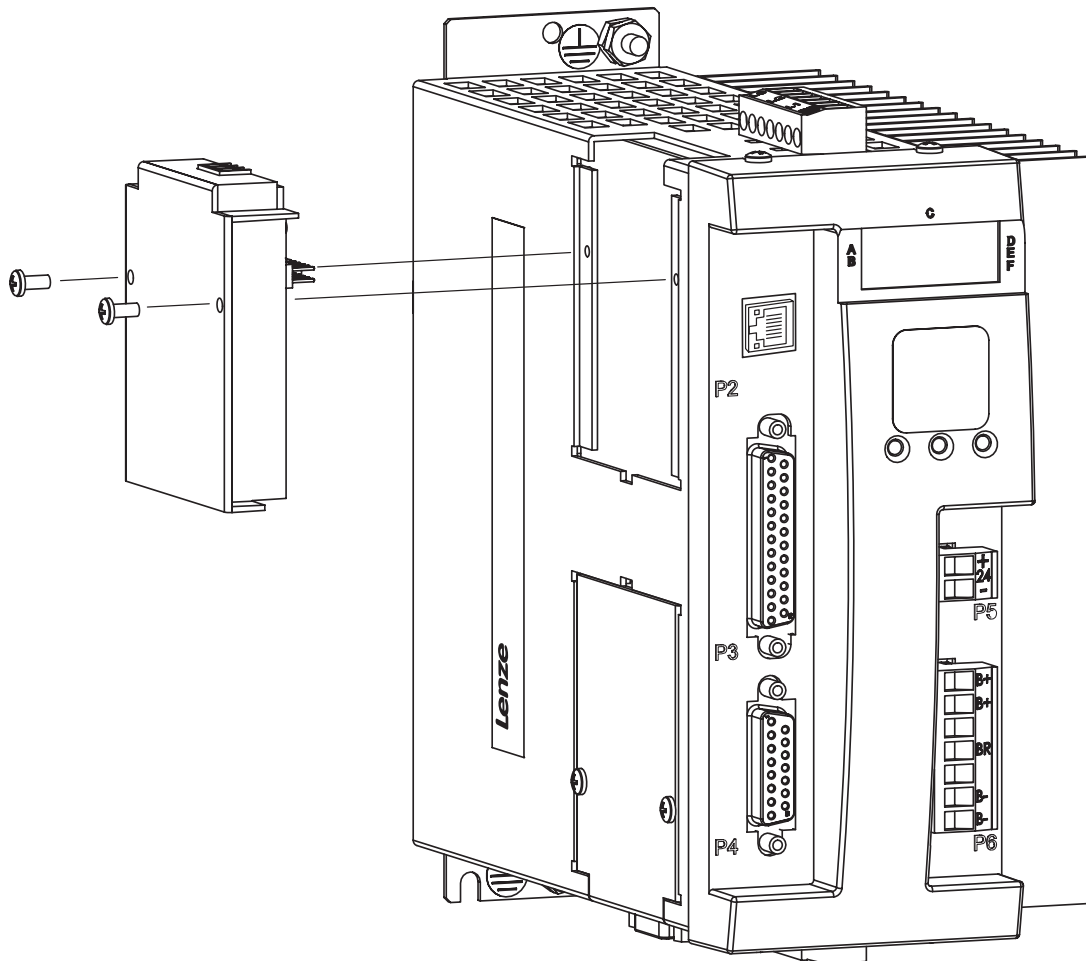


Figure 1: Installation of CAN Communications Module

2.2 Electrical Installation

Table 1 and Figure 2 illustrate the pinout of the PositionServo CAN Module connector. This connector provides 2-wire plus isolated ground connection to the network.

Table 1: CAN Bus Interface Pin Assignments

Terminal	Name	Description
1	ICOM	Isolated Common
2	CAN L	CAN Bus Low
3	CAN H	CAN Bus High



Figure 2: CAN Bus Interface Pinout

Connections and Shielding

Figure 3 illustrates the connection of the cables for a PositionServo drive in a CAN master/slave network. A 120ohm (1%) termination is necessary between the high and low terminals of the first and last slaves in the network.

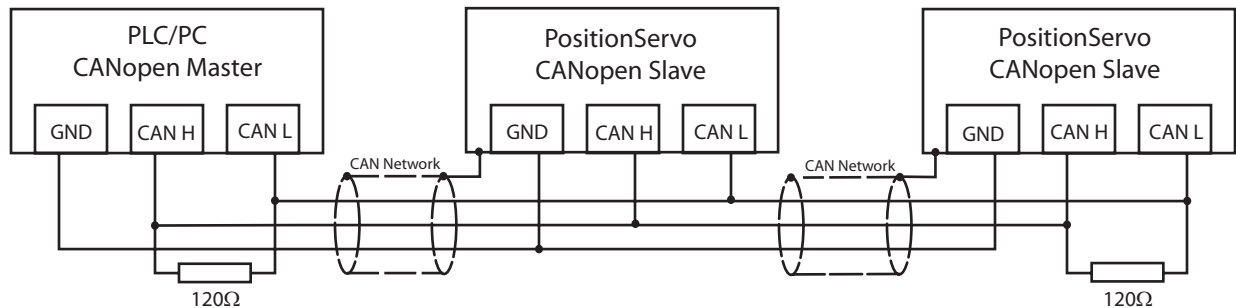


Figure 3: 120Ω (1%) Termination in CAN Network

3 Introduction

This reference guide assumes that the reader has a working knowledge of CANopen protocol and familiarity with the programming and operation of motion control equipment. This guide is intended as a reference only. The optional CANopen communication module (P/N E94ZACAN1) is required for the PositionServo drive to communicate on a CAN network.

3.1 CAN Overview

The backbone of CANopen is CAN, a serial bus network originally designed by Robert Bosch GmbH to coordinate multiple control systems in automobiles. The CAN model lends itself to distributed control. Any device can broadcast messages on the network. Each device receives all messages and uses filters to accept only the appropriate messages. Thus, a single message can reach multiple nodes, reducing the number of messages that need to be sent. This also greatly reduces bandwidth required for addressing, allowing distributed control at real-time speeds across the entire system.

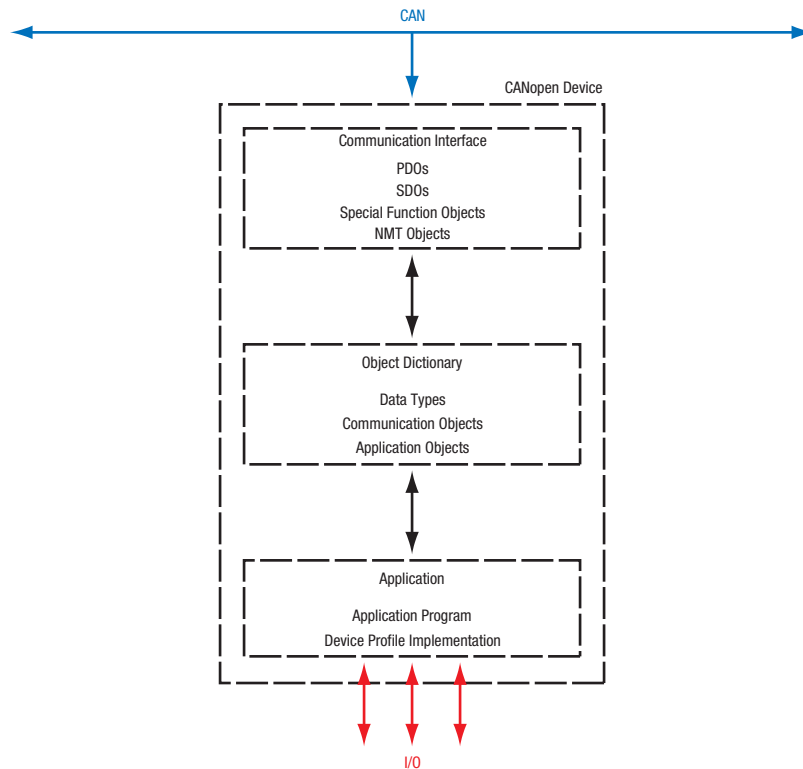


Figure 4: CANbus Network

CAN and CANopen Architecture

CAN specifies the data link and physical connection layers of a fast, reliable network. The CANopen profiles specify how various types of devices, including motion control devices, can use the CAN network in a more efficient manner.

In a CANopen motion control system, control loops are closed on the individual amplifiers, not across the network. A master application coordinates multiple devices, using the network to transmit commands and receive status information. Each device can transmit to the master or any other device on the network. CANopen provides the protocol for mapping device and master internal commands to messages that can be shared across the network. A CANopen network can support up to 127 nodes. Each node has a seven-bit node ID in the range of 1-127. (Node ID 0 is reserved and should not be used.)

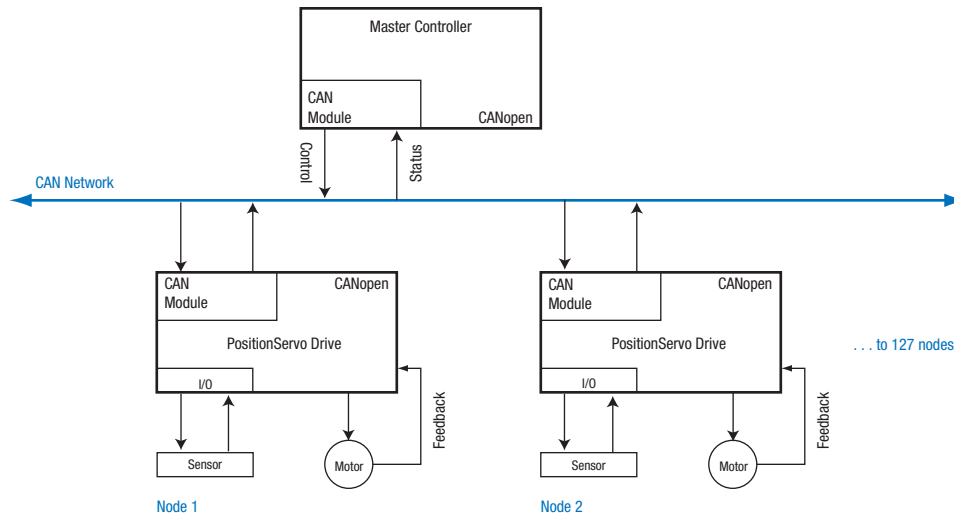


Figure 5: CANopen Network

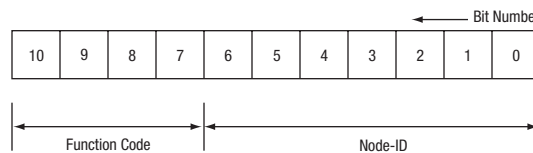


Figure 6: 11-bit CAN Identifier

Example of a CANopen Move Sequence:

- CANopen master transmits a control word to initialize all devices.
- Devices transmit messages indicating their status (in this example, all are operational).
- CANopen master transmits a message instructing devices to perform homing operations.
- Devices indicate that homing is complete.
- CANopen master transmits messages instructing devices to enter position profile mode (point-to-point motion mode) and issues first set of point-to-point move coordinates.
- Devices execute their moves, using local position, velocity, and current loops, and then transmit actual position information back to the network.
- CANopen master issues next set of position coordinates.

3.2 PositionServo Drive Configuration

Before AC Tech drives can be used in a CANopen network they need to be properly connected and configured. Refer to the PositionServo User's Manual for details on hardware connection.

There are a few parameters that need to be configured before the PositionServo can operate in a CANopen network. These parameters are listed under the <Communication> <CAN> folder in the MotionView software program. Alternatively these parameters can be reached from the drive's front display and keypad. CAN related parameters are explained herein:

- CAN Control Enabled/Disabled: Use this parameter to enable or disable CAN followed by reboot. This parameter takes effect after the drive has been re-booted (power cycled).
- CAN baud rate 10k- 1000k: Parameter takes effect after drive has been re-booted (power cycled).
- CAN address 1-127: sets drive's CAN ID. This parameter takes effect after the drive has been re-booted (power cycled).

- **CAN Boot Up Mode** Pre-Operational, Operational or Pseudo Master modes are available after power up.
 - **Pre-Operational** default mode for CAN Open slave. Drive will await message from master to enter Operational mode
 - **Operational** drive will enter Operational mode immediately after power up without receiving activation message from master. This feature is useful in a master-less network.
 - **Pseudo Master** in this mode drive will send activation message (with specified delay, see below) for all CAN slaves waiting in Pre-Operational mode. This mode is useful when emulating master functionality and activating passive slaves. Only one drive can be configured as the pseudo master and only when there is no other master device.
- **CAN Boot up delay** If drive is configured for Pseudo Master mode it will send activation message with delay specified in this parameter. Delay is used to allow specified slaves to boot up and configure their hardware to listen to the Master messages.

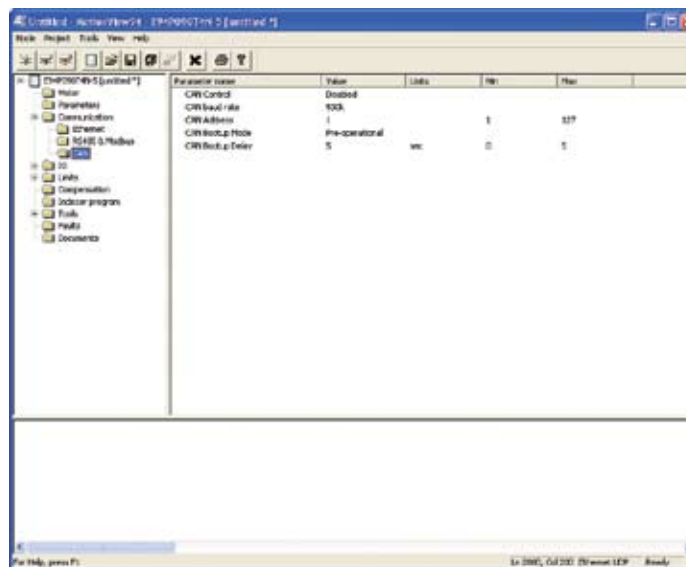


Figure 7: MotionView CANopen Communication Parameters

CANopen configuration parameters in MotionView

To configure the drive for CANopen operations follow steps:

1. Start MotionView and perform connection to the drive.
2. Open <Communication><CAN> folder to see CAN related parameters.
3. Set proper communication speed
4. Set drive's CAN ID
5. Set the [CAN Bootup Mode]. Select pre-operational mode if the drive is going to be part of CANopen network with a CANopen Master controller (or the other drive is set to Pseudo-Master mode)
6. If you selected "Pseudo Master Mode" in step 5, set the [CAN Bootup Delay].
7. Set the [CAN Control] parameter to "Enable"
8. Re-boot the drive.
9. After drive boots it enters selected boot up mode and is ready for operation under CAN control
10. To turn off CAN mode, set parameter [CAN Control] to disable and re-boot the drive. Drive will return to normal operation mode thereafter.

**NOTE:**

1. The user program is disabled while in CAN control mode and attempting to access it will result in a warning message. The program however is not erased from the memory and will be made available for execution upon return from CANopen mode ([CAN Control] set to Disable).
2. MotionView can be connected to drive for monitoring of parameters, reviewing faults, oscilloscope operations etc. Drive's variables can be viewed from <Tools><Diagnostic> folder.

All CAN parameters can be accessed from the front panel. The parameter names as seen from the built in display are:

Name	Description	Input	Value
CANb	CAN baud rate	1	10k
		2	25k
		3	50k
		4	125k
		5	250k
		6	500k
		7	800k
		8	1000k
CAnA	CAN address	1 - 127	
Cano	Bootup Op mode	0	pre-operational
		1	operational
		2	pseudo master mode
CAnE	CAN control parameter	0	disabled
		1	enabled

**NOTE:**

CAnE , CAnA and CANb require a reboot to change.

3.3 CAN Protocol

The physical layer of CAN is a differentially driven, two-wire bus, terminated by 120-ohm resistors at each end. The maximum bit rate supported by CAN is 1,000,000 bits/second for up to 25 meters. Lower bit rates may be used for longer network lengths.

The CAN Message

CANopen messages are transmitted within CAN messages.

CAN Message Format

CAN messages are communicated over the bus in the form of network packets. Each packet consists of an identifier (CAN message ID), control bits, and zero to eight bytes of data. (The CAN message is sometimes referred to as a communication object or COB and the CAN message ID as a COB-ID.)

CRC Error Checking

Each packet is sent with CRC (cyclic redundancy check) information to allow controllers to identify and re-send incorrectly formatted packets.

CAN Message ID

Every CAN message has a CAN message ID. The message ID plays two important roles:

- It provides the criteria by which the message is accepted or rejected by a node.
- It determines the message's priority.

CAN Message Priority

The priority of a CAN message is encoded in the message ID. The lower the value of the message ID, the higher the priority of the message. When two or more devices attempt to transmit packets at the same time, the packet with the highest priority succeeds. The other devices back off and re-attempt later.

Objects and Dictionaries

The primary means of controlling a device on a CANopen network is by writing to device parameters, and reading device status information. For this purpose, each device defines a group of parameters that can be written, and status values that can be read. These parameters and status values are collectively referred to as the device's objects. These objects define and control every aspect of a device's identity and operation. Some objects define basic information such as device type, model, and serial number. Others are used to check device status and deliver motion commands. The entire set of objects defined by a device is called the device's Object Dictionary (OD). Every device on a CANopen network must define an object dictionary, and nearly every CANopen network message involves reading values from or writing values to the object dictionaries of devices on the network.

Object Dictionary as Interface

The object dictionary is an interface between a device and other entities on the network.

CANopen Profiles and the Object Dictionary

The CANopen profiles specify the mandatory and optional objects that comprise most of an object dictionary. The Communication Profile specifies how all devices must communicate with the CAN network. The Communication Profile specifies dictionary objects that set up a device's ability to send and receive messages. The device profiles specify how to access particular functions of a device. The Profile for Drives and Motion Control specifies objects used to control device homing and position control. In addition to the objects specified in the Application Layer and Communication Profile and device profiles, CANopen allows manufacturers to add device-specific objects to a dictionary.

Object Dictionary Structure

An object dictionary is a lookup table. Each object is identified by a 16-bit index with an eight-bit sub-index. Most objects represent simple data types, such as 16-bit integers, 32-bit integers, and strings. These can be accessed directly by the 16-bit index.

Other objects use the sub-index to represent groups of related parameters. For instance, the Motor Data object (index 0x6410, paragraph 6.5, page 47) has 24 sub-index objects defining basic motor characteristics such as motor type, motor wiring configuration, and hall sensor type. (The subindex provides up to 255 sub-entries for each index.)

The organization of the dictionary is specified in the profiles, as shown herein.

Index Range	Objects
0000	not used
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Complex Data Types
0060-007F	Device Profile Specific Static Data Types (including those specific to motion control)
0080-009F	Device Profile Specific Complex Data Types (including those specific to motion control)
00A0-0FFF	Reserved for further use
1000-1FFF	Communication Profile Area (DS 301)
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardized Device Profile Area (including Profile for Motion Control)
A000-FFFF	Reserved for future use

3.4 Accessing the Object Dictionary

CANopen provides two methods to access a device’s object dictionary:

- The Service Data Object (SDO)
- The Process Data Object (PDO)

Each can be described as a channel for access to an object dictionary.

3.4.1 SDOs and PDOs

The basic characteristics of PDOs and SDOs are listed in Table 2. For help deciding whether to use an SDO or a PDO refer to paragraph 3.4.4 “SDO or PDO? Design Considerations”.

Table 2: Basic PDO & SDO Characteristics

SDO	PDO
The SDO protocol allows any object in the object dictionary to be accessed, regardless of the object’s size. This comes at the cost of significant protocol overhead.	One PDO message can transfer up to eight bytes of data in a CAN message. There is no additional protocol overhead for PDO messages.
Transfer is always confirmed.	PDO transfers are unconfirmed.
Has direct, unlimited access to the object dictionary.	Requires prior setup, wherein the CANopen master application uses SDOs to map each byte of the PDO message to one or more objects. Thus, the message itself does not need to identify the objects, leaving more bytes available for data.
Employs a client/server communication model, where the CANopen master is the sole client of the device object dictionary being accessed.	Employs a peer-to-peer communication model. Any network can initiate a PDO communication, and multiple nodes can receive it.
An SDO has two CAN message identifiers: a transmit identifier for messages from the device to the CANopen master, and a receive identifier for messages from the CANopen master.	Transmit PDOs are used to send data from the device, and receive PDOs are used to receive data.
SDOs can be used to access the object dictionary directly.	A PDO can be used only after it has been configured using SDO transfers.
Best suited for device configuration, PDO mapping, and other infrequent, low priority communication between the CANopen master and individual devices. Such transfers tend to involve the setting up of basic node services; thus, the term service data object.	Best suited for high-priority transfer of small amounts of data, such as delivery of set points from the CANopen master or broadcast of a device’s status. Such transfers tend to relate directly to the application process; thus, the term process data object.
For more information about SDOs, refer to “SDOs: Description and Examples”, paragraph 3.4.2, page 17	For more information about PDOs, refer to “PDOs: Description and Examples”, paragraph 3.4.3, page 18

The Communication Profile requires the support of at least one SDO per device. (Without an SDO, there would be no way to access the object dictionary.) It also specifies default parameters for four PDOs. AC Tech CANopen amplifiers each support 1 SDO and 16 PDOs (eight transmit PDOs and eight receive PDOs).

3.4.2 SDOs: Description and Examples

Each PositionServo amplifier provides one SDO. The CANopen master can use this SDO to configure, monitor, and control the device by reading from and writing to its object dictionary.

SDO CAN Message IDs

The SDO protocol uses two CAN message identifiers. One ID is for messages sent from the CANopen master (SDO client) to the amplifier (SDO server). The other ID is for messages sent from the SDO server to the SDO client.

The CAN message ID numbers for these two messages are fixed by the CANopen protocol. They are based on the device's node ID (which ranges from 1 to 127). The ID used for messages from the SDO client to the SDO server (i.e. from the CANopen master to the amplifier) is the hex value 0x600 + the node ID. The message from the SDO server to the SDO client is 0x580 + the node ID. For example, an amplifier with node ID 7 uses CAN message IDs 0x587 and 0x607 for its SDO protocol.

Client/ Server Communication

The SDO employs a client/server communication model. The CANopen master is the sole client. The device is the server. The CANopen master application should provide a client SDO for each device under its control. The CAN message ID of an SDO message sent from the CANopen master to a device should match the device's receive SDO message identifier. In response, the CANopen master should expect an SDO message whose CAN message ID matches the device's transmit SDO message identifier.

SDO Message Format

The SDO uses a series of CAN messages to send the segments that make up a block of data. The full details of the SDO protocol are described in the CANopen Application Layer and Communication Profile.

Confirmation

Because an SDO transfer is always confirmed, each SDO transfer requires at least two CAN messages (one from the master and one from the slave).

Confirmation Example

To update an object that holds an eight-byte long value requires six CAN messages:

1. The master sends a message to the device indicating its intentions to update the object.
2. The message includes the object's index and sub-index values as well as the size (in bytes) of the data to be transferred.
3. The device responds to the CANopen master indicating that it is ready to receive the data.
4. The CANopen master sends one byte of message header information and the first 7 bytes of data. (Because SDO transfers use one byte of the CAN message data for header information, the largest amount of data that can be passed in any single message is 7 bytes.)
5. The device responds indicating that it received the data and is ready for more.
6. The CANopen master sends the remaining byte of data.
7. The device responds indicating success.

Segmented, Expedited and Block Transfers

As in the example above, most SDO transfers consist of an initial transfer request from the client, followed by series of confirmed eight-byte messages. Each message contains one byte of header information and a segment (up to seven bytes long) of the data being transferred. For the transfer of short blocks of data (four bytes or less), the Communication Profile specifies an expedited SDO method. The entire data block is included in the initial SDO message (for uploads) or in the response (for downloads). Thus, the entire transfer is completed in two messages.

The Communication Profile also describes a method called block SDO transfers, where many segments can be transferred with a single acknowledgment at the end of the transfer. AC Tech CANopen amplifiers do not require the use of the block transfer protocol.

Using an SDO

To use an SDO, the CANopen master needs an SDO client to communicate with the SDO on each device.

3.4.3 PDOs: Description and Examples

PositionServo amplifiers provide eight transmit PDOs and eight receive PDOs. A transmit PDO is used to transmit information from the device to the network. A receive PDO is used to update the device.

Default PDO Message Identifiers

The Communication Profile reserves four CAN message identifiers for transmit PDOs and four identifiers for receive PDOs. Refer to the sections on Receive PDO Communication Parameters and Transmit PDO Communication Parameters.

The first four transmit PDOs and receive PDOs provided in AC Tech CANopen amplifiers use these default addresses. The addresses of the remaining four transmit PDOs and receive PDOs are null by default. Any PDO message identifier can be reconfigured by the programmer.

PDO Peer-to-Peer Communication

Peer-to-peer relationships match the transmit PDO identifier of the sending node to a receive PDO identifier of one or more other nodes on the network. Any device can broadcast a PDO message using one of its eight transmit PDOs. The CAN identifier of the outgoing message matches the ID of the sending PDO. Any node with a matching receive PDO identifier will accept the message.

PDO Peer-to-Peer Example:

Node 1 = transmit PDO 1, has a CAN message ID of 0x0189. Node 2 = receive PDO 1 has a matching ID, as does Node 3. They both accept the message. Other nodes do not have a matching receive PDO, so they do not accept the message.

PDO Mapping

For optimal of the CAN message's eight-byte data area use PDO mapping. For each byte in the PDO message, mapping uses the SDO to configure dictionary objects in both the sending and the receiving node to know:

- The index and sub-index which objects are to be accessed
- The type of data
- The length of the data

The PDO message itself carries no transfer control information, leaving all eight bytes available for data. (Contrast this with the SDO, which uses one byte of the CAN message data area to describe the objects being written or read, and the length of the data.)

Default PDO Mappings

The Profile for Drives and Motion Control specifies default mappings for the first eight transmit PDOs and the first eight receive PDOs. AC Tech CANopen amplifiers are shipped with these default PDO mappings. These default PDO mappings can be re-mapped by the programmer.

Mappable Objects

Not all objects in a device's object dictionary can be mapped to a PDO. This manual notes this ability (or lack thereof) in the description of each object.

Dynamic PDO Mapping

AC Tech supports dynamic PDO mapping, which allows the CANopen master to change the mapping of a PDO during operation. For instance, a PDO might use one mapping in Homing Mode, and another mapping in Profile Position Mode.

PDO Transmission Modes

PDOs can be sent in one of two transmission modes:

- Synchronous: Messages are sent only after receipt of a specified number of synchronization (SYNC) objects, sent at regular intervals by a designated synchronization device. (For more information on the SYNC object, see SYNC and high-resolution Time Stamp messages)
- Asynchronous: The receipt of SYNC messages does not govern message transmission.

Synchronous transmission can be cyclic, where the message is sent after a predefined number of SYNC messages, or acyclic, where the message is triggered by some internal event but does not get sent until the receipt of a SYNC message.

PDO Triggering Modes

The transmission of a transmit PDO message from a node can be triggered in one of three ways:

Trigger	Description
Event	Message transmission is triggered by the occurrence of an object specific event. For synchronous PDOs this is the expiration of the specified transmission period, synchronized by the reception of the SYNC object. For acyclicly transmitted synchronous PDOs and asynchronous PDOs the triggering of a message transmission is a device specific event specified in the device profile.
SYNC message	For synchronous PDOs, the message is transmitted after a specified number of SYNC cycles have occurred.
Remote Request	The transmission of an asynchronous PDO is initiated upon receipt of a remote request initiated by any other device.

PDO Examples

The programmer has broad discretion in the use of PDOs. Consider some of the default receive PDO mappings specified in the Profile for Drives and Motion Control:

PDO	Default Objects Mapped	Purpose
Receive PDO 1	Control Word (0x6040)	Controls the state of the device
Receive PDO 2	Control Word (0x6040) Mode of Operation (0x6060)	Controls the state and operating mode of the device
Receive PDO 3	Control Word (0x6040) Target Position (0x607a)	Controls the state and target position of the amplifier in profile position mode

Here are some other examples:

- On the device designated as the SYNC message and time stamp producer, map a transmit PDO to transmit the high-resolution time stamp message on a periodic basis. Map receive PDOs on other devices to receive this object.
- Another transmit PDO could transmit general amplifier status updates.

3.4.4 SDO or PDO? Design Considerations

Differences Between SDO and PDO

As stated earlier, SDOs and PDOs can both be described as channels through which CAN messages are sent, and both provide access to a device’s object dictionary. However, each has characteristics that make it more appropriate for certain types of data transfers. Table 3 provides a review of the differences between SDOs and PDOs, and some design considerations indicated by those differences.

Table 3: SDO & PDO Design Considerations

SDO	PDO	Design Considerations
The accessed device always confirms SDO messages. This makes SDOs slower.	PDO messages are unconfirmed. This makes PDOs faster.	To transfer 8 bytes or less at real-time speed, use a PDO. For instance, to receive control instructions and transmit status updates. To transfer large amounts of low priority data, use the SDO. Also, if confirmation is absolutely required, use an SDO.
One SDO transfer can send long blocks of data, using as many CAN messages as required.	A PDO transfer can only send small amounts of data (up to eight bytes) in a single CAN message. Mapping allows very efficient use of those eight bytes.	
Asynchronous.	Synchronous or asynchronous. Cyclic or acyclic.	Use PDO when synchronous or broadcast communications are required. For instance, to communicate set points from the master to multiple devices for a multi-axis move, or to have a device broadcast its status.
The SDO employs a client-server communication model. The CANopen master is the client. It reads from and writes to the object dictionaries of devices. The device being accessed is the server.	The PDO employs a peer-to-peer communication model. Any device can send a PDO message, and a PDO message can be received and processed by multiple devices..	
All communications can be performed through the SDO without using any PDOs.	The CANopen master application uses SDO messages to map the content of the PDO, at a cost of increased CPUcycles on the CANopen master and increased bus traffic.	If the application does not benefit from the use of a PDO for a certain transfer, consider using SDO to avoid the extra overhead. For instance, if an object’s value is updated only once (as with many configuration objects), the SDO is more efficient. If the object’s value is updated repeatedly, a PDO is more efficient.

3.4.5 Mapping a PDO

Reasons for mapping or remapping a PDO include:

- changes to the amplifier’s Manufacturer Status Register object (index 0x1002, paragraph 6.2, page 36)
- changes in the CANopen status word
- amplifier I/O change
- amplifier PVT status changes

Two objects in the device’s object dictionary define a PDO:

- A PDO’s communication object defines the PDO’s CAN message ID and its communication type (synchronous or asynchronous) and triggering type (event-drive or cyclic).
- A PDOs mapping object maps every data byte in the PDO message to an object in the device’s object dictionary.

Mapping a PDO is the process of configuring the PDO’s communication and mapping objects.

To Map a Receive PDO

The general procedure for mapping a receive PDO is listed in Table 4. The procedure for mapping a transmit PDO is similar.

Table 4: Mapping a Receive PDO

Stage	Step	Sub-Steps / Comments
1	Disable the PDO	In the PDO's mapping object (Receive PDO Mapping Parameters, index 0x1601), set the sub-index 0 (NUMBER OF MAPPED OBJECTS) to zero. This disables the PDO
2	Set the communication parameters	If necessary, set the PDO's CAN message ID (PDO COB-ID) using sub-index 1 of the PDO's RECEIVE PDO Communication Parameters (index 0x1401). Choose the PDO's transmission type (PDO TYPE) in sub-index 2 of object 0x1401. A value in the range [0-240] = synchronous; [254-255] = asynchronous.
3	Map the data	Using the PDO's mapping parameters (sub-indexes 1-4 of Receive PDO Mapping Parameters, index 0x1601), you can map up to 4 objects (whose contents must total to no more than 8 bytes), as follows: In bits 0-7 of the mapping value, enter the size (in bits) of the object to be mapped, as specified in the object dictionary. In bits 8-15, enter the sub-index of the object to be mapped. Clear bits 8-15 if the object is a simple variable. In bits 16-31, enter the index of the object to be mapped.
4	Set the number of mapped objects and enable the PDO	In the PDO's Receive PDO Mapping Parameters (index 0x1601), set sub-index 0 (NUMBER OF MAPPED OBJECTS) to the actual number of objects mapped. This properly configures the PDO. Also, the presence of a non-zero value in the NUMBER OF MAPPED OBJECTS object enables the PDO

Example: Mapping a Receive PDO

This example illustrates the general procedure for mapping a receive PDO. In the example, the second receive PDO is mapped to the device's Control Word object (index 0x6040) to receive device state change commands and to the Mode of Operation object (index 0x6060) to receive mode change commands.

3.5 Objects that Define SDO's and PDO's

To define an SDO or PDO, use the objects included in Table 5.

Table 5: Objects the Define an SDO or PDO

Object	Index	Sub-Index
Receive Object		
Receive PDO Communication Parameters	0x1400 - 0x1407	--
PDO COB_ID Index	0x1400-7	1
PDO Type	0x1400-7	2
Receive PDO Mapping Parameters	0x1600 - 0x1607	--
Number of Mapped Objects	0x1600-7	0
PDO Mapping	0x1600-7	1-8
Transmit Object		
Transmit PDO Communication Parameters	0x1800 - 0x1807	--
PDO COB-ID	0x1800-7	1
PDO Type	0x1800-7	2
Transmit PDO Mapping Parameters	0x1A00 - 0x1A07	--
Number of Mapped Objects	0x1A00-7	0
PDO Mapping	0x1A00-7	1-8

Object				Index	Sub-Index
RECEIVE PDO COMMUNICATION PARAMETERS				0x1200	1
Type	Access	Units	Range	Map PDO	Memory
Record	RW	--	--	NO	--
Description These objects allow configuration of the communication parameters of each receive PDO. Subindex 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index																														
PDO COB-ID				0x1200	1																														
Type	Access	Units	Range	Map PDO	Memory																														
Unsigned 32	RW	--	Refer to Description	NO	R																														
Description CAN message ID used by the PDO. The ID is formatted as follows: <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-10</td> <td>Give the 11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.</td> </tr> <tr> <td>11-28</td> <td>Give the upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.</td> </tr> <tr> <td>29</td> <td>Defines the identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.</td> </tr> <tr> <td>30</td> <td>Reserved for future use.</td> </tr> <tr> <td>31</td> <td>Identifies the PDO as valid if clear. If set, the PDO is disabled and its mapping may be changed.</td> </tr> </tbody> </table> Default Values The default values for this object are specified in the DS-301 CANopen specification. These values are: <table border="1"> <thead> <tr> <th>Index</th> <th>Default ID</th> </tr> </thead> <tbody> <tr> <td>0x1400</td> <td>0x00000200 + amplifier CAN node ID.</td> </tr> <tr> <td>0x1401</td> <td>0x00000300 + amplifier CAN node ID.</td> </tr> <tr> <td>0x1402</td> <td>0x00000400 + amplifier CAN node ID.</td> </tr> <tr> <td>0x1403</td> <td>0x00000500 + amplifier CAN node ID.</td> </tr> <tr> <td>0x1404</td> <td>0x80000000</td> </tr> <tr> <td>0x1405</td> <td>0x80000000</td> </tr> <tr> <td>0x1406</td> <td>0x80000000</td> </tr> <tr> <td>0x1407</td> <td>0x80000000</td> </tr> </tbody> </table>						Bit	Description	0-10	Give the 11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.	11-28	Give the upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.	29	Defines the identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.	30	Reserved for future use.	31	Identifies the PDO as valid if clear. If set, the PDO is disabled and its mapping may be changed.	Index	Default ID	0x1400	0x00000200 + amplifier CAN node ID.	0x1401	0x00000300 + amplifier CAN node ID.	0x1402	0x00000400 + amplifier CAN node ID.	0x1403	0x00000500 + amplifier CAN node ID.	0x1404	0x80000000	0x1405	0x80000000	0x1406	0x80000000	0x1407	0x80000000
Bit	Description																																		
0-10	Give the 11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.																																		
11-28	Give the upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.																																		
29	Defines the identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.																																		
30	Reserved for future use.																																		
31	Identifies the PDO as valid if clear. If set, the PDO is disabled and its mapping may be changed.																																		
Index	Default ID																																		
0x1400	0x00000200 + amplifier CAN node ID.																																		
0x1401	0x00000300 + amplifier CAN node ID.																																		
0x1402	0x00000400 + amplifier CAN node ID.																																		
0x1403	0x00000500 + amplifier CAN node ID.																																		
0x1404	0x80000000																																		
0x1405	0x80000000																																		
0x1406	0x80000000																																		
0x1407	0x80000000																																		

Object				Index	Sub-Index								
PDO Type				0x1400 – 7	2								
Type	Access	Units	Range	Map PDO	Memory								
Unsigned 8	RW	--	Refer to Description	NO	R								
Description This object controls the behavior of the PDO when new data is received. The following codes are defined for receive PDOs: <table border="1"> <thead> <tr> <th>Code</th> <th>Behavior</th> </tr> </thead> <tbody> <tr> <td>0-240</td> <td>The received data is held until the next SYNC message. When the SYNC message is received the data is applied.</td> </tr> <tr> <td>241-253</td> <td>Reserved.</td> </tr> <tr> <td>254-255</td> <td>The received data is applied to its mapped objects immediately upon reception.</td> </tr> </tbody> </table>						Code	Behavior	0-240	The received data is held until the next SYNC message. When the SYNC message is received the data is applied.	241-253	Reserved.	254-255	The received data is applied to its mapped objects immediately upon reception.
Code	Behavior												
0-240	The received data is held until the next SYNC message. When the SYNC message is received the data is applied.												
241-253	Reserved.												
254-255	The received data is applied to its mapped objects immediately upon reception.												

Object				Index	Sub-Index
Receive PDO Mapping Parameters				0x1600 – 0x1607	--
Type	Access	Units	Range	Map PDO	Memory
Record	RW	--	--	NO	--
Description These objects allow the mapping of each of the receive PDO objects to be configured.					

Object				Index	Sub-Index
Number of Mapped Objects				0x1600 – 7	0
Type	Access	Units	Range	Map PDO	Memory
Unsigned 8	RW	--	0 - 8	NO	R
Description This value gives the total number of objects mapped to this PDO. It can be set to 0 to disable the PDO operation and must be set to 0 before changing the PDO mapping. Once the PDO mapping has been established by configuring the objects in sub-indexes 1 – 8, this value should be updated to indicate the actual number of objects mapped to the PDO.					

Object				Index	Sub-Index								
PDO Mapping				0x1600 – 7	1 - 8								
Type	Access	Units	Range	Map PDO	Memory								
Unsigned 32	RW	--	Refer to Description	NO	R								
Description When a PDO message is received, the data passed with the PDO message (up to 8 bytes) is used to update the objects mapped to the PDO. The values in the PDO mapping objects identify which object(s) the PDO data maps to. The first object is specified by the value in sub-index 1; the second object is identified by sub-index 2, etc. Each of the PDO mapping values consist of a 32-bit value structured as follows: <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-7</td> <td>Size (in bits) of the object being mapped. Must match the actual object size as defined in the object dictionary.</td> </tr> <tr> <td>8-15</td> <td>Sub-index of the object to be mapped.</td> </tr> <tr> <td>16-31</td> <td>Index of the object to be mapped.</td> </tr> </tbody> </table>						Bit	Description	0-7	Size (in bits) of the object being mapped. Must match the actual object size as defined in the object dictionary.	8-15	Sub-index of the object to be mapped.	16-31	Index of the object to be mapped.
Bit	Description												
0-7	Size (in bits) of the object being mapped. Must match the actual object size as defined in the object dictionary.												
8-15	Sub-index of the object to be mapped.												
16-31	Index of the object to be mapped.												

Object				Index	Sub-Index
Transmit PDO Communication Parameters				0x1800 – 0x1807	--
Type	Access	Units	Range	Map PDO	Memory
Record	RW	--	--	NO	--
Description These objects allow configuration of communication parameters of each transmit PDO object. Sub-index 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index																														
PDO COB-ID				0x1800 – 7	1																														
Type	Access	Units	Range	Map PDO	Memory																														
Unsigned 32	RW	--	Refer to Description	NO	R																														
Description This object holds the CAN object ID used by the PDO. The ID is formatted as follows: <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-10</td> <td>11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.</td> </tr> <tr> <td>11-28</td> <td>Upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.</td> </tr> <tr> <td>29</td> <td>Identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.</td> </tr> <tr> <td>30</td> <td>If set, remote transmit requests (RTR) are not allowed on this PDO. If clear, the PDO is transmitted in response to a remote request.</td> </tr> <tr> <td>31</td> <td>Identifies the PDO as valid if clear. If set, the PDO is disabled and its mapping may be changed.</td> </tr> </tbody> </table> Default Values The default values for this object are specified in the DS-301 CANopen specification. These values are: <table border="1"> <thead> <tr> <th>Index</th> <th>Default ID</th> </tr> </thead> <tbody> <tr> <td>0x1800</td> <td>0x00000180 + amplifier CAN node ID.</td> </tr> <tr> <td>0x1801</td> <td>0x00000280 + amplifier CAN node ID.</td> </tr> <tr> <td>0x1802</td> <td>0x00000380 + amplifier CAN node ID.</td> </tr> <tr> <td>0x1803</td> <td>0x00000480 + amplifier CAN node ID.</td> </tr> <tr> <td>0x1804</td> <td>0x80000000</td> </tr> <tr> <td>0x1805</td> <td>0x80000000</td> </tr> <tr> <td>0x1806</td> <td>0x80000000</td> </tr> <tr> <td>0x1807</td> <td>0x80000000</td> </tr> </tbody> </table>						Bit	Description	0-10	11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.	11-28	Upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.	29	Identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.	30	If set, remote transmit requests (RTR) are not allowed on this PDO. If clear, the PDO is transmitted in response to a remote request.	31	Identifies the PDO as valid if clear. If set, the PDO is disabled and its mapping may be changed.	Index	Default ID	0x1800	0x00000180 + amplifier CAN node ID.	0x1801	0x00000280 + amplifier CAN node ID.	0x1802	0x00000380 + amplifier CAN node ID.	0x1803	0x00000480 + amplifier CAN node ID.	0x1804	0x80000000	0x1805	0x80000000	0x1806	0x80000000	0x1807	0x80000000
Bit	Description																																		
0-10	11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.																																		
11-28	Upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.																																		
29	Identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.																																		
30	If set, remote transmit requests (RTR) are not allowed on this PDO. If clear, the PDO is transmitted in response to a remote request.																																		
31	Identifies the PDO as valid if clear. If set, the PDO is disabled and its mapping may be changed.																																		
Index	Default ID																																		
0x1800	0x00000180 + amplifier CAN node ID.																																		
0x1801	0x00000280 + amplifier CAN node ID.																																		
0x1802	0x00000380 + amplifier CAN node ID.																																		
0x1803	0x00000480 + amplifier CAN node ID.																																		
0x1804	0x80000000																																		
0x1805	0x80000000																																		
0x1806	0x80000000																																		
0x1807	0x80000000																																		

Object				Index	Sub-Index														
PDO Type				0x1800 – 7	2														
Type	Access	Units	Range	Map PDO	Memory														
Unsigned 8	RW	--	Refer to Description	EVENT	R														
Description This object identifies which events trigger a PDO transmission: <table border="1"> <thead> <tr> <th>Code</th> <th>Behavior</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PDO is transmitted on the next SYNC message following a PDO event. See PDO Events, below, for a description of a PDO event.</td> </tr> <tr> <td>1-240</td> <td>The PDO is transmitted every N SYNC messages, where N is the PDO type code. For example, a PDO with type code 7 would be transmitted on every 7th SYNC message.</td> </tr> <tr> <td>241-251</td> <td>Reserved.</td> </tr> <tr> <td>252</td> <td>The PDO is transmitted on the SYNC message following a remote request.</td> </tr> <tr> <td>253</td> <td>The PDO is transmitted immediately in response to a remote request.</td> </tr> <tr> <td>254-255</td> <td>The PDO is transmitted immediately in response to an internal PDO event.</td> </tr> </tbody> </table>						Code	Behavior	0	The PDO is transmitted on the next SYNC message following a PDO event. See PDO Events, below, for a description of a PDO event.	1-240	The PDO is transmitted every N SYNC messages, where N is the PDO type code. For example, a PDO with type code 7 would be transmitted on every 7th SYNC message.	241-251	Reserved.	252	The PDO is transmitted on the SYNC message following a remote request.	253	The PDO is transmitted immediately in response to a remote request.	254-255	The PDO is transmitted immediately in response to an internal PDO event.
Code	Behavior																		
0	The PDO is transmitted on the next SYNC message following a PDO event. See PDO Events, below, for a description of a PDO event.																		
1-240	The PDO is transmitted every N SYNC messages, where N is the PDO type code. For example, a PDO with type code 7 would be transmitted on every 7th SYNC message.																		
241-251	Reserved.																		
252	The PDO is transmitted on the SYNC message following a remote request.																		
253	The PDO is transmitted immediately in response to a remote request.																		
254-255	The PDO is transmitted immediately in response to an internal PDO event.																		

PDO Events

Some objects in the object dictionary have special PDO events associated with them. If such an object is mapped to a transmit PDO, then the PDO may be configured with a code that relies on this event to trigger its transmission. The codes that use PDO events are 0, 254, and 255.

An example of an object that has a PDO event associated with it is the Device Status object (index 0x6041). This object triggers an event to any mapped transmit PDO each time its value changes.

A transmit PDO which included this object in its mapping would have its event signaled each time the status register changed. Most objects in the object dictionary do not have PDO events associated with them. Those that do are identified by the word EVENT in the PDO Mapping fields of their descriptions.

Object				Index	Sub-Index
Transmit PDO Mapping Parameters				0x1A00 – 0x1A07	--
Type	Access	Units	Range	Map PDO	Memory
Record	RW	--	--	NO	--
Description These objects allow the mapping of each of the transmit PDO objects to be configured.					

Object				Index	Sub-Index
Number of Mapped Objects				0x1A00 – 7	0
Type	Access	Units	Range	Map PDO	Memory
Unsigned 8	RW	--	0 - 8	NO	R
Description Total number of objects mapped to this PDO. It can be set to 0 to disable the PDO operation, and must be set to 0 before changing the PDO mapping. Once the PDO mapping has been established by configuring the objects in sub-indices 1 – 4, this value should be updated to indicate the actual number of objects mapped to the PDO.					

Object				Index	Sub-Index								
PDO Mapping				0x1A00 – 7	1 - 8								
Type	Access	Units	Range	Map PDO	Memory								
Unsigned 32	RW	--	Refer to Description	NO	R								
Description When a PDO message is transmitted, the data passed with the PDO message (up to 8 bytes) is gathered from the objects mapped to the PDO. The values in the PDO Mapping objects identify which object(s) the PDO data maps to. The first object is specified by the value in sub-index 1; the second object is identified by sub-index 2, etc. Each of the PDO mapping values consist of a 32-bit value structured as follows: <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-7</td> <td>Size (in bits) of the object being mapped. This value must match the actual object size as defined in the object dictionary.</td> </tr> <tr> <td>8-15</td> <td>Sub-index of the object to be mapped.</td> </tr> <tr> <td>16-31</td> <td>Index of the object to be mapped.</td> </tr> </tbody> </table>						Bit	Description	0-7	Size (in bits) of the object being mapped. This value must match the actual object size as defined in the object dictionary.	8-15	Sub-index of the object to be mapped.	16-31	Index of the object to be mapped.
Bit	Description												
0-7	Size (in bits) of the object being mapped. This value must match the actual object size as defined in the object dictionary.												
8-15	Sub-index of the object to be mapped.												
16-31	Index of the object to be mapped.												

4 Network Management

This chapter describes the messages, methods, and objects used to manage devices on a CANopen network.

4.1 Network Management Overview

4.2 Network Management Objects

4.1 Network Management Overview

This section describes the objects, messages, and methods used to control the CANopen network.

4.1.1 Network Management Services and Objects

Network management services on the CANopen network include device state control, device monitoring, synchronization, and emergency handling. Special communication objects, as summarized herein, provide these services.

Object	Description
Network Management (NMT)	This object provides services to control the state of the device, including the initialization, starting, monitoring, resetting, and stopping of nodes. It also provides device-monitoring services (node-guarding and heartbeat).
Synchronization (SYNC)	Broadcast periodically by a specified device or the CANopen master to allow synchronized activity among multiple devices. The CAN message ID of the SYNC message is 80.
Time Stamp	Broadcast periodically by a specified device or the CANopen master to allow devices to synchronize their clocks.
Emergency	Transmitted by a device when an internal error occurs.

Network Manager Node

In general applications, a single node (such as a PC) is designated as the network manager. The network manager runs the software that issues all NMT messages. The network manager node can be the same node that runs the CANopen master application.

4.1.2 General Device State Control

State Machine

Every CANopen device implements a simple state machine. The machine defines three states (pre-operational, operational and stopped). The network manager application uses NMT messages to interact with the state machine and control state changes.

Device States

The following states are defined for AC Technology CANopen amplifiers:

State	Description
Pre-operational	Every node enters this state after power-up or reset. In this state, the device is not functional, but will communicate over the CANopen network. PDO transfers are not allowed in pre-operational state, but SDO transfers may be used.
Operational	This is the normal operating state for all devices. SDO and PDO transfers are both allowed.
Stopped	No communication is allowed in this state except for network management messages. Neither SDO nor PDO transfers may be used.

State Control Messages

NMT messages can be used to control state changes on network devices. The following NMT messages are sent by the network manager to control these state changes. Each of these messages can be either sent to a single node (by node ID), or broadcast to all nodes:

NMT Message	Cause/Effect
Reset	Causes each receiving node to perform a soft reset and come up in pre-operational state.
Reset communications	Causes each receiving node to reset its CANopen network interface to power-on state, and enter pre-operational state. This is not a full device reset, just a reset of the CANopen interface.
Pre-operational	Causes the receiving node(s) to enter pre-operational state. No reset is performed.
Start	Causes the node(s) to enter operational state.
Stop	Causes the node(s) to enter stopped state.

4.1.3 Device Monitoring Monitoring Protocols

In addition to controlling state machines, NMT messages provide services for monitoring devices on the network. Monitoring services use one of two protocols: heartbeat and node guarding.

Heartbeat Protocol

The heartbeat protocol allows the network manager application to detect problems with a device or its network connection. The CANopen master configures the device to periodically transmit a heartbeat message indicating the device's current state (pre-operational, operational, or stopped). The network manager monitors the heartbeat messages. Failure to receive a node's heartbeat messages indicates a problem with the device or its connection to the network.

Node-guarding Protocol

The node-guarding protocol is similar to the heartbeat, but it allows both the device and the network manager to monitor the connection between them. The network manager configures the device (node) to expect node-guarding messages at some interval. The network manager then sends a message to the configured device at that frequency, and the device responds with a node-guarding message. This allows both the network manager and the device to identify a network failure if the guarding messages stop.

SYNC and High-resolution Time Stamp Messages

The SYNC message is a standard CANopen message used to synchronize multiple devices and to trigger the synchronous transmission of PDOs. In addition, to allow more accurate synchronization of device clocks, CANopen amplifiers use the optional high-resolution time stamp message specified in the Communication Profile. Normally, a single device produces both the SYNC message and the high-resolution time stamp message.

4.1.4 Time Stamp PDOs

The device designated as the time stamp producer should have a transmit PDO mapped for the high-resolution time stamp message. This PDO should be configured for synchronous transmission, based on the SYNC message. It is recommended to send this message approximately every 100 milliseconds.

Every other device (all time stamp consumers) should have a receive PDO mapped for the high resolution time stamp message. The message ID of each receive PDO used to receive a time stamp should match the ID of the transmit PDO used to send the time stamp.

Configuring the devices in this fashion causes the time stamp producer to generate a transmit PDO for every N sync messages. This PDO is received by each of the time stamp consumers on the network and causes them to update their internal system times based on the message content. The result is that all devices on the network act as though they share the same clock input, and remain tightly synchronized.

4.1.5 Emergency Messages

A device sends an 8-byte emergency message (EMCY) when an error occurs in the device. It contains information about the error type, and AC Tech-specific information. A device need only send one EMCY message per event. Any device can be configured to accept EMCY messages.

EMCY Message Structure

The EMCY message is structured as follows:

Bytes Description

- 0, 1 Emergency error code. See EMCY Message CANopen Error Codes
- 2 Error register object value See Error Register object index 0x1001
- 3 Reserved for future use (0 for now).
- 4,5 Bit mask of active error conditions on the amplifier EMCY: AC Tech-Specific Error Conditions.
- 6,7 Reserved for future use (0 for now).

EMCY Message CANopen Error Codes

Bytes 0 and 1 of the EMCY message describe the standard CANopen error codes. AC Tech amplifiers support Error codes 00xx and 10xx.

Error Code (hex)	Description	Error Code (hex)	Description
00xx	Error Reset or No Error	62xx	User Software
10xx	Generic Error	63xx	Data Set
20xx	Current	70xx	Additional Modules
21xx	Current, device input side	80xx	Monitoring
22xx	Current inside the device	81xx	Communication
23xx	Current, device output side	8110	CAN Overrun (Objects lost)
30xx	Voltage	8120	CAN in Error Passive Mode
31xx	Mains Voltage	8130	Life Guard Error or Heartbeat Error
32xx	Voltage inside the device	8140	recovered from bus off
33xx	Output Voltage	8150	Transmit COB-ID
40xx	Temperature	82xx	Protocol Error
41xx	Ambient Temperature	8210	PDO not processed due to length error
42xx	Device Temperature	8220	PDO length exceeded
50xx	Device Hardware	90xx	External Error
60xx	Device Software	F0xx	Additional Functions
61xx	Internal Software	FFxx	Device specific. XX represents fault number specified in Programmer's manual

4.2 Network Management Objects

This section describes objects closely related to network management. They include:

COB-ID SYNC MESSAGE INDEX 0X1005

PRODUCER HEARTBEAT TIME INDEX 0X1017

EMERGENCY OBJECT ID INDEX 0X1014

EMERGENCY OBJECT ID INHIBIT TIME INDEX 0X1015

Object				Index	Sub-Index												
COB-ID SYNC MESSAGE				0X1005	--												
Type	Access	Units	Range	Map PDO	Memory												
Unsigned 32	RW	--	See SYNC ID Format	NO	R												
<p>Description</p> <p>This object establishes an amplifier as the SYNC producer and defines the CAN object ID (COBID) associated with the SYNC message. The SYNC message is a standard CANopen message type used to synchronize multiple devices on a CANopen network.</p> <p>SYNC ID Format: The SYNC message ID is formatted as follows:</p> <table border="1"> <thead> <tr> <th>Bits</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0-10</td> <td>Give the 11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.</td> </tr> <tr> <td>11-28</td> <td>Give the upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.</td> </tr> <tr> <td>29</td> <td>Identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.</td> </tr> <tr> <td>30</td> <td>If set, the amplifier is configured as the SYNC message producer. This bit should be set in at most one amplifier on a network.</td> </tr> <tr> <td>31</td> <td>Reserved</td> </tr> </tbody> </table>						Bits	Description	0-10	Give the 11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.	11-28	Give the upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.	29	Identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.	30	If set, the amplifier is configured as the SYNC message producer. This bit should be set in at most one amplifier on a network.	31	Reserved
Bits	Description																
0-10	Give the 11-bit identifier for standard (CAN 2.0A) identifiers, or the lower 11 bits for extended (CAN 2.0B) identifiers.																
11-28	Give the upper 18 bits of extended identifiers. For standard identifiers these bits should be written as zeros.																
29	Identifier format. This bit is clear for standard (11-bit) identifiers, and set for extended (29-bit) identifiers.																
30	If set, the amplifier is configured as the SYNC message producer. This bit should be set in at most one amplifier on a network.																
31	Reserved																

Object				Index	Sub-Index
PRODUCER HEARTBEAT TIME				0X1017	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 16	RW	milliseconds	--	NO	R
<p>Description</p> <p>This object gives the frequency at which the amplifier will produce heartbeat messages. This object may be set to zero to disable heartbeat production. Note that only one of the two nodeguarding methods may be used at once. If this object is non-zero, then the heartbeat protocol is used regardless of the settings of the node-guarding time and lifetime factor.</p>					

Object				Index	Sub-Index
EMERGENCY OBJECT ID				0X1014	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	RW	--	--	NO	R
<p>Description</p> <p>CAN message ID used with the emergency object. Refer to the CANopen Application Layer and Communication Profile (DS 301).</p>					

Object				Index	Sub-Index
EMERGENCY OBJECT ID INHIBIT TIME				0X1015	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 16	RW	milliseconds	--	NO	R
<p>Description</p> <p>Inhibit time for the emergency object. Refer to the CANopen Application Layer and Communication Profile (DS 301).</p>					

5 Device Configuration and Control through Native Variables List

5.1 Native Control

5.2 Objects to Access Drive Variables

5.1 Native Control

Every aspect of the PositionServo can be manipulated by writing or reading the drive's internal variable(s). All variables are addressed by their respective index number. Variables are listed in the "complete list of variables" in the PositionServo Programmer's Manual.

Every variable can be interpreted as 32 bit Integer or as DOUBLE. Each variable has its native format inside and without regards to how the value was sent, it will be casted to its natural format by the drive. For example: Drive variable #30 is the drive's current limit. Its natural format is float. However this variable can be set as Integer type as well. Of course the fractional portion of the value can't be changed by Integer type but if integer precision is enough it can be used.

All variables are located in RAM but some of them have non-volatile copy in EPM memory. Implementation of the CANopen interface provides 4 types of objects to access the drive's variables. Objects with indexes 0x2000 – 0x23FF read or write RAM time copies of the variables as 32 bit integers. Objects with indexes 0x2400 – 0x27FF read or write RAM time copies of the variables as Float numbers. Objects with indexes 0x3000-0x33FF write RAM and EPM copies and read EPM copies of the variables as Integer 32 numbers. Objects with indexes 0x3400-0x37FF write RAM and EPM copies and read EPM copies of the variables as Float numbers.

This organization gives the user unified access to variables and simplifies implementation of communication software.

5.2 Objects to Access the Drive's RAM Variables

Object				Index	Sub-Index
RAM VARIABLES				0X2000-0X23FF	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	RW	--	--	NO	R
Description Objects in this range Read or Write corresponding internal drive's RAM copies of the variables as Integer 32 values. Object Index to access particular variable calculated as: $\text{Object Index} = 0x2000 + \text{VarID}$, where: VarID is the variable ordinal index from the variable table. (Variable table is provided in Programmer's Manual). For Units and Range of every particular variable please refer to Complete list of Variables in Programmer's Manual.					

Object				Index	Sub-Index
RAM VARIABLES				0X2400-0X27FF	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	R/W	--	--	NO	R
Description Objects in this range Read or Write corresponding internal drive's RAM copies of the variables as Float values. Object Index to access particular variable calculated as: $\text{Object Index} = 0x2400 + \text{VarID}$, where: VarID is the variable ordinal index from the variable table. (Variable table is provided in Programmer's Manual). For Units and Range of every particular variable please refer to Complete list of Variables in Programmer's Manual.					

Object				Index	Sub-Index
RAM VARIABLES				0X3000-0X33FF	--
Type	Access	Units	Range	Map PDO	Memory
Float	R/W	--	--	NO	RF
Description Objects in this range: <ul style="list-style-type: none"> • Write corresponding internal drive's RAM and EPM copies of the variables as Integer 32 values. • Read corresponding internal drive's EPM copies of the variables as Integer 32 values Object Index to access particular variable calculated as: $\text{Object Index} = 0x2000 + \text{VarID}$, where: VarID is the variable ordinal index from the variable table. (Variable table is provided in Programmer's Manual). For Units and Range of every particular variable please refer to Complete list of Variables in Programmer's Manual.					

Object				Index	Sub-Index
RAM VARIABLES				0X3400-0X37FF	--
Type	Access	Units	Range	Map PDO	Memory
Float	R/W	--	--	NO	RF
Description Objects in this range: <ul style="list-style-type: none"> • Write corresponding internal drive's RAM and EPM copies of the variables as Float values. • Read corresponding internal drive's EPM copies of the variables as Float values Object Index to access particular variable calculated as: $\text{Object Index} = 0x2000 + \text{VarID}$, where: VarID is the variable ordinal index from the variable table. (Variable table is provided in Programmer's Manual). For Units and Range of every particular variable please refer to Complete list of Variables in Programmer's Manual.					

6 Device Control, Configuration and Status

This chapter describes a wide range of device control, configuration, and status methods and objects.

- 6.1 Device Control and Status Overview
- 6.2 Device Control and Status Objects
- 6.3 Error Management Objects
- 6.4 Basic Amplifier Configuration Objects
- 6.5 Basic Motor Configuration Objects

6.1 Device Control and Status Overview

This section describes the objects and functions used to control the status of an amplifier including:

- 6.1.1 Control Word, Status Word, and Device Control Function
- 6.1.2 State Changes Diagram

6.1.1 Control Word, Status Word, and Device Control Function

Device Control Function Block

The Profile for Drives and Motion Control describes control of the amplifier in terms of a control function block with two major sub-elements: the operation modes and the state machine.

Control and Status Words

As illustrated in Figure 8, the Control Word object (index 0x6040, paragraph 6.2, page 34) manages device mode and state changes. The Status Word object (index 0x6041, paragraph 6.2, page 34) identifies the current state of the amplifier. Other factors affecting control functions include: digital input signals, fault conditions, and settings in various dictionary objects.

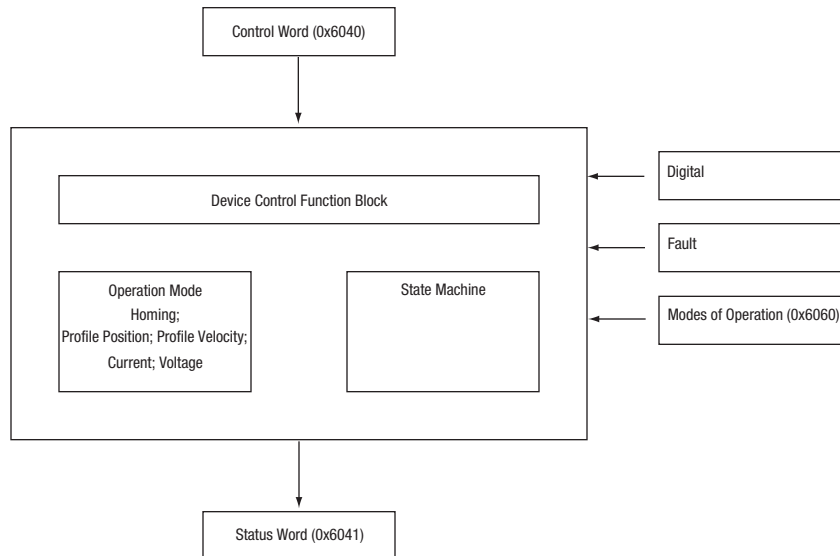


Figure 8: Control and Status Words

Operation Modes

As mentioned elsewhere in this manual, AC Tech CANopen amplifiers support these operation modes:

- Homing
- Profile position
- Profile velocity
- Current follower
- Velocity follower

State Machine Nesting and States

Note that the Communication Profile also specifies a state machine, with three states: preoperational, operational, and stopped. The entire device control function block described in this chapter, including the device state machine, operates in the operational state of the Communication Profile state machine.

The state machine describes the status and possible control sequences of the drive. The state also determines which commands are accepted. States are described herein:

State	Description
Not Ready to Switch On	Low-level power (e.g. \sim 15V, 5V) has been applied to the drive. The drive is being initialized or is running self-test. A brake, if present, is applied in this state. The drive function is disabled.
Switch On Disabled	Drive initialization is complete. The drive parameters have been set up. Drive parameters may be changed. The drive function is disabled.
Ready to Switch On	The drive parameters may be changed. The drive function is disabled.
Switched On	High voltage has been applied to the drive. The power amplifier is ready. The drive parameters may be changed. The drive function is disabled.
Operation Enable	No faults have been detected. The drive function is enabled and power is applied to the motor. The drive parameters may be changed. (This corresponds to normal operation of the drive.)

Quick Stop Active	The drive parameters may be changed. The quick stop function is being executed. The drive function is enabled and power is applied to the motor. If the 'Quick-Stop-Option-Code' is switched to 5 (Stay in Quick-Stop), the amplifier cannot exit the Quick-Stop-State, but can be transmitted to 'Operation Enable' with the command 'Enable Operation.'
Fault Reaction Active	The drive parameters may be changed. A non-fatal fault has occurred in the drive. The quick stop function is being executed. The drive function is enabled and power is applied to the motor.
Fault	The drive parameters may be changed. A fault has occurred in the drive. The drive function is disabled.

6.1.2 State Changes Diagram

The diagram illustrated in Figure 9 is from the Profile for Drives and Motion Control and shows the possible state change sequences of an amplifier. Each transition is numbered and described in the legend herein.

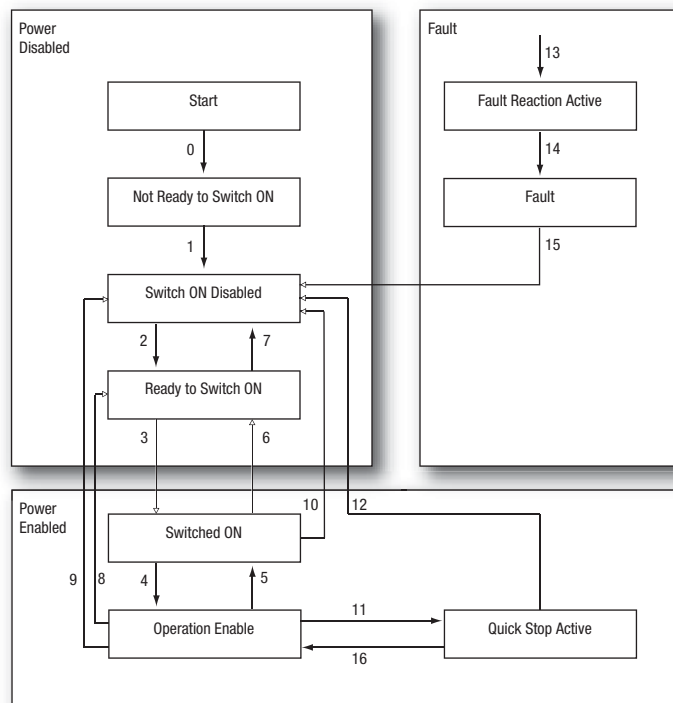


Figure 9: State Changes

State Changes Diagram Legend

#	From State	To State	Event/Action
0	Startup	Not Ready to Switch On	Event: Reset. Action: The drive self-tests and/or self-initializes
1	Not Ready to Switch On	Switch On Disabled	Event: The drive has self-tested and/or initialized successfully. Action: Activate communication and process data monitoring
2	Switch On Disabled	Ready to Switch On	Event: 'Shutdown' command received from host. Action: None
3	Ready to Switch On	Switched On	Event: 'Switch On' command received from host. Action: The power section is switched on if it is not already switched on
4	Switched On	Operation Enable	Event: 'Enable Operation' command received from host. Action: The drive function is enabled.
5	Operation Enable	Switched On	Event: 'Disable Operation' command received from host. Action: The drive operation is disabled.
6	Switched On	Ready to Switch On	Event: 'Shutdown' command received from host. Action: The power section is switched off.
7	Ready to Switch On	Switch On Disabled	Event: 'Quick stop' command received from host. Action: None
8	Operation Enable	Ready to Switch On	Event: 'Shutdown' command received from host. Action: The power section is switched off immediately, and the motor is free to rotate if unbraked
9	Operation Enable	Switch On Disabled	Event: 'Disable Voltage' command received from host. Action: The power section is switched off immediately, and the motor is free to rotate if unbraked
10	Switched On	Switch On Disabled	Event: 'Disable Voltage' or 'Quick Stop' command received from host. Action: The power section is switched off immediately, and the motor is free to rotate if unbraked
11	Operation Enable	Quick Stop Active	Event: 'Quick Stop' command received from host. \
			Action: The Quick Stop function is executed.
12	Quick Stop Active	Switch On Disabled	Event: 'Quick Stop' is completed or 'Disable Voltage' command received from host. This transition is possible if the Quick-Stop-Option-Code is not 5 (Stay in Quick-Stop) Action: The power section is switched off.
13	FAULT	Fault Reaction Active	Event: A fatal fault has occurred in the drive. Action: Execute appropriate fault reaction.
14	Fault Reaction Active	Fault	Event: The fault reaction is completed. Action: The drive function is disabled. The power section may be switched off.
15	Fault	Switch On Disabled	Event: 'Fault Reset' command received from host. Action: A reset of the fault condition is carried out if no fault exists currently on the drive. After leaving the 'Fault' state the Bit 'Fault Reset' of the Control Word has to be cleared by the host.
16	Quick Stop Active	Operation Enable	Event: 'Enable Operation' command received from host. This transition is possible if the Quick-Stop-Option-Code is 5, 6, 7, or 8 (see the Quick Stop Option Code object, index 0x6085, paragraph 6.2, page 35). Action: The drive function is enabled.

6.2 Device Control and Status Objects

This section describes the objects used to control the status of an amplifier including:

CONTROL WORD	INDEX 0X6040
STATUS WORD	INDEX 0X6041
QUICK STOP OPTION CODE	INDEX 0X605A
SHUTDOWN OPTION CODE	INDEX 0X605B
DISABLE OPERATION OPTION CODE	INDEX 0X605C
MODE OF OPERATION	INDEX 0X6060
MODE OF OPERATION DISPLAY	INDEX 0X6061
REFERENCE SOURCE	INDEX 0X2025
MANUFACTURER STATUS REGISTER	INDEX 0X1002

Object				Index	Sub-Index																
CONTROL WORD				0X6040	--																
Type	Access	Units	Range	Map PDO	Memory																
Unsigned 16	RW	--	See Description	YES	R																
<p>Description</p> <p>This object is used to controls the state of the amplifier. It can be used to enable / disable the amplifier output, start, and abort moves in all operating modes, and clear fault conditions.</p> <p>Control Word Bit Mapping: The value programmed into this object is bit-mapped as follows:</p> <table> <thead> <tr> <th>Bits</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Switch On. This bit must be set to enable the amplifier.</td> </tr> <tr> <td>1</td> <td>Enable Voltage. This bit must be set to enable the amplifier.</td> </tr> <tr> <td>2</td> <td>Quick Stop. If this bit is clear, then the amplifier is commanded to perform a quick stop.</td> </tr> <tr> <td>3</td> <td>Enable Operation. This bit must be set to enable the amplifier.</td> </tr> <tr> <td>4-6</td> <td>Operation mode specific. Descriptions appear in the sections that describe the various operating modes</td> </tr> <tr> <td>7</td> <td>Reset Fault. A low-to-high transition of this bit makes the amplifier attempt to clear any latched fault condition.</td> </tr> <tr> <td>8-15</td> <td>Reserved for future use.</td> </tr> </tbody> </table>						Bits	Description	0	Switch On. This bit must be set to enable the amplifier.	1	Enable Voltage. This bit must be set to enable the amplifier.	2	Quick Stop. If this bit is clear, then the amplifier is commanded to perform a quick stop.	3	Enable Operation. This bit must be set to enable the amplifier.	4-6	Operation mode specific. Descriptions appear in the sections that describe the various operating modes	7	Reset Fault. A low-to-high transition of this bit makes the amplifier attempt to clear any latched fault condition.	8-15	Reserved for future use.
Bits	Description																				
0	Switch On. This bit must be set to enable the amplifier.																				
1	Enable Voltage. This bit must be set to enable the amplifier.																				
2	Quick Stop. If this bit is clear, then the amplifier is commanded to perform a quick stop.																				
3	Enable Operation. This bit must be set to enable the amplifier.																				
4-6	Operation mode specific. Descriptions appear in the sections that describe the various operating modes																				
7	Reset Fault. A low-to-high transition of this bit makes the amplifier attempt to clear any latched fault condition.																				
8-15	Reserved for future use.																				

Object				Index	Sub-Index																												
STATUS WORD				0X6041	--																												
Type	Access	Units	Range	Map PDO	Memory																												
Unsigned 16	RO	--	Refer to Description	YES	--																												
<p>Description</p> <p>This object identifies the current state of the amplifier and is bit-mapped as follows:</p> <table> <thead> <tr> <th>Bits</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Ready to switch on.</td> </tr> <tr> <td>1</td> <td>Switched on.</td> </tr> <tr> <td>2</td> <td>Operation Enabled. Set when the amplifier is enabled.</td> </tr> <tr> <td>3</td> <td>Fault. If set, a latched fault condition is present in the amplifier.</td> </tr> <tr> <td>4</td> <td>Voltage enabled. Set if the amplifier bus voltage is above the minimum necessary for normal operation.</td> </tr> <tr> <td>5</td> <td>Quick Stop. When clear, the amplifier is performing a quick stop.</td> </tr> <tr> <td>6</td> <td>Switch on disabled.</td> </tr> <tr> <td>8</td> <td>Set if the last trajectory was aborted rather than finishing normally.</td> </tr> <tr> <td></td> <td>Remote. Set when the amplifier is being controlled by the CANopen interface. When clear, the amplifier may be monitored through this interface, but some other input source is controlling it.</td> </tr> <tr> <td>10</td> <td>Target Reached. This bit is set when the motor has come to rest at the target position. This bit is cleared when a trajectory is running, or when the position error is greater then the tracking window value.</td> </tr> <tr> <td>11</td> <td>Internal Limit Active. This bit is set when amplifier limits current.</td> </tr> <tr> <td>12-13</td> <td>The meanings of these bits are operation mode specific.</td> </tr> <tr> <td>14-15</td> <td>Set when the amplifier is performing a move and cleared when the trajectory finishes. This bit is cleared immediately at the end of the move, not after the motor has settled into position.</td> </tr> </tbody> </table>						Bits	Description	0	Ready to switch on.	1	Switched on.	2	Operation Enabled. Set when the amplifier is enabled.	3	Fault. If set, a latched fault condition is present in the amplifier.	4	Voltage enabled. Set if the amplifier bus voltage is above the minimum necessary for normal operation.	5	Quick Stop. When clear, the amplifier is performing a quick stop.	6	Switch on disabled.	8	Set if the last trajectory was aborted rather than finishing normally.		Remote. Set when the amplifier is being controlled by the CANopen interface. When clear, the amplifier may be monitored through this interface, but some other input source is controlling it.	10	Target Reached. This bit is set when the motor has come to rest at the target position. This bit is cleared when a trajectory is running, or when the position error is greater then the tracking window value.	11	Internal Limit Active. This bit is set when amplifier limits current.	12-13	The meanings of these bits are operation mode specific.	14-15	Set when the amplifier is performing a move and cleared when the trajectory finishes. This bit is cleared immediately at the end of the move, not after the motor has settled into position.
Bits	Description																																
0	Ready to switch on.																																
1	Switched on.																																
2	Operation Enabled. Set when the amplifier is enabled.																																
3	Fault. If set, a latched fault condition is present in the amplifier.																																
4	Voltage enabled. Set if the amplifier bus voltage is above the minimum necessary for normal operation.																																
5	Quick Stop. When clear, the amplifier is performing a quick stop.																																
6	Switch on disabled.																																
8	Set if the last trajectory was aborted rather than finishing normally.																																
	Remote. Set when the amplifier is being controlled by the CANopen interface. When clear, the amplifier may be monitored through this interface, but some other input source is controlling it.																																
10	Target Reached. This bit is set when the motor has come to rest at the target position. This bit is cleared when a trajectory is running, or when the position error is greater then the tracking window value.																																
11	Internal Limit Active. This bit is set when amplifier limits current.																																
12-13	The meanings of these bits are operation mode specific.																																
14-15	Set when the amplifier is performing a move and cleared when the trajectory finishes. This bit is cleared immediately at the end of the move, not after the motor has settled into position.																																

Object				Index	Sub-Index												
QUICK STOP OPTION CODE				0X605A	--												
Type	Access	Units	Range	Map PDO	Memory												
Unsigned 16	RW	--	Refer to Description	NO	R												
<p>Description</p> <p>This object defines the behavior of the amplifier when a quick stop command is issued. The following values are defined:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable the amplifier's outputs</td> </tr> <tr> <td>1</td> <td>Slow down using the slow down ramp (i.e. the normal move deceleration value). When the move has been successfully aborted the amplifier's state will transition to the 'switch on disabled' state.</td> </tr> <tr> <td>2</td> <td>Slow down using the quick stop ramp, then transition to 'switch on disabled'.</td> </tr> <tr> <td>5</td> <td>Slow down using the slow down ramp. The amplifier state will remain in the 'quick stop' state after the move has been finished.</td> </tr> <tr> <td>6</td> <td>Slow down using the quick stop ramp and stay in 'quick stop' state.</td> </tr> </tbody> </table> <p>All other values will produce unspecified results and should not be used.</p>						Value	Description	0	Disable the amplifier's outputs	1	Slow down using the slow down ramp (i.e. the normal move deceleration value). When the move has been successfully aborted the amplifier's state will transition to the 'switch on disabled' state.	2	Slow down using the quick stop ramp, then transition to 'switch on disabled'.	5	Slow down using the slow down ramp. The amplifier state will remain in the 'quick stop' state after the move has been finished.	6	Slow down using the quick stop ramp and stay in 'quick stop' state.
Value	Description																
0	Disable the amplifier's outputs																
1	Slow down using the slow down ramp (i.e. the normal move deceleration value). When the move has been successfully aborted the amplifier's state will transition to the 'switch on disabled' state.																
2	Slow down using the quick stop ramp, then transition to 'switch on disabled'.																
5	Slow down using the slow down ramp. The amplifier state will remain in the 'quick stop' state after the move has been finished.																
6	Slow down using the quick stop ramp and stay in 'quick stop' state.																

Object				Index	Sub-Index						
SHUTDOWN OPTION CODE				0X605B	--						
Type	Access	Units	Range	Map PDO	Memory						
Unsigned 16	RW	--	Refer to Description	NO	R						
<p>Description</p> <p>This object defines the behavior of the amplifier when the amplifier's state is changed from 'operation enabled' to 'ready to switch on'. The following values are defined:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable the amplifier's outputs</td> </tr> <tr> <td>1</td> <td>Slow down using the slow down ramp (i.e. the normal move deceleration value).</td> </tr> </tbody> </table> <p>All other values will produce unspecified results and should not be used.</p>						Value	Description	0	Disable the amplifier's outputs	1	Slow down using the slow down ramp (i.e. the normal move deceleration value).
Value	Description										
0	Disable the amplifier's outputs										
1	Slow down using the slow down ramp (i.e. the normal move deceleration value).										

Object				Index	Sub-Index						
DISABLE OPERATION OPTION CODE				0X605C	--						
Type	Access	Units	Range	Map PDO	Memory						
Unsigned 16	RW	--	Refer to Description	NO	R						
<p>Description</p> <p>This object defines the behavior of the amplifier when the amplifier's state is changed from 'operation enabled' to 'switched on'. The following values are defined:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disable the amplifier's outputs</td> </tr> <tr> <td>1</td> <td>Slow down using the slow down ramp (i.e. the normal move deceleration value).</td> </tr> </tbody> </table> <p>All other values will produce unspecified results and should not be used.</p>						Value	Description	0	Disable the amplifier's outputs	1	Slow down using the slow down ramp (i.e. the normal move deceleration value).
Value	Description										
0	Disable the amplifier's outputs										
1	Slow down using the slow down ramp (i.e. the normal move deceleration value).										

Object				Index	Sub-Index												
MODE OF OPERATION				0X6060	--												
Type	Access	Units	Range	Map PDO	Memory												
Unsigned 8	RW	--	Refer to Description	YES	R												
<p>Description</p> <p>This object selects the amplifier's mode of operation. The modes of operation presently supported by this device are:</p> <table border="1"> <thead> <tr> <th>Mode</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>-1</td> <td>Single loop Current follower</td> </tr> <tr> <td>-2</td> <td>Single loop Velocity follower</td> </tr> <tr> <td>1</td> <td>Profile Position mode</td> </tr> <tr> <td>3</td> <td>Profile Velocity mode</td> </tr> <tr> <td>6</td> <td>Homing mode</td> </tr> </tbody> </table> <p>The amplifier will not accept other values. Note that there may be some delay between setting the mode of operation and the amplifier assuming that mode. To read the active mode of operation, use object 0x6061.</p>						Mode	Description	-1	Single loop Current follower	-2	Single loop Velocity follower	1	Profile Position mode	3	Profile Velocity mode	6	Homing mode
Mode	Description																
-1	Single loop Current follower																
-2	Single loop Velocity follower																
1	Profile Position mode																
3	Profile Velocity mode																
6	Homing mode																

Object				Index	Sub-Index
MODE OF OPERATION DISPLAY				0X6061	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 8	RO	--	Refer to Description	YES	--
Description This object displays the current mode of operation. Refer to Mode of Operation (index 0x6060, paragraph 6.2, page 35).					

Object				Index	Sub-Index						
REFERENCE SOURCE				0X2025	--						
Type	Access	Units	Range	Map PDO	Memory						
Unsigned 16	RW	--	Refer to Description	YES	--						
Description This object configures type of the reference for Velocity follower or Current follower modes. Refer to Mode of Operation (index 0x6060, paragraph 6.2, page 35) ,NON PROFILED OPERATING MODES (paragraph 8, page 62) and Control Loop Configuration (paragraph 7, page 51). Possible values are: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reference configured to be Analog input #1</td> </tr> <tr> <td>1</td> <td>Reference configured to a digital value. Object number depends on the operating mode</td> </tr> </tbody> </table>						Value	Description	0	Reference configured to be Analog input #1	1	Reference configured to a digital value. Object number depends on the operating mode
Value	Description										
0	Reference configured to be Analog input #1										
1	Reference configured to a digital value. Object number depends on the operating mode										

Object				Index	Sub-Index																																																																		
MANUFACTURER STATUS REGISTER				0X1002	--																																																																		
Type	Access	Units	Range	Map PDO	Memory																																																																		
Unsigned 32	RO	--	Refer to Description	YES	--																																																																		
Description This 32-bit object is a bit-mapped status register with the following fields: <table border="1"> <thead> <tr> <th>Bit in register</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0</td><td>Set when drive enabled</td></tr> <tr><td>1</td><td>Set if DSP subsystem at any fault</td></tr> <tr><td>2</td><td>Set if drive has a valid program</td></tr> <tr><td>3</td><td>Set if byte-code or system or DSP at any fault</td></tr> <tr><td>4</td><td>Set if drive has a valid source code</td></tr> <tr><td>5</td><td>Set if motion completed and target position is within specified limits</td></tr> <tr><td>6</td><td>Set when scope is triggered and data collected</td></tr> <tr><td>7</td><td>Set if motion stack is full</td></tr> <tr><td>8</td><td>Set if motion stack is empty</td></tr> <tr><td>9</td><td>Set if byte-code halted</td></tr> <tr><td>10</td><td>Set if byte-code is running</td></tr> <tr><td>11</td><td>Set if byte-code is set to run in step mode</td></tr> <tr><td>12</td><td>Set if byte-code is reached the end of program</td></tr> <tr><td>13</td><td>Set if current limit is reached</td></tr> <tr><td>14</td><td>Set if byte-code at fault</td></tr> <tr><td>15</td><td>Set if no valid motor selected</td></tr> <tr><td>16</td><td>Set if byte-code at arithmetic fault</td></tr> <tr><td>17</td><td>Set if byte-code at user fault</td></tr> <tr><td>18</td><td>Set if DSP initialization completed</td></tr> <tr><td>19</td><td>Set if registration has been triggered</td></tr> <tr><td>20</td><td>Set if registration variable was updated from DSP after last trigger</td></tr> <tr><td>21</td><td>Set if motion module at fault</td></tr> <tr><td>22</td><td>Set if motion suspended</td></tr> <tr><td>23</td><td>Set if program requested to suspend motion</td></tr> <tr><td>24</td><td>Set if system waits completion of motion</td></tr> <tr><td>25</td><td>Set if motion command completed and motion Queue is empty</td></tr> <tr><td>26</td><td>Set if byte-code task requested reset</td></tr> <tr><td>27</td><td>If set interface control is disabled. This flag is set/clear by ICONTROL ON/OFF statement.</td></tr> <tr><td>28</td><td>Set if positive limit switch reached</td></tr> <tr><td>29</td><td>Set if negative limit switch reached</td></tr> <tr><td>30</td><td>Events disabled. All events disabled when this flag is set. After executing EVENTS ON all events previously enabled by EVENT EventName ON statements become enabled again</td></tr> <tr><td>31</td><td>Set if 'under voltage' condition detected</td></tr> </tbody> </table>						Bit in register	Description	0	Set when drive enabled	1	Set if DSP subsystem at any fault	2	Set if drive has a valid program	3	Set if byte-code or system or DSP at any fault	4	Set if drive has a valid source code	5	Set if motion completed and target position is within specified limits	6	Set when scope is triggered and data collected	7	Set if motion stack is full	8	Set if motion stack is empty	9	Set if byte-code halted	10	Set if byte-code is running	11	Set if byte-code is set to run in step mode	12	Set if byte-code is reached the end of program	13	Set if current limit is reached	14	Set if byte-code at fault	15	Set if no valid motor selected	16	Set if byte-code at arithmetic fault	17	Set if byte-code at user fault	18	Set if DSP initialization completed	19	Set if registration has been triggered	20	Set if registration variable was updated from DSP after last trigger	21	Set if motion module at fault	22	Set if motion suspended	23	Set if program requested to suspend motion	24	Set if system waits completion of motion	25	Set if motion command completed and motion Queue is empty	26	Set if byte-code task requested reset	27	If set interface control is disabled. This flag is set/clear by ICONTROL ON/OFF statement.	28	Set if positive limit switch reached	29	Set if negative limit switch reached	30	Events disabled. All events disabled when this flag is set. After executing EVENTS ON all events previously enabled by EVENT EventName ON statements become enabled again	31	Set if 'under voltage' condition detected
Bit in register	Description																																																																						
0	Set when drive enabled																																																																						
1	Set if DSP subsystem at any fault																																																																						
2	Set if drive has a valid program																																																																						
3	Set if byte-code or system or DSP at any fault																																																																						
4	Set if drive has a valid source code																																																																						
5	Set if motion completed and target position is within specified limits																																																																						
6	Set when scope is triggered and data collected																																																																						
7	Set if motion stack is full																																																																						
8	Set if motion stack is empty																																																																						
9	Set if byte-code halted																																																																						
10	Set if byte-code is running																																																																						
11	Set if byte-code is set to run in step mode																																																																						
12	Set if byte-code is reached the end of program																																																																						
13	Set if current limit is reached																																																																						
14	Set if byte-code at fault																																																																						
15	Set if no valid motor selected																																																																						
16	Set if byte-code at arithmetic fault																																																																						
17	Set if byte-code at user fault																																																																						
18	Set if DSP initialization completed																																																																						
19	Set if registration has been triggered																																																																						
20	Set if registration variable was updated from DSP after last trigger																																																																						
21	Set if motion module at fault																																																																						
22	Set if motion suspended																																																																						
23	Set if program requested to suspend motion																																																																						
24	Set if system waits completion of motion																																																																						
25	Set if motion command completed and motion Queue is empty																																																																						
26	Set if byte-code task requested reset																																																																						
27	If set interface control is disabled. This flag is set/clear by ICONTROL ON/OFF statement.																																																																						
28	Set if positive limit switch reached																																																																						
29	Set if negative limit switch reached																																																																						
30	Events disabled. All events disabled when this flag is set. After executing EVENTS ON all events previously enabled by EVENT EventName ON statements become enabled again																																																																						
31	Set if 'under voltage' condition detected																																																																						

Object				Index	Sub-Index
EXTENDED STATUS REGISTER				0X2053	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	R0	--	Refer to Description	YES	--
Description This 32-bit object is a bit-mapped extended status register with the following fields: Bit in register Description 0 DSP subsystem in run mode 1 Velocity in specified velocity window 2 Registration input detected 3 DSP system in fault state 4 DSP system ready to run 5 Velocity within Zero limits 6 Reserved 7 Reserved 8 Reserved 9 Encoder lost 10 Encoder data invalid 11 Regen output ON (active) 12 Motor powered 13 Over-current flag 14 Reserved 15 Reserved 16 Event processor is running 17 Events are set to run in step mode 18 Reserved 19 Set if parameter's flash file is not valid (checksum doesn't match) 20 Set if indexer program protected by password 21 If set then PositionServo is in Test Mode					

6.3 Error Management Objects

This section describes the objects used to view error status and define error limits and error handling. They include:

PRE-DEFINED ERROR OBJECT	INDEX 0X1003	
NUMBER OF ERRORS	INDEX 0X1003	SUB-INDEX 0
STANDARD ERROR FIELD	INDEX 0X1003	SUB-INDEX 1-8
ERROR REGISTER	INDEX 0X1001	
FAULT STATUS	INDEX 0X2009	

This is the FAULT History access:

Object				Index	Sub-Index
PRE-DEFINED ERROR OBJECT				0X1003	--
Type	Access	Units	Range	Map PDO	Memory
Array	RW	--	--	NO	R
Description This object provides an error history. Each sub-index object holds an error that has occurred on the device and has been signaled via the Emergency Object. Refer to Emergency Messages. The entry at sub-index 0 contains the number of errors that are recorded in the array starting at sub-index 1. Each new error is stored at sub-index 1. Older errors move down the list.					

Object				Index	Sub-Index
NUMBER OF ERRORS				0X1003	0
Type	Access	Units	Range	Map PDO	Memory
Unsigned 8	RW	--	0 - 8	NO	R
Description This object provides the number of errors in the error history (number of sub-index objects 1-8). Writing a 0 deletes the error history (empties the array). Writing a value higher than 0 results in an error.					

Object				Index	Sub-Index
STANDARD ERROR FIELD				0X1003	1-8
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	RW	--	--	NO	R
Description One sub-index object for each error found, up to 8 errors. Each is composed of a 16-bit error code and a 16-bit additional error information field. The error code is contained in the lower 2 bytes (LSB) and the additional information is included in the upper 2 bytes (MSB).					

Object				Index	Sub-Index																		
ERROR REGISTER				0X1001	--																		
Type	Access	Units	Range	Map PDO	Memory																		
Unsigned 8	RO	--	Refer to Description	YES	--																		
Description This object is a bit-mapped list of error conditions present in the amplifier. The bits used in this register are mapped as follows: <table border="0"> <tr> <td><u>Bits</u></td> <td><u>Description</u></td> </tr> <tr> <td>0</td> <td>Generic error. This bit is set any time there is an error condition in the amplifier.</td> </tr> <tr> <td>1</td> <td>Current error. Indicates either a short circuit on the motor outputs, or excessive current beyond amplifier capability was detected.</td> </tr> <tr> <td>2</td> <td>Voltage error. The DC bus voltage supplied to the amplifier is either over or under the amplifier's limits.</td> </tr> <tr> <td>3</td> <td>Temperature error. Either the amplifier or motor is over temperature. Note that the amplifier will only detect a motor over temperature condition if an amplifier input has been configured to detect this condition.</td> </tr> <tr> <td>4</td> <td>Communication error. The amplifier does not presently use this bit.</td> </tr> <tr> <td>5</td> <td>Reserved. Undefined</td> </tr> <tr> <td>6</td> <td>Reserved. Always 0.</td> </tr> <tr> <td>7</td> <td>The following errors cause this bit to be set; tracking error, limit switch active.</td> </tr> </table>						<u>Bits</u>	<u>Description</u>	0	Generic error. This bit is set any time there is an error condition in the amplifier.	1	Current error. Indicates either a short circuit on the motor outputs, or excessive current beyond amplifier capability was detected.	2	Voltage error. The DC bus voltage supplied to the amplifier is either over or under the amplifier's limits.	3	Temperature error. Either the amplifier or motor is over temperature. Note that the amplifier will only detect a motor over temperature condition if an amplifier input has been configured to detect this condition.	4	Communication error. The amplifier does not presently use this bit.	5	Reserved. Undefined	6	Reserved. Always 0.	7	The following errors cause this bit to be set; tracking error, limit switch active.
<u>Bits</u>	<u>Description</u>																						
0	Generic error. This bit is set any time there is an error condition in the amplifier.																						
1	Current error. Indicates either a short circuit on the motor outputs, or excessive current beyond amplifier capability was detected.																						
2	Voltage error. The DC bus voltage supplied to the amplifier is either over or under the amplifier's limits.																						
3	Temperature error. Either the amplifier or motor is over temperature. Note that the amplifier will only detect a motor over temperature condition if an amplifier input has been configured to detect this condition.																						
4	Communication error. The amplifier does not presently use this bit.																						
5	Reserved. Undefined																						
6	Reserved. Always 0.																						
7	The following errors cause this bit to be set; tracking error, limit switch active.																						

Object				Index	Sub-Index																																																																																										
FAULT STATUS				0X2009	--																																																																																										
Type	Access	Units	Range	Map PDO	Memory																																																																																										
Unsigned 32	R0	--	Refer to Description	YES	R																																																																																										
<p>Description</p> <p>This object allows latching fault conditions to be viewed. When the object is read, any set bit will indicate a latching fault condition:</p> <table border="1"> <thead> <tr> <th>Fault ID</th> <th>Associated Flags*</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>1</td><td>1, 3</td><td>Overvoltage</td></tr> <tr><td>2</td><td>1, 3</td><td>Invalid hall sensors code</td></tr> <tr><td>3</td><td>1, 3</td><td>Overcurrent</td></tr> <tr><td>4</td><td>1, 3</td><td>Overtemperature</td></tr> <tr><td>5</td><td>1, 3</td><td>Reserved</td></tr> <tr><td>6</td><td>1, 3</td><td>Overspeed. (Overspeed limit set by motor capability in motor file)</td></tr> <tr><td>7</td><td>1, 3</td><td>Position error excess.</td></tr> <tr><td>8</td><td>1, 3</td><td>Attempt to enable while motor data array invalid or motor was not selected.</td></tr> <tr><td>9</td><td>1,3</td><td>Motor overtemperature switch activated</td></tr> <tr><td>10</td><td>1,3</td><td>Subprocessor error</td></tr> <tr><td>11-13</td><td>-</td><td>Reserved</td></tr> <tr><td>14</td><td>1,3</td><td>Undervoltage</td></tr> <tr><td>15</td><td>1,3</td><td>Hardware current trip protection</td></tr> <tr><td>16</td><td>-</td><td>Reserved</td></tr> <tr><td>17</td><td>3</td><td>Unrecoverable error.</td></tr> <tr><td>18</td><td>16</td><td>Division by zero</td></tr> <tr><td>19</td><td>16</td><td>Arithmetic overflow</td></tr> <tr><td>20</td><td>3</td><td>Subroutine stack overflow. Exceeded 16 levels subroutines stack depth.</td></tr> <tr><td>21</td><td>3</td><td>Subroutine stack underflow. Attempt to execute RETURN statement without preceding call to subroutine.</td></tr> <tr><td>22</td><td>3</td><td>Variable evaluation stack overflow. Expression too complicated for compiler to process.</td></tr> <tr><td>23</td><td>21</td><td>Motion Queue overflow. 32 levels depth exceeded</td></tr> <tr><td>24</td><td>21</td><td>Motion Queue underflow. Last queued MDV statement has non 0 target velocity</td></tr> <tr><td>25</td><td>3</td><td>Unknown opcode. Byte code interpreter error</td></tr> <tr><td>26</td><td>3</td><td>Unknown byte code. Byte code interpreter error</td></tr> <tr><td>27</td><td>21</td><td>Drive disabled. Attempt to execute motion while drive is disabled.</td></tr> <tr><td>28</td><td>16, 21</td><td>Accel too big. Motion statement parameters result in calculating too big Accel value for system to handle</td></tr> <tr><td>29</td><td>16, 21</td><td>Accel too small. Motion statement parameters result in calculating too small Accel value for system to handle</td></tr> <tr><td>30</td><td>16, 21</td><td>Velocity too big. Motion statement parameters result in calculating too big velocity value for system to handle</td></tr> <tr><td>31</td><td>16, 21</td><td>Velocity too small. Motion statement parameters result in calculating too small velocity value for system to handle</td></tr> </tbody> </table> <p>* Associated Flags in Status Register (Object Index 0x1002, paragraph 6.2, page 36)</p>						Fault ID	Associated Flags*	Description	1	1, 3	Overvoltage	2	1, 3	Invalid hall sensors code	3	1, 3	Overcurrent	4	1, 3	Overtemperature	5	1, 3	Reserved	6	1, 3	Overspeed. (Overspeed limit set by motor capability in motor file)	7	1, 3	Position error excess.	8	1, 3	Attempt to enable while motor data array invalid or motor was not selected.	9	1,3	Motor overtemperature switch activated	10	1,3	Subprocessor error	11-13	-	Reserved	14	1,3	Undervoltage	15	1,3	Hardware current trip protection	16	-	Reserved	17	3	Unrecoverable error.	18	16	Division by zero	19	16	Arithmetic overflow	20	3	Subroutine stack overflow. Exceeded 16 levels subroutines stack depth.	21	3	Subroutine stack underflow. Attempt to execute RETURN statement without preceding call to subroutine.	22	3	Variable evaluation stack overflow. Expression too complicated for compiler to process.	23	21	Motion Queue overflow. 32 levels depth exceeded	24	21	Motion Queue underflow. Last queued MDV statement has non 0 target velocity	25	3	Unknown opcode. Byte code interpreter error	26	3	Unknown byte code. Byte code interpreter error	27	21	Drive disabled. Attempt to execute motion while drive is disabled.	28	16, 21	Accel too big. Motion statement parameters result in calculating too big Accel value for system to handle	29	16, 21	Accel too small. Motion statement parameters result in calculating too small Accel value for system to handle	30	16, 21	Velocity too big. Motion statement parameters result in calculating too big velocity value for system to handle	31	16, 21	Velocity too small. Motion statement parameters result in calculating too small velocity value for system to handle
Fault ID	Associated Flags*	Description																																																																																													
1	1, 3	Overvoltage																																																																																													
2	1, 3	Invalid hall sensors code																																																																																													
3	1, 3	Overcurrent																																																																																													
4	1, 3	Overtemperature																																																																																													
5	1, 3	Reserved																																																																																													
6	1, 3	Overspeed. (Overspeed limit set by motor capability in motor file)																																																																																													
7	1, 3	Position error excess.																																																																																													
8	1, 3	Attempt to enable while motor data array invalid or motor was not selected.																																																																																													
9	1,3	Motor overtemperature switch activated																																																																																													
10	1,3	Subprocessor error																																																																																													
11-13	-	Reserved																																																																																													
14	1,3	Undervoltage																																																																																													
15	1,3	Hardware current trip protection																																																																																													
16	-	Reserved																																																																																													
17	3	Unrecoverable error.																																																																																													
18	16	Division by zero																																																																																													
19	16	Arithmetic overflow																																																																																													
20	3	Subroutine stack overflow. Exceeded 16 levels subroutines stack depth.																																																																																													
21	3	Subroutine stack underflow. Attempt to execute RETURN statement without preceding call to subroutine.																																																																																													
22	3	Variable evaluation stack overflow. Expression too complicated for compiler to process.																																																																																													
23	21	Motion Queue overflow. 32 levels depth exceeded																																																																																													
24	21	Motion Queue underflow. Last queued MDV statement has non 0 target velocity																																																																																													
25	3	Unknown opcode. Byte code interpreter error																																																																																													
26	3	Unknown byte code. Byte code interpreter error																																																																																													
27	21	Drive disabled. Attempt to execute motion while drive is disabled.																																																																																													
28	16, 21	Accel too big. Motion statement parameters result in calculating too big Accel value for system to handle																																																																																													
29	16, 21	Accel too small. Motion statement parameters result in calculating too small Accel value for system to handle																																																																																													
30	16, 21	Velocity too big. Motion statement parameters result in calculating too big velocity value for system to handle																																																																																													
31	16, 21	Velocity too small. Motion statement parameters result in calculating too small velocity value for system to handle																																																																																													

6.4 Basic Amplifier Configuration Objects

Objects described in this section provide access to basic amplifier parameters. They include:

DEVICE TYPE	INDEX 0X1000	
DEVICE NAME	INDEX 0X1008	
HARDWARE VERSION STRING	INDEX 0X1009	
SOFTWARE VERSION NUMBER	INDEX 0X100A	
IDENTITY OBJECT	INDEX 0X1018	
VENDOR ID	INDEX 0X1018	SUB-INDEX 1
PRODUCT CODE	INDEX 0X1018	SUB-INDEX 2
REVISION NUMBER	INDEX 0X1018	SUB-INDEX 3
SERIAL NUMBER	INDEX 0X1018	SUB-INDEX 4
AMPLIFIER NAME	INDEX 0X2060	
CANOPEN NETWORK CONFIGURATION	INDEX 0X20EA	
SUPPORTED DRIVE MODES	INDEX 0X6502	
AMPLIFIER MODEL NUMBER	INDEX 0X6503	
AMPLIFIER MANUFACTURER	INDEX 0X6504	
MANUFACTURER'S WEB ADDRESS	INDEX 0X6505	
AMPLIFIER DATA	INDEX 0X6510	
AMPLIFIER SERIAL NUMBER	INDEX 0X6510	SUB-INDEX 1
AMPLIFIER DATE CODE	INDEX 0X6510	SUB-INDEX 2
AMPLIFIER PEAK CURRENT	INDEX 0X6510	SUB-INDEX 3
AMPLIFIER CONTINUOUS CURRENT	INDEX 0X6510	SUB-INDEX 4
AMPLIFIER PEAK CURRENT TIME	INDEX 0X6510	SUB-INDEX 5
AMPLIFIER MAXIMUM VOLTAGE	INDEX 0X6510	SUB-INDEX 6
AMPLIFIER MINIMUM VOLTAGE	INDEX 0X6510	SUB-INDEX 7
AMPLIFIER MAXIMUM TEMPERATURE	INDEX 0X6510	SUB-INDEX 8
AMPLIFIER CURRENT LOOP PERIOD	INDEX 0X6510	SUB-INDEX 9
AMPLIFIER SERVO LOOP PERIOD	INDEX 0X6510	SUB-INDEX 10
AMPLIFIER TYPE CODE	INDEX 0X6510	SUB-INDEX 11
DEVICE TYPE	INDEX 0X67FF	

Object				Index	Sub-Index
DEVICE TYPE				0X1000	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	RO	--	Refer to Description	NO	--
Description Describes the type of device and its functionality. This 32-bit value is composed of two 16-bit components. The lower two bytes identify the device profile supported by the device. This amplifier supports the DSP402 device profile, indicated by the value 0x0192. The upper two bytes give detailed information about the type of motors the drive can control. The bit mapping of this value is defined by the Profile for Drives and Motion Control. CANopen amplifiers, this value is 0x0002, indicating that supports servo devices.					

Object				Index	Sub-Index
DEVICE NAME				0X1008	--
Type	Access	Units	Range	Map PDO	Memory
Visible String	R0	--	--	NO	--
Description An ASCII string which gives the amplifier's model number.					

Object				Index	Sub-Index
HARDWARE VERSION STRING				0X1009	--
Type	Access	Units	Range	Map PDO	Memory
String	Const	--	--	NO	--
Description Describes amplifier hardware version.					

Object				Index	Sub-Index
SOFTWARE VERSION NUMBER				0X100A	--
Type	Access	Units	Range	Map PDO	Memory
Visible String	R0	--	--	NO	--
Description Contains an ASCII string listing the software version number of the amplifier.					

Object				Index	Sub-Index
IDENTITY OBJECT				0X1018	--
Type	Access	Units	Range	Map PDO	Memory
Record	R0	--	--	NO	--
Description This object can uniquely identify an amplifier by unique manufacturer ID, serial number, and product revision information. Sub-index 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index
VENDOR ID				0X1018	1
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	R0	--	0x0000019C	NO	--
Description A unique identifier assigned to AC Technology Corp. The value of this identifier is fixed at: 0x0000019C					

Object				Index	Sub-Index				
PRODUCT CODE				0X1018	2				
Type	Access	Units	Range	Map PDO	Memory				
Unsigned 32	R0	--	Refer to Description	NO	--				
Description A value that uniquely identifies the amplifier type. The currently defined values for this object are: <table border="0" style="width: 100%;"> <tr> <td style="width: 10%;"><u>Value</u></td> <td><u>Product</u></td> </tr> <tr> <td>940</td> <td>940 Position Servo</td> </tr> </table>						<u>Value</u>	<u>Product</u>	940	940 Position Servo
<u>Value</u>	<u>Product</u>								
940	940 Position Servo								

Object				Index	Sub-Index
REVISION NUMBER				0X1018	3
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	R0	--	--	NO	--
Description Identifies the revision of the CANopen interface.					

Object				Index	Sub-Index
SERIAL NUMBER				0X1018	4
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	R0	--	--	NO	--
Description Identifies the amplifier's serial number.					

Object				Index	Sub-Index
AMPLIFIER NAME				0X2002	--
Type	Access	Units	Range	Map PDO	Memory
Visible string	RW	--	--	NO	F
Description This object may be used to assign a name an amplifier. The data written here is stored to flash memory and is not used by the amplifier. Although this object is documented as holding a string (i.e. ASCII data), any values may be written here.					

Object				Index	Sub-Index
CANOPEN NETWORK CONFIGURATION				0X20EA	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 16	RW	--	Refer to Description	NO	RF
Description This object is used to configure the CANopen network bit rate and node ID for the amplifier. Values written here are stored to flash memory. The new network configuration will not take effect until the amplifier is reset.					
<u>Bit</u>	<u>Description</u>				
0-6	Node ID value.				
7	Reserved for future use.				
8-11	Reserved				
12-15	Network bit rate setting.				
On power-up (or after a reset), the amplifier will determine its node ID programmed in bits 0-6. Note that the node ID value zero is not a legal CANopen ID, and will result in unspecified action. The network bit rate is encoded as one of the following values:					
<u>Code</u>	<u>Bit Rate (bits / second)</u>				
0	1,000,000				
1	800,000				
2	500,000				
3	250,000				
4	125,000				
5	50,000				
6	20,000				
7-15	Reserved for future use				

Object				Index	Sub-Index																				
SUPPORTED DRIVE MODES				0X6502	--																				
Type	Access	Units	Range	Map PDO	Memory																				
Unsigned 32	R0	--	Refer to Description	NO	--																				
<p>Description</p> <p>This bit-mapped value gives the modes of operation supported by the amplifier. The standard device profile (DSP402) defines several modes of operation. Each mode is assigned one bit in this variable. A drive indicates its support for the mode of operation by setting the corresponding bit. The modes of operation supported by this device, and their corresponding bits in this object, are as follows:</p> <table border="0"> <tr> <td>Bit</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Position profile mode</td> </tr> <tr> <td>1</td> <td>Velocity mode</td> </tr> <tr> <td>2</td> <td>Profile velocity mode</td> </tr> <tr> <td>3</td> <td>Profile torque mode</td> </tr> <tr> <td>5</td> <td>Homing mode</td> </tr> <tr> <td>6</td> <td>Interpolated Position Mode</td> </tr> <tr> <td>7-15</td> <td>Reserved</td> </tr> <tr> <td>16</td> <td>Single loop current follower (AC Technology specific)</td> </tr> <tr> <td>17</td> <td>Single loop velocity follower (AC Technology specific)</td> </tr> </table> <p>The current version of amplifier firmware supports only modes that are <i>initialized</i>. Their bits are <u>st</u> in the word and therefore the expected value of this object is 0x00030025.</p>						Bit	Description	0	Position profile mode	1	Velocity mode	2	Profile velocity mode	3	Profile torque mode	5	Homing mode	6	Interpolated Position Mode	7-15	Reserved	16	Single loop current follower (AC Technology specific)	17	Single loop velocity follower (AC Technology specific)
Bit	Description																								
0	Position profile mode																								
1	Velocity mode																								
2	Profile velocity mode																								
3	Profile torque mode																								
5	Homing mode																								
6	Interpolated Position Mode																								
7-15	Reserved																								
16	Single loop current follower (AC Technology specific)																								
17	Single loop velocity follower (AC Technology specific)																								

Object				Index	Sub-Index
AMPLIFIER MODEL NUMBER				0X6503	--
Type	Access	Units	Range	Map PDO	Memory
Visible String	R0	--	--	NO	--
<p>Description</p> <p>This ASCII string gives the amplifier model number (ID string).</p>					

Object				Index	Sub-Index
AMPLIFIER MANUFACTURER				0X6504	--
Type	Access	Units	Range	Map PDO	Memory
Visible String	R0	--	--	NO	--
<p>Description</p> <p>This ASCII string identifies the amplifier's manufacturer as "AC Tech Corp."</p>					

Object				Index	Sub-Index
MANUFACTURER'S WEB ADDRESS				0X6505	--
Type	Access	Units	Range	Map PDO	Memory
Visible String	R0	--	--	NO	--
<p>Description</p> <p>This ASCII string gives the web address of AC Technology Corp. (www.actech.com)</p>					

Object				Index	Sub-Index
AMPLIFIER DATA				0X6510	--
Type	Access	Units	Range	Map PDO	Memory
Record	R0	--	--	NO	--
<p>Description</p> <p>This record lists various amplifier parameters. Sub-index 0 contains the number of sub-elements of this record.</p>					

Object				Index	Sub-Index
AMPLIFIER SERIAL NUMBER				0X6510	1
Type	Access	Units	Range	Map PDO	Memory
Integer 32	R0	--	--	NO	--
Description Gives the amplifier serial number.					

Object				Index	Sub-Index
AMPLIFIER BUILD AND DATE CODE NUMBER				0X6510	2
Type	Access	Units	Range	Map PDO	Memory
Visible String	R0	--	--	NO	--
Description This ASCII string gives the manufacturing build code.					

Object				Index	Sub-Index
AMPLIFIER PEAK CURRENT				0X6510	3
Type	Access	Units	Range	Map PDO	Memory
Integer 16	R0	0.01 A	--	NO	--
Description The amplifier's peak current rating in 0.01-amp units. Peak current rating for PositionServo amplifiers are 3X continues current (sub-index 4).					

Object				Index	Sub-Index
AMPLIFIER CONTINUOUS CURRENT				0X6510	4
Type	Access	Units	Range	Map PDO	Memory
Integer 16	R0	0.01 A	--	NO	--
Description The amplifier's continuous current rating in 0.01-amp units.					

Object				Index	Sub-Index
AMPLIFIER PEAK CURRENT TIME				0X6510	5
Type	Access	Units	Range	Map PDO	Memory
Integer 16	R0	milliseconds	--	NO	--
Description The time the amplifier is rated to output peak current in milliseconds. Expected value for PositionServo amplifier 2000 (mS)					

Object				Index	Sub-Index
AMPLIFIER MAXIMUM VOLTAGE				0X6510	6
Type	Access	Units	Range	Map PDO	Memory
Integer 16	R0	0.1 V	--	NO	--
Description Maximum bus voltage rating for amplifier in 0.1-volt units.					

Object				Index	Sub-Index
AMPLIFIER MINIMUM VOLTAGE				0X6510	7
Type	Access	Units	Range	Map PDO	Memory
Integer 16	R0	0.1 V	--	NO	--
Description Minimum bus voltage rating for amplifier in 0.1-volt units. Over-voltage hysteresis for amplifier in 0.1-volt units.					

Object				Index	Sub-Index
AMPLIFIER MAXIMUM TEMPERATURE				0X6510	8
Type	Access	Units	Range	Map PDO	Memory
Integer 16	R0	degrees C	--	NO	--
Description Temperature limit for amplifier in degrees centigrade. PositionServo has set to 100 degree centigrade.					

Object				Index	Sub-Index
AMPLIFIER CURRENT LOOP PERIOD				0X6510	9
Type	Access	Units	Range	Map PDO	Memory
Integer 16	R0	10ns	--	NO	--
Description Current loop update period in 10-nanosecond units.					

Object				Index	Sub-Index
AMPLIFIER SERVO LOOP PERIOD				0X6510	10
Type	Access	Units	Range	Map PDO	Memory
Integer 16	R0	--	--	NO	--
Description Servo loop update period as a multiple of the current loop period. (4 for PositionServo models.)					

Object				Index	Sub-Index
AMPLIFIER TYPE CODE				0X6510	11
Type	Access	Units	Range	Map PDO	Memory
String	R0	--	Refer to Description	NO	--
Description 3 digit string "X XX" that identifies the type of amplifier in terms of mains voltage (V) and output current capability (A rms).					
<u>Digit 1</u>	<u>Description</u>	<u>Digits 2 & 3</u>	<u>Description</u>		
X	B = 240V AC C = 480V AC	XX	02 = 2A rms 04 = 4A rms 05 = 5A rms	06 = 6A rms 08 = 8A rms 09 = 9A rms	10 = 10A rms 12 = 12A rms 18 = 18A rms

Object				Index	Sub-Index
DEVICE TYPE				0X67FF	--
Type	Access	Units	Range	Map PDO	Memory
Unsigned 32	R0	--	--	NO	--
Description Holds the same data as object 0x1000. Repeated as required by the CANopen specification.					

6.5 Basic Motor Configuration Objects

Objects described in this section provide access to basic motor parameters. They include:

MOTOR MODEL NUMBER	INDEX 0X6403	
MOTOR MANUFACTURER	INDEX 0X6404	
MOTOR DATA	INDEX 0X6410	
MOTOR TYPE	INDEX 0X6410	SUB-INDEX 1
MOTOR TYPE	INDEX 0X6410	SUB-INDEX 2
SYMBOLIC MOTOR MODEL	INDEX 0X6410	SUB-INDEX 3
MOTOR VENDOR NAME	INDEX 0X6410	SUB-INDEX 4
FEEDBACK CONFIGURATION	INDEX 0X6410	SUB-INDEX 5
HALLCODE INDEX	INDEX 0X6410	SUB-INDEX 6
HALL OFFSET	INDEX 0X6410	SUB-INDEX 7
Zero OFFSET	INDEX 0X6410	SUB-INDEX 8
ICTRL (RESERVED)	INDEX 0X6410	SUB-INDEX 9
MOTOR INERTIA	INDEX 0X6410	SUB-INDEX 10
MOTOR BACK EMF	INDEX 0X6410	SUB-INDEX 11
MOTOR TORQUE CONSTANT	INDEX 0X6410	SUB-INDEX 12
MOTOR INDUCTANCE	INDEX 0X6410	SUB-INDEX 13
MOTOR RESISTANCE	INDEX 0X6410	SUB-INDEX 14
MOTOR MAX CONT. CURRENT	INDEX 0X6410	SUB-INDEX 15
MOTOR MAX VELOCITY	INDEX 0X6410	SUB-INDEX 16
MOTOR POLES	INDEX 0X6410	SUB-INDEX 17
ENCODER COUNT	INDEX 0X6410	SUB-INDEX 18
MOTOR NOMINAL TERMINAL VOLTAGE	INDEX 0X6410	SUB-INDEX 19
MOTOR FEEDBACK DEVICE TYPE	INDEX 0X6410	SUB-INDEX 19

Object				Index	Sub-Index
MOTOR MODEL NUMBER				0X6403	--
Type	Access	Units	Range	Map PDO	Memory
Visible String	RW	--	--	NO	F
Description This object gives a location to store the motor's model number for future reference. M_MODEL. Note that this parameter is always stored to non-volatile memory on the amplifier. The programmed value is preserved across power cycles.					

Object				Index	Sub-Index
RESERVED				0X6404	--
Type	Access	Units	Range	Map PDO	Memory
Visible String	RO	--	--	NO	F
Description Reserved for future use.					

Object				Index	Sub-Index
MOTOR DATA				0X6410	--
Type	Access	Units	Range	Map PDO	Memory
Record	RW	--	--	NO	--
Description This record holds a variety of motor parameters. Note that all motor parameters are stored to non-volatile memory on the amplifier. The programmed values are preserved across power cycles. Sub-index 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index
MOTOR ID				0X6410	1
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	Refer to Description	NO	F
Description Returns motor catalog ID. This field needs to be set to "-1" for any motor which doesn't have AC Technology Corp. catalog ID assigned.					

Object				Index	Sub-Index								
MOTOR TYPE				0X6410	2								
Type	Access	Units	Range	Map PDO	Memory								
Integer 16	RW	--	Refer to Description	NO	F								
Description M_SERIES Defines the type of motor connected to the amplifier. For proper operation, this object must be set to one of the following values: <table border="0" style="width: 100%;"> <tr> <td style="width: 10%;"><u>Type</u></td> <td><u>Description</u></td> </tr> <tr> <td>1-999</td> <td>Rotary AC Synchronous servo motor</td> </tr> <tr> <td>1000-1999</td> <td>Linear AC Synchronous servo motor</td> </tr> <tr> <td>2000-2999</td> <td>Rotary AC induction motor</td> </tr> </table>						<u>Type</u>	<u>Description</u>	1-999	Rotary AC Synchronous servo motor	1000-1999	Linear AC Synchronous servo motor	2000-2999	Rotary AC induction motor
<u>Type</u>	<u>Description</u>												
1-999	Rotary AC Synchronous servo motor												
1000-1999	Linear AC Synchronous servo motor												
2000-2999	Rotary AC induction motor												

Object				Index	Sub-Index
RESERVED				0X6410	3
Type	Access	Units	Range	Map PDO	Memory
Visible String	RO	--	Refer to Description	NO	F
Description This index is reserved for future use.					

Object				Index	Sub-Index
MOTOR VENDOR NAME				0X6410	4
Type	Access	Units	Range	Map PDO	Memory
Visible String	RW	--	--	NO	F
Description Defines symbolic motor's vendor name. Example: "Lenze"					

Object				Index	Sub-Index
FEEDBACK CONFIGURATION				0X6410	5
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	Refer to Description	NO	F
Description Describes motor's feedback device configuration data as follows:					
<u>Value</u>	<u>Description</u>				
0	reserved				
1	encoder feedback				
2	resolver feedback				
3	Absolute encoder (BiSS, SPI)				
4	Absolute encoder (EnDat)				
5	Absolute encoder (HyperFace)				

Object				Index	Sub-Index
HALL CODE				0X6410	6
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 - 5	NO	F
Description Defines motor hallcode index. Index shows hall sensors mapping to the corresponding motor phases U,V or W:					
<u>Code</u>	<u>Hall ordering</u>				
0	U V W				
1	U W V				
2	V U W				
3	V W U				
4	W V U				
5	W U V				

Object				Index	Sub-Index
HALL OFFSET				0X6410	7
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RO	--	0	NO	F
Description Reserved for future use by AC Technology Corp. Must be set to 0.					

Object				Index	Sub-Index
Zero OFFSET				0X6410	8
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	0.1 mech. degree	36000 - 36000	NO	F
Description Description $(\text{Value} / 100) * (65536/360)$ Value/100 must be modulo 360 - positive. Sets the resolver offset value in 0.01 mechanical degree. Must be 0 for motors with incremental encoders.					

Object				Index	Sub-Index
ICTRL (RESERVED)				0X6410	9
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	--	NO	F
Description Reserved. Must be set to 0.					

Object				Index	Sub-Index
MOTOR INERTIA				0X6410	10
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	0.000001 kg / cm ²	0 - 2,147,483,647	NO	F
Description M_JM - kg / m ² Value in * 10e2 - gives kg/m ² - write to PID Motor inertia in units of 0.000001 kg / cm ² .					

Object				Index	Sub-Index
MOTOR BACK EMF				0X6410	11
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	0.01 V/kRPM	0 - 2,147,483,647	NO	F
Description M_KE - Value * 100 ->PID Motor back-EMF constant. Units are 0.01 V/kRPM.					

Object				Index	Sub-Index
MOTOR TORQUE CONSTANT				0X6410	12
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	0.001 Nm / Amp	0 - 2,147,483,647	NO	F
Description M_KT Value * 1000 - > PID Motor torque constant, units: 0.001 Nm / Amp.					

Object				Index	Sub-Index
MOTOR INDUCTANCE				0X6410	13
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	0.01 milli Henry	0 - 32,767	NO	F
Description M_LS Value *100 -> PID Motor winding inductance, in 0.01-milli Henry units.					

Object				Index	Sub-Index
MOTOR RESISTANCE				0X6410	14
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	0.01 Ohm	0 - 32,767	NO	F
Description M_RS Value * 100 -> PID Motor winding resistance, in 0.01-Ohm units.					

Object				Index	Sub-Index
MOTOR MAX CONT. CURRENT				0X6410	15
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	0.01 A(Rms)	0 - 32,767	NO	F
Description M_MAXCURRENT Value * 100 -> PID Motor continuous RMS current per phase in 0.01 A Rms units.					

Object				Index	Sub-Index
MOTOR MAX VELOCITY				0X6410	16
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	1 RPM	0 - 32,767	NO	F
Description M_MAXVELOCITY -> direct Motor maximum velocity in RPM.					

Object				Index	Sub-Index
MOTOR POLES				0X6410	17
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 - 32,767	NO	F
Description M_NPOLES -> direct Number of motor poles per rotation. For proper operation this value must be set correctly for the motor being controlled. This parameter is only used for rotary motors. For linear motors its value needs to be set to 2.					

Object				Index	Sub-Index
ENCODER COUNTS				0X6410	18
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts / rev	0 - 2,147,483,647	NO	F
Description M_ENCODER -> Value *4 For rotary motors gives the number of encoder counts / motor revolution. For linear motors this parameter represents ration between pole pair pitch and linear encoder resolution. Value to write for linear motors can be calculated as follows: Encoder counts = Lpp / LEres where Lpp - pole pair pitch (S-S or N-N distance in linear motor) LEres – linear encoder resolution.					

Object				Index	Sub-Index
MOTOR NOMINAL TERMINAL VOLTAGE				0X6410	19
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	1 Volt	0 - 32,767	NO	F
Description M_TERMVOLTAGE - direct Sets motor normal terminal voltage in 1 Volt units.					

Object				Index	Sub-Index														
MOTOR FEEDBACK DEVICE TYPE				0X6410	20														
Type	Access	Units	Range	Map PDO	Memory														
Integer 16	RW	--	0 - 32,767	NO	F														
Description Describes motor's feedback device configuration data as follows: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>reserved</td> </tr> <tr> <td>1</td> <td>encoder feedback</td> </tr> <tr> <td>2</td> <td>resolver feedback</td> </tr> <tr> <td>3</td> <td>Absolute encoder (BiSS, SPI)</td> </tr> <tr> <td>4</td> <td>Absolute encoder (EnDat)</td> </tr> <tr> <td>5</td> <td>Absolute encoder (HyperFace)</td> </tr> </tbody> </table>						Value	Description	0	reserved	1	encoder feedback	2	resolver feedback	3	Absolute encoder (BiSS, SPI)	4	Absolute encoder (EnDat)	5	Absolute encoder (HyperFace)
Value	Description																		
0	reserved																		
1	encoder feedback																		
2	resolver feedback																		
3	Absolute encoder (BiSS, SPI)																		
4	Absolute encoder (EnDat)																		
5	Absolute encoder (HyperFace)																		

7 Control Loops

This chapter describes the nested control loop model used by AC Tech amplifiers to control the position of the motor.

- 7.1 Control Loop Configuration
- 7.2 Position Loop Configuration Objects
- 7.3 Velocity Loop Configuration Objects
- 7.4 Current Loop Configuration Objects

7.1 Control Loop Configuration

This section provides an overview of the control loops. Topics include:

- 7.1.1 Nested Control Loops
- 7.1.2 The Position Loop
- 7.1.3 The Velocity Loop
- 7.1.4 The Current Loop

7.1.1 Nested Control Loops

AC Tech amplifiers use up to three nested control loops - current, velocity, and position – to control a motor in three associated operating modes. In position mode, The amplifier uses all three loops, as shown in Figure 10. The loops are nested: the current loop within the velocity loop, within the position loop. Stated another way, the position loop drives the velocity loop, which drives the current loop

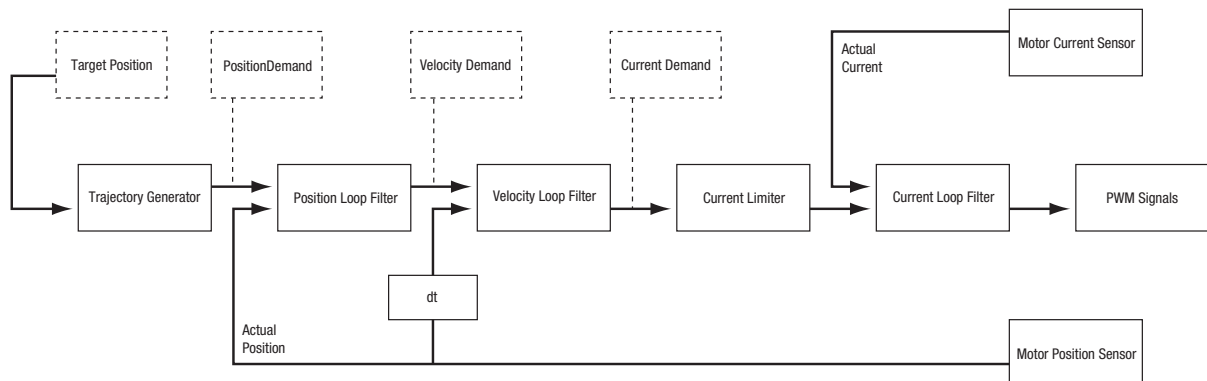


Figure 10: Nested Loops

Basic Attributes of All Control Loops

These loops (and servo control loops in general) share several common attributes including command input, limits, feedback, gain and outputs.

Loop Attribute Description

Command Input	Every loop is given a value which it will attempt to control. For example, the velocity loop receives a velocity demand that is the desired motor speed.
Limits	Limits are set on current loop to protect the motor and/or mechanical system.
Feedback	The nature of servo control loops is that they receive feedback from the device they are controlling. For example, the position loop uses the actual motor position as feedback.
Gains	These are constant values that are used in the mathematical equation of the servo loop. The values of these gains can be adjusted during amplifier setup to improve the loop performance. Adjusting these values is often referred to as tuning the loop.
Output	The loop generates a control signal. This signal can be used as the command signal to another control loop or the input to a power amplifier.

7.1.2 The Position Loop

The CANopen master provides a target position to the amplifier's internal trajectory generator. In turn the generator provides the position loop a position demand and velocity and acceleration limit values. The position loop applies corrective gains in response to feedback to forward a velocity demand to the velocity loop. The inputs to the position loop vary with different operating modes.

Trajectory Generator Inputs

The inputs to the trajectory generator include profile position, velocity, and acceleration values. They are accessed through different sets of mode-specific objects as summarized in Table 7.

Table 7: Trajectory Generator Inputs

Mode	Input Object Name	Object ID	Description
Homing	Homing Method	0x6098	Defines the method to find the motor home position
	Homing Speeds	0x6099	The sub-index objects of 0x6099 hold the two velocities (fast and slow) used when homing
	Homing Acceleration	0x609A	Defines the acceleration used for all homing moves
	Home Offset	0x607C	Used in homing mode as an offset between the home sensor position and the zero position
Profile Position	Motion Profile Type	0x6086	Selects the type of trajectory profile to use. Choices are trapezoidal and s-curve.
	Target Position	0x607A	Destination position of the move
	Profile Velocity	0x6081	The velocity that the trajectory generator attempts to achieve when running in position profile mode
	Profile Acceleration	0x6083	Acceleration that the trajectory generator attempts to achieve when running in position profile mode
	Profile Deceleration	0x6084	Deceleration that the trajectory generator attempts to achieve at the end of a trapezoidal profile when running in position profile mode

Position Loop Feedback

The feedback to the loop is the actual motor position, obtained from a position sensor attached to the motor (most often a quadrature encoder). This is provided by the Position Actual Value object (index 0x6063, paragraph 7.2, page 55).

Position Loop Gains

The following gains are used by the position loop to calculate the output value: PP, PI, PL, and PD.

Gain	Description
PP - Position loop proportional	The loop calculates its position error as the difference between the Position Actual Value and the Position Demand Value. This error in turn is multiplied by the proportional gain value. The primary effect of this gain is to reduce the following error.
PI – Integral	The position error gets accumulated. The primary effect of this gain is to decrease steady error. Accumulated error multiplies on PI gain value and added to the velocity demand
PL - Limit	Limit influence produced by PI gain to allow faster settling time
PD – Differential	The position error in previous step gets subtracted from the current position error forming error change rate. Error rate of change multiplies on PD gain and added to velocity demand. This gain primarily contributes to stability of the loop acting as a “shock absorber”.

These gains are accessed through the sub-index objects of the Position Loop Gains object (index 0x60FB, paragraph 7.2, page 57).

7.1.3 The Velocity Loop

As shown the summing junction takes the velocity demand from position loop, subtracts the actual velocity, represented by the feedback signal, and produces an error signal. This error signal is then processed using the integral and proportional gains to produce a current demand.

During normal operation in position mode velocity loop driven by the position loop. When device placed in velocity mode velocity loop can be driven by analog reference, or value taken from internal variable manipulated by any of the device’s interfaces such as RS-232, RS-485, Ethernet, CAN etc. Set of limiting parameters for this occasion is set by following objects:

Object Name	Object ID	Object Note
*Analog Input #1 Velocity scaling factor	0x2424	(used only without position loop)
*Velocity Loop Acceleration	0x244C	(used only without position loop)
*Velocity Loop Deceleration	0x244D	(used only without position loop)

*Not used when the velocity loop is controlled by the position loop.

Velocity Loop Input

If drive is in position mode then output of the position loop is the input of the velocity loop . If drive is in velocity follower mode then input of the velocity loop could be an internal object (index 0x248B) or analog input #1. Selection is done using the Reference Source object (index 0x2025, paragraph 6.2, page 36).

Velocity Loop Gains

The velocity loop uses the following gains. Refer to the Velocity Loop Gains object (index 0x60F9, paragraph 7.3, page 60).

Gain	Name	Description
Vp	Velocity loop proportional	The velocity error (the difference between the actual and the limited commanded velocity) is multiplied by this gain. The primary effect of this gain is to increase bandwidth (or decrease the step-response time) as the gain is increased.
Vw	Velocity wide	This is additional proportional gain applied to output of velocity loop to allow wider range of gain values. This gain can range from -16 to +4
Vi	Velocity loop integral	The integral of the velocity error is multiplied by this value. Integral gain reduces the velocity error to zero over time. It controls the DC accuracy of the loop, or the flatness of the top of a square wave signal. The error integral is the accumulated sum of the velocity error value over time.

7.1.4 The Current Loop

Current loop starts from current limiter. The current limiter accepts a current demand from the velocity loop, applies limits, and passes a limited current value to the summing junction. The summing junction takes the commanded current, subtracts the actual current (represented by the feedback signal), and produces an error signal. This error signal is then processed using the integral and proportional gains to produce a voltage command. This command is then applied to the amplifier's power stage.

Current Loop Input

If drive is in position or velocity mode then output of the velocity loop is the input of the current loop. If drive is in current mode then input of the current loop could be an internal object (index 0x248B) or analog input #1. Selection is done by the Reference Source object (index 0x2025). In case if analog loop is driven by analog input #1 scale value applied by object Analog Input #1 Current scaling factor (index 0x2423)

Current Loop Limits

The commanded current value is first reduced based on a set of current limit parameters designed to protect the motor. These current limits are accessed through the following objects:

Object Name	Object ID	Description
Peak Current Limit 16	0x241F	Maximum current that can be generated by the amplifier if carrier frequency selected is 16kHz for a short duration of time. This value cannot exceed the peak current rating of the amplifier.
Peak Current Limit 8	0x2420	Maximum current that can be generated by the amplifier if carrier frequency selected is 8kHz for a short duration of time. This value cannot exceed the peak current rating of the amplifier.
Continuous Current Limit	0x241E	Maximum current that can be constantly generated by the amplifier.
Analog Input # 1 CSF	0x2423	(CSF: Current Scaling Factor) Current scaling value when driven by analog input #1

7.2 Position Loop Configuration Objects

This section describes the objects used to configure the position control loop.

POSITION DEMAND VALUE	INDEX 0X6062	
POSITION ACTUAL VALUE	INDEX 0X6063	
POSITION ACTUAL VALUE	INDEX 0X6064	
FOLLOWING ERROR WINDOW	INDEX 0X6065	
FOLLOWING ERROR TIME	INDEX 0X6066	
POSITION WINDOW	INDEX 0X6067	
ACTUAL VELOCITY	INDEX 0X6069	
VELOCITY DEMAND VALUE	INDEX 0X606B	
ACTUAL VELOCITY	INDEX 0X606C	
FOLLOWING ERROR	INDEX 0X60F4	
POSITION LOOP GAINS	INDEX 0X60FB	
POSITION LOOP PROPORTIONAL GAIN	INDEX 0X60FB	SUB-INDEX 1
POSITION LOOP INTEGRAL GAIN	INDEX 0X60FB	SUB-INDEX 2
POSITION LOOP DERIVATIVE GAIN	INDEX 0X60FB	SUB-INDEX 3
POSITION LOOP INTEGRAL GAIN LIMIT	INDEX 0X60FB	SUB-INDEX 4
SOFTWARE POSITION LIMITS	INDEX 0X607D	
NEGATIVE SOFTWARE LIMIT POSITION	INDEX 0X607D	SUB-INDEX 1
POSITIVE SOFTWARE LIMIT POSITION	INDEX 0X607D	SUB-INDEX 2
SOFTWARE LIMITS MODE	INDEX 0X607D	SUB-INDEX 3

Object				Index	Sub-Index
POSITION DEMAND VALUE				0X6062	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	R0	encoder counts	--	YES	--
Description This is the motor position (in units of encoder counts) to which the amplifier is currently trying to move the axis. This value is updated every servo cycle based on the amplifier's internal trajectory generator. Units: encoder counts.					

Object				Index	Sub-Index
POSITION ACTUAL VALUE				0X6063	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts	--	YES	R
Description This is the actual motor position in units of encoder counts as calculated by the amplifier every servo cycle based on the state of the encoder input lines.					

Object				Index	Sub-Index
POSITION ACTUAL VALUE				0X6064	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts	--	YES	R
Description This object holds the same value as Position Actual Value object (index 0x6063).					

Object				Index	Sub-Index
FOLLOWING ERROR WINDOW				0X6065	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts	0 – 32767	YES	RF
Description This object holds allowable following error. If at any time following error exceeds this value for time more then specified by Following Error Time object (index 0x6066) amplifier will enter fault state.					

Object				Index	Sub-Index
FOLLOWING ERROR TIME				0X6066	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	ms	0 – 8000	YES	RF
Description This object holds allowable following error time i.e maximum time following error can set by object index 0X6065 before fault will be generated.					

Object				Index	Sub-Index
POSITION WINDOW				0X6067	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts	0 - 2,147,483,647	YES	RF
Description Size of the amplifier's tracking window. The "target reached" bit of the amplifier's status word is set when the amplifier is not running a trajectory, and the position error has been within the tracking window for the programmed tracking window time. Bit #5 in manufacturer status window is affected by this object as well. Bit #5 is set when current profile command is completed and motor actual position is within specified position window.					

Object				Index	Sub-Index
ACTUAL VELOCITY				0X6069	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RO	0.1 enc counts / sec	--	YES	--
Description Actual motor velocity in units of 0.1 encoder counts / second.					

Object				Index	Sub-Index
VELOCITY DEMAND VALUE				0X606B	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RO	0.1 enc counts / sec	--	YES	--
Description Velocity that the velocity loop is currently trying to attain. When the amplifier is running in homing or profile position mode, the velocity demand value is the output of the position loop, and the input to the velocity loop. AC Tech CANopen amplifiers support some modes in which the velocity demand is produced from a source other then the position loop. In these modes the demand velocity comes from the analog reference input, or the internal velocity preset memory location.					

Object				Index	Sub-Index
ACTUAL VELOCITY				0X606C	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	R0	0.1 enc counts / sec	--	YES	--
Description This object contains exactly the same information as object 0x6069.					

Object				Index	Sub-Index
FOLLOWING ERROR				0X60F4	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	R0	encoder counts	--	YES	--
Description This object gives the difference, in units of encoder counts, between the Position Actual Value object (index 0x6063) and the Position Demand Value object (index 0x60fc). This value is calculated as part of the position control loop. It is also the value that the various tracking windows are compared to. Refer to the Position Window object (index 0x6067, paragraph 7.2, page 56), and the Error Window object (index 0x6065, page 56).					

Object				Index	Sub-Index
POSITION LOOP GAINS				0X60FB	--
Type	Access	Units	Range	Map PDO	Memory
Record	RW	--	--	YES	--
Description This object contains the various gain values used to optimize the position control loop. Sub-index 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index
POSITION LOOP PROPORTIONAL GAIN				0X60FB	1
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 –32,767	YES	RF
Description This gain value is multiplied by the position loop error. The position loop error is the difference between the instantaneous commanded position and the actual motor position.					

Object				Index	Sub-Index
POSITION LOOP INTEGRAL GAIN				0X60FB	2
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 –16,383	YES	RF
Description This value is multiplied by position loop error over the time. The position loop error is the difference between the instantaneous commanded position and the actual motor position.					

Object				Index	Sub-Index
POSITION LOOP DERIVATIVE GAIN				0X60FB	3
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 –32,767	YES	RF
Description This value is multiplied by position loop error's difference. Error difference is taking by subtracting loop error value in last servo cycle and loop error value in present servo cycle.					

Object				Index	Sub-Index
POSITION LOOP INTEGRAL GAIN LIMIT				0X60FB	4
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 –2000	YES	RF
Description This value limits influence of the integral gain (object 0x60FB sub index 2.) term. This value internally scaled to 1/2000 meaning that 2000 would represent 100% of the term influence.					

Object				Index	Sub-Index
SOFTWARE POSITION LIMITS				0X607D	--
Type	Access	Units	Range	Map PDO	Memory
Array	RW	--	--	YES	--
Description This array holds the two software position limit values Negative Software Limit Position and Positive Software Limit Position. Sub-index 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index
NEGATIVE SOFTWARE LIMIT POSITION				0X607D	1
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts	--	YES	RF
Description The Software Position Limits array holds the two software position limit values: Negative Software Limit Position and Positive Software Limit Position. Sub-index 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index
POSITIVE SOFTWARE LIMIT POSITION				0X607D	2
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts	--	YES	RF
Description The Software Position Limits array holds the two software position limit values: Negative Software Limit Position and Positive Software Limit Position. Sub-index 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index
SOFTWARE LIMITS MODE				0X607D	3
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	--	0-2	YES	RF
Description Enables or Disables software position limits. Possible settings 0-disabled or 1-enabled(will disable and fault when hit), 2- enabled (will decelerate to stop and generate fault).					

7.3 Velocity Loop Configuration Objects

This section describes the objects used to configure the velocity control loop including:

VELOCITY LOOP MAXIMUM ACCELERATION	INDEX 0X244C	
VELOCITY LOOP MAXIMUM DECELERATION	INDEX 0X244D	
VELOCITY LIMITS ENABLED	INDEX 0X204B	
ANALOG INPUT VELOCITY SCALE	INDEX 0X2424	
PROGRAMMED VELOCITY	INDEX 0X248B	
VELOCITY LOOP GAINS	INDEX 0X60F9	
VELOCITY LOOP PROPORTIONAL GAIN	INDEX 0X60F9	SUB-INDEX 1
VELOCITY LOOP INTEGRAL GAIN	INDEX 0X60F9	SUB-INDEX 2
VELOCITY LOOP GAIN SCALER	INDEX 0X60F9	SUB-INDEX 3

Object				Index	Sub-Index
VELOCITY LOOP MAXIMUM ACCELERATION				0X244C	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	RPM*Sec	0.1 – 5,000,000	YES	RF
Description This acceleration value limits the maximum rate of change of the commanded velocity input to the velocity loop. This limit only applies when the absolute value of the velocity change is positive (i.e. the speed is increasing in either direction). Units are RPM*Sec. This value is only used if velocity demand is produced by a source other than the Position Loop. The value is applied if Accel/ Decel limiting is enabled by object 0x204B.					

Object				Index	Sub-Index
VELOCITY LOOP MAXIMUM DECELERATION				0X244D	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	RPM*Sec	0.1 – 5,000,000	YES	RF
Description This acceleration value limits the maximum rate of change of the commanded velocity input to the velocity loop. This limit only applies when the absolute value of the velocity change is negative (i.e. the speed is decreasing in either direction). Units are RPM*Sec. This value is only used if velocity demand is produced by a source other than the Position Loop. The value is applied if Accel/Decel limiting is enabled by object Velocity Limits Enabled (index 0x204B).					

Object				Index	Sub-Index
VELOCITY LIMITS ENABLED				0X204B	--
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 – 1	YES	RF
Description This object enables or disables acceleration and deceleration limiting for sources other than the position loop output. Possible values: 0 – limits disabled and 1 – limits enabled.					

Object				Index	Sub-Index
ANALOG INPUT VELOCITY SCALE				0X2424	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	RPM/V	-2000 to +2000	YES	--
Description When analog input #1 is used as velocity loop reference (set velocity) ,this object gives the scaling factor in RPM/V units.					

Object				Index	Sub-Index
PROGRAMMED VELOCITY				0X248B	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	RPS	-350 to +350	YES	RF
Description Gives the commanded velocity value when running in velocity follower mode (Operating mode= -2)					

Object				Index	Sub-Index
VELOCITY LOOP GAINS				0X60F9	--
Type	Access	Units	Range	Map PDO	Memory
Record	RW	--	--	YES	--
Description This object contains the various gain values used to optimize the velocity control loop. Sub-index 0 contains the number of sub-elements of this record.					

Object				Index	Sub-Index
VELOCITY LOOP PROPORTIONAL GAIN				0X60F9	1
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 – 32,767	YES	RF
Description This gain value is multiplied by the velocity error value. The velocity error is the difference between the desired and actual motor velocity.					

Object				Index	Sub-Index
VELOCITY LOOP INTEGRAL GAIN				0X60F9	2
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	0 – 32,767	YES	RF
Description This gain value is multiplied by the integral of the velocity loop error.					

Object				Index	Sub-Index
VELOCITY LOOP GAIN SCALER				0X60F9	3
Type	Access	Units	Range	Map PDO	Memory
Integer 16	RW	--	-16 to +4	YES	RF
Description A shift value that allows more range on the velocity loop gains when using a different encoders. The output of the velocity loop is multiplied by velocity loop gain scaler after the other gains have been used.					

7.4 Current Loop Configuration Objects

This section describes the objects used to configure the current control loop including:

USER PEAK CURRENT LIMIT16	INDEX 0X241F
USER PEAK CURRENT LIMIT8	INDEX 0X2420
USER CONTINUOUS CURRENT LIMIT	INDEX 0X241E
ACTUAL MOTOR CURRENT	INDEX 0X24BC
PROGRAMMED CURRENT	INDEX 0X248B
ANALOG INPUT CURRENT SCALE	INDEX 0X2423

Object				Index	Sub-Index
USER PEAK CURRENT LIMIT16				0X241F	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	Amps (RMS)	0 – 50	YES	RF
Description Specifies a peak current limit in phase Amps (RMS) when drive PWM carrier frequency is set to 16kHz					

Object				Index	Sub-Index
USER PEAK CURRENT LIMIT8				0X2420	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	Amps (RMS)	0 – 50	YES	RF
Description Specifies a peak current limit in phase Amps (RMS) when drive PWM carrier frequency is set to 8kHz.					

Object				Index	Sub-Index
USER CONTINUOUS CURRENT LIMIT				0X241E	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	Amps (RMS)	0 – 50	YES	RF
Description Specifies a continues current limit in phase Amps (RMS).					

Object				Index	Sub-Index
ACTUAL MOTOR CURRENT				0X24BC	--
Type	Access	Units	Range	Map PDO	Memory
Float	RO	Amps (RMS)	--	YES	--
Description Actual motor RMS per phase current.					

Object				Index	Sub-Index
PROGRAMMED CURRENT				0X248B	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	Amps (RMS)	-50 to +50	YES	RF
Description This object gives the programmed current value (0.01 Amps) used when running in current (torque) mode.					

Object				Index	Sub-Index
ANALOG INPUT CURRENT SCALE				0X2423	--
Type	Access	Units	Range	Map PDO	Memory
Float	RW	Amps (RMS)	-5 to +5	YES	--
Description When analog input #1 is used as a current loop reference (set current), this object gives the scaling factor in A/V units.					

8 Non Profiled Operating Modes

This chapter describes the operation of an amplifier in non-profiled modes such as velocity follower and current follower. Contents include:

- 8.1 Current Follower Mode
- 8.2 Velocity Follower Mode

8.1 Current Follower Mode

Current follower mode is set by setting the Mode of Operation object (index 0x6060, paragraph 6.2, page 35) to -1. In this mode the current loop input is disconnected from the velocity loop output. Reference for the current loop could be the value of the Programmed Current object (index 0x248B) or analog input #1. If the analog signal #1 is used for the current loop reference then the analog signal at the analog input #1 is multiplied by the Analog Input Current Scale object (index 0x2423) and then fed to the current loop summing junction (reference). The Reference Source object (index 0x2025) controls the reference source configuration.

8.2 Velocity Follower Mode

Velocity follower mode is set by setting object Mode of Operation (index 0x6060, paragraph 6.2, page 35) to -2. In this mode the velocity loop input is disconnected from the position loop output. Reference for the velocity loop could be the value of the Programmed Velocity object (index 0x248B) or analog input #1. If analog signal #1 is used for the velocity loop reference then the analog signal at the analog input #1 is multiplied by the Analog Input Velocity Scale object (index 0x2424) and then fed to the velocity reference limiter and then to the velocity loop summing junction. The Reference Source object (index 0x2025) controls the reference source configuration. The velocity reference limiter has 3 objects: Velocity Loop Maximum Acceleration (index 0x244C); Velocity Loop Maximum Deceleration (index 0x244D) and Velocity Limit Enabled (index 0x204B).

9 Homing Mode

9.1 Homing Mode Operation

This section describes control of the amplifier in homing mode. Contents of this section include:

- 9.1.1 Homing Overview
- 9.1.2 Homing Methods
- 9.1.3 Method 1: Homing on the Negative Limit Switch
- 9.1.4 Method 2: Homing on the Positive Limit Switch
- 9.1.5 Methods 3-4: Homing on the Positive Limit Switch and Index Pulse
- 9.1.6 Methods 5-6: Homing on the Negative Limit Switch and Index Pulse
- 9.1.7 Methods 7-14: Homing on the Home Switch and Index Pulse
- 9.1.8 Methods 15, 16, 20, 22, 24, 26, 28 & 30: Reserved
- 9.1.9 Methods 17-18: Homing without an Index Pulse
- 9.1.10 Methods 19, 21, 23, 25, 27 & 29: Homing without an Index Pulse
- 9.1.11 Methods 31-32: Reserved
- 9.1.12 Methods 33-34: Homing on the Index Pulse
- 9.1.13 Method 35: Homing on the Current Position

9.1.1 Homing Overview

What is Homing? Homing is the method by which a drive seeks the home position (also called the datum, reference point, or zero point). There are various methods of achieving this using:

- limit switches at the ends of travel, or
- a dedicated home switch.

Most of the methods also use the index pulse input from an incremental encoder.

The Homing Function

The homing function provides a set of trajectory parameters to the position loop, as shown in Figure 11. The parameters are generated by the homing function and are not directly accessible through CANopen dictionary objects. They include the profile mode and velocity, acceleration, and deceleration data.

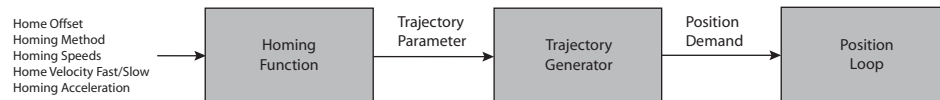


Figure 11: Homing Function

Initiating and Verifying a Homing Sequence

For the amplifier to operate in homing mode, the Mode of Operation object (index 0x6060, paragraph 6.2, page 35) should be set to 6. A homing move is started by setting bit 4 of the Control Word object (index 0x6040, paragraph 6.2, page 34). The results of a homing operation can be accessed in the Status Word (index 0x6041, paragraph 6.2, page 34).

Home Offset

The home offset is the difference between the zero position for the application and the machine home position (found during homing). During homing the home position is found and once the homing is completed the zero position is offset from the home position by adding the Home Offset to the home position. All subsequent absolute moves shall be taken relative to this new zero position. This is illustrated in Figure 12.

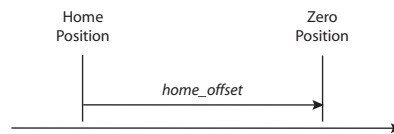


Figure 12: Home Offset

Homing Speeds

There are two homing speeds: fast and slow. The fast speed is used to find the home switch and the slow speed is used to find the index pulse. Refer to the Homing Speeds object (index 0x6099, paragraph 9.2, page 70)

Homing Acceleration

Homing Acceleration (index 0x609A) establishes the acceleration to be used for all accelerations and decelerations with the standard homing modes. Note that in homing, it is not possible to program a separate deceleration rate.

9.1.2 Homing Methods

There are several homing methods, each supported by objects described later in this chapter. Each method establishes the:

- Homing signal (positive limit switch, negative limit switch, home switch)
- Direction of actuation and, where appropriate, the position of the index pulse.

Legend to Homing Method Descriptions

Homing method descriptions and diagrams in this manual are based on those in the CANopen Profile for Drives and Motion Control (DSP 402). As highlighted in the example below, each homing method diagram shows the motor in the starting position on a mechanical stage. The arrow line indicates direction of motion, and the circled number indicates the homing method (the mode selected in the Homing Method object). The location of the circled method number indicates the home position reached with that method. Solid line stems on the index pulse line indicate index pulse locations. Longer dashed lines overlay these stems as a visual aid. Finally, the relevant limit switch is represented, showing the active and inactive zones and transition.

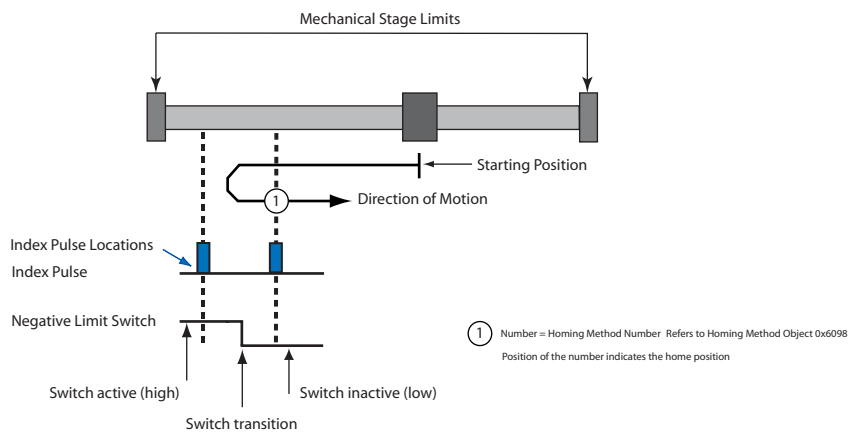


Figure 13: Homing Method Descriptions

Note that in the homing method descriptions, negative motion is leftward and positive motion is rightward.

9.1.3 Homing Method 1: Homing on the Negative Limit Switch

Using this method, the initial direction of movement is leftward if the negative limit switch is inactive (here shown as low). The home position is at the first index pulse to the right of the position where the negative limit switch becomes inactive.

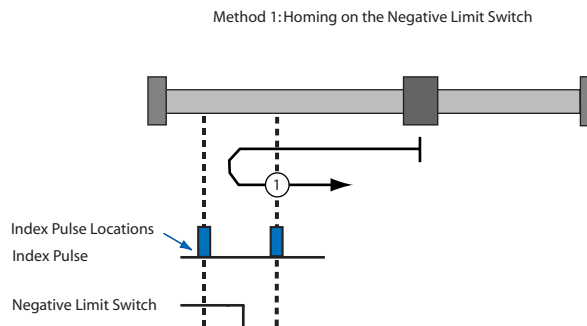


Figure 14: Homing on the Negative Limit Switch

9.1.4 Homing Method 2: Homing on the Positive Limit Switch

Using this method the initial direction of movement is rightward if the positive limit switch is inactive (here shown as low). The position of home is is at the first index pulse to the left of the position where the positive limit switch becomes inactive.

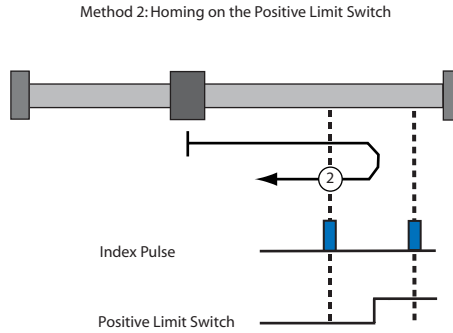


Figure 15: Homing on the Positive Limit Switch

9.1.5 Homing Method 3 and 4: Homing on the Positive Home Switch and Index Pulse

Using methods 3 or 4, the initial direction of movement depends on the state of the home switch. The home position is at the index pulse to either to the left or the right of the point where the home switch changes state. If the initial position is located so that the direction of movement must reverse during homing, the point at which the reversal takes place is anywhere after a change of state of the home switch.

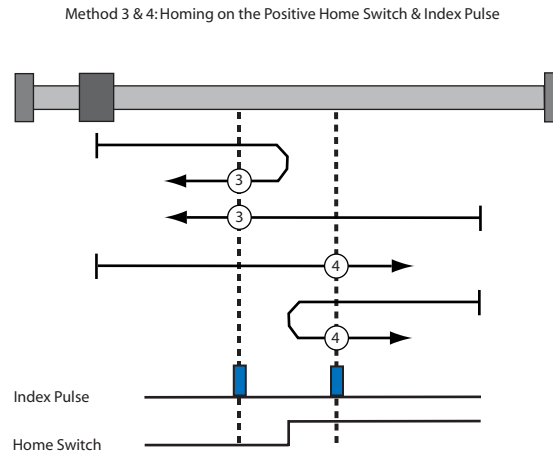


Figure 16: Homing on the Positive Home Switch & Index Pulse

9.1.6 Homing Methods 5 and 6: Homing on the Negative Home Switch and Index Pulse

Using methods 5 or 6, the initial direction of movement depends on the state of the home switch. The home position is at the index pulse to either to the left or the right of the point where the home switch changes state. If the initial position is located so that the direction of movement must reverse during homing, the point at which the reversal takes place is anywhere after a change of state of the home switch.

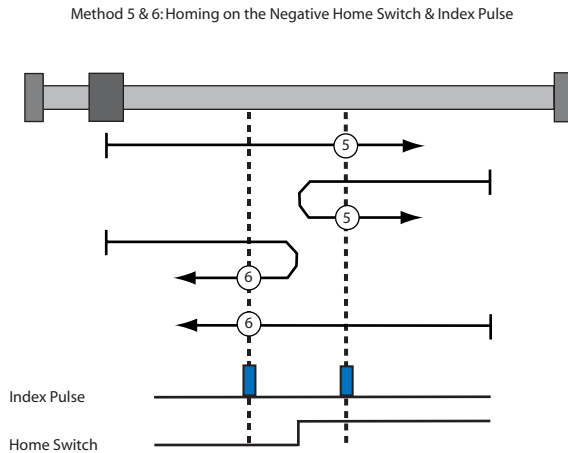


Figure 17: Homing on the Negative Home Switch & Index Pulse

9.1.7 Homing Methods 7-14: Homing on the Home Switch and Index Pulse

These methods use a home switch, which is active over only a portion of the travel. In effect, the switch has a momentary action as the axis sweeps past the switch.

Using methods 7 to 10, the initial direction of movement is to the right. Using methods 11 to 14 the initial direction of movement is to the left, unless the home switch is active at the start of the motion. In this case the initial direction of motion depends on the edge being sought. The home position is at the index pulse on either side of the rising or falling edges of the home switch, as shown in the following two diagrams. If the initial direction of movement leads away from the home switch, the drive must reverse on encountering the relevant limit switch.

Figure 18 illustrates a homing sequence on the home switch and index pulse with a positive initial move.

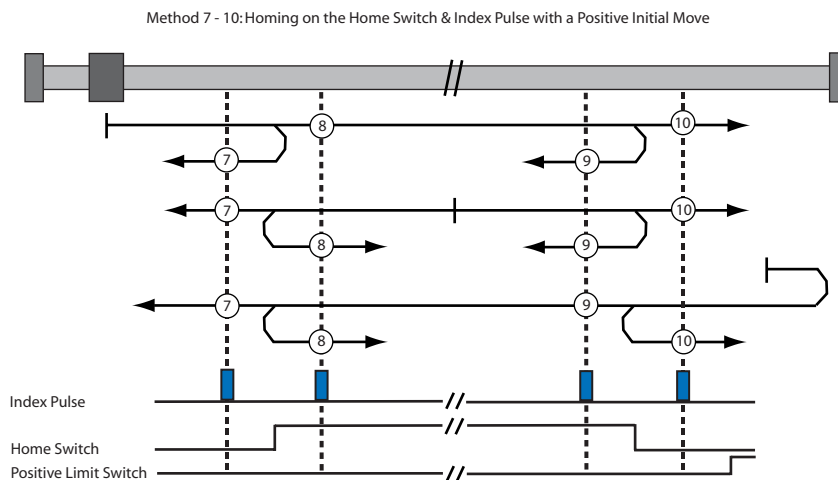


Figure 18: Homing on the Home Switch & Index Pulse w/ Positive Initial Move

Figure 19 illustrates a homing sequence on the home switch and index pulse with a negative initial move.

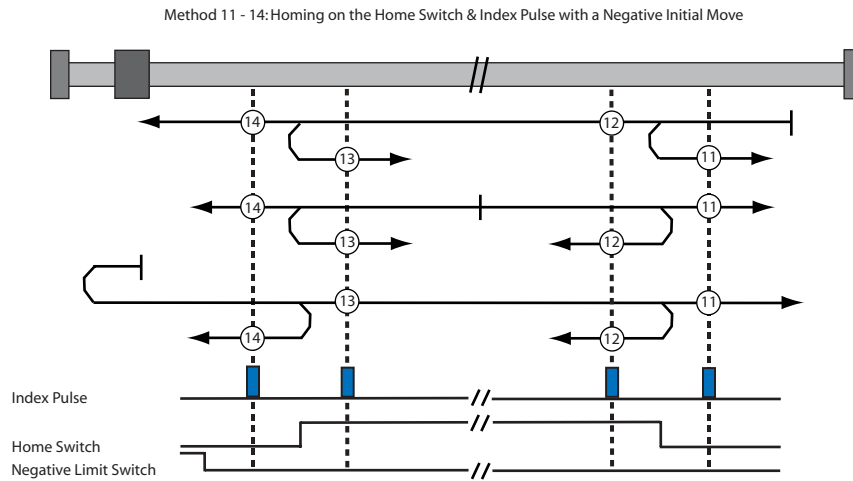


Figure 19: Homing on the Home Switch & Index Pulse w/ Negative Initial Move

9.1.8 Homing Methods 15, 16, 20, 22, 24, 26, 28, and 30: Reserved

Homing methods 15, 16, 20, 22, 24, 26, 28 and 30 are reserved for future use.

9.1.9 Homing Methods 17 and 18: Homing without an Index Pulse

These methods are similar to methods 1-2, except that the home position is not dependent on the index pulse but only on the relevant limit switch translation. Method 17 uses the negative limit switch, and method 18 uses the positive limit switch.

9.1.10 Homing Methods 19, 21, 23, 25, 27, and 29: Homing without an Index Pulse

These methods are similar to methods 1 to 14, except that the home position does not depend on the index pulse. Instead, it depends on the relevant home or limit switch transitions. For example, method 19 is similar to method 3 as shown in the following diagram.

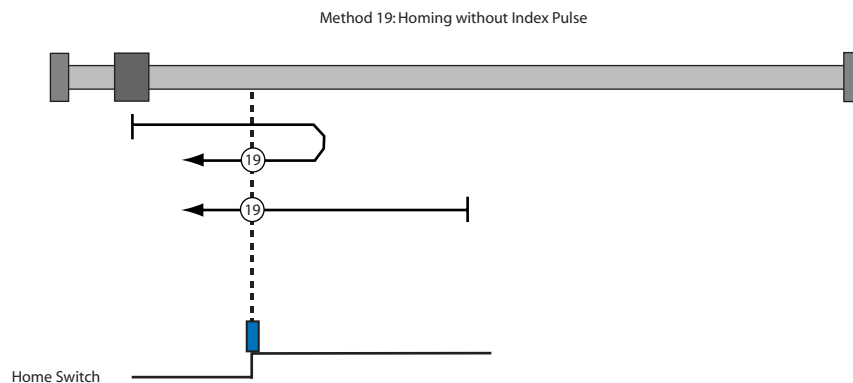


Figure 20: Homing without an Index Pulse

This means method 19 and 20 (as described in the Profile for Drives and Motion Control) both imply the same home algorithm and location, because methods 3 and 4 are only different in which index pulse the locate. Likewise, 22, 24, 26, 28, and 30 (as described in the Profile for Drives and Motion Control) are redundant. For this reason, in AC Tech amplifiers, the following redundant home methods are reserved: 20, 22, 24, 26, 28, and 30. The equivalent home method (one less than each of these values) should be used instead.

9.1.11 Homing Methods 31 and 32: Reserved

Homing methods 31 and 32 are reserved for future use.

9.1.12 Homing Methods 33 and 34: Homing on the Index Pulse

Using methods 33 or 34 the direction of homing is negative or positive respectively. The home position is at the index pulse found in the selected direction.

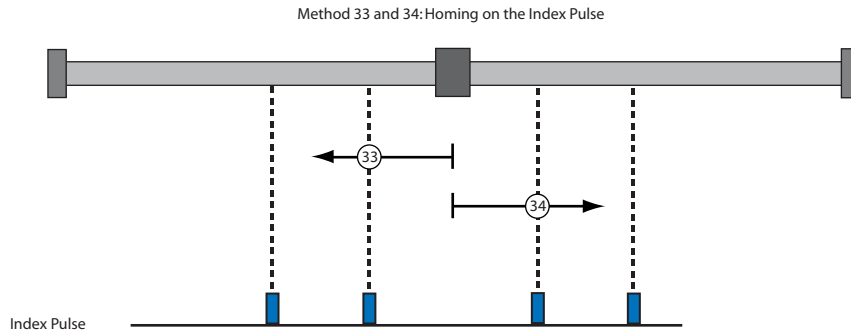


Figure 21: Homing on the Index Pulse

9.1.13 Homing Method 35: Homing on the Current Position

In homing method 35, the current position is the homing position.

9.2 Homing Mode Operation Objects

This section describes the objects that control the operation of the amplifier in homing mode. They include:

HOMING METHOD	INDEX 0X6098	
HOMING SPEEDS	INDEX 0X6099	
HOME VELOCITY – FAST	INDEX 0X6099	SUB-INDEX 1
HOME VELOCITY – SLOW	INDEX 0X6099	SUB-INDEX 2
HOMING ACCELERATION	INDEX 0X609A	
HOME OFFSET	INDEX 0X607C	

Object				Index	Sub-Index																																																																				
HOMING METHOD				0X6098	--																																																																				
Type	Access	Units	Range	Map PDO	Memory																																																																				
Integer 8	RW	--	Refer to Description	YES	RF																																																																				
<p>Description</p> <p>Defines the method to find the motor home position in homing mode. Supported methods:</p> <table> <thead> <tr> <th>Mode</th> <th>Home position</th> </tr> </thead> <tbody> <tr><td>0</td><td>The current position.</td></tr> <tr><td>1</td><td>The location of the first encoder index pulse on the positive side of the negative limit switch.</td></tr> <tr><td>2</td><td>The location of the first encoder index pulse on the negative side of the positive limit switch.</td></tr> <tr><td>3</td><td>The location of the first index pulse on the negative side of a positive home switch. A positive home switch is one that goes active at some position, and remains active for all positions greater than that one.</td></tr> <tr><td>4</td><td>The location of the first index pulse on the positive side of a positive home switch.</td></tr> <tr><td>5</td><td>The location of the first index pulse on the positive side of a negative home switch. A negative home switch is one that goes active at some position, and remains active for all positions less than that one.</td></tr> <tr><td>6</td><td>The location of the first index pulse on the negative side of a negative home switch.</td></tr> <tr><td>7</td><td>The location of the first index pulse on the negative side of the negative edge of an intermittent home switch. An intermittent home switch is one that is only active for a limited range of travel.</td></tr> <tr><td>8</td><td>The location of the first index pulse on the positive side of the negative edge of an intermittent home switch.</td></tr> <tr><td>9</td><td>The location of the first index pulse on the negative side of the positive edge of an intermittent home switch.</td></tr> <tr><td>10</td><td>The location of the first index pulse on the positive side of the positive edge of an intermittent home switch.</td></tr> <tr><td>11</td><td>The location of the first index pulse on the positive side of the positive edge of an intermittent home switch.</td></tr> <tr><td>12</td><td>The location of the first index pulse on the negative side of the positive edge of an intermittent home switch.</td></tr> <tr><td>13</td><td>The location of the first index pulse on the positive side of the negative edge of an intermittent home switch.</td></tr> <tr><td>14</td><td>The location of the first index pulse on the negative side of the negative edge of an intermittent home switch.</td></tr> <tr><td>15-16</td><td>Reserved for future use.</td></tr> <tr><td>17</td><td>The edge of a negative limit switch.</td></tr> <tr><td>18</td><td>The edge of a positive limit switch.</td></tr> <tr><td>19</td><td>The edge of a positive home switch.</td></tr> <tr><td>20</td><td>Reserved for future use.</td></tr> <tr><td>21</td><td>The edge of a negative home switch.</td></tr> <tr><td>22</td><td>Reserved for future use.</td></tr> <tr><td>23</td><td>The negative edge of an intermittent home switch.</td></tr> <tr><td>24</td><td>Reserved for future use.</td></tr> <tr><td>25</td><td>Positive edge of an intermittent home switch.</td></tr> <tr><td>26</td><td>Reserved for future use.</td></tr> <tr><td>27</td><td>The positive edge of an intermittent home switch.</td></tr> <tr><td>28</td><td>Reserved for future use.</td></tr> <tr><td>29</td><td>Negative edge of an intermittent home switch.</td></tr> <tr><td>30-32</td><td>Reserved for future use.</td></tr> <tr><td>33</td><td>The first index pulse on the negative side of the current position.</td></tr> <tr><td>34</td><td>The first index pulse on the positive side of the current position.</td></tr> <tr><td>35</td><td>Set current position to home and move to new zero position (including home offset). This is the same as mode 0 except that mode 0 does not do the final move to the home position.</td></tr> </tbody> </table> <p>Note that these homing methods only define the location of the home position. The zero position is always the home position adjusted by the homing offset. Refer to Homing Methods.</p>						Mode	Home position	0	The current position.	1	The location of the first encoder index pulse on the positive side of the negative limit switch.	2	The location of the first encoder index pulse on the negative side of the positive limit switch.	3	The location of the first index pulse on the negative side of a positive home switch. A positive home switch is one that goes active at some position, and remains active for all positions greater than that one.	4	The location of the first index pulse on the positive side of a positive home switch.	5	The location of the first index pulse on the positive side of a negative home switch. A negative home switch is one that goes active at some position, and remains active for all positions less than that one.	6	The location of the first index pulse on the negative side of a negative home switch.	7	The location of the first index pulse on the negative side of the negative edge of an intermittent home switch. An intermittent home switch is one that is only active for a limited range of travel.	8	The location of the first index pulse on the positive side of the negative edge of an intermittent home switch.	9	The location of the first index pulse on the negative side of the positive edge of an intermittent home switch.	10	The location of the first index pulse on the positive side of the positive edge of an intermittent home switch.	11	The location of the first index pulse on the positive side of the positive edge of an intermittent home switch.	12	The location of the first index pulse on the negative side of the positive edge of an intermittent home switch.	13	The location of the first index pulse on the positive side of the negative edge of an intermittent home switch.	14	The location of the first index pulse on the negative side of the negative edge of an intermittent home switch.	15-16	Reserved for future use.	17	The edge of a negative limit switch.	18	The edge of a positive limit switch.	19	The edge of a positive home switch.	20	Reserved for future use.	21	The edge of a negative home switch.	22	Reserved for future use.	23	The negative edge of an intermittent home switch.	24	Reserved for future use.	25	Positive edge of an intermittent home switch.	26	Reserved for future use.	27	The positive edge of an intermittent home switch.	28	Reserved for future use.	29	Negative edge of an intermittent home switch.	30-32	Reserved for future use.	33	The first index pulse on the negative side of the current position.	34	The first index pulse on the positive side of the current position.	35	Set current position to home and move to new zero position (including home offset). This is the same as mode 0 except that mode 0 does not do the final move to the home position.
Mode	Home position																																																																								
0	The current position.																																																																								
1	The location of the first encoder index pulse on the positive side of the negative limit switch.																																																																								
2	The location of the first encoder index pulse on the negative side of the positive limit switch.																																																																								
3	The location of the first index pulse on the negative side of a positive home switch. A positive home switch is one that goes active at some position, and remains active for all positions greater than that one.																																																																								
4	The location of the first index pulse on the positive side of a positive home switch.																																																																								
5	The location of the first index pulse on the positive side of a negative home switch. A negative home switch is one that goes active at some position, and remains active for all positions less than that one.																																																																								
6	The location of the first index pulse on the negative side of a negative home switch.																																																																								
7	The location of the first index pulse on the negative side of the negative edge of an intermittent home switch. An intermittent home switch is one that is only active for a limited range of travel.																																																																								
8	The location of the first index pulse on the positive side of the negative edge of an intermittent home switch.																																																																								
9	The location of the first index pulse on the negative side of the positive edge of an intermittent home switch.																																																																								
10	The location of the first index pulse on the positive side of the positive edge of an intermittent home switch.																																																																								
11	The location of the first index pulse on the positive side of the positive edge of an intermittent home switch.																																																																								
12	The location of the first index pulse on the negative side of the positive edge of an intermittent home switch.																																																																								
13	The location of the first index pulse on the positive side of the negative edge of an intermittent home switch.																																																																								
14	The location of the first index pulse on the negative side of the negative edge of an intermittent home switch.																																																																								
15-16	Reserved for future use.																																																																								
17	The edge of a negative limit switch.																																																																								
18	The edge of a positive limit switch.																																																																								
19	The edge of a positive home switch.																																																																								
20	Reserved for future use.																																																																								
21	The edge of a negative home switch.																																																																								
22	Reserved for future use.																																																																								
23	The negative edge of an intermittent home switch.																																																																								
24	Reserved for future use.																																																																								
25	Positive edge of an intermittent home switch.																																																																								
26	Reserved for future use.																																																																								
27	The positive edge of an intermittent home switch.																																																																								
28	Reserved for future use.																																																																								
29	Negative edge of an intermittent home switch.																																																																								
30-32	Reserved for future use.																																																																								
33	The first index pulse on the negative side of the current position.																																																																								
34	The first index pulse on the positive side of the current position.																																																																								
35	Set current position to home and move to new zero position (including home offset). This is the same as mode 0 except that mode 0 does not do the final move to the home position.																																																																								

Object				Index	Sub-Index
HOMING SPEEDS				0X6099	--
Type	Access	Units	Range	Map PDO	Memory
Array	RW	--	--	YES	--
Description This array holds the two velocities used when homing. Sub-index 0 contains the number of subelements of this record.					

Object				Index	Sub-Index
HOME VELOCITY – FAST				0X6099	1
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	0.1 encoder cnts/sec	0 – 500,000,000	YES	RF
Description This velocity value is used during segments of the homing procedure that may be handled at high speed. Generally, this means move in which the home sensor is being located, but the edge of the sensor is not being found. Units are 0.1 encoder counts / second.					

Object				Index	Sub-Index
HOME VELOCITY – SLOW				0X6099	2
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	0.1 encoder cnts/sec	0 – 500,000,000	YES	RF
Description This velocity value is used for homing segment that require low speed such as cases where the edge of a homing sensor is being sought. Units are 0.1 encoder counts / second.					

Object				Index	Sub-Index
HOMING ACCELERATION				0X609A	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	10 encoder cnts/sec ²	0 – 200,000,000	YES	RF
Description This value defines the acceleration used for all homing moves. It is specified in units of 10 encoder counts / second ² . The same acceleration value is used at the beginning and ending of moves (i.e. there is no separate deceleration value).					

Object				Index	Sub-Index
HOME OFFSET				0X607C	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts	--	YES	RF
Description This offset value (in units of encoder counts) is used in homing mode as an offset between the home sensor position and the zero position. Refer to Home Offset for more information.					

10 Profile Position and Profile Velocity Mode Operation

This chapter describes the operation of an amplifier in profile position and profile velocity modes. Contents include:

- 10.1 Profile Position Mode Operation
- 10.2 Profile Velocity Mode Operation

10.1 Profile Position Mode Operation Overview

This section provides an overview of profile position mode operation. Contents of this section include:

- 10.1.1 Point-to-Point Motion Profiles
- 10.1.2 Handling a Series of Point-to-Point Moves
- 10.1.3 Point-to-Point Move Parameters and Related Data
- 10.1.4 Point-to-Point Move Sequence Examples

10.1.1 Point-to-Point Motion Profiles

In profile position mode, an amplifier receives set points from the trajectory generator to define a target position and moves the axis to that position at a specified velocity and acceleration. This is known as a point-to-point move. For the amplifier to operate in profile position mode, the Mode of Operation object (index 0x6060, paragraph 6.2, page 35) should be set to 0x0001. AC tech amplifiers also support special multi-segment type of moves where target position and velocities can be defined as set allowing complicated segment motion profiles.

Trapezoidal and S-curve Motion Profiles

In a point-to-point move, the rate of change in acceleration is known as jerk. Some applications can tolerate jerk, whereas in others, high rates of jerk can cause excessive mechanical wear or material damage. To support systems with varying levels of jerk tolerance, the profile position mode supports two motion profiles: the trapezoidal profile, which has unlimited jerk, and the jerklimited S-curve profile.

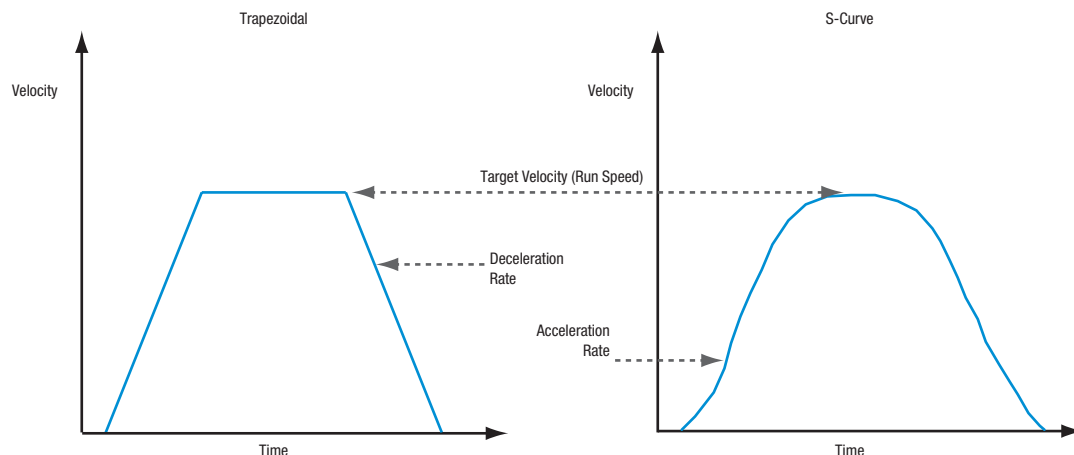


Figure 22: Trapezoidal & S-Curve Motion Profiles

In a trapezoidal profile, jerk is unlimited at the corners of the profile (start of the move, target velocity, start of deceleration, and end of the move). S-curve profiling limits jerk or “smoothes” the motion. Note that an S-curve profile move does support an independent deceleration rate. The Motion Profile Type object (index 0x6086) controls which type of profile is used.

Relative vs. Absolute Moves

In a relative move, the target position is added to the instantaneous commanded position, and the result is the destination of the move. In an absolute move, the target position is offset from the home position. The instantaneous commanded position (called the demand position in the CANopen specification) is the output of the trajectory generator. During the course of the profile this position changes constantly. It is possible to update the profile's target position while the move is in progress. If this update is performed as a relative move, then the target position value will be added to the instantaneous commanded position at the time the update is received. This type of update is most useful when the motor needs to be moved a set distance beyond the point where some asynchronous event occurs.

10.1.2 Handling a Series of Point-to-Point Moves

There are two methods for handling a series of point-to-point moves:

- As a series of discrete profiles (supported in both trapezoidal and S-curve profile moves)
- As one continuous profile (supported in both trapezoidal and S-curve profile moves)

General descriptions of the two methods follow. Detailed procedures appear later in the chapter.

A Series of Discrete Profiles

The simplest way to handle a series of point-to-point moves is to start a move to a particular position, wait for the move to finish, and then start the next move. As shown in Figure 23, each move is discrete. The motor accelerates, runs at target velocity, and then decelerates to zero before the next move begins.

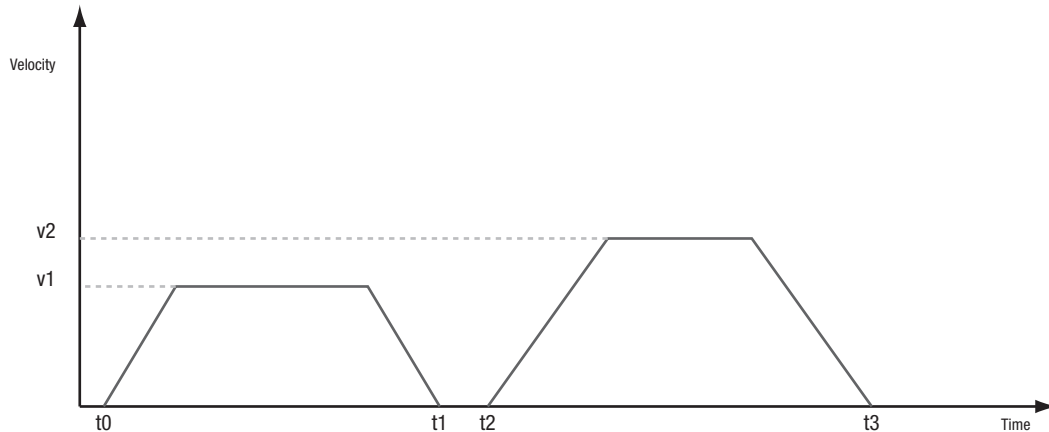


Figure 23: Discrete Profiles

The Profile for Drives and Motion Control refers to this method as the “single setpoint” method. AC Tech CANopen amplifiers allow use of this method with both trapezoidal and S-curve profile moves.

One Continuous Profile (segmented moves)

Alternately, a series of trapezoidal or S-curved profile moves can be treated as a continuous move. AC Tech amplifiers use special technique called Segmented Moves. Each segment of the complex move consists of Target position and Target velocity. As shown in Figure 24, the motor does not stop between moves. Instead, motion profile defined as set of Target Position – Profile Velocity pairs. If this method is used then profile acceleration and deceleration are not used and calculated automatically by amplifier. Their values are calculated based on segment start – segment stop velocities and segment length in position units. This method gives user greater flexibility rather than change point on the fly method since whole profile can be described before motion is started. At the same time loading the next value of the target position and target velocity could be done on the fly. This is particularly useful in systems where calculation by the motion master must also be done on the fly. The motion buffer for the segmented move is 32 entries deep.

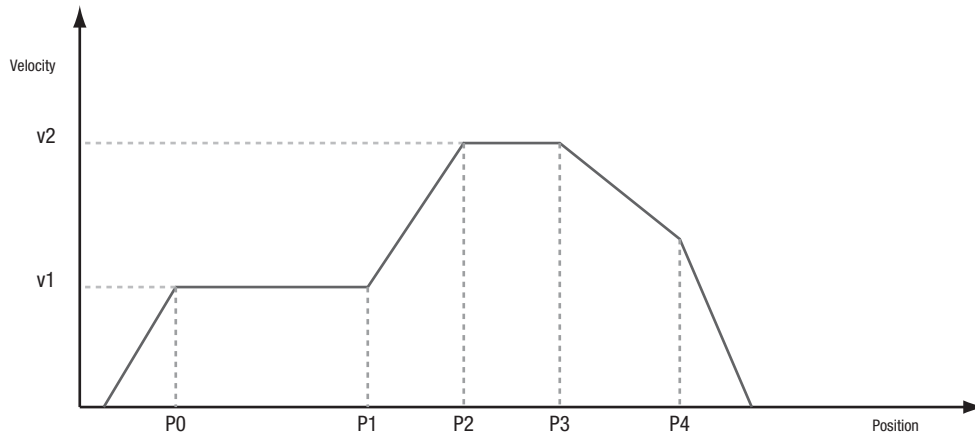


Figure 24: One Continuous Profile

The Profile for Drives and Motion Control refers to this method as the “set of setpoints” method. AC Tech CANopen amplifiers allow use of this method with trapezoidal or S-curve types of moves.

10.1.3 Point-to-Point Move Parameters and Related Data

Move Parameters

Each point-to-point move is controlled by a set of parameters, accessed through the following objects.

Object Name	Object ID	Description
Target Position	0x607A	When running in position profile mode, this object holds the destination position of the trajectory generator.
Profile Velocity	0x6081	Velocity that the trajectory generator will attempt to achieve when running in position profile mode.
Profile Acceleration	0x6083	Acceleration that the trajectory generator attempts to achieve when running in position profile mode.
Profile Deceleration	0x6084	Note that an S-curve profile move does not use a deceleration rate. Instead, the acceleration rate is applied to both the acceleration and deceleration of the move.
Quick Stop Deceleration	0x6085	Deceleration value used when a trajectory needs to be stopped as the result of a quick stop command. Note that unlike most trajectory configuration values, this value is NOT buffered. Therefore, if the value of this object is updated during an abort, the new value is used immediately.
Motion Profile Type	0x6086	Trapezoidal or S-Curve

The Point-to-Point Move Buffer

In profile position mode, the amplifier uses a buffer to store the parameters (listed in the Move Parameters table) for the next point-to-point move, or for next move segment consisting of the Target position-Profile velocity pair. The move buffer can be modified at any point before a control sequence copies the “next-move” parameters to the active move registers.

Move-Related Control Word and Status Word Bit Settings

An amplifier's Control Word (index 0x6040) and Status Word (index 0x6041) play an important role in the initiation and control of point-to-point move sequences, as described herein.

Object Name	Object ID	Bit#	Bit Name	Description
Control Word	0x6040	4	New Setpoint	The transition of bit 4 from 0 to 1 is what causes the amplifier to copy a set of move parameters from the buffer to the active register, thus starting the next move.
		5	Motion Suspended	Allows or not start motion on change bit #4. If this bit is set then transition of bit #4 copies motion parameters to motion buffer but doesn't start the motion until this bit is cleared. If there is an entries in motion buffer clearing this bit will start the motion.
		6	Absolute/ Relative	Value = 0: Move is absolute (based on home position). Value = 1: Move is relative (based on current commanded position).
		8	Halt	Value = 1: Interrupts the motion of the drive. Wait for release to continue.
Status Word	0x6041	10	Target Reached	Amplifier sets bit 10 to 1 when target position has been reached. Amplifier clears bit 10 to zero when new target is received.
		12	Setpoint Acknowledged	Set by the amplifier when Control Word bit 4 goes from 0 to 1. Cleared when Control Word bit 4 is cleared if there is more room in motion buffer (buffer depth is 32 entry). If there is no room in the buffer (buffer full) bit will stay set until buffer has at least one vacant entry.

Refer to paragraph 6.2, page 34 for more Control Word (index 0x6040) and Status Word (index 0x6041) information.

10.1.4 Point-To-Point Move Sequence Examples

Figures 25 and 26 illustrate how to perform:

- A series of moves treated as a Series of Discrete Profiles
- A series of trapezoidal or S-curve position multi-segment moves treated as One Continuous Profile

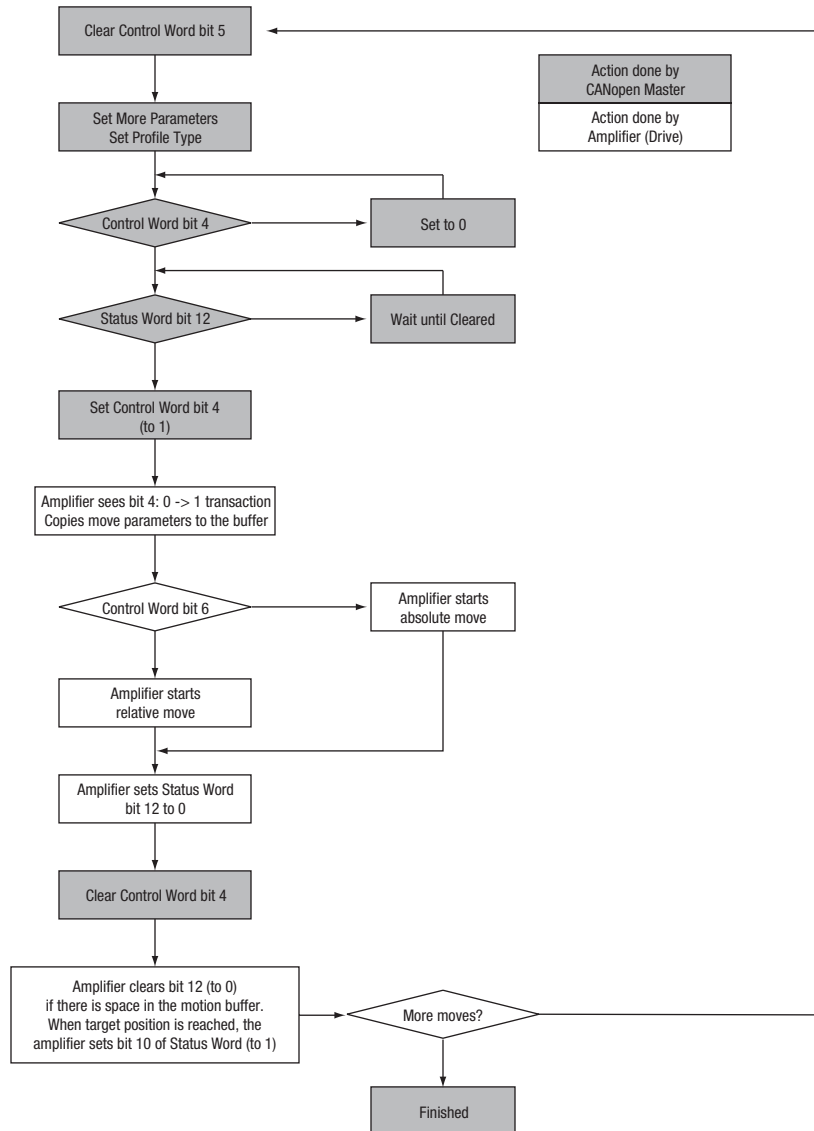


Figure 25: A series of moves treated as a Series of Discrete Profiles

Notes:

1. This type of move is supported as a trapezoidal profile or S-curve profile.
2. Control Word bit 4 is "new setpoint." It needs to be 0 because the move will be triggered by a 0->1 transition.
3. Bit 4, value of 1 indicates that valid data has been sent to amplifier and new move should begin. Bit 5 is "change set immediately." A value of 1 tells the amplifier to update the current profile immediately by copying the contents of the move buffer to the active registers (without waiting for move to finish).
4. Amplifier must detect bit 4 0-1 transition to begin move. Bit 5 value 1 allows immediate update.
5. Control word bit 6: value 0 causes absolute move; value 1 causes relative move.
6. Status Word bit 12 is "setpoint acknowledge." A value of 1 indicates the amplifier has received a setpoint and has started the move.
7. Control Word bit 4 is "new setpoint." It needs to be 0 to allow the next move will be triggered by a 0->1 transition. Also, the 1->0 transition causes the amplifier to clear bit 12.
8. Amplifier detects 0->1 transition of Control Word bit 4 and clears bit 12 in response. When the motor reaches the target position, the amplifier sets Status Word bit 10 ("target reached") to 1.

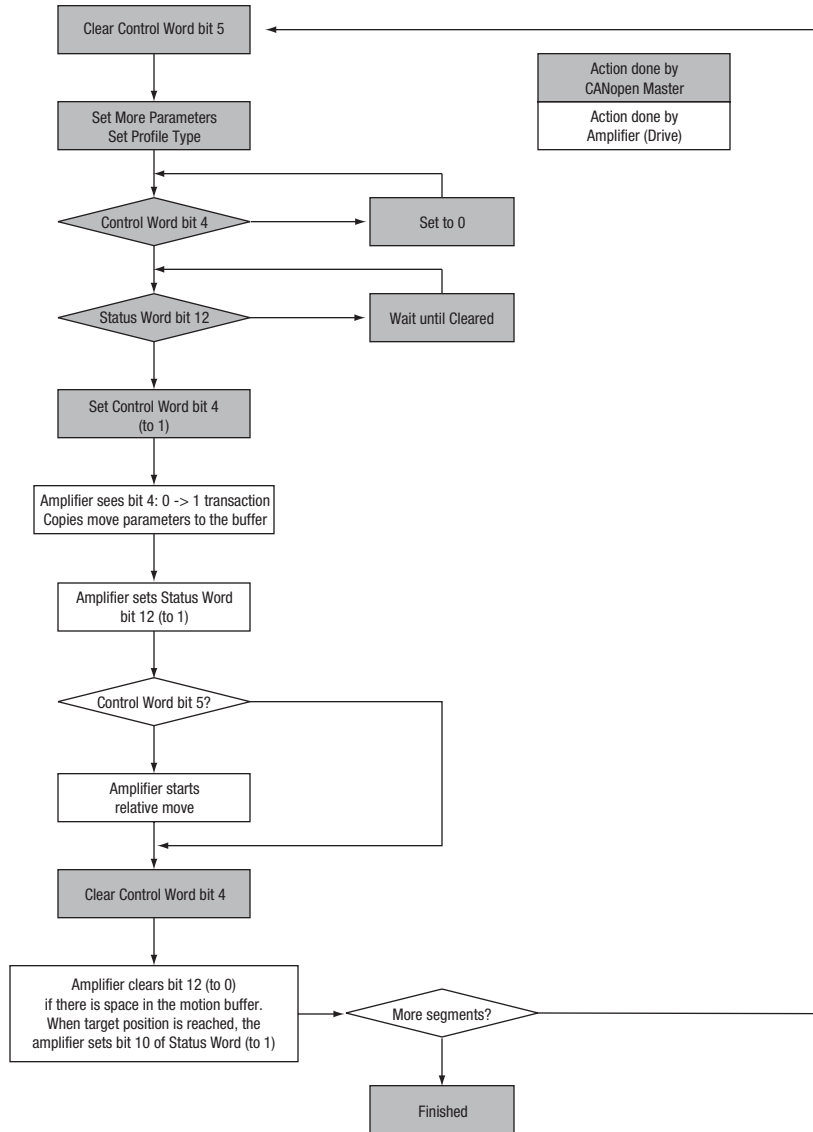


Figure 26: A series of trapezoidal or S-curve position multi-segment moves treated as One Continuous Profile.

10.2 Profile Velocity Mode Operation

10.2.1 Position and Velocity Loops

In profile velocity mode, both the velocity and position loops are used. Profile velocity moves are controlled by some of the same gains and limits objects used in profile position mode. In addition, a Target Velocity object (index 0x60FF) provides the target velocity. For the amplifier to operate in profile velocity mode, the Mode of Operation object (index 0x6060, paragraph 6.2, page 35) should be set to 0x0003.

Controlling motion in Profile Velocity Mode

In profile velocity mode, motion is started by giving a non-zero value to the Target Velocity (index 0x60FF). Motion is stopped by setting this object to zero. In profile velocity mode, the target velocity is updated as soon as the Target Velocity object (index 0x60FF) is set. In this mode, Control Word bits 4, 5, and 6 are not used. To start a move in profile velocity mode, set the profile parameters (profile accel, profile decel, and target velocity).

10.3 Profile Position, Profile Velocity Mode Objects.

This section describes the objects that control the operation of the amplifier in profile position mode. They include:

TARGET POSITION	INDEX 0X607A
PROFILE VELOCITY	INDEX 0X6081
TARGET VELOCITY	INDEX 0X60FF
PROFILE ACCELERATION	INDEX 0X6083
PROFILE DECELERATION	INDEX 0X6084
QUICK STOP DECELERATION	INDEX 0X6085
MOTION PROFILE TYPE	INDEX 0X6086

Object				Index	Sub-Index
TARGET POSITION				0X607A	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	encoder counts	--	YES	RF
<p>Description</p> <p>When running in position profile mode, this object holds the destination position of the trajectory generator. The units of this object are fixed at encoder counts.</p> <p>Note that the target position programmed here is not passed to the internal trajectory generator until the move has been started or updated using the Control Word. Refer to paragraph 10.1, page 71 "Profile Position Mode Operation Overview" for more information.</p>					

Object				Index	Sub-Index
PROFILE VELOCITY				0X6081	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	0.1 encoder cnts/sec	0 – 500,000,000	YES	RF
<p>Description</p> <p>Velocity that the trajectory generator attempts to achieve when running in position profile mode.</p> <p>Note that the value programmed here is not passed to the internal trajectory generator until the move has been started or updated using the Control Word. Refer to paragraph 10.1, page 71 "Profile Position Mode Operation Overview" for more information.</p>					

Object				Index	Sub-Index
TARGET VELOCITY				0X60FF	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	0.1 encoder cnts/sec	-500,000,000 to +500,000,000	YES	R
<p>Description</p> <p>In profile velocity mode, this object is an input to the amplifier's internal trajectory generator. Any change to the target velocity triggers an immediate update to the trajectory generator.</p> <p>Note that this is different from the way the profile position works. In that mode, changing the trajectory input parameters doesn't affect the trajectory generator until bit 4 of the Control Word object (index 0x6040) has been changed from 0 to 1.</p>					

Object				Index	Sub-Index
PROFILE ACCELERATION				0X6083	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	10 encoder cnts/sec ²	0 -200,000,000	YES	RF
<p>Description</p> <p>In profile position mode, this value (specified in units of 10 encoder counts / second²) is the acceleration that the trajectory generator attempts to achieve.</p> <p>Note that the value programmed here is not passed to the internal trajectory generator until the move has been started or updated using the Control Word. Refer to paragraph 10.1, page 71 "Profile Position Mode Operation Overview" for more information.</p>					

Object				Index	Sub-Index
PROFILE ACCELERATION				0X6083	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	10 encoder cnts/sec ²	0 -200,000,000	YES	RF
<p>Description</p> <p>In profile position mode, this value (specified in units of 10 encoder counts / second²) is the acceleration that the trajectory generator attempts to achieve.</p> <p>Note that the value programmed here is not passed to the internal trajectory generator until the move has been started or updated using the Control Word. Refer to paragraph 10.1, page 71 "Profile Position Mode Operation Overview" for more information.</p>					

Object				Index	Sub-Index
PROFILE DECELERATION				0X6084	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	10 encoder cnts/sec ²	0 -200,000,000	YES	RF
<p>Description</p> <p>This value (specified in units of 10 encoder counts / second²) is the deceleration that the trajectory generator attempts to achieve at the end of a trapezoidal profile when running in position profile mode. Note that this value is only used when running trapezoidal or profile position special velocity mode profiles. The S-curve profile generator uses the Profile Acceleration object (index 0x6083) as the acceleration target for both the start and end of moves.</p> <p>Note that the value programmed here is not passed to the internal trajectory generator until the move has been started or updated using the Control Word. Refer to paragraph 10.1, page 71 "Profile Position Mode Operation Overview" for more information.</p>					

Object				Index	Sub-Index
QUICK STOP DECELERATION				0X6085	--
Type	Access	Units	Range	Map PDO	Memory
Integer 32	RW	10 encoder cnts/sec ²	0 -200,000,000	YES	RF
<p>Description</p> <p>Deceleration value used when a trajectory needs to be stopped as the result of a quick stop command. When a quick stop command is issued, the demand velocity is decreased by this value until it reaches zero. This occurs in all position modes (homing, profile position, and interpolated position modes), and for all trajectory generators (trapezoidal and s-curve).</p> <p>Note that unlike most trajectory configuration values, this value is NOT buffered. Therefore, if the value of this object is updated during an abort, the new value is used immediately.</p>					

Object				Index	Sub-Index										
MOTION PROFILE TYPE				0X6086	--										
Type	Access	Units	Range	Map PDO	Memory										
Integer 16	RW	--	Refer to Description	YES	RF										
<p>Description</p> <p>Type of trajectory profile to use when running in profile position mode. Supported values are:</p> <table border="1"> <thead> <tr> <th>Mode</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Trapezoidal profile mode.</td> </tr> <tr> <td>3</td> <td>S-curve profile mode.</td> </tr> <tr> <td>-1</td> <td>Trapezoidal multi-segmented move</td> </tr> <tr> <td>-2</td> <td>S-curve multi-segmented move</td> </tr> </tbody> </table> <p>The amplifier will not accept other values. See Profile Position Mode Operation Overview, for more information.</p> <p>Note that the value programmed here is not passed to the internal trajectory generator until the move has been started or updated using the Control Word bit #4. Refer to paragraph 10.1, page 71 "Profile Position Mode Operation Overview" for more information.</p>						Mode	Description	0	Trapezoidal profile mode.	3	S-curve profile mode.	-1	Trapezoidal multi-segmented move	-2	S-curve multi-segmented move
Mode	Description														
0	Trapezoidal profile mode.														
3	S-curve profile mode.														
-1	Trapezoidal multi-segmented move														
-2	S-curve multi-segmented move														



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



MotionView[®]
OnBoard

PositionServo CANopen Communication Module Communications Interface Reference Guide

About These Instructions

This documentation applies to the optional CANopen communications module for the PositionServo drive and should be used in conjunction with the PositionServo User Manual (Document S94PM01) that shipped with the drive. These documents should be read in their entirety as they contain important technical data and describe the installation and operation of the drive.

© 2008 Lenze AC Tech Corporation

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from Lenze AC Tech Corporation. The information and technical data in this manual are subject to change without notice. Lenze AC Tech Corporation makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. Lenze AC Tech Corporation assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

CANopen® and CiA® are registered trademarks of the CiA (CAN in Automation e.V.).

PositionServo™ is a registered trademark of Lenze AC Tech Corporation.



1	Safety Information.....	1
1.1	Warnings, Cautions and Notes.....	1
1.1.1	General	1
1.1.2	Application	1
1.1.3	Installation.....	1
1.1.4	Electrical Connection	2
1.1.5	Operation	2
2	Introduction.....	3
2.1	Fieldbus Overview	3
2.2	Module Specification	3
2.3	Module Identification Label.....	3
3	Installation	4
3.3	Electrical Installation	5
3.3.1	Cable Types.....	5
3.3.2	Network Limitations.....	5
3.3.3	Connections and Shielding	6
3.3.4	Network Termination	7
3.3.5	Network Schematic	7
4	Commissioning	8
4.1	Overview.....	8
4.2	Configuring the Network.....	8
4.2.1	Master Support Files.....	8
4.2.2	CANopen Master Setup Procedure	8
4.3	Configuring the PositionServo CANopen Module	9
4.3.1	Connecting.....	9
4.3.2	Connect to the Drive with MotionView OnBoard	9
4.3.3	Communication Module Selection.....	10
4.3.4	CANopen Node Settings.....	10
4.3.5	Node Address	11
4.3.6	Baud / Data Rate	11
4.3.7	CAN Bootup Mode	12
4.3.8	CAN Bootup Delay	12
4.3.9	Data Mapping.....	13
4.3.10	Re-Initialising	13
4.3.11	Non-Module Parameter Settings.....	13



Contents

5.	CANopen Object Dictionary	14
5.1	What is the CANopen Object Dictionary?.....	14
5.1.1	Object Format.....	14
5.1.2	Object Dictionary Layout.....	14
5.1.3	Accessing the Object Dictionary.....	15
5.2	Communication Profile Area	15
5.2.1	Device Type.....	16
5.2.2	Error Register	16
5.2.3	Pre-defined Error Field.....	16
5.2.4	SYNC COB ID.....	17
5.2.5	Manufacture Device Name.....	17
5.2.6	Manufacture Hardware Version.....	17
5.2.7	Manufacture Software Version.....	17
5.2.8	Emergency Message COB ID	17
5.2.9	Inhibit Time EMCY	18
5.2.10	Producer Heartbeat Time	18
5.2.11	Identity Object	18
5.2.12	RxPDO 1 to 8 Communication Parameters	19
5.2.13	RxPDO Mapping Parameters.....	20
5.2.14	TxPDO 1 to 8 Communication Parameters	20
5.2.15	TxPDO Mapping Parameters.....	21
5.3	Manufacture Specific Profile Area.....	22
5.3.1	Data Format, Size and Memory Area.....	22
5.4	Endian Format.....	22
5.5	Object Access	22
6.	Service Data Objects.....	23
6.1	What are Service Data Objects?	23
6.2	SDO Message Identifiers	23
6.3	PID Access.....	23
6.4	SDO Abort Codes.....	23
6.5	SDO Message Frame.....	24
6.5.1	Specifier.....	25
6.5.2	Multiplexor	25
6.5.3	Data	25
6.6	SDO Access Examples.....	26
6.6.1	Example 1: Read Velocity Accel Limit.....	26
6.6.2	Example 2: Write to Velocity Accel Limit	26
6.6.3	Example 3: Read User Variable V0 Least Significant Byte (LSB).....	27
6.6.4	Example 4: Write to User Variable V0 Least Significant Byte (LSB).....	27
6.6.5	Example 5: Read APOS.....	28
6.6.6	Example 6: Write to APOS.....	28



7	Process Data Objects	29
7.1	What are Process Data Objects?.....	29
7.2	PDO Configuration in MotionView	29
7.2.1	COB ID and Mode	29
7.2.2	Transmission Type	30
7.2.3	Event Time	31
7.2.4	Inhibit Time	31
7.3	Mapping PDOs	32
7.3.1	Amount and Size of PDOs	32
7.3.2	Receive (Rx) PDOs	32
7.3.3	Transmit (Tx) PDOs.....	33
8	Emergency Objects.....	34
8.1	What is an Emergency Object?.....	34
8.2	Emergency Object format.....	34
8.2.1	Error Code	34
8.2.2	Error Register	35
8.2.3	Manufacture Specific Error Field.....	35
8.3	Emergency Object Examples	36
8.3.1	Example 1: Hardware Disable	36
8.3.2	Example 2: Limit Switch	37
8.3.3	Example 3: CAN Receive buffer overrun.....	37
9	Drive Control and Status	38
9.1	Overview.....	38
9.2	Control BITS	38
9.2.1	Software Enable/Disable.....	38
9.2.2	Drive Reset (Cold Boot)	38
9.2.3	Suspend Motion	38
9.2.4	Stop Motion.....	39
9.3	Status Word	39
9.3.1	Status Flags Register	39
9.3.2	Extended Status Bits	40
10	Advanced Features	41
10.1	CAN Baud Rate.....	41
10.2	CAN Node Address	41
10.3	CAN Boot-up Mode.....	41
10.4	CAN Boot-up Delay.....	42
10.5	Communication Module Selection.....	42
10.6	PDO Configuration	43
10.6.1	PDO COB-ID, Activation and Transmission Type.....	43
10.6.2	PDO Mapping	45
10.6.3	TPDO Event Time and Inhibit Time.....	45
11	Reference	46
11.1	PID List with CANopen Values.....	46



1 Safety Information

1.1 Warnings, Cautions and Notes

1.1.1 General

Some parts of Lenze controllers (frequency inverters, servo inverters, DC controllers) can be live, moving and rotating. 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.

1.1.2 Application

Drive controllers are components 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 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 drive as directed) is only allowed when there is compliance to 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. In the case of radio interference, special measures may be necessary for drive controllers.

1.1.3 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! When installing the drive ensure optimal airflow by observing all clearance distances in the drive's user manual. Do not expose the drive to excessive: vibration, temperature, humidity, sunlight, dust, pollutants, corrosive chemicals or other hazardous environments.



Safety Information

1.1.4 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 in accordance with the appropriate regulations (e.g. cable cross-sections, fuses, PE connection). Additional information can be obtained from the regulatory documentation.

The regulatory 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.

1.1.5 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, do not touch the live components and power connection until the capacitors have discharged. Please observe the corresponding notes on the controller.
- Do not continuously cycle input power to the controller more than once every three minutes.
- Close all protective covers and doors during operation.



WARNING!

Network control permits automatic starting and stopping 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.

Table 1: Pictographs used in these instructions

Pictograph	Signal word	Meaning	Consequences 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 for persons	Death or injury
	STOP!	Possible damage to equipment	Damage to drive system or its surroundings
	NOTE	Useful tip: If observed, it will make using the drive easier	



2 Introduction

The following information is provided to explain how the PositionServo drive operates on a CANopen network; it is not intended to explain how CANopen itself works. Therefore, a working knowledge of CANopen is assumed, as well as familiarity with the operation of the PositionServo drive.

2.1 Fieldbus Overview

The CANopen fieldbus is an internationally accepted communications protocol designed for commercial and industrial installations of factory automation and motion control applications. High data transfer rates combined with its efficient data formatting, permit the coordination and control of multi-node applications.

2.2 Module Specification

- Supported baudrates: 1Mbps*, 800kbps*, 500kbps, 250kbps, 125kbps, 50kps, 20kbps, 10kbps.
- Service Data Object (SDOs) supported
- Process Data Objects (PDOs) supported, maximum of 16 (8 x RPDO, 8 x TPDO)
- Synchronous, asynchronous and change of state communication modes (transmission types) supported
- Heartbeat function supported
- Emergency objects supported
- Master and or master-less node configuration supported.

NOTE: * Baudrates only supported with hardware revision 1B10 or higher.

2.3 Module Identification Label

Figure 1 illustrates the labels on the PositionServo CANopen communications module. The PositionServo CANopen module is identifiable by:

- One label affixed to the side of the module.
- The TYPE identifier in the center of the label: E94ZACAN1.
- The port (interface) identifier, P21, on the right hand side of label.

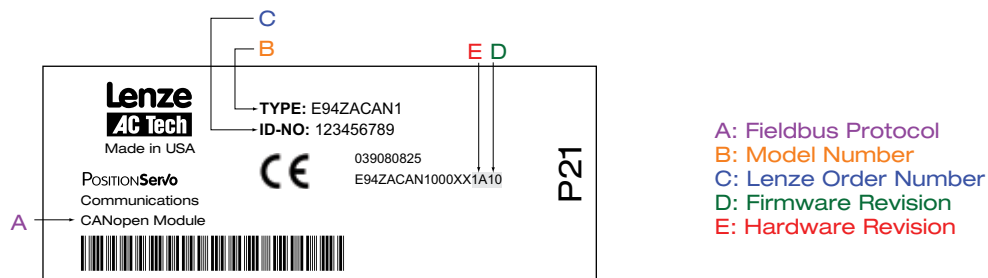


Figure 1: PositionServo CAN Module Label

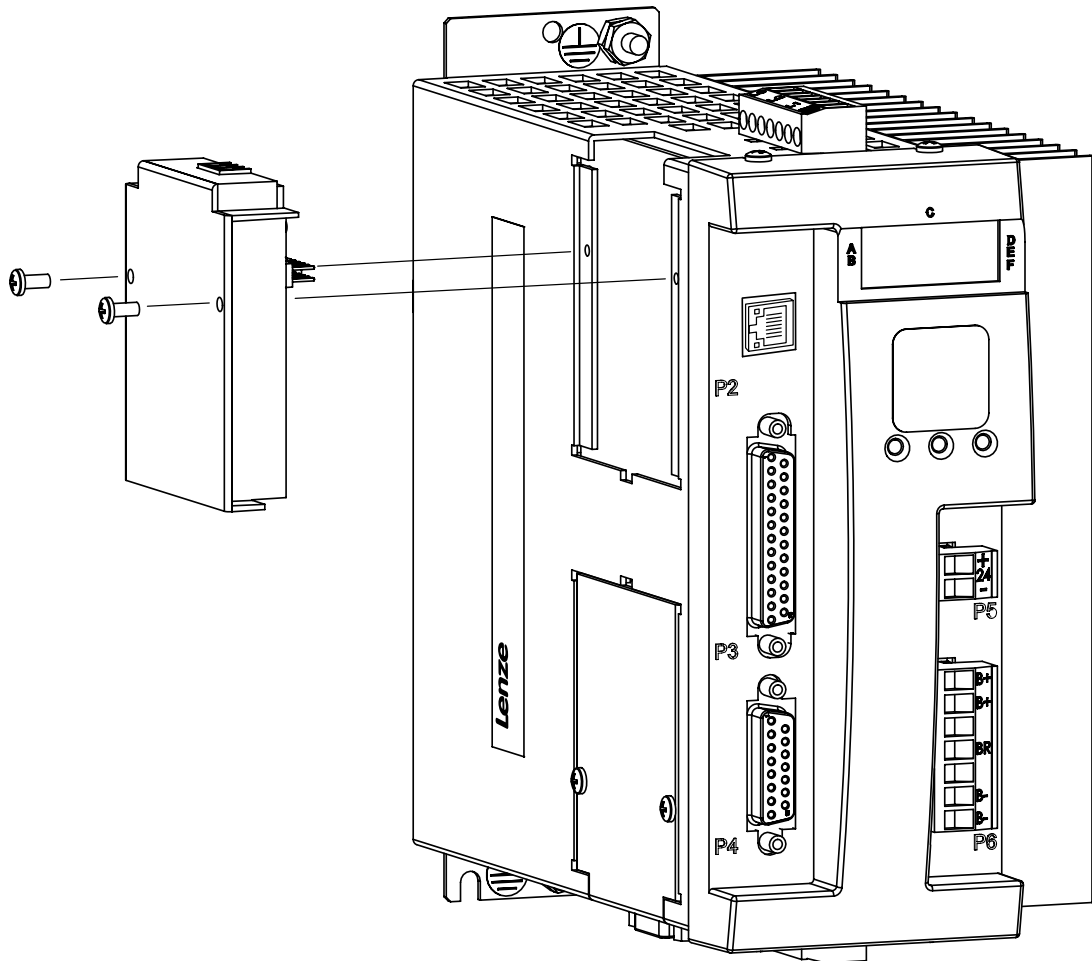


Installation

3 Installation

3.1 Mechanical Installation

1. Ensure that for reasons of safety, the AC supply, DC supply and +24VDC backup supply have been disconnected before opening the option bay cover.
2. Remove the two COMM module screws that secure Option Bay 1. With a flat head screwdriver, lift the Option Bay 1 cover plate and remove.
3. Fit the 20-pin header into the module before fitting the module into the drive.
4. Install the CAN COMM Module (E94ZACAN1) in Option Bay 1.
5. Replace the two COMM module screws (max torque: 0.3Nm/3lb-in) to secure Option Bay 1 in place.



S921a

Figure 2: Installation of CAN Communications Module



3.2 CANopen Module

Installed in Option Bay 1 as P21, the CANopen module (E94ZACAN1) is optically isolated from the rest of the drive's circuitry. The 3-pin CANopen module is for HW/SW 1A10 and the 5-pin CANopen module is for HW/SW 1B10 or higher. Table 2 lists the pinouts of the PositionServo CAN Module connector. This connector provides 2-wire plus isolated ground connection to the network.

Table 2 CANopen Interface Pin Assignments

3-pin			5-pin		
Pin	Name	Function	Pin	Name	Function
1	CAN_GND	CAN Ground	1	CAN_GND	CAN Ground
2	CAN L	CAN Bus Low	2	CAN L	CAN Bus Low
3	CAN H	CAN Bus High	3	Shield	
			4	CAN H	CAN Bus High
			5	NC	No connection

3.3 Electrical Installation

3.3.1 Cable Types

Due to the high data rates used on CANopen networks it is paramount that correctly specified quality cable is used. The use of low quality cable will result in excess signal attenuation and data loss.

3.3.2 Network Limitations

There are several limiting factors that must be taken into consideration when designing a CANopen network, however, here is a simple checklist:

- CANopen networks are limited to a maximum of 127 nodes.
- Only 32 nodes may be connected on a single network segment.
- A network may be built up from one or several segments with the use of network repeaters.
- Maximum total network length is governed by the data rate used. Refer to Table 3.
- Minimum of 1 meter of cable between nodes.
- Use fibre optic segments to:
 - Extend networks beyond normal cable limitations.
 - Overcome different ground potential problems.
 - Overcome very high electromagnetic interference.
- Spurs or T connections while sometimes useful reduce the network quality, therefore is strongly advised not to use spurs as extreme care must be taken during network design phase so as to avoid problems.



Installation

Table 3: Network Length Specifications

Baud Rate	Maximum Network Length
10kbps	5000 meters
20kbps	2500 meters
50kbps	1000 meters
125kbps	500 meters
250kbps	250 meters
500kbps	100 meters
800kbps	50 meters
1Mbps	25 meters

3.3.3 Connections and Shielding

To ensure good system noise immunity all network cables should be correctly grounded:

- Minimum recommendation of grounding is that the network cable is grounded once in every cubical.
- Ideally the network cable should be grounded on or as near to each drive as possible.
- For wiring of cable to the connector plug the unscreened cable cores should be kept as short as possible; recommended maximum of 20mm.



NOTE

As per the CiA specification (DRP303-1) it is recommend that the CAN_GND be connected on all nodes. If this is not possible due to application or cable restrictions then it is recommend that the CAN_GND terminal be connected to chassis/earth (PE).

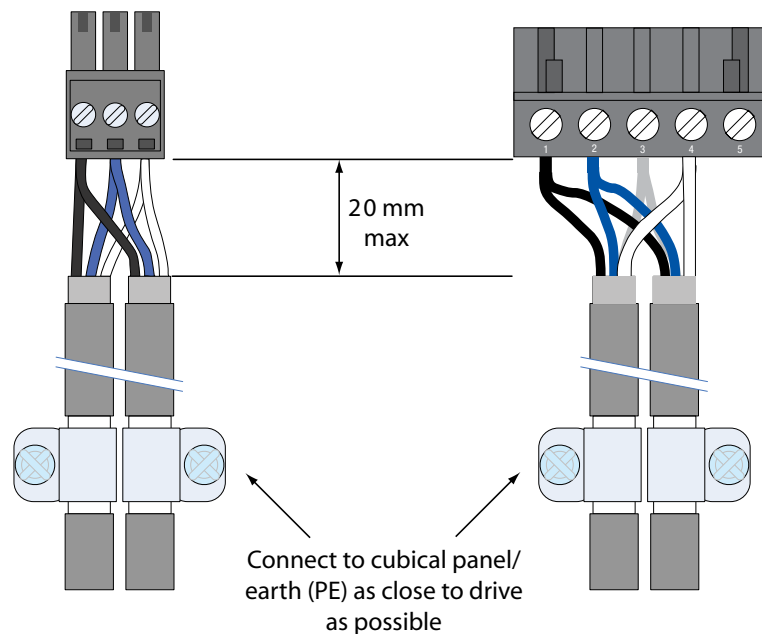


Figure 3: Connector Wiring Diagram



3.3.4 Network Termination

In high speed fieldbus networks such as CANopen it is essential to install the specified termination resistors, i.e. one at both ends of a network segment. Failure to do so will result in signals being reflected back along the cable which will cause data corruption. A 120Ω $\frac{1}{4}W$ resistor should be fitted to both ends of a network segment across the CAN_L and CAN_H lines.

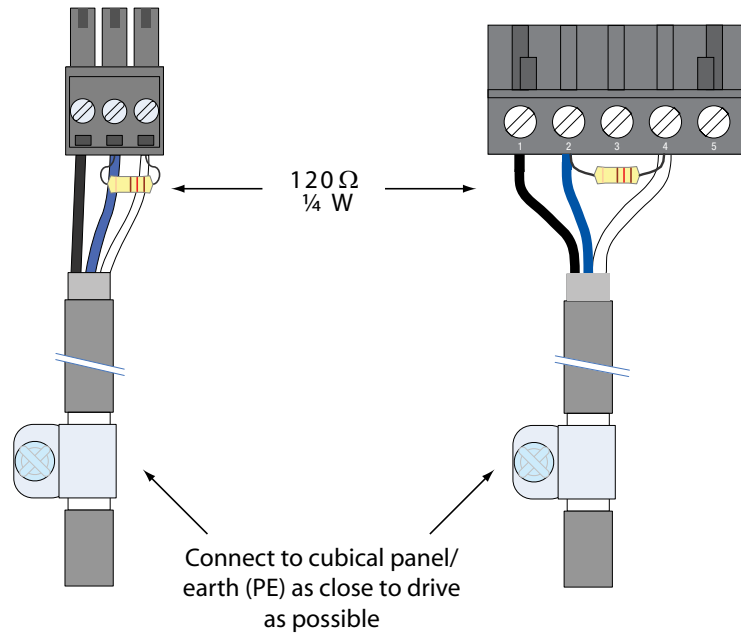


Figure 4: Termination Resistor Wiring Diagram

3.3.5 Network Schematic

Figure 5 illustrates an example CANopen network wiring diagram for the PositionServo.

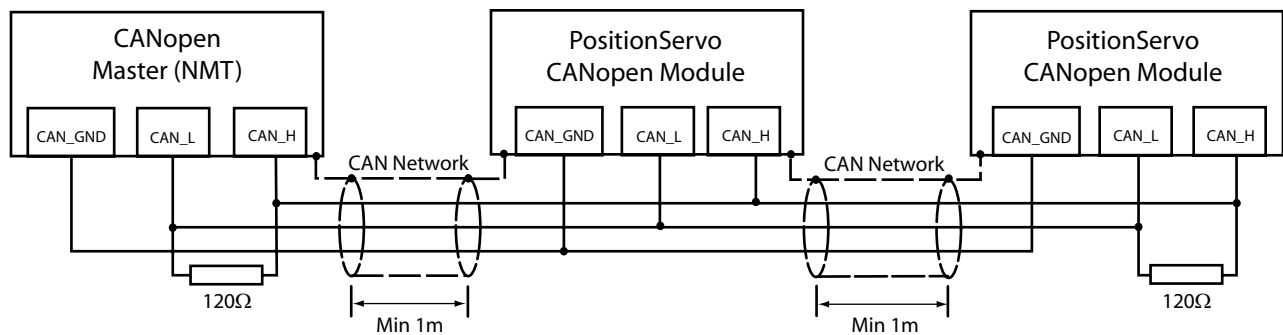


Figure 5: Network Wiring Diagram



NOTE

As per the CiA specification (DRP303-1) it is recommended that the CAN_GND be connected on all nodes. If this is not possible due to application or cable restrictions then it is recommended that the CAN_GND terminal be connected to chassis/earth (PE).



Commissioning

4 Commissioning

4.1 Overview

It is assumed that the user has familiarised themselves with how to set parameters using MotionView software. Refer to the PositionServo Programming Manual (PM94M01) for details on MotionView software.

The details that follow provide a step-by-step guide to quickly and easily set-up a PositionServo drive to communicate on a CANopen fieldbus network, in a basic format. There are many more features and settings available for the CAN option module, for details on these refer to the fuller description in the sections that follow.

4.2 Configuring the Network

By its very nature CANopen is a very open and flexible protocol and as such there are several methods that can be used to control a CANopen network, these are typically:

- **Master – Slave:** The master controls slave devices by utilising Process Data Objects (PDOs) and/or Service Data Objects (SDOs). In addition to slave node control, the master can and is often used to configure the slave node.
- **Peer – Peer:** This is when there is typically no master on the network and devices are preconfigured to communicate directly with each other by means of matching PDO COB IDs. However, most CANopen nodes will start in a “pre-operational” state and therefore it is usually the responsibility of one node to act as a pseudo master and produce a network telegram to enable all other nodes to go to an “operational” state.
- **Combination of both Master-Slave and Peer-Peer.**

4.2.1 Master Support Files

Many CANopen master configuration software utilise EDS (Electronic Data Sheet) files to configure the network profile and communications with the relevant devices. EDS files are text files that contain information about objects supported by the device. Device icon files are also supplied for use with the CANopen configuration software.

The PositionServo EDS files are available on the CD ROM that ships with the module and on the Lenze-AC Tech website.

4.2.2 CANopen Master Setup Procedure

The method for configuring master devices differs greatly between manufacturers. Provided herein is a very basic, generic guide to setting up a network master.

1. Launch the Master configuration software.
2. Install/Import the required EDS support file(s) using the wizard tool if provided.
3. Setup master CANopen port with required criteria such as node address and baudrate etc.
4. Add or “drag and drop” the required slave devices from the EDS library to the CANopen network which is typically depicted on screen.



5. Configure the slave node address, ensuring that each node has a unique and individual address.
6. Configure each slave's PDO data mapping relationship to the master interface.
7. If utilising Master based slave configuration refer to section 5, the Object Dictionary, for a list and description of supported objects.
8. If required configure the master to produce a Sync telegram on a regular time period.
9. If required configure the master to set all slave devices to go to the "Operational State" on network startup.
10. Save the configuration and download to the master.

4.3 Configuring the PositionServo CANopen Module

4.3.1 Connecting

With the drive power disconnected, install the CANopen module and connect the network cable as instructed in the preceding sections. Ensure the drive Run/Enable terminal is disabled then apply the correct voltage to the drive (refer to drive's user manual for voltage supply details).

4.3.2 Connect to the Drive with MotionView OnBoard

Refer to the PositionServo User Manual, section 6.2 for full details on configuring and connecting a drive via MotionView OnBoard (MVOB) software. Contained herein is a brief description of launching MVOB and communicating with the drive.

1. Open the PC's web browser. Enter the drive's default IP address [192.168.124.120] in the browser's Address window.
2. The authentication screen may be displayed if the PC does not have Java RTE version 1.4 or higher. If so, to remedy this situation, download the latest Java RTE from <http://www.java.com>.
3. When MotionView has finished installing, a Java icon entitled [MotionView OnBoard] will appear on your desktop and the MVOB splash screen is displayed. Click [Run] to enter the MotionView program.
4. Once MotionView has launched, verify motor is safe to operate, click [YES, I have] then select [Connect] from the Main toolbar (top left). The Connection dialog box will appear.
5. Select [Discover] to find the drive(s) on the network available for connection.

[Discover] may fail to find the drive's IP address on a computer with both a wireless network card and a wired network card (or a PC with more than one network connection). If this happens, try one of the following remedies:

Disable the wireless network card and then use [Discover].
Type in the drive's IP address manually at the box [IP Address].
Then click [Connect]

6. Highlight the drive (or drives) to be connected and click [Connect] in the dialog box.



Commissioning

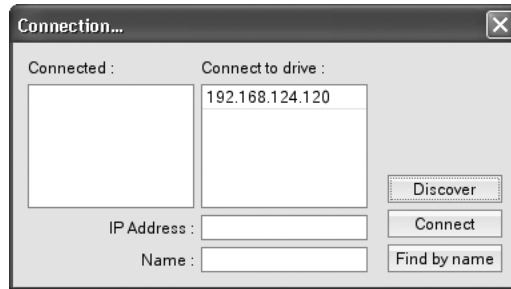


Figure 6: Connection Box with Discovered Drive

In the lower left of the MotionView display, the Message Window will contain the connection status message. The message “Successfully connected to drive B04402200450_192.168.124.120” indicates that the drive B04402200450 with IP address 192.168.124.120 is connected.

4.3.3 Communication Module Selection

In the left-hand node tree of MotionView OnBoard, click on the [Communications] folder. Using the drop down menu, select [CANopen Simple] as the required fieldbus selection. The Important Message box (to REBOOT) is displayed because the Communication setting has been changed (from None to CANopen Simple in this example). Click [Ok] to dismiss the dialog box. Reboot the drive.



Figure 7: Fieldbus Selection

4.3.4 CANopen Node Settings

The PositionServo CANopen node settings, can be accessed by clicking on the [CAN] folder and [CANopen] sub folder icons. To access the PositionServo CAN node settings of Node Address and Baud Rate, click on the [CAN] folder icon.



NOTE

Making any changes in either the [CAN] or [CANopen] folder will prompt the “Important Message” box to be displayed to inform of the required reboot.



Some parameter(s) change will take an effect after REBOOT.

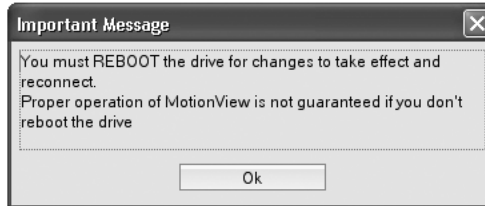


Figure 8: REBOOT Message

4.3.5 Node Address

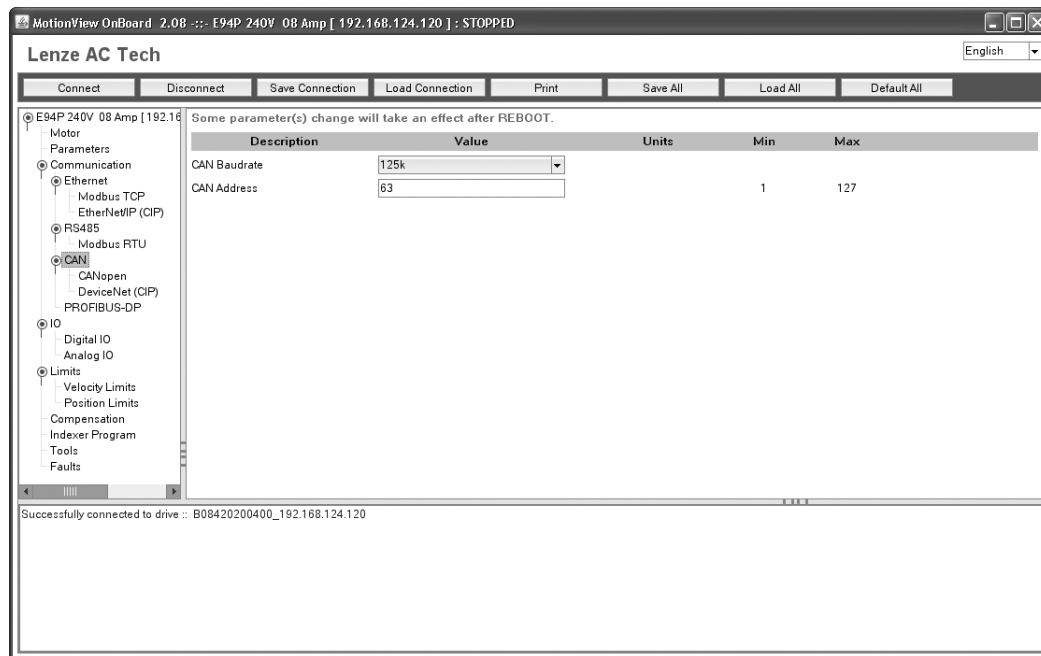


Figure 9: CAN Node Address and Baud Rate

The default address is 63. The permissible address range is: 0 – 127.

Each node on the network must have an individual address, if two or more nodes have duplicate addresses this may prevent the network from functioning correctly. The node address is also accessible from the drive keypad using parameter " CAN ".

4.3.6 Baud / Data Rate

The default baudrate is 125kbps (PID234=4). The permissible baudrate range is: 10kbps to 1Mbps (PID234 range 1-8).

The baud rate is also accessible from the drive keypad using parameter " CANb ".



Commissioning

4.3.7 CAN Bootup Mode

The PositionServo CANopen Bootup Mode is accessed by clicking on the [CANopen] folder.



Figure 10: CAN Bootup Mode

The default mode is Pre-Operational. The permissible modes are listed in Table 4.

Table 4: PID236 CAN Bootup Mode

Mode	Comment
Pre-Operational	Drive enters the "PRE_OPERATION" state after bootup
Operational	Drive enters the "OPERATIONAL" state after bootup
Pseudo Master	In addition to the drive entering the "OPERATIONAL" state after bootup the drive will also (after the delay period set by the CAN Bootup Delay parameter) broadcast a command for all other CANopen nodes to go in to the "OPERATIONAL" state

The Bootup Mode is also accessible from the drive keypad using parameter "CANM".

4.3.8 CAN Bootup Delay

The PositionServo CANopen Bootup Delay is accessed by clicking on the [CANopen] folder.



Figure 11: CAN Bootup Delay

The default delay time is 3 seconds. The permissible delay time is: 0 – 5 (sec). The Bootup Delay is only functional when the drive is used in pseudo master mode (4.3.7). The Bootup Delay parameter sets the time delay from when the drive itself boots up and is fully functional to when it broadcasts the NMT command for all slave devices to go to the operational state.

The CAN Bootup Delay is also accessible from the drive keypad using parameter "CANE".



4.3.9 Data Mapping

- The PositionServo CANopen module can support up to 8 Process Data Objects (PDOs) in each direction.
- PDO Configuration is described in full in section 7, Process Data Objects.
- The default mappings for PositionServo CANopen is 4 RxPDO and 4 TxPDO each with one mapped object. The configuration is shown in Table 5.

Table 5: Default mapped PDOs

RxPDO	Mapped Function	Transmission Type	Data Format		TxPDO	Mapped Function	Transmission Type	Data Format
1	2064 – VAR_V0	255	RAM Integer		1	2036 – VAR_STATUS	255	RAM Integer
2	248B – VAR_IREF	255	RAM FLOAT		2	2053 – VAR_EXSTATUS	1	RAM Integer
3	2042 – VAR_OUTPUS	255	RAM Integer		3	24BC – VAR_PHCUR	1	RAM FLOAT
4	2458 – VAR_AOUT	255	RAM FLOAT		4	24D7 – VAR_APOS	1	RAM FLOAT

4.3.10 Re-Initialising

To activate any changes made the drive has to be re-initialised. Hence the warning within MotionView

Some parameter(s) change will take an effect after REBOOT.

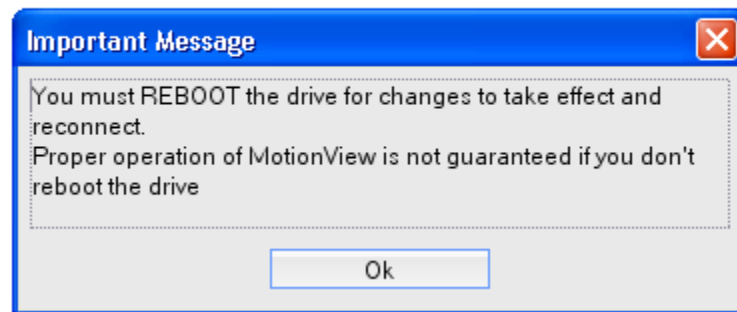


Figure 12: REBOOT Message

This can be done by cycling the power to the drive.

4.3.11 Non-Module Parameter Settings

In addition to configuring the CAN option module and depending upon the application there may be several drive based parameters that will need to be set using MotionView or an Indexer program or via CANopen Service Data Object (SDOs). Such as:

- PID34 – Drive Mode (VAR_DRIVEMODE)
- PID37 – Reference (VAR_REFERENCE)
- PID29 – Enable switch funtion (VAR_ENABLE_SWITCH_TYPE)



CANopen Object Dictionary

5. CANopen Object Dictionary

5.1 What is the CANopen Object Dictionary?

- CANopen is an object based protocol and as such everything is orientated around these objects.
- The Object Dictionary is essentially a grouping of objects accessible via the network in an ordered and pre-defined fashion.
- EDS (Electronic Data Sheets) files are text files that contain a complete list of a nodes supported Object Dictionary. These are used by some CANopen network configuration software tools to simplify network setup and commissioning.

5.1.1 Object Format

Each object within the dictionary consists of 6 parts as shown in Table 6.

Table 6: Format of the Object Dictionary

Part	Description
Index	Denotes the object's position / address within the Object Dictionary
Object Code	Denotes the data format, such as ARRAY or VARIABLE
Name	Provides a simple textual description of the function of that particular object
Type	Denotes the data type such as boolean, Floating point, signed or unsigned integer etc
Attribute	Defines the access rights for a particular object
M/O	Defines whether the object is M andatory or O ptional according to the CANopen specification

An object may also contain “sub-indexes” if the object in question is a complex object such as an array or record.

5.1.2 Object Dictionary Layout

The overall layout of the Object Dictionary is shown in Table 7.

Table 7: Object Dictionary Structure

Index (hex)	Object
0000	not used
0001 - 001F	Static Data Types
0020 - 003F	Complex Data Types
0040 - 005F	Manufacturer Specific Complex Data Types
0060 - 007F	Device Profile Specific Static Data Types
0080 - 009F	Device Profile Specific Complex Data Types
00A0 - 0FFF	Reserved for future use
1000 - 1FFF	Communication Profile Area
2000 - 5FFF	Manufacturer Specific Profile Area
6000 - 9FFF	Standardised Device Profile Area (not supported)
A000 - BFFF	Standardised Interface Profile Area (not supported)
C000 - FFFF	Reserved for future use



5.1.3 Accessing the Object Dictionary

The Object Dictionary can be accessed primarily in two ways:

- Confirmed access using Service Data Object (SDO) access
- Unconfirmed access using Process Data Object (PDO) access, i.e. no *handshake*.

Every message sent via CANopen network contains an address to identify the node the message is destined for. This address is called the CAN Object Identifier (COB ID). Each communication object has a specific identifier depending upon its type/function. Refer to Table 8 for message types and associated COB IDs.

Table 8: Messages and Associated COB IDs

COB ID	Object	Description
0x00	NMT Service	Network Management Telegram: These are used for network services such as instructing a node to go to different operational states and are typically produced by a network master
0x80	SYNC	Synchronization message: Typically generated by a network master and used to control actions of slave nodes
0x100	Time Stamp	High resolution method for synchronising the internal clocks of network nodes. NOTE: the PositionServo does not support this function
0x81 – 0xFF	Emergency	Used to transmit error messages
0x181 – 0x1FF	TxPDO 1	Process Data Object: Fast exchange of process data e.g. actual velocity or position values
0x281 – 0x2FF	TxPDO 2	
0x381 – 0x3FF	TxPDO 3	
0x481 – 0x4FF	TxPDO 4	
0x201 – 0x27F	RxPDO 1	
0x301 – 0x37F	RxPDO 2	
0x401 – 0x47F	RxPDO 3	
0x501 – 0x57F	RxPDO 4	
0x581 – 0x5FF	TxSDO	Service Data Object: Used for general parameter access, PDO configuration and non time-critical acyclic data access
0x601 – 0x67F	RxSDO	
0x701 – 0x77F	Error Control	PositionServo supports the “Heartbeat” protocol, in which the drive produces a basic network status message on a pre-defined time period

5.2 Communication Profile Area

Table 9 lists the communication objects supported by PositionServo.

Table 9: Communication Profile Objects

Object	Name	Object	Name
0x1000	Device type	0x1014	Emergency message COB ID
0x1001	Error register	0x1015	Inhibit time EMCY
0x1003	Pre-defined error field	0x1017	Producer heartbeat time
0x1005	SYNC COB ID	0x1018	Identity object
0x1008	Manufacture Device Name	0x1400 to 0x1407	RxPDO 1 to 8 parameters
0x1009	Manufacture hardware version	0x1600 to 0x1607	RxPDO 1 to 8 Mapping Parameters
0x100A	Manufacture software version	0x1800 to 0x1807	TxPDO 1 to 8 parameters
		0x1A00 to 0x1A07	TxPDO 1 to 8 Mapping Parameters



CANopen Object Dictionary

5.2.1 Device Type

Device Type			
Object Index:	0x1000	Sub-index:	00
Default:	00020192	Range:	--
Access:	RO	Type:	Unsigned 32
Units:	--	PDO mappable:	No

This object describes the device type and its functionality. The 32-bit value is composed of two components.

Table10: Device Type format

Byte 3	Byte 2	Byte 1	Byte 0
Additional information		Device Profile Supported	
0x0002		0x0192	

- Bytes 0 and 1: A value 0x0192 (402 decimal) indicates the DS402 device profile.
- Bytes 2 and 3: These give detailed information about the type device the CANopen node is. A value of 0x0002, indicates that the node is a servo drive.

5.2.2 Error Register

Error Register			
Object Index:	0x1001	Sub-index:	00
Default:	--	Range:	--
Access:	RO	Type:	Unsigned 8
Units:	--	PDO mappable:	Yes

The error register is used to indicate that an error has occurred. If a bit is set to 1, then an error has occurred. The error register also forms part of the emergency object, refer to section 8 for details.

5.2.3 Pre-defined Error Field

Number of Errors			
Object Index:	0x1003	Sub-index:	0x00
Default:	0	Range:	0 – 1
Access:	RW	Type:	Unsigned 8
Units:	--	PDO mappable:	No

Standard Error Field			
Object Index:	0x1003	Sub-index:	0x01
Default:	--	Range:	--
Access:	RO	Type:	Unsigned 32
Units:	--	PDO mappable:	No

This object holds the error code that has occurred on the device and has been signaled via the Emergency Object. In doing so it provides an error history.

- Sub-index 0 contains the number of errors recorded
- Sub-index 1 contains the value of the last record trip code
- Writing a “0” to sub-index 0 deletes the error history



5.2.4 SYNC COB ID

SYNC COB ID			
Object Index:	0x1005	Sub-index:	00
Default:	0x80	Range:	--
Access:	RW	Type:	Unsigned 32
Units:	--	PDO mappable:	No

This object defines the COB-ID of the Synchronisation Object the drive will use for PDOs that use transmission types 0-254.

5.2.5 Manufacture Device Name

Manufacture Device Name			
Object Index:	0x1008	Sub-index:	00
Default:	--	Range:	--
Access:	RO	Type:	Visible string
Units:	--	PDO mappable:	No

This objects contains the manufacturer device name. If read it will return the string "94P"

5.2.6 Manufacture Hardware Version

Manufacture Hardware Version			
Object Index:	0x1009	Sub-index:	00
Default:	--	Range:	--
Access:	RO	Type:	Visible string
Units:	--	PDO mappable:	No

This objects contains the manufacturer hardware version. If read it will return the value of the drive hardware version, e.g. "450"

5.2.7 Manufacture Software Version

Manufacture Software Version			
Object Index:	0x100A	Sub-index:	00
Default:	--	Range:	--
Access:	RO	Type:	Visible string
Units:	--	PDO mappable:	No

This objects contains the manufacturer software version. If read it will return the value of the drive firmware version, e.g. "430"

5.2.8 Emergency Message COB ID

Emergency Message COB ID			
Object Index:	0x1014	Sub-index:	00
Default:	--	Range:	--
Access:	RO	Type:	Unsigned 32
Units:	--	PDO mappable:	No

This object defines the identifier used for the Emergency object transmitted by the PositionServo during an error or a trip state. The identifier is set automatically: Emergency message COB ID = 0x80 + node address.



5.2.9 Inhibit Time EMCY

Inhibit Time EMCY			
Object Index:	0x1015	Sub-index:	00
Default:	0	Range:	--
Access:	RW	Type:	Unsigned 16
Units:	100µs	PDO mappable:	No

This object specifies the inhibit time for the Emergency Object transmitted by the PositionServo. If a value greater than 0 is set then once an Emergency Object has been transmitted all other Emergency Object transmissions are prohibited until the specified time has elapsed.

5.2.10 Producer Heartbeat Time

Producer heartbeat time			
Object Index:	0x1017	Sub-index:	00
Default:	--	Range:	--
Access:	RW	Type:	Unsigned 16
Units:	ms	PDO mappable:	No

The PositionServo can be enabled to be a “heartbeat producer”, i.e. produce a heartbeat message cyclically. This message is received by one or more “heartbeat consumer” devices, usually the CANopen master, and indicates to the master controller that the slave device is communicating without problem. If the heartbeat message is not received within the defined time period, a “heartbeat event” will be generated in the master controller, allowing it to take appropriate action to ensure system safety is maintained. Setting the heartbeat time to 0 disables the heartbeat function.

5.2.11 Identity Object

Number of entries			
Object Index:	0x1018	Sub-index:	0x00
Default:	--	Range:	--
Access:	RO	Type:	Unsigned 8
Units:	--	PDO mappable:	No

Vendor ID			
Object Index:	0x1018	Sub-index:	0x01
Default:	--	Range:	--
Access:	RO	Type:	Unsigned 32
Units:	--	PDO mappable:	No

Product code			
Object Index:	0x1018	Sub-index:	0x02
Default:	--	Range:	--
Access:	RO	Type:	Unsigned 32
Units:	--	PDO mappable:	No

Revision number			
Object Index:	0x1018	Sub-index:	0x03
Default:	--	Range:	--
Access:	RO	Type:	Unsigned 32
Units:	--	PDO mappable:	No



Serial number			
Object Index:	0x1018	Sub-index:	0x04
Default:	--	Range:	--
Access:	RO	Type:	Unsigned 32
Units:	--	PDO mappable:	No

This object holds general information about the PositionServo to provide a standard method of differentiating and identifying different devices on the network.

- Sub-index 0: Lists how many elements there are to the Identity
- Sub-index 1: Vendor ID for Lenze AC Tech which is 0x19C
- Sub-index 2: Lenze AC Tech product code for PositionServo which is 0x3AC (940 decimal)
- Sub-index 3: CANopen module revision number
- Sub-index 4: Serial number for the PositionServo drive

5.2.12 RxPDO 1 to 8 Communication Parameters

Receive PDO Parameter			
Object Index:	0x1400 - 7	Sub-index:	0x00
Default:	0	Range:	--
Access:	RO	Type:	Unsigned 8
Units:	--	PDO mappable:	No

COB ID used by PDO			
Object Index:	0x1400 - 7	Sub-index:	0x01
Default:	--	Range:	--
Access:	RW	Type:	Unsigned 32
Units:	--	PDO mappable:	No

Transmission type			
Object Index:	0x1400 - 7	Sub-index:	0x02
Default:	--	Range:	0 - 255
Access:	RW	Type:	Unsigned 8
Units:	--	PDO mappable:	No

These objects are used to configure the RxPDOs. Also refer to section 7 for full details on Process Data Objects (PDOs).

- Sub-index 0: Specifies how many sub indexes there are to this object
- Sub-index 1: Specifies the Communication Object ID used by the PDO
- Sub-index 2: Specifies the transmission type used by the PDO

Table 11: Supported RxPDO Transmission Types

Transmission Type	Scheduling	Description
0 – 240	Synchronous	The received data is held until the next SYNC message. When the SYNC message is received the data is applied
254 – 255	Asynchronous, timer trigger	The received data is applied to its mapped objects immediately upon reception



5.2.13 RxPDO Mapping Parameters

Number of mapped objects in the PDO			
Object Index:	0x1600 - 7	Sub-index:	0x00
Default:	0	Range:	--
Access:	RW	Type:	Unsigned 8
Units:	--	PDO mappable:	No

COB ID used by PDO			
Object Index:	0x1600 - 7	Sub-index:	0x01 - 8
Default:	--	Range:	--
Access:	RW	Type:	Unsigned 32
Units:	--	PDO mappable:	No

These objects are used to map the objects within the RxPDOs. Also see section 7 for full details on Process Data Objects (PDOs).

- Sub-index 0: Specifies how many objects are mapped in each RxPDO
- Sub-index 1 - 8: Contain the objects number that are mapped

5.2.14 TxPDO 1 to 8 Communication Parameters

Receive PDO Parameter			
Object Index:	0x1800 - 7	Sub-index:	0x00
Default:	0	Range:	--
Access:	RO	Type:	Unsigned 8
Units:	--	PDO mappable:	No

COB ID used by PDO			
Object Index:	0x1800 - 7	Sub-index:	0x01
Default:	--	Range:	--
Access:	RW	Type:	Unsigned 32
Units:	--	PDO mappable:	No

Transmission type			
Object Index:	0x1800 - 7	Sub-index:	0x02
Default:	--	Range:	0 - 255
Access:	RW	Type:	Unsigned 8
Units:	--	PDO mappable:	No

Inhibit time			
Object Index:	0x1800 - 7	Sub-index:	0x03
Default:	0	Range:	--
Access:	RW	Type:	Unsigned 16
Units:	100µs	PDO mappable:	No

Event timer			
Object Index:	0x1800 - 7	Sub-index:	0x05
Default:	0	Range:	--
Access:	RW	Type:	Unsigned 16
Units:	ms	PDO mappable:	No



These objects are used to configure the TxPDOs. Also refer to section 7 for full details on Process Data Objects (PDOs).

- Sub-index 0: Specifies how many sub indexes there are to this object
- Sub-index 1: Specifies the Communication Object ID used by the PDO
- Sub-index 2: Specifies the transmission type used by the PDO

Table 12: Supported TxPDO Transmission Types

Transmission Type	Scheduling	Description
0	Acyclic, Synchronous	If the source data has changed, the TxPDO is transmitted on reception of a SYNC object
1 – 240	Cyclic, synchronous	The PDO is transferred synchronously and cyclically. The transmission type indicates the number of SYNC objects that are necessary to trigger TxPDOs.
254/255	Asynchronous, timer trigger	The PDO is Event driven. Events are created by the following: <ul style="list-style-type: none"> • When the value of a mapped object within the PDO changes • If the Event timer is configured and used to generate periodic transmissions

- Sub-index 3: Setting a value greater than 0 automatically activates the function so that on a TxPDO transmission an “inhibit timer” is started and the next transmission will not occur until the timer expires
- Sub-index 4: Is not support by PositionServo
- Sub-index 5: Event Timer can be used to generate TxPDOs. Setting a time greater than 0 enables this feature and sets the fixed interval for the TxPDO to be transmitted.

5.2.15 TxPDO Mapping Parameters

Number of mapped objects in the PDO			
Object Index:	0x1A00 - 7	Sub-index:	0x00
Default:	0	Range:	--
Access:	RW	Type:	Unsigned 8
Units:	--	PDO mappable:	No

COB ID used by PDO			
Object Index:	0x1A00 - 7	Sub-index:	0x01 - 8
Default:	--	Range:	--
Access:	RW	Type:	Unsigned 32
Units:	--	PDO mappable:	No

These objects are used to map the objects within the RxPDOs. Also see section 7 for full details on Process Data Objects (PDOs).

- Sub-index 0: Specifies how many objects are mapped in each TxPDO
- Sub-index 1 - 8: Contains the objects number that are mapped.



5.3 Manufacture Specific Profile Area

Objects in the range 0x2000 to 0x5FFF are free for manufacturers to utilise. In the case of PositionServo, these objects are used to provide access to drive parameters, user variables and Index program commands. Details on these functions, variables and parameters are provided in the PositionServo User and Programming manuals.

Furthermore the PositionServo supports several data areas, types and sizes. To simplify this, the Manufacture Specific Profile Area is divided and structured in a logical manner.

5.3.1 Data Format, Size and Memory Area

All PositionServo drive parameters are 32-bit in size but can be accessed in two different formats:

- IEEE Floating Point (FLOAT or REAL).
- 32-bit integer (DWORD or DINT). 16-bit access is possible for a limited range of integer format objects where the 32-bit object is divided into two sub-indexes.

Furthermore, PositionServo parameters exist in each of the two formats in both RAM (volatile) and EPM (non-volatile) areas. Therefore the memory addresses are divided into four ranges according to their format and memory type as shown in Table 13.

Table 13: Object Address Ranges

Object Offset	0x2000	0x2400	0x3000	0x3400
Type	RAM	RAM	EPM	EPM
Format	32-bit Integer	Float	32-bit Integer	Float

The CANopen object for a drive parameter can be calculated using:

$$\text{CANopen}_{\text{Object}} = \text{PID}_{\text{Number}} + \text{Object}_{\text{Offset}}$$

where:

$\text{PID}_{\text{Number}}$ = PositionServo **P**arameter **I**ndex **N**umber in hexadecimal.

$\text{Object}_{\text{Offset}}$ = Memory offset per Table 15.

5.4 Endian Format

CANopen uses “little-endian” representation for object data. This means that when a numerical value that is larger than a single byte is transmitted, the LEAST significant byte (LSB) is sent first, e.g.

- 16-bit integer value 0x1234 = 2 bytes of 0x34 and 0x12
- 32-bit integer value 0x12345678 = 4 bytes of 0x78, 0x56, 0x34 and 0x12

5.5 Object Access

- Care should be taken when accessing drive registers from multiple sources such as multiple clients or the drive Indexer program as data could be over written or out of sequence.
- Writing to the EPM area of memory simultaneously writes to the RAM area too.
- Writing to the EPM area of memory should be done conservatively as the EEPROM (EPM) has a typical life expectancy of 1 million writes.



6. Service Data Objects

6.1 What are Service Data Objects?

- Service Data Objects (SDOs) is the name given to the method used to provide non-cyclic access to all objects in the CANopen object dictionary.
- SDO messages are always instigated from a host device such as a CANopen master/client.
- Slave/server devices only have an SDO server, allowing them to respond to SDO messages initiated from elsewhere.
- Each SDO message allows 1 object to be accessed in the CANopen object dictionary.
- SDO messages can handle up to 4 data bytes, i.e. one 32-bit data value to be handled on each message.
- SDO communications are supported in PRE-OPERATIONAL and OPERATIONAL states.

6.2 SDO Message Identifiers

Each message format on a CAN based network has a CAN Objects Identifier (COB ID) associated with it. For SDOs there are two COB IDs:

- Master/client messages use 0x600 + intended slave/server node address
- Slave/server response messages use 0x580 + their node address

6.3 PID Access

- SDO communications provide access to all objects in the CANopen objects dictionary. The dictionary includes an object range 0x2000 to 0x5FFF for manufacture specific features.
- Every PositionServo parameter (PID) is available as a CANopen object in this range.
- A full list of the manufacture objects is available in section 11.1, PID List with CANopen Values.
- The EDS file also contains a full list all support objects.

6.4 SDO Abort Codes

If an invalid SDO message is received the drive will respond with a SDO Abort Code as per the CANopen DS301 protocol specification. Table 14 lists the SDO Abort Codes.

Table 14: SDO Abort Codes

Abort Code (Hex)	Description
0503 0000	Toggle bit not alternated
0504 0000	SDO protocol timed out
0504 0001	Client/server command specifier not valid or unknown
0504 0002	Invalid block size (block mode only)
0504 0003	Invalid sequence number (block mode only)
0504 0004	CRC error (block mode only)



SDO Access

Abort Code (Hex)	Description
0504 0005	Out of memory
0601 0000	Unsupported access to an object
0601 0001	Attempt to read a write-only object
0601 0002	Attempt to write to a read-only object
0602 0000	Object does not exist in the object dictionary
0604 0041	Object cannot be mapped to the PDO
0604 0042	The number and length of the objects to be mapped would exceed PDO length
0604 0043	General parameter incompatibility
0604 0047	General internal incompatibility in the device
0606 0000	Access failed due to a hardware error
0607 0010	Data type does not match, length of service parameter does not match
0607 0012	Data type does not match, length of service parameter too high
0607 0013	Data type does not match, length of service parameter too low
0609 0011	Sub-index does not exist
0609 0030	Value range of parameter exceeded (only for write-access)
0609 0031	Value of parameter written too high
0609 0032	Value of parameter written too low
0609 0036	Maximum value is less than minimum value
0800 0000	General error
0800 0020	Data cannot be transferred or stored to the application
0800 0021	Data cannot be transferred or stored to the application because of local control
0800 0022	Data cannot be transferred or stored to the application because of the present device state
0800 0023	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error)

6.5 SDO Message Frame

For full details of the SDO messaging system and frame construction including CRC, refer to the official CANopen specification DS301. Here is a simplified overview.

- Every SDO request and response message contains 8 bytes of data.
- SDO messages can be segmented and handled as a block when working with complex object data values or when data values are larger than 32-bit. However, because all PositionServo objects values are no greater than 32-bit, segmented and block transfer are not required and a full SDO message is achieved with 2 messages (request and response). This is known as “Expedited SDO Transfers”.
- The commonly used term “SDO Download” implies a master/client to slave/server write.
- The commonly used term “SDO Upload” implies a master/client to slave/server read.

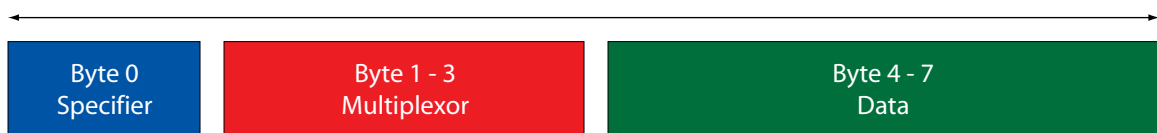


Figure 13: Expedited SDO Message Frame



6.5.1 Specifier

Table 15 lists the BIT functions with the Specifier byte.

Table 15: Specifier

Byte	Bit	Function	Description	
0	0	S: Size indicator	Only used for write requests and read responses 0 – data size not indicated 1 – data size in N field	
	1	E: Transfer type	Only used for write requests and read responses 0 – SDO is segmented 1 – SDO is expedited	
	2	N: Number of empty bytes	Only used for write requests and read responses N indicates the number of empty data bytes within the Data field (bytes 4-7)	
	3			
	4	Reserved	Not used, always 0	
	5	CCS / SCS: identifies the type of SDO	CCS - Client 0 – download segment request 1 – initiate download request 2 – initiate upload request 3 – upload segment request 4 – abort transfer request 5 – block upload 6 – block download	SCS - Server 0 – upload segment response 1 – download segment response 2 – initiate upload response 3 – initiate download response 4 – abort transfer request 5 – block download 6 – block upload
	6			
7				

6.5.2 Multiplexor

Table 16 lists the functions with the Multiplexor bytes

Table 16: Multiplexor

Byte	Function	Description
1 – 2	Index	CAN object number to be accessed
3	Sub-index	CAN object sub-index number to be accessed

6.5.3 Data

Four bytes used to hold data during master/client writes and slave/server read responses.



SDO Access

6.6 SDO Access Examples

6.6.1 Example 1: Read Velocity Accel Limit

- Drive address = 1
- Read Velocity Accel Limit, PID76 / VAR_ACCEL_LIMIT (default value = 1000)
- Uses the SDO Initiate Upload command
- Use object 0x204C sub-index 1 (32-bit integer)

Table 17a: SDO master/client read request

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x601	0x40	0x4C 0x20	0x01	0x00 0x00 0x00 0x00
Client to server #1	CCS = 2	Object 0x204C	Sub-index 0x01	Data 0

Table 17b: SDO slave/server read response

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x581	0x42	0x4C 0x20	0x01	0xE8 0x03 0x00 0x00
Slave #1 to master	S = 1 E = 1 N = 0 SCS = 2	Object 0x204C	Sub-index 0x01	Data 0x03E8 (1000dec)

6.6.2 Example 2: Write to Velocity Accel Limit

- Drive address = 1
- Write Velocity Accel Limit, PID76 / VAR_ACCEL_LIMIT with a value of 1500
- Uses the SDO Initiate Download command
- Use object 0x204C sub-index 1 (32-bit integer)

Table 18a: SDO master/client write request

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x601	0x23	0x4C 0x20	0x01	0xDC 0x05 0x00 0x00
Client to server #1	S = 1 E = 1 N = 0 CCS = 1	Object 0x204C	Sub-index 0x01	Data 0x05DC (1500dec)

Table 18b: SDO slave/server write response

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x581	0x60	0x4C 0x20	0x01	0x00 0x00 0x00 0x00
Slave #1 to master	SCS = 3	Object 0x204C	Sub-index 0x01	Data 0



6.6.3 Example 3: Read User Variable V0 Least Significant Byte (LSB)

- Drive address = 1
- Read User Variable V0 LSB, PID100 / VAR_V0
- Uses the SDO Initiate Upload command
- Use object 0x2066 sub-index 2 (16-bit integer)

Table 19a: SDO master/client read request

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x601	0x40	0x66 0x20	0x02	0x00 0x00 0x00 0x00
Client to server #1	CCS = 2	Object 0x2066	Sub-index 0x02	Data 0

Table 19b: SDO slave/server read response

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x581	0x4B	0x66 0x20	0x02	0xD2 0x04 0x00 0x00
Slave #1 to master	S = 1 E = 1 N = 2 SCS = 2	Object 0x2066	Sub-index 0x02	Data 0x04D2 (1234dec)

6.6.4 Example 4: Write to User Variable V0 Least Significant Byte (LSB)

- Drive address = 1
- Write User Variable V0 LSB, PID100 / VAR_V0 with a value of 5678
- Uses the SDO Initiate Download command
- Use object 0x2066 sub-index 2 (16-bit integer)

Table 20a: SDO master/client read request

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x601	0x2B	0x66 0x20	0x02	0x2E 0x16 0x00 0x00
Client to server #1	S = 1 E = 1 N = 2 CCS = 1	Object 0x2066	Sub-index 0x02	Data 0x162E (5678dec)

Table 20b: SDO slave/server read response

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x581	0x60	0x66 0x20	0x02	0x00 0x00 0x00 0x00
Slave #1 to master	SCS = 3	Object 0x2066	Sub-index 0x02	Data 0



SDO Access

6.6.5 Example 5: Read APOS

- Drive address = 1
- Read Variable APOS, PID215 / VAR_APOS
- Uses the SDO Initiate Upload command
- Use object 0x24D7 sub-index 0 (IEEE 32-bit FLOAT)

Table 21a: SDO master/client read request

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x601	0x40	0xD7 0x24	0x00	0x00 0x00 0x00 0x00
Client to server #1	CCS = 2	Object 0x24D7	Sub-index 0x00	Data 0

Table 21b: SDO slave/server read response

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x581	0x43	0xD7 0x24	0x00	0X00 0x27 0x0B 0x40
Slave #1 to master	S = 1 E = 1 N = 0 SCS = 2	Object 0x24D7	Sub-index 0x00	Data 0x400B2700 (2.174255 _{dec})

6.6.6 Example 6: Write to APOS

- Drive address = 1
- Write APOS, PID215 / VAR_APOS with a value of 1234.5678
- Uses the SDO Initiate Download command
- Use object 0x24D7 sub-index 0 (IEEE 32-bit FLOAT)

Table 22a: SDO master/client read request

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x601	0x23	0xD7 0x24	0x00	0x2B 0x52 0x9A 0x44
Client to server #1	S = 1 E = 1 N = 0 CCS = 1	Object 0x24D7	Sub-index 0x00	Data 0x449A522B (1234.5678 _{dec})

Table 22b: SDO slave/server read response

COB ID	Byte 0	Bytes 1-2	Byte 3	Bytes 4-7
0x581	0x60	0xD7 0x24	0x00	0x00 0x00 0x00 0x00
Slave #1 to master	SCS = 3	Object 0x24D7	Sub-index 0x00	Data 0



7 Process Data Objects

7.1 What are Process Data Objects?

- Process Data Objects (PDOs) is the name given to the method used to transfer routine process data between the network nodes.
- Process Data Objects are usually pre-configured in the CANopen master and downloaded to the PositionServo during network initialisation. In addition to this the PositionServo supports dynamic PDO mapping meaning that the PDO configuration can be changed “live” (instantly) without the need to re-initialise the drive.
- Alternatively, PDOs can be easily configured directly from within the MotionView platform. This is the recommended method when using a basic CAN master that does not support EDS files or when creating a peer to peer (masterless) network.
- The terms “RxPDO” (Receive) and “TxPDO” (Transmit) describe the direction of data as seen by the individual nodes.



NOTE

- Dynamic PDO mapping is only supported when editing the PDO configuration from an external source, i.e. a CANopen master. Editing PDO mapping through the MotionView platform requires the drive to be re-initialised by means of a reboot / power cycle or issuing a node reset command from a CAN NMT capable master.
- PositionServo CANopen settings that are edited by an external source, i.e. a CANopen master are not saved on power off or drive reboot and will therefore have to be re-configured by the CAN master.
- PositionServo CANopen settings that are edited from within the MotionView platform are saved within the drive.

7.2 PDO Configuration in MotionView

7.2.1 COB ID and Mode

PDOs 1 to 4 have default Object Identifiers (COB ID) assigned based on the CANopen DS301 specification. Refer to Table 23.

PDOs 5 to 8 are disabled by default and do not have any default COB IDs.

Table 23: PDO Default COB IDs

PDO	COB ID
RxPDO1	512 + node address
RxPDO2	768 + node address
RxPDO3	1024 + node address
RxPDO4	1280 + node address
TxPDO1	384 + node address
TxPDO2	640 + node address
TxPDO3	896 + node address
TxPDO4	1152 + node address



PDO Access

To edit the COB ID un-tick the default option to unlock the COB ID number.

The Mode function controls whether PDOs are Enabled or Disabled.

The screenshot shows two configuration panels: RxPDO #1 (blue) and TxPDO #1 (green). In both panels, the 'COB ID' field contains the value 575 (for Rx) and 447 (for Tx). To the right of each field is a checked checkbox labeled 'Default'. The 'Mode' dropdown menu is set to 'Enable', and the 'Transmission Type' field contains the value 255.

Figure 14a: COB ID is locked to Default

The screenshot shows the same two configuration panels as Figure 14a. In this version, the 'Default' checkboxes for both RxPDO and TxPDO are unchecked, indicating that the COB ID is unlocked and can be edited.

Figure 14b: COB ID is unlocked

7.2.2 Transmission Type

The Transmission Type defines the scheduling of a PDO. Furthermore the Transmission Type can be divided in two categories: Synchronous and Asynchronous. Table 24 lists the supported Transmission Types.

- **Synchronous:** Messages are processed only after receipt of a specified number of synchronization (SYNC) object. The Sync object is sent at regular intervals by a designated synchronization device such as a CANopen master.
- **Asynchronous:** The receipt of SYNC messages does not govern message processing.

Table 24: Support Transmission Types

	Transmission Type	Scheduling	Description
RxPDO	0 – 240	Synchronous	The received data is held until the next SYNC message. When the SYNC message is received the data is applied
	254 – 255	Asynchronous, timer trigger	The received data is applied to its mapped objects immediately upon reception
TxPDO	0	Acyclic, Synchronous	If the source data has changed, the TxPDO is transmitted on reception of a SYNC object
	1 – 240	Cyclic, synchronous	The PDO is transferred synchronously and cyclically. The transmission type indicates the number of SYNC objects that are necessary to trigger TxPDOs.
	254/255	Asynchronous, timer trigger	The PDO is Event driven. Events are created by the following: <ul style="list-style-type: none"> • When the value of a mapped object within the PDO changes • If the Event timer is configured and used to generate periodic transmissions



7.2.3 Event Time

Independent of the Transmission Type, an Event Timer can be used to generate TxPDOs.

- Event Timer value of 0 disables the Event Timer function.
- Event Timer value greater than 0 sets the fixed interval for the TxPDO to be transmitted.

TxPDO #1		
COB ID	447	<input checked="" type="checkbox"/> Default
Mode	Enable	
Transmission Type	255	
Event Time	0	ms
Inhibit Time	0	(x100us)

Figure 15: Event Timer

7.2.4 Inhibit Time

The Inhibit Time is used to prohibit the TxPDO being transmitted “back to back”, i.e. Sets the minimum time between TxPDO transmissions. Such transmission are likely when using a Transmission Type that utilise Change Of State (COS).

Setting a value greater than 0 automatically activates the function so that on a TxPDO transmission an “inhibit timer” is started and the next transmission will not occur until the timer expires.

TxPDO #1		
COB ID	447	<input checked="" type="checkbox"/> Default
Mode	Enable	
Transmission Type	255	
Event Time	0	ms
Inhibit Time	0	(x100us)

Figure 16: Inhibit Time



PDO Access

7.3 Mapping PDOs

7.3.1 Amount and Size of PDOs

The CANopen module supports up to 8 RxPDOs and 8 TxPDOs. Each PDO can map up to 64-bits of data, in the case of PositionServo, the MotionView interface automatically controls how many object mapping selectors are available to prevent overmapping / oversizing the PDO.

7.3.2 Receive (Rx) PDOs

The CANopen module can map up to a maximum of 8 RxPDOs.

RxPDO mapping is set via the MVOB [Communications] [CANOpen] folder. Each mapped object selector lists all the CANopen objects that are available.

RxPDO #1

COB ID	575	<input checked="" type="checkbox"/> Default
Mode	Enable	
Transmission Type	255	
1st Mapped Object	2042 2 2rw OUTPUTS_LSB	
2nd Mapped Object	2042 3 2rw OUTPUTS_MSB	
3rd Mapped Object	2064 2 2rw V0_LSB	
4th Mapped Object	2064 3 2rw V0_MSB	

Figure 17: CAN RxPDO Mapping - Four 16-bit Objects

RxPDO #1

COB ID	575	<input checked="" type="checkbox"/> Default
Mode	Enable	
Transmission Type	255	
1st Mapped Object	2034 1 4w ENABLE	
2nd Mapped Object	2034 2 2w ENABLE_LSB	
3rd Mapped Object	2042 2 2rw OUTPUTS_LSB	
4th Mapped Object	Not Used	

Figure 18: CAN RxPDO Mapping - 64bit limit reached



7.3.3 Transmit (Tx) PDOs

The CANopen module can map up to a maximum of 8 TxPDOs.

TxPDO mapping is set via the MVOB [Communications] [CANopen] folder. Each mapped object selector lists all the CANopen objects that are available.

TxPDO #1	
COB ID	447 <input checked="" type="checkbox"/> Default
Mode	Enable
Transmission Type	255
Event Time	0 ms
Inhibit Time	0 (x100us)
1st Mapped Object	2053 2 2r EXSTATUS_LSB
2nd Mapped Object	2053 3 2r EXSTATUS_MSB
3rd Mapped Object	2041 2 2r INPUTS_LSB
4th Mapped Object	2041 3 2r INPUTS_MSB

Figure 19: CAN TxPDO Mapping - Four 16-bit Objects

TxPDO #1	
COB ID	447 <input checked="" type="checkbox"/> Default
Mode	Enable
Transmission Type	255
Event Time	0 ms
Inhibit Time	0 (x100us)
1st Mapped Object	2009 1 4r FAULT_STATUS
2nd Mapped Object	2036 2 2r STATUS_LSB
3rd Mapped Object	2036 2 2r STATUS_LSB
4th Mapped Object	Not Used

Figure 20: CAN TxPDO Mapping - 64bit limit reached



Emergency Objects

8 Emergency Objects

8.1 What is an Emergency Object?

- Emergency Objects is the name given to the method used to provide diagnostic information when an error or trip(s) occurs.
- Emergency Objects are produced by slave devices upon a fault or error condition so that appropriate action can be taken by the network master.
- Each Emergency Objects has a unique identifier (COB ID) based on the object range (0x80) + the device producing the objects' node address.
- When transmitted the Emergency Object message carries the errors codes as defined by the CANopen DS301 specification and when applicable additional manufacture specific error data.
- An emergency object is transmitted only once per 'error event'. As long as no new errors occur on a device no further emergency objects will be transmitted.
- When a PositionServo generates an Emergency Object the drives error code will be duplicated in object 0x1003

8.2 Emergency Object format

The Emergency Object consists of 8 bytes and is structured as illustrated in Figure 21.



Figure 21: Structure of an Emergency Object

8.2.1 Error Code

Table 25 lists all the error codes that are support by PositionServo.

Table 25: Emergency Error Codes

Error Code	Definitions
0x0000	No error or Error reset
0x1000	Generic Error, will be followed by the PositionServo trip number in the Manufacture Specific Error Field byte 3
0x6080	Device Software error, will be followed additional error codes in the Manufacture Specific Error Field bytes 3 to 4, refer to Table 26
0x8130	Heartbeat Error
0x8210	PDO not processed due to length error



8.2.2 Error Register

The Error register byte is also available as object 0x1001.

The error register is used to indicate that an error has occurred. If bit 0 is set to 1, then an error has occurred.

In addition to object 0x1001, object 0x1003 records the last drive trip to occur. Refer to section 5.2.3 for further details.

8.2.3 Manufacture Specific Error Field

Table 26 lists Manufacture Error Field codes for a Device Software Error Emergency Object (0x6080).

Table 26: Manufacture Error Field codes for 0x6080

Error Code	Description
0x0101	Error Register: Object 0x1001 sub 0x00 not found
0x0102	Heartbeat: Object 0x1017 sub 0x00 not found
0x0105	Fail to init CAN interface
0x0106	Fail to set receive filter for NMT master message
0x0107	Fail to set receive filter for SYNC message
0x0108	Fail to set receive filter for SDO requests
0x010A	Fail to send Boot-up Message
0x010B	Fail to transmit first heartbeat message after it is enabled or time expired
0x010C	Fail to reply to node guarding message
0x010D	Fail to transmit emergency Error code 0x8130
0x010E	Fail to transmit emergency Error code 0x8210 (Emergency signaled by RPDO)
0x0121	SDO Manager related Object 0x1F00 sub 0x00
0x0121	Object 0x1F01 sub 0x00 not found
0x0122	Fail to set filter for dynamic SDO requests
0x0181	Number of Heartbeat Consumer: Object 0x1016 sub 0x01 not found
0x0181	Number of Emergency Consumer: Object 0x1028 sub 0x01 not found
0x0301	Fail to transmit response to SDO abort message
0x0302	Fail to transmit response to SDO response message (Byte)
0x0303	Fail to transmit response to SDO response message
0x0304	Fail to transmit response to SDO expedited write conformation message
0x0305	Fail to transmit response to SDO segmented write conformation message
0x0306	Fail to transmit response to SDO expedited Communication parameter read request
0x0308	Fail to set receive filter for RPDO
0x0309	Fail to transmit response PDO Mapped parameter read request
0x030A	Fail to transmit response to Parameter Read request
0x030B	Fail to transmit response to SYNC ID read/write request Object 0x1005 sub 0x00
0x030C	Fail to transmit response to Node Guarding read request Object 0x100C sub 0x00
0x030D	Fail to transmit SDO response message



Emergency Objects

Error Code	Description
0x0336	No TPDO are available (Communication parameters)
0x0337	No RPDO are available (Communication parameters)
0x0338	No TPDO are available (Mapped Parameters)
0x0339	No RPDO are available (Mapped Parameters)
0x0402	Fail to set receive filter for RPDO
0x0403	Fail to transmit PDO
0x0404	In the incoming message length of PDO message is wrong or PDO number is not valid
0x0405	TPDO trigger is wrong
0x0406	TPDO number is wrong
0x0407	TPDO number is wrong
0x0501	Not enough storage space in Non-Volatile memory
0x0701	NMT command is not valid
0x0702	NMT transmit queue overrun
0x2201	Length of Rx message is greater than 8
0xC105	CAN Rx queue overrun
0xF200	Fail to transmit emergency Error code 0x0000 (with no error and error reset)

8.3 Emergency Object Examples

8.3.1 Example 1: Hardware Disable

The drive trips “F036” due to hardware disable while Indexer Code software enable is still active.

Table 27: Emergency Object for trip “F036”

COB ID	Bytes 0-1	Byte 2	Bytes 3-7
0x81	0x00 0x10	0x01	0x24 0x00 0x00 0x00
EMCY source	Error Code	Error Register	Manufacture Specific Error Field
Node 1	0x1000	0x01	0x24 (36 _{dec})

Emergency Objects



8.3.2 Example 2: Limit Switch

The drive trips “F032” due to positive limit switch.

Table 28: Emergency Object for trip “F032”

COB ID	Bytes 0-1	Byte 2	Bytes 3-7
0x81	0x00 0x10	0x01	0x20 0x00 0x00 0x00
EMCY source	Error Code	Error Register	Manufacture Specific Error Field
Node 1	0x1000	0x01	0x20 (32 _{dec})

8.3.3 Example 3: CAN Receive buffer overrun

Drive receive buffer overrun due to excess incoming data.

Table 29: Emergency Object for CAN Rx Buffer Overrun

COB ID	Bytes 0-1	Byte 2	Bytes 3-7
0x81	0x80 0x60	0x01	0x05 0xC1 0x00 0x00
EMCY source	Error Code	Error Register	Manufacture Specific Error Field
Node 1	0x6080	0x01	0xC105 (49413 _{dec})



Drive Control and Status

9 Drive Control and Status

9.1 Overview

The control and status words provide a means for the digital control and monitoring of the drive using a single data word. Each control bit has a particular function and provides a method of controlling the output functions of the drive, such as run and direction. Each bit in the status word provides feedback about the drive's state of health and operational condition.

9.2 Control BITS

There are several control bits available within PositionServo that can be written to through PDO or SDO communications. Some of the most commonly used ones are listed as follows, for a complete list of drive control functions refer to the PositionServo Programming Manual.

9.2.1 Software Enable/Disable

PID52 - Enable			
Default:	N/A	Range:	0 - 1
Access:	WO	Type:	Integer

This is the VAR_ENABLE function.

- 0 - disable
- 1 - enable

This function is the default mapping for TxPDO 1.

9.2.2 Drive Reset (Cold Boot)

PID53 - Reset			
Default:	N/A	Range:	0 - 1
Access:	WO	Type:	Integer

This is the VAR_RESET function.

- 0 - no action
- 1 - reset drive

9.2.3 Suspend Motion

PID91 - Suspend Motion			
Default:	0	Range:	0 - 1
Access:	RW	Type:	Integer

This is the VAR_SUSPEND_MOTION function.

- 0 - motion enabled
- 1 - motion disabled



9.2.4 Stop Motion

PID136 - Stop Motion			
Default:	N/A	Range:	0 - 1
Access:	WO	Type:	Integer

This is the VAR_STOP_MOTION function.

- 0 - no action
- 1 - stops motion

9.3 Status Word

There are several status words and individual status bits/flags available within PositionServo that can be read from through PDO or SDO communications.

9.3.1 Status Flags Register

PID54 - DSTATUS			
Default:	N/A	Range:	
Access:	RO	Type:	Integer

This is the VAR_DSTATUS function.

Table 30: DSTATUS Register

Bit in register	Description
0	Set when drive enabled
1	Set if DSP subsystem at any fault
2	Set if drive has a valid program
3	Set if byte-code or system or DSP at any fault
4	Set if drive has a valid source code
5	Set if motion completed and target position is within specified limits
6	Set when scope is triggered and data collected
7	Set if motion stack is full
8	Set if motion stack is empty
9	Set if byte-code halted
10	Set if byte-code is running
11	Set if byte-code is set to run in step mode
12	Set if byte-code is reached the end of program
13	Set if current limit is reached
14	Set if byte-code at fault
15	Set if no valid motor selected
16	Set if byte-code at arithmetic fault
17	Set if byte-code at user fault
18	Set if DSP initialization completed
19	Set if registration has been triggered



Drive Control and Status

Bit in register	Description
20	Set if registration variable was updated from DSP after last trigger
21	Set if motion module at fault
22	Set if motion suspended
23	Set if program requested to suspend motion
24	Set if system waits completion of motion
25	Set if motion command completed and motion Queue is empty
26	Set if byte-code task requested reset
27	If set interface control is disabled. This flag is set/clear by ICONTROL ON/OFF statement.
28	Set if positive limit switch reached
29	Set if negative limit switch reached
30	Events disabled. All events disabled when this flag is set. After executing EVENTS ON all events previously enabled by EVENT EventName ON statements become enabled again

9.3.2 Extended Status Bits

PID84 - DEXSTATUS			
Default:	N/A	Range:	
Access:	RO	Type:	Integer

This is the VAR_EXSTATUS function

Table 31: Encoding for Extended Status Bits

Bit #	Function	Comment
0	Reserved	
1	Velocity in specified window	Velocity in limits as per parameter #59: VAR_VLIMIT_SPEEDWND
2-4	Reserved	
5	Velocity at 0 (zero)	Velocity 0: Zero defined by parameter #58: VAR_VLIMIT_ZEROSPEED
6,7	Reserved	
8	Bus voltage below under-voltage limit	Utilized to indicate drive is operating from +24V keep alive and a valid DC bus voltage level is not present.
9,10	Reserved	
11	Regen circuit is on	Drive regeneration circuit is active. Drive will be dissipating power through the braking resistor (if fitted).
12-20	Reserved	
21	Set if homing operation in progress	Drive executing Pre-defined homing function (refer to section 2.15, PS Programming Manual, PM94M01).
22	Set if system homed	Drive completed Pre-defined homing function (refer to section 2.15, PS Programming Manual, PM94M01).
23	If set then last fault will remain on the display until re-enabled.	User can set this bit to retain fault code on the display until re-enabled. It is useful if there is a fault handler routine. When the fault handler is exited, the fault number on the display will be replaced by current status (usually DiS if bit #24 is not set). Setting bit #24 retains diagnostics on the display.
24	Set if EIP IO exclusive owner connection is established. Cleared if closed.	Checks if drive is controlled by EthernetIP master. Use bit #25 and bit #26 to process "lost of connection" condition (if needed) in the user's program
25	Set if EIP IO exclusive owner connection times out. Cleared if exc. owner conn exists.	Checks if connection with Ethernet/IP master is lost. Use bit #26 and bit #25 to process "lost of connection" condition (if needed) in the user's program
26-30	Reserved	



10 Advanced Features

10.1 CAN Baud Rate

PID234 - CAN_BAUD_EPM			
Default:	63	Range:	0 - 127
Access:	RW	Type:	Integer

This is VAR_ CAN_OPERMODE_EPM. Its function is the same as the CAN Boot-up Mode from within MotionView as described in section 4.3.5.

If editing from within MotionView, select the appropriate baud rate from the drop down menu. If editing PID234 directly, use Table 32 for the correct settings.

Table 32: CAN Baud Rate

PID234 Value	Baud Rate
1	10kbps
2	20kbps
3	50kbps
4	125kbps
5	250kbps
6	500kbps
7	800kbps
8	1Mbps

10.2 CAN Node Address

PID235 - CAN_ADD_EPM			
Default:	63	Range:	0 - 127
Access:	RW	Type:	Integer

This is VAR_ CAN_OPERMODE_EPM. Its function is the same as the CAN Boot-up Mode from within MotionView as described in section 4.3.5.

Set PID235 to the required value. The default address is 63. The permissible address range is: 0 – 127. Each node on the network must have an individual address, if two or more nodes have duplicate addresses this may prevent the network from functioning correctly.

10.3 CAN Boot-up Mode

PID236 - CAN_OPERMODE_EPM			
Default:	0	Range:	0 - 2
Access:	RW	Type:	Integer

This is VAR_ CAN_OPERMODE_EPM. Its function is the same as the CAN Boot-up Mode from within MotionView as described in section 4.3.7



Advanced Features

Table 33: PID236 CAN Bootup Mode

PID236 Value	Mode	Comment
0	Pre-Operational	Drive enters the "PRE_OPERATION" state after bootup
1	Operational	Drive enters the "OPERATIONAL" state after bootup
2	Pseudo Master	In addition to the drive entering the "OPERATIONAL" state after bootup the drive will also (after the delay period set by the CAN Bootup Delay parameter) broadcast a command for all other CANopen nodes to go in to the "OPERATIONAL" state

10.4 CAN Boot-up Delay

PID237 - CAN_OPERDELAY_EPM			
Default:	0	Range:	0 - 5
Access:	RW	Type:	Integer

This is VAR_CAN_OPERDELAY_EPM. Its function is the same as the CAN Boot-up Delay from within MotionView as described in section 4.3.8

Set PID237 to the required value. The default delay time is 3 seconds. The permissible delay time is: 0 – 5(sec). The Bootup Delay function is only functional when the drive is used in pseudo master mode. The Bootup Delay parameter sets the time delay from when the drive itself boots up and is fully functional to when it broadcasts the NMT command for all slave devices to go to the operational state.

10.5 Communication Module Selection

PID238 - CAN_ENABLE_EPM			
Default:	0	Range:	0 - 4
Access:	RW	Type:	Integer

This is VAR_CAN_ENABLE_EPM. Its function is the same as the module section from within MotionView as described in section 4.3.3.

Table 34: PID238 Serial Fieldbus Selection

PID238 Value	Fieldbus Selection
0	None / RS485
1	CANopen Simple
2	Reserved
3	DeviceNet
4	PROFIBUS-DP

The Fieldbus Selection is also accessible from the drive keypad using parameter "CANF".



10.6 PDO Configuration

The drive's PDOs can be configured in 4 ways:

- With MotionView simple to use drop down menus
- Direct from a CAN NMT master*
- From within the PositionServo Index program
- Via Ethernet communications



NOTE: * - PDO configuration settings from a NMT CAN master is volatile.

Configuring the PDO setting via all other sources actual interfaces with the following parameters

10.6.1 PDO COB-ID, Activation and Transmission Type

PID311 to PID318 - RPDO1 to 8 COM			
Default:	Various	Range:	0 - 0xFFFFFFFF
Access:	RW	Type:	Integer

PID351 to PID358 - TPDO1 to 8 COM			
Default:	Various	Range:	0 - 0xFFFFFFFF
Access:	RW	Type:	Integer

The PDO COM Parameter is divided in to the functions listed in Table 35.



Advanced Features

Table 35: PDO COM Functions

Byte	Nibble	Bit	Function	Description																					
0	0	0	COB-ID	0x000 - Set if default Mode is enabled 0x001 to 0xFFFF - COB-ID set for non-default mode																					
		1																							
		2																							
		3																							
	1	4																							
		5																							
		6																							
1	2	7			Mode	4 - Default 5 - non-default / unlocked																			
		8																							
		9																							
		10																							
	3	11																							
		12																							
		13																							
2	4	14	Transmission Type	<table border="1"> <thead> <tr> <th></th> <th>Transmission Type</th> <th>Scheduling</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td rowspan="2">RxPDO</td> <td>0 – 240</td> <td>Synchronous</td> <td>The received data is held until the next SYNC message. When the SYNC message is received the data is applied</td> </tr> <tr> <td>254 – 255</td> <td>Asynchronous, timer trigger</td> <td>The received data is applied to its mapped objects immediately upon reception</td> </tr> <tr> <td rowspan="3">TxPDO</td> <td>0</td> <td>Acyclic, Synchronous</td> <td>If the source data has changed, the TxPDO is transmitted on reception of a SYNC object</td> </tr> <tr> <td>1 – 240</td> <td>Cyclic, synchronous</td> <td>The PDO is transferred synchronously and cyclically. The transmission type indicates the number of SYNC objects that are necessary to trigger TxPDOs.</td> </tr> <tr> <td>254/255</td> <td>Asynchronous, timer trigger</td> <td>The PDO is Event driven. Events are created by the following: <ul style="list-style-type: none"> • When the value of a mapped object within the PDO changes • If the Event timer is configured and used to generate periodic transmissions </td> </tr> </tbody> </table>		Transmission Type	Scheduling	Description	RxPDO	0 – 240	Synchronous	The received data is held until the next SYNC message. When the SYNC message is received the data is applied	254 – 255	Asynchronous, timer trigger	The received data is applied to its mapped objects immediately upon reception	TxPDO	0	Acyclic, Synchronous	If the source data has changed, the TxPDO is transmitted on reception of a SYNC object	1 – 240	Cyclic, synchronous	The PDO is transferred synchronously and cyclically. The transmission type indicates the number of SYNC objects that are necessary to trigger TxPDOs.	254/255	Asynchronous, timer trigger	The PDO is Event driven. Events are created by the following: <ul style="list-style-type: none"> • When the value of a mapped object within the PDO changes • If the Event timer is configured and used to generate periodic transmissions
					Transmission Type	Scheduling	Description																		
		RxPDO			0 – 240	Synchronous	The received data is held until the next SYNC message. When the SYNC message is received the data is applied																		
	254 – 255				Asynchronous, timer trigger	The received data is applied to its mapped objects immediately upon reception																			
	TxPDO	0			Acyclic, Synchronous	If the source data has changed, the TxPDO is transmitted on reception of a SYNC object																			
		1 – 240			Cyclic, synchronous	The PDO is transferred synchronously and cyclically. The transmission type indicates the number of SYNC objects that are necessary to trigger TxPDOs.																			
		254/255			Asynchronous, timer trigger	The PDO is Event driven. Events are created by the following: <ul style="list-style-type: none"> • When the value of a mapped object within the PDO changes • If the Event timer is configured and used to generate periodic transmissions 																			
	16																								
	17																								
	5	19			18	Reserved	Not used																		
					20																				
21																									
22																									
		23																							



10.6.2 PDO Mapping

RxPDO Mapping

Mapping PIDs 319 - 348 are configured internally by the GUI. Users do **not** set these PIDs directly. Refer to section 7.3 for proper operation.

TxPDO Mapping

Mapping PIDs 359 - 388 are configured internally by the GUI. Users do **not** set these PIDs directly. Refer to section 7.3 for proper operation.

10.6.3 TPDO Event Time and Inhibit Time

PID391 to PID398 - TPDO1-8 Event Times			
Default:	0	Range:	0 - 65535
Access:	RW	Type:	Integer

Independent of the Transmission Type, an Event Timer can be used to generate TxPDOs.

- Event Timer value of 0 disables the Event Timer function.
- Event Timer value greater than 0 sets the fixed interval in milliseconds (ms) for the TxPDO to be transmitted.

PID399 to PID406 - TPDO1-8 Inhibit Times			
Default:	0	Range:	0 - 65535
Access:	RW	Type:	Integer

The Inhibit Time is used to prohibit the TxPDO being transmitted “back to back”, i.e. Sets the minimum time (in multiples of 100 microseconds, μs) between TxPDO transmissions. Such transmission are likely when using a Transmission Type that utilise Change Of State (COS).

Setting a value greater than 0 automatically activates the function so that on a TxPDO transmission an “inhibit timer” is started and the next transmission will not occur until the timer expires.



11 Reference

11.1 PID List with CANopen Values

This is a condensed PID List to show the corresponding CANopen Registers for PIDs 1-413. For the complete variable list refer to the PositionServo Programming Manual (PM94P01 or PM94M01).

These variables can be accessed from the user's program or any supported communications interface protocol. From the user program, any variable can be accessed by either its variable name or by its index value (using the syntax: @<VARINDEX> , where <VARINDEX> is the variable index from the PID List. From the communications interface any variable can be accessed by its index value.

The column "**Type**" indicates the type of variable:

- mtr Motor: denotes a motor value
- mtn Motion: writing to an "mtn" variable could cause the start of motion ⚠
- vel Velocity: denotes a velocity or velocity scaling value

The column "**Format**" provides the native format of the variable:

- W 32 bit integer
- F float (real)

When setting a variable via an external device the value can be addressed as floating or integer. The value will automatically adjusted to fit its given form.

The column "**EPM**" shows if a variable has a non-volatile storage space in the EPM memory:

- Y Variable has non-volatile storage Space in EPM
- N Variable does not exist in EPM memory

The user's program uses a RAM (volatile) 'copy' of the variables stored on the EPM. At power up all RAM copies of the variables are initialized with the EPM values. The EPM's values are not affected by changing the variables in the user's program. When the user's program reads a variable it always reads from the RAM (volatile) copy of the variable. Communications Interface functions can change both the volatile and non-volatile copy of the variable. If the host interface requests a change to the EPM (non-volatile) value, this change is done both in the user program's RAM memory as well as in the EPM. Interface functions have the choice of reading from the RAM (volatile) or from the EPM (non-volatile) copy of the variable.

The column "**Access**" lists the user's access rights to a variable:

- R read only
- W write only
- R/W read/write

Writing to an R (read-only) variable or reading from a W (write-only) variable will not work.

The column "**Units**" shows units of the variable. Units unique to this manual that are used for motion are:

- UU user units
- EC encoder counts
- S seconds
- PPS pulses per sample. Sample time is 512 μ s - servo loop rate
- PPSS pulses per sample per sample. Sample time is 512 μ s - servo loop rate

Parameter Reference



Table 36: PID List with CANopen Values

Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
1	VAR_IDSTRING			N	R	Drive's identification string			2001	2401	3001	3401
2	VAR_NAME			Y	R/W	Drive's symbolic name			2002	2402	3002	3402
3	VAR_SERIAL_NUMBER				R	Drive's serial number			2003	2403	3003	3403
4	VAR_MEM_INDEX				R/W	Position in RAM file	(0 - 32767)		2004	2404	3004	3404
5	VAR_MEM_VALUE				R/W	Value to be read or written to the RAM file			2005	2405	3005	3405
6	VAR_MEM_INDEX_INCREMENT				R/W	Holds value the MEM_INDEX will modify once the R/W operation is complete			2006	2406	3006	3406
7	VAR_VELOCITY_ACTUAL		F	N	R	Actual measured motor velocity		UU/sec	2007	2407	3007	3407
8	VAR_RSVD_2								2008	2408	3008	3408
9	VAR_DEFAULT				R	Drive Default Settings			2009	2409	3009	3409
10	VAR_M_ID	mtr		Y	R/W*	Motor ID			200A	240A	300A	340A
11	VAR_M_MODEL	mtr		Y	R/W*	Motor model			200B	240B	300B	340B
12	VAR_M_VENDOR	mtr		Y	R/W*	Motor vendor			200C	240C	300C	340C
13	VAR_M_ESET	mtr		Y	R/W*	Motor Feedback Resolver: *Positive for CW	0 - none 1 - Positive for CW		200D	240D	300D	340D
14	VAR_M_HALLCODE	mtr		Y	R/W*	Hallcode index	Range: 0 - 5		200E	240E	300E	340E
15	VAR_M_HOFFSET	mtr		Y	R/W*	Reserved			200F	240F	300F	340F
16	VAR_M_ZOFFSET	mtr		Y	R/W*	Resolver Offset	Range: 0 - 360		2010	2410	3010	3410
17	VAR_M_ICTRL	mtr		Y	R/W*	Reserved			2011	2411	3011	3411
18	VAR_M_JM	mtr		Y	R/W*	Motor moment of inertia Jm	Range: 0 - 0.1	Kgm2	2012	2412	3012	3412
19	VAR_M_KE	mtr		Y	R/W*	Motor voltage or back EMF constant Ke	Range: 1 - 500	V/Krpm	2013	2413	3013	3413
20	VAR_M_KT	mtr		Y	R/W*	Motor torque or force constant Kt	Range: 0.01 - 10	Nm/A	2014	2414	3014	3414
21	VAR_M_LS	mtr		Y	R/W*	Motor phase-to-phase inductance Lm	Range: 0.1 - 500	mH	2015	2415	3015	3415
22	VAR_M_RS	mtr		Y	R/W*	Motor phase-to-phase resistance Rm	Range: 0.01 - 500	[Ohm]	2016	2416	3016	3416
23	VAR_M_MAXCURRENT	mtr		Y	R/W*	Motor's max current(RMS)	Range: 0.5 - 50	[A]mp	2017	2417	3017	3417
24	VAR_M_MAXVELOCITY	mtr		Y	R/W*	Motor's max velocity	Range: 500 - 20000	RPM	2018	2418	3018	3418
25	VAR_M_NPOLES	mtr		Y	R/W*	Motor's poles number	Range: 2 - 200		2019	2419	3019	3419
26	VAR_M_ENCODER	mtr		Y	R/W*	Encoder resolution	Range: 256 - 65536 * 12/ Npoles	PPR	201A	241A	301A	341A
27	VAR_M_TERMVOLTAGE	mtr		Y	R/W*	Nominal Motor's terminal voltage	Range: 50 - 800	[V]olt	201B	241B	301B	341B
28	VAR_M_FEEDBACK	mtr		Y	R/W*	Feedback type	1 - Encoder 2 - Resolver		201C	241C	301C	341C
29	VAR_ENABLE_SWITCH_TYPE		W	Y	R/W	Enable switch function	0 - inhibit only 1 - Run	Bit	201D	241D	301D	341D
30	VAR_CURRENTLIMIT		F	Y	R/W	Current limit		[A]mp	201E	241E	301E	341E
31	VAR_PEAKCURRENTLIMIT16		F	Y	R/W	Peak current limit for 16kHz operation		[A]mp	201F	241F	301F	341F
32	VAR_PEAKCURRENTLIMIT		F	Y	R/W	Peak current limit for 8kHz operation		[A]mp	2020	2420	3020	3420
33	VAR_PWMFREQUENCY		W	Y	R/W	PWM frequency selection	0 - 16kHz 1 - 8kHz		2021	2421	3021	3421
34	VAR_DRIVEMODE		W	Y	R/W	Drive mode	0 - torque 1 - velocity 2 - position		2022	2422	3022	3422
35	VAR_CURRENT_SCALE		F	Y	R/W	Analog input #1 current reference scale	Range: Model Dependent	A/V	2023	2423	3023	3423
36	VAR_VELOCITY_SCALE	vel	F	Y	R/W	Analog input #1 velocity reference scale	Range: -10000 to +10000	RPM/V	2024	2424	3024	3424
37	VAR_REFERENCE		W	Y	R/W	Reference selection	1 - internal source 0 - external		2025	2425	3025	3425
38	VAR_STEPINPUTTYPE		W	Y	R/W	Selects how position reference inputs operating	0 - Quadrature inputs (A/B) 1 - Step & Direction		2026	2426	3026	3426



Parameter Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
39	VAR_MOTORTHERMALPROTECT		W	Y	R/W	Motor thermal protection function	0 - disabled 1 - enabled		2027	2427	3027	3427
40	VAR_MOTORPTCRESTANCE		F	Y	R/W	Motor thermal protection PTC cut-off resistance		[Ohm]	2028	2428	3028	3428
41	VAR_SECONDCODER		W	Y	R/W	Second encoder	0 - disabled 1 - enabled		2029	2429	3029	3429
42	VAR_REGENDUTY		W	Y	R/W	Regen circuit PWM duty cycle in %	Range: 1-100%	%	202A	242A	302A	342A
43	VAR_ENCODERREPEATSRC		W	Y	R/W	Selects source for repeat buffers	0 - Model 940 - Encoder Port P4 Model 941 - 2nd Encoder Option Bay 1 - Model 940 - 2nd Encoder Option Bay Model 941 - Resolver Port P4		202B	242B	302B	342B
44	VAR_VP_GAIN	vel	W	Y	R/W	Velocity loop Proportional gain	Range: 0 - 32767		202C	242C	302C	342C
45	VAR_VI_GAIN	vel	W	Y	R/W	Velocity loop Integral gain	Range: 0 - 32767		202D	242D	302D	342D
46	VAR_PP_GAIN		W	Y	R/W	Position loop Proportional gain	Range: 0 - 32767		202E	242E	302E	342E
47	VAR_PI_GAIN		W	Y	R/W	Position loop Integral gain	Range: 0 - 16383		202F	242F	302F	342F
48	VAR_PD_GAIN		W	Y	R/W	Position loop Differential gain	Range: 0 - 32767		2030	2430	3030	3430
49	VAR_PI_LIMIT		W	Y	R/W	Position loop integral gain limit	Range: 0 - 20000		2031	2431	3031	3431
50	VAR_SEI_GAIN								2032	2432	3032	3432
51	VAR_VREG_WINDOW	vel	W	Y	R/W	Gains scaling coefficient	Range: -16 to +4		2033	2433	3033	3433
52	VAR_ENABLE		W	N	W	Software Enable/Disable	0 - disable 1 - enable		2034	2434	3034	3434
53	VAR_RESET		W	N	W	Drive's reset (Disables drive, Stops running program if any, reset active fault)	0 - no action 1 - reset drive		2035	2435	3035	3435
54	VAR_STATUS		W	N	R	Drive's status register			2036	2436	3036	3436
55	VAR_BCF_SIZE		W	Y	R	User's program Byte-code size		Bytes	2037	2437	3037	3437
56	VAR_AUTOBOOT		W	Y	R/W	User's program autostart flag	0 - program started manually (MotionView or interface) 1 - program started automatically after drive booted		2038	2438	3038	3438
57	VAR_GROUPID		W	Y	R/W	Network group ID	Range: 1 - 32767		2039	2439	3039	3439
58	VAR_VLIMIT_ZEROSPEED		F	Y	R/W	Zero Speed window	Range: 0 - 100	Rpm	203A	243A	303A	343A
59	VAR_VLIMIT_SPEEDWND		F	Y	R/W	At Speed window	Range: 10 - 10000	Rpm	203B	243B	303B	343B
60	VAR_VLIMIT_ATSPEED		F	Y	R/W	Target Velocity for At Speed window	Range: -10000 - +10000	Rpm	203C	243C	303C	343C
61	VAR_PLIMIT_POSERROR		W	Y	R/W	Position error	Range: 1 - 32767	EC	203D	243D	303D	343D
62	VAR_PLIMIT_ERRORTIME		F	Y	R/W	Position error time (time which position error has to remain to set-off position error fault)	Range: 0.25 - 8000	mS	203E	243E	303E	343E
63	VAR_PLIMIT_SEPOSERROR		W	Y	R/W	Second encoder Position error	Range: 1 - 32767	EC	203F	243F	303F	343F
64	VAR_PLIMIT_SEERRORTIME		F	Y	R/W	Second encoder Position error time (time which position error has to remain to set-off position error fault)	Range: 0.25 - 8000	mS	2040	2440	3040	3440
65	VAR_INPUTS		W	N	R	Digital inputs states. A1 occupies Bit 0, A2-Bit 1 ... C4 - bit 11.			2041	2441	3041	3441

Parameter Reference



Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
66	VAR_OUTPUT		W	N	R/W	Digital outputs states. Writing to this variables sets/resets digital outputs except outputs which have been assigned special function.	Output 1 Bit 0 Output 2 Bit 1 Output 3 Bit 2 Output 4 Bit 3		2042	2442	3042	3442
67	VAR_IP_ADDRESS		W	Y	R/W	Ethernet IP address. IP address changes at next boot up. 32 bit value			2043	2443	3043	3443
68	VAR_IP_MASK		W	Y	R/W	Ethernet IP NetMask. Mask changes at next boot up. 32 bit value			2044	2444	3044	3444
69	VAR_IP_GATEWAY		W	Y	R/W	Ethernet Gateway IP address. Address changes at next boot up. 32 bit value			2045	2445	3045	3445
70	VAR_IP_DHCP		W	Y	R/W	Use DHCP	0 - manual 1 - use DHCP service		2046	2446	3046	3446
71	VAR_AIN1		F	N	R	Analog Input AIN1 current value		[V]olt	2047	2447	3047	3447
72	VAR_AIN2		F	N	R	Analog Input AIN2 current value		[V]olt	2048	2448	3048	3448
73	VAR_BUSVOLTAGE		F	N	R	Bus voltage		[V]olt	2049	2449	3049	3449
74	VAR_HTEMP		F	N	R	Heatsink temperature	Returns: 0 - for temperatures < 40C and actual heat sink temperature for temperatures >40 C	[c]	204A	244A	304A	344A
75	VAR_ENABLE_ACCELDECEL	vel		Y	R/W	Enable Accel/Decel function for velocity mode	0 - disable 1 - enable		204B	244B	304B	344B
76	VAR_ACCEL_LIMIT	vel	F	Y	R/W	Accel value for velocity mode	Range: 0.1 - 5000000	Rpm*Sec	204C	244C	304C	344C
77	VAR_DECEL_LIMIT	vel	F	Y	R/W	Decel value for velocity mode	Range: 0.1 - 5000000	Rpm*Sec	204D	244D	304D	344D
78	VAR_FAULT_RESET		W	Y	R/W	Reset fault configuration	0 - on activation of Enable/Inhibit input (A3) 1 - on deactivation of Enable/Inhibit input (A3)		204E	244E	304E	344E
79	VAR_M2SRATIO_MASTER		W	Y	R/W	Master to system ratio	Master counts range: -32767 - +32767		204F	244F	304F	344F
80	VAR_M2SRATIO_SYSTEM		W	Y	R/W	Master to system ratio	System counts range: 1 - 32767		2050	2450	3050	3450
81	VAR_S2PRATIO_SECOND		W	Y	R/W	Secondary encoder to prime encoder ratio	Second counts range: -32767 - +32767		2051	2451	3051	3451
82	VAR_S2PRATIO_PRIME		W	Y	R/W	Secondary encoder to prime encoder ratio	Prime counts range: 1 - 32767		2052	2452	3052	3452
83	VAR_EXSTATUS		W	N	R	Extended status. Lower word copy of DSP status flags.			2053	2453	3053	3453
84	VAR_HLS_MODE		W	Y	R/W	Hardware limit switches	0 - not used 1 - stop and fault 2 - fault		2054	2454	3054	3454
85	VAR_AOUT_FUNCTION		W	Y	R/W	Analog output function range: 0 - 8	0 - Not assigned 1 - Phase Current (RMS) 2 - Phase Current (Peak Value) 3 - Motor Velocity 4 - Phase Current R 5 - Phase Current S 6 - Phase Current T 7 - Iq current 8 - Id current		2055	2455	3055	3455
86	VAR_AOUT_VELSCALE		F	Y	R/W	Analog output scale for velocity quantities.	Range: 0 - 10	mV/Rpm	2056	2456	3056	3456



Parameter Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
87	VAR_AOUT_CURSCALE		F	Y	R/W	Analog output scale for current related quantities.	Range: 0 - 10	V/A	2057	2457	3057	3457
88	VAR_AOUT		F	N	W	Analog output value.(Used if VAR #85 is set to 0)	Range: 0 - 10	V	2058	2458	3058	3458
89	VAR_AIN1_DEADBAND		F	Y	R/W	Analog input #1 dead-band. Applied when used as current or velocity reference.	Range: 0 - 100	mV	2059	2459	3059	3459
90	VAR_AIN1_OFFSET			Y	R/W	Analog input #1 offset. Applied when used as current/velocity reference	Range: -10,000 to +10,000	mV	205A	245A	305A	345A
91	VAR_SUSPEND_MOTION		W	N	R/W	Suspend motion. Suspends motion produced by trajectory generator. Current move will be completed before motion is suspended.	0 - motion suspended 1 - motion resumed		205B	245B	305B	345B
92	VAR_MOVEP	mtn	W	N	W	Target position for absolute move. Writing value executes Move to position as per MOVEP statement using current values of acceleration, deceleration and max velocity.		UU	205C	245C	305C	345C
93	VAR_MOVED	mtn	W	N	W	Incremental position. Writing value <0> executes Incremental move as per MOVED statement using current values of acceleration, deceleration and max velocity		UU	205D	245D	305D	345D
94	VAR_MDV_DISTANCE		F	N	W	Distance for MDV move		UU	205E	245E	305E	345E
95	VAR_MDV_VELOCITY	mtn	F	N	W	Velocity for MDV move. Writing to this variable executes MDV move with Distance value last written to variable #94		UU	205F	245F	305F	345F
96	VAR_MOVE_PW1	mtn	W	N	W	Writing value executes Move in positive direction while input true (active). Value specifies input #	0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4		2060	2460	3060	3460
97	VAR_MOVE_PW0	mtn	W	N	W	Writing value executes Move in positive direction while input false (not active). Value specifies input #	0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4		2061	2461	3061	3461
98	VAR_MOVE_NW1	mtn	F	N	W	Writing value executes Move negative direction while input true (active). Value specifies input #	0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4		2062	2462	3062	3462
99	VAR_MOVE_NW0	mtn	F	N	W	Writing value executes Move negative direction while input false (not active). Value specifies input #	0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4		2063	2463	3063	3463
100	VAR_V0		F	Y	R/W	User variable			2064	2464	3064	3464
101	VAR_V1		F	Y	R/W	User variable			2065	2465	3065	3465
102	VAR_V2		F	Y	R/W	User variable			2066	2466	3066	3466
103	VAR_V3		F	Y	R/W	User variable			2067	2467	3067	3467
104	VAR_V4		F	Y	R/W	User variable			2068	2468	3068	3468
105	VAR_V5		F	Y	R/W	User variable			2069	2469	3069	3469
106	VAR_V6		F	Y	R/W	User variable			206A	246A	306A	346A
107	VAR_V7		F	Y	R/W	User variable			206B	246B	306B	346B
108	VAR_V8		F	Y	R/W	User variable			206C	246C	306C	346C
109	VAR_V9		F	Y	R/W	User variable			206D	246D	306D	346D
110	VAR_V10		F	Y	R/W	User variable			206E	246E	306E	346E
111	VAR_V11		F	Y	R/W	User variable			206F	246F	306F	346F
112	VAR_V12		F	Y	R/W	User variable			2070	2470	3070	3470
113	VAR_V13		F	Y	R/W	User variable			2071	2471	3071	3471
114	VAR_V14		F	Y	R/W	User variable			2072	2472	3072	3472
115	VAR_V15		F	Y	R/W	User variable			2073	2473	3073	3473
116	VAR_V16		F	Y	R/W	User variable			2074	2474	3074	3474
117	VAR_V17		F	Y	R/W	User variable			2075	2475	3075	3475
118	VAR_V18		F	Y	R/W	User variable			2076	2476	3076	3476
119	VAR_V19		F	Y	R/W	User variable			2077	2477	3077	3477
120	VAR_V20		F	Y	R/W	User variable			2078	2478	3078	3478
121	VAR_V21		F	Y	R/W	User variable			2079	2479	3079	3479
122	VAR_V22		F	Y	R/W	User variable			207A	247A	307A	347A
123	VAR_V23		F	Y	R/W	User variable			207B	247B	307B	347B
124	VAR_V24		F	Y	R/W	User variable			207C	247C	307C	347C
125	VAR_V25		F	Y	R/W	User variable			207D	247D	307D	347D
126	VAR_V26		F	Y	R/W	User variable			207E	247E	307E	347E
127	VAR_V27		F	Y	R/W	User variable			207F	247F	307F	347F

Parameter Reference



Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
128	VAR_V28		F	Y	R/W	User variable			2080	2480	3080	3480
129	VAR_V29		F	Y	R/W	User variable			2081	2481	3081	3481
130	VAR_V30		F	Y	R/W	User variable			2082	2482	3082	3482
131	VAR_V31		F	Y	R/W	User variable			2083	2483	3083	3483
132	VAR_MOVEDR_DISTANCE		F	N	W	Registered move distance.	Incremental motion as per MOVEDR statement	UU	2084	2484	3084	3484
133	VAR_MOVEDR_DISPLACEMENT	mtn	F	N	W	Registered move displacement. Writing to this variable executes the move MOVEDR using value set by #132		UU	2085	2485	3085	3485
134	VAR_MOVEPR_DISTANCE		F	N	W	Registered move distance.	Absolute motion as per MOVEPR statement	UU	2086	2486	3086	3486
135	VAR_MOVEPR_DISPLACEMENT	mtn	F	N	W	Registered move displacement. Writing to this variable makes the move MOVEPR using value set by #134		UU	2087	2487	3087	3487
136	VAR_STOP_MOTION		W	N	W	Stops motion	0 - no action 1 - stops motion		2088	2488	3088	3488
137	VAR_START_PROGRAM		W	N	W	Starts user program	0 - no action 1 - starts program		2089	2489	3089	3489
138	VAR_VEL_MODE_ON		W	N	W	Turns on Profile Velocity for Internal Position Mode	0 - normal operation 1 - velocity mode on		208A	248A	308A	348A
139	VAR_IREF		F	N	W	Reference for Internal Torque or Velocity Mode	0 - Internal Velocity mode 1 - Internal Torque mode	"RPS Amps"	208B	248B	308B	348B
140	VAR_NV0		F	N	R/W	User defined Network variable			208C	248C	308C	348C
141	VAR_NV1		F	N	R/W	User defined Network variable			208D	248D	308D	348D
142	VAR_NV2		F	N	R/W	User defined Network variable			208E	248E	308E	348E
143	VAR_NV3		F	N	R/W	User defined Network variable			208F	248F	308F	348F
144	VAR_NV4		F	N	R/W	User defined Network variable			2090	2490	3090	3490
145	VAR_NV5		F	N	R/W	User defined Network variable			2091	2491	3091	3491
146	VAR_NV6		F	N	R/W	User defined Network variable			2092	2492	3092	3492
147	VAR_NV7		F	N	R/W	User defined Network variable			2093	2493	3093	3493
148	VAR_NV8		F	N	R/W	User defined Network variable			2094	2494	3094	3494
149	VAR_NV9		F	N	R/W	User defined Network variable			2095	2495	3095	3495
150	VAR_NV10		F	N	R/W	User defined Network variable			2096	2496	3096	3496
151	VAR_NV11		F	N	R/W	User defined Network variable			2097	2497	3097	3497
152	VAR_NV12		F	N	R/W	User defined Network variable			2098	2498	3098	3498
153	VAR_NV13		F	N	R/W	User defined Network variable			2099	2499	3099	3499
154	VAR_NV14		F	N	R/W	User defined Network variable			209A	249A	309A	349A
155	VAR_NV15		F	N	R/W	User defined Network variable			209B	249B	309B	349B
156	VAR_NV16		F	N	R/W	User defined Network variable			209C	249C	309C	349C
157	VAR_NV17		F	N	R/W	User defined Network variable			209D	249D	309D	349D
158	VAR_NV18		F	N	R/W	User defined Network variable			209E	249E	309E	349E
159	VAR_NV19		F	N	R/W	User defined Network variable			209F	249F	309F	349F
160	VAR_NV20		F	N	R/W	User defined Network variable			20A0	24A0	30A0	34A0
161	VAR_NV21		F	N	R/W	User defined Network variable			20A1	24A1	30A1	34A1
162	VAR_NV22		F	N	R/W	User defined Network variable			20A2	24A2	30A2	34A2
163	VAR_NV23		F	N	R/W	User defined Network variable			20A3	24A3	30A3	34A3
164	VAR_NV24		F	N	R/W	User defined Network variable			20A4	24A4	30A4	34A4
165	VAR_NV25		F	N	R/W	User defined Network variable			20A5	24A5	30A5	34A5
166	VAR_NV26		F	N	R/W	User defined Network variable			20A6	24A6	30A6	34A6
167	VAR_NV27		F	N	R/W	User defined Network variable			20A7	24A7	30A7	34A7
168	VAR_NV28		F	N	R/W	User defined Network variable			20A8	24A8	30A8	34A8
169	VAR_NV29		F	N	R/W	User defined Network variable			20A9	24A9	30A9	34A9
170	VAR_NV30		F	N	R/W	User defined Network variable			20AA	24AA	30AA	34AA



Parameter Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
171	VAR_NV31		F	N	R/W	User defined Network variable			20AB	24AB	30AB	34AB
172	VAR_SERIAL_ADDRESS		W	Y	R/W	RS485 drive ID	Range: 0 - 254		20AC	24AC	30AC	34AC
173	VAR_MODBUS_BAUDRATE		W	Y	R/W	Baud rate for ModBus operations	0 - 2400 1 - 4800 2 - 9600 3 - 19200 4 - 38400 5 - 57600 6 - 115200		20AD	24AD	30AD	34AD
174	VAR_MODBUS_DELAY		W	Y	R/W	ModBus reply delay in mS	Range: 0 - 1000	mS	20AE	24AE	30AE	34AE
175	VAR_RS485_CONFIG		W	Y	R/W	Rs485 configuration	0 - normal IP over PPP 1 - ModBus1 - 4800		20AF	24AF	30AF	34AF
176	VAR_PPP_BAUDRATE		W	Y	R/W	RS232/485 (normal mode) baud rate	1 - 4800 2 - 9600 3 - 19200 4 - 38400 5 - 57600 6 - 115200		20B0	24B0	30B0	34B0
177	VAR_MOVEPS		F	N	W	Same as variable #92 but using S-curve acceleration/deceleration			20B1	24B1	30B1	34B1
178	VAR_MOVEDS		F	N	W	Same as variable #93 but using S-curve acceleration/deceleration			20B2	24B2	30B2	34B2
179	VAR_MDVS_VELOCITY	mtn		N	W	Velocity for MDV move using S-curve accel/ deceleration. Writing to this variable executes MDV move with Distance value last written to variable #94 (unless motion is suspended by #91).		UU	20B3	24B3	30B3	34B3
180	VAR_MAXVEL		F	N	R/W	Max velocity for motion profile		UU/S	20B4	24B4	30B4	34B4
181	VAR_ACCEL		F	N	R/W	Accel value for indexing		UU/S2	20B5	24B5	30B5	34B5
182	VAR_DECEL		F	N	R/W	Decel value for indexing		UU/S2	20B6	24B6	30B6	34B6
183	VAR_QDECEL		F	N	R/W	Quick decel value		UU/S2	20B7	24B7	30B7	34B7
184	VAR_INPOSLIM		W	N	R/W	Sets window for "In Position" Limits		UU	20B8	24B8	30B8	34B8
185	VAR_VEL		F	N	R/W	Velocity reference for "Profiled" velocity		UU/S	20B9	24B9	30B9	34B9
186	VAR_UNITS		F	Y	R/W	User units			20BA	24BA	30BA	34BA
187	VAR_MECOUNTER		W	N	R/W	A/B inputs reference counter value		Count	20BB	24BB	30BB	34BB
188	VAR_PHCUR		F	N	R	Phase current		A	20BC	24BC	30BC	34BC
189	VAR_POS_PULSES		W	N	R/W	Target position in encoder pulses		EC	20BD	24BD	30BD	34BD
190	VAR_APOS_PULSES		W	N	R/W	Actual position in encoder pulses		EC	20BE	24BE	30BE	34BE
191	VAR_POSEERR_PULSES		W	N	R	Position error in encoder pulses		EC	20BF	24BF	30BF	34BF
192	VAR_CURRENT_VEL_PPS		F	N	R	Set-point (target) velocity in PPS		PPS	20C0	24C0	30C0	34C0
193	VAR_CURRENT_ACCEL_PPSS		F	N	R	Set-point (target) acceleration (demanded value)		PPSS	20C1	24C1	30C1	34C1
194	VAR_IN0_DEBOUNCE		W	Y	R/W	Input A1 de-bounce time in mS	Range: 0 - 1000	mS	20C2	24C2	30C2	34C2
195	VAR_IN1_DEBOUNCE		W	Y	R/W	Input A2 de-bounce time in mS	Range: 0 - 1000	mS	20C3	24C3	30C3	34C3
196	VAR_IN2_DEBOUNCE		W	Y	R/W	Input A3 de-bounce time in mS	Range: 0 - 1000	mS	20C4	24C4	30C4	34C4
197	VAR_IN3_DEBOUNCE		W	Y	R/W	Input A4 de-bounce time in mS	Range: 0 - 1000	mS	20C5	24C5	30C5	34C5
198	VAR_IN4_DEBOUNCE		W	Y	R/W	Input B1 de-bounce time in mS	Range: 0 - 1000	mS	20C6	24C6	30C6	34C6
199	VAR_IN5_DEBOUNCE		W	Y	R/W	Input B2 de-bounce time in mS	Range: 0 - 1000	mS	20C7	24C7	30C7	34C7
200	VAR_IN6_DEBOUNCE		W	Y	R/W	Input B3 de-bounce time in mS	Range: 0 - 1000	mS	20C8	24C8	30C8	34C8
201	VAR_IN7_DEBOUNCE		W	Y	R/W	Input B4 de-bounce time in mS	Range: 0 - 1000	mS	20C9	24C9	30C9	34C9
202	VAR_IN8_DEBOUNCE		W	Y	R/W	Input C1 de-bounce time in mS	Range: 0 - 1000	mS	20CA	24CA	30CA	34CA
203	VAR_IN9_DEBOUNCE		W	Y	R/W	Input C2 de-bounce time in mS	Range: 0 - 1000	mS	20CB	24CB	30CB	34CB
204	VAR_IN10_DEBOUNCE		W	Y	R/W	Input C3 de-bounce time in mS	Range: 0 - 1000	mS	20CC	24CC	30CC	34CC

Parameter Reference



Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
205	VAR_IN11_DEBOUNCE		W	Y	R/W	Input C4 de-bounce time in mS	Range: 0 - 1000	mS	20CD	24CD	30CD	34CD
206	VAR_OUT1_FUNCTION		W	Y	R/W	Programmable Output function	0 - Not Assigned 1 - Zero Speed 2 - In Speed Window 3 - Current Limit 4 - Run time fault 5 - Ready 6 - Brake 7 - In position		20CE	24CE	30CE	34CE
207	VAR_OUT2_FUNCTION		W	Y	R/W	Programmable Output Function			20CF	24CF	30CF	34CF
208	VAR_OUT3_FUNCTION		W	Y	R/W	Programmable Output Function			20D0	24D0	30D0	34D0
209	VAR_OUT4_FUNCTION		W	Y	R/W	Programmable Output Function			20D1	24D1	30D1	34D1
210	VAR_HALLCODE		W	N	R	Current hall code	Bit 0 - Hall 1 Bit 1 - Hall 2 Bit 2 - Hall 3		20D2	24D2	30D2	34D2
211	VAR_ENCODER		W	N	R	Primary encoder current value		EC	20D3	24D3	30D3	34D3
212	VAR_RPOS_PULSES		W	N	R	Registration position		EC	20D4	24D4	30D4	34D4
213	VAR_RPOS		F	N	R	Registration position		UU	20D5	24D5	30D5	34D5
214	VAR_POS		F	N	R/W	Target position		UU	20D6	24D6	30D6	34D6
215	VAR_APOS		F	N	R/W	Actual position		UU	20D7	24D7	30D7	34D7
216	VAR_POSEERROR		W	N	R	Position error		EC	20D8	24D8	30D8	34D8
217	VAR_CURRENT_VEL		F	N	R	Set-point (target) velocity (demanded value)		UU/S	20D9	24D9	30D9	34D9
218	VAR_CURRENT_ACCEL		F	N	R	Set-point (target) acceleration (demanded value)		UU/S ²	20DA	24DA	30DA	34DA
219	VAR_TPOS_ADVANCE		W	N	W	Target position advance. Every write to this variable adds value to the Target position summing point. Value gets added once per write. This variable useful when loop is driven by Master encoder signals and trying to correct phase. Value is in encoder counts		EC	20DB	24DB	30DB	34DB
220	VAR_JOINDEX		W	N	R/W	Same as INDEX variable in user's program.			20DC	24DC	30DC	34DC
221	VAR_PSLIMIT_PULSES		W	Y	R/W	Positive Software limit switch value in Encoder counts		EC	20DD	24DD	30DD	34DD
222	VAR_NSLIMIT_PULSES		W	Y	R/W	Negative Software limit switch value in Encoder counts		EC	20DE	24DE	30DE	34DE
223	VAR_SLS_MODE		W	Y	R/W	Soft limit switch action code:	0 - no action 1 - Fault 2 - Stop and fault (When loop is driven by trajectory generator only. With all other sources same action as 1)		20DF	24DF	30DF	34DF
224	VAR_PSLIMIT		F	Y	R/W	Same as var 221 but value in User Units		UU	20E0	24E0	30E0	34E0
225	VAR_NSLIMIT		F	Y	R/W	Same as var 222 but value in User Units		UU	20E1	24E1	30E1	34E1
226	VAR_SE_APOS_PULSES		W	N	R	2nd encoder actual position in encoder counts		EC	20E2	24E2	30E2	34E2
227	VAR_SE_POSEERROR_PULSES		W	N	R	2nd encoder position error in encoder counts		EC	20E3	24E3	30E3	34E3
228	VAR_MODBUS_PARITY		W	Y	R/W	Parity for Modbus Control:	0 - No Parity 1 - Odd Parity 2 - Even Parity		20E4	24E4	30E4	34E4
229	VAR_MODBUS_STOPBITS		W	Y	R/W	Number of Stopbits for Modbus Control	0 - 1.0 1 - 1.5 2 - 2.0		20E5	24E5	30E5	34E5
230	VAR_M_NOMINALVEL		F	Y	R/W	Induction Motor Nominal Velocity	Range: 500 - 20000 RPM	RPM	20E6	24E6	30E6	34E6
231	VAR_M_COSPHI		F	Y	R/W	Induction Motor Cosine Phi	Range: 0 - 1.0		20E7	24E7	30E7	34E7
232	VAR_M_BASEFREQUENCY		F	Y	R/W	Induction Motor Base Frequency	Range: 0 - 400Hz	Hz	20E8	24E8	30E8	34E8
233	VAR_M_SERIES					Induction Motor Series			20E9	24E9	30E9	34E9



Parameter Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
234	VAR_CAN_BAUD_EPM		W	Y	R/W	CAN Bus Parameter: Baud Rate: 1 - 8	1 - 10k 2 - 20k 3 - 50k 4 - 125k 5 - 250k 6 - 500k 7 - 800k 8 - 1000k		20EA	24EA	30EA	34EA
235	VAR_CAN_ADDR_EPM		W	Y	R/W	CAN Bus Parameter: Address	Range: 1-127		20EB	24EB	30EB	34EB
236	VAR_CAN_OPERMODE_EPM		W	Y	R/W	CAN Bus Parameter: Boot-up Mode (Operational State Control)	0 - enters into pre-operational state 1 - enters into operational state 2 - pseudo NMT: sends NMT Start Node command after delay (set by variable 237)		20EC	24EC	30EC	34EC
237	VAR_CAN_OPERDELAY_EPM		W	Y	R/W	CAN Bus Parameter: pseudo NMT mode delay time in seconds	Refer to variable 236	sec	20ED	24ED	30ED	34ED
238	VAR_CAN_ENABLE_EPM		W	Y	R/W	CAN Bus Parameter: Mode Control	0 - Disable CAN interface 1 - Enable CAN interface in DS301 mode 2 - Enable CAN interface in DS402 mode 3 - Enable DeviceNet 4 - Enable PROFIBUS DP		20EE	24EE	30EE	34EE
239	VAR_HOME_ACCEL		F	Y		Homing Mode: ACCEL rate	Range: 0 - 1000000.0	UU/sec ²	20EF	24EF	30EF	34EF
240	VAR_HOME_OFFSET		F	Y	R/W	Homing Mode: Home Position Offset	Range: -32767 to +32767	UU	20F0	24F0	30F0	34F0
241	VAR_HOME_OFFSET_PULSES		W	Y	R/W	Homing Mode: Home Position Offset in encoder counts	Range: +/- 2147418112	EC	20F1	24F1	30F1	34F1
242	VAR_HOME_FAST_VEL		F	Y	R/W	Homing Mode: Fast Velocity	Range: -10000 to +10000	UU/sec	20F2	24F2	30F2	34F2
243	VAR_HOME_SLOW_VEL		F	Y	R/W	Homing Mode: Slow Velocity	Range: -10000 to +10000	UU/sec	20F3	24F3	30F3	34F3
244	VAR_HOME_METHOD		W	Y	R/W	Homing Mode: Homing Method	Range: 1 - 35		20F4	24F4	30F4	34F4
245	VAR_START_HOMING		W	N	W	Homing Mode: Start Homing	0 - No action 1 - Start Homing		20F5	24F5	30F5	34F5
246	VAR_HOME_SWITCH_INPUT		W	Y	R/W	Homing Mode: Switch Input Assignment:	Range: 0-11 0-3: A1-A4 4-7: B1-B4 8-11: C1-C4		20F6	24F6	30F6	34F6
247	VAR_M_VALIDATE_MOTOR		W	N	W	Makes Drive accept Motor's parameters	0 - No action 1 - Validate Motor Data		20F7	24F7	30F7	34F7
248	VAR_M_I2T		F	Y	R/W	Motor			20F8	24F8	30F8	34F8
249	VAR_M_EABSOLUTE		F	Y	R/W	Motor			20F9	24F9	30F9	34F9
250	VAR_M_ABSWAP		F	Y	R/W	Motor Encoder Feedback: B leads A	0 - No Action 1 - B leads A for forward checked (active)		20FA	24FA	30FA	34FA
251	VAR_M_HALLS_INVERTED		F	Y	R/W	Motor Encoder Feedback: Halls	0 - No Action 1 - Inverted Halls Box checked (active)		20FB	24FB	30FB	34FB
252	RESERVED					Do NOT use			20FC	24FC	30FC	34FC
253	RESERVED					Do NOT use			20FD	24FD	30FD	34FD
254	RESERVED					Do NOT use			20FE	24FE	30FE	34FE
255	RESERVED					Do NOT use			20FF	24FF	30FF	34FF

Parameter Reference



Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
256	RESERVED					Do NOT use			2100	2500	3100	3500
257	RESERVED					Do NOT use			2101	2501	3101	3501
258	RESERVED					Do NOT use			2102	2502	3102	3502
259	RESOLVER_EMU_TRK		W	Y	R/W	Resolver Emulation Track Number	Range: 0 - 15 0 - 1024 1 - 256 2 - 360 3 - 400 4 - 500 5 - 512 6 - 720 7 - 800 8 - 1000 9 - 1024 10 - 2000 11 - 2048 12 - 2500 13 - 2880 14 - 250 15 - 4096		2103	2503	3103	3503
260	RESERVED					Do NOT use			2104	2504	3104	3504
261	VAR_CIP_LINK_A_IN_CTRL		W	Y	R/W	Datalink "A" for input assembly			2105	2505	3105	3505
262	VAR_CIP_LINK_B_IN_CTRL		W	Y	R/W	Datalink "B" for input assembly			2106	2506	3106	3506
263	VAR_CIP_LINK_C_IN_CTRL		W	Y	R/W	Datalink "C" for input assembly			2107	2507	3107	3507
264	VAR_CIP_LINK_D_IN_CTRL		W	Y	R/W	Datalink "D" for input assembly			2108	2508	3108	3508
265	VAR_CIP_LINK_A_OUT_CTRL		W	Y	R/W	Datalink "A" for output assembly			2109	2509	3109	3509
266	VAR_CIP_LINK_B_OUT_CTRL		W	Y	R/W	Datalink "B" for output assembly			210A	250A	310A	350A
267	VAR_CIP_LINK_C_OUT_CTRL		W	Y	R/W	Datalink "C" for output assembly			210B	250B	310B	350B
268	VAR_CIP_LINK_D_OUT_CTRL		W	Y	R/W	Datalink "D" for output assembly			210C	250C	310C	350C
269	VAR_CIP_DAT_REG_CTRL		W	Y	R/W	Data format control for Ethernet/IP assemblies			210D	250D	310D	350D
270	VAR_CIP_CTRL_REG		W	Y	R/W	Control register for control via Ethernet/IP			210E	250E	310E	350E
271	VAR_CIP_STATUS_REG		W	N	R	Status register 2 (Fromat for Ethernet/IP)			210F	250F	310F	350F
272	VAR_CIP_HEART_BEAT		W	Y	R/W	CIP Heart beat timer (Ethernet/IP)			2110	2510	3110	3510
273	VAR_EIP_MCAST_TTL		W	Y	R/W	Ethernet/IP multicast "time to leave" parameter			2111	2511	3111	3511
274	VAR_EIP_MCAST_CTRL		W	Y	R/W	Multicast enable/disable control register			2112	2512	3112	3512
275	EIP_MCAST_ADDRESS		W	Y	R/W	Multicast address	Default = 239.192.15.32		2113	2513	3113	3513
276	DNET_SCALE_POLL_IO		W	Y	R/W	DeviceNet polled IO data scale factor			2114	2514	3114	3514
277	TCP_REPLY_DELAY		W	Y	R/W	TCP reply delay value			2115	2515	3115	3515
278	RESERVED					Do NOT use			2116	2516	3116	3516
279	RESERVED					Do NOT use			2117	2517	3117	3517
280	RESERVED					Do NOT use			2118	2518	3118	3518
281	RESERVED					Do NOT use			2119	2519	3119	3519
282	RESERVED					Do NOT use			211A	251A	311A	351A
283	PBUS_ADDR		W	Y	R/W	Profibus address			211B	251B	311B	351B
284	PBUS_DOUT_SIZE		W	Y	R/W	Number of Profibus Data Out channels	Range: 0 - 12		211C	251C	311C	351C
285	PBUS_DIN_SIZE		W	Y	R/W	Number of Profibus Data In channels	Range: 0 - 12		211D	251D	311D	351D
286	PBUS_OUT_LINK1		W	Y	R/W	Profibus Data Out, Channel link 1 PID map			211E	251E	311E	351E
287	PBUS_OUT_LINK2		W	Y	R/W	Profibus Data Out, Channel link 2 PID map			211F	251F	311F	351F
288	PBUS_OUT_LINK3		W	Y	R/W	Profibus Data Out, Channel link 3 PID map			2120	2520	3120	3520
289	PBUS_OUT_LINK4		W	Y	R/W	Profibus Data Out, Channel link 4 PID map			2121	2521	3121	3521
290	PBUS_OUT_LINK5		W	Y	R/W	Profibus Data Out, Channel link 5 PID map			2122	2522	3122	3522
291	PBUS_OUT_LINK6		W	Y	R/W	Profibus Data Out, Channel link 6 PID map			2123	2523	3123	3523
292	PBUS_OUT_LINK7		W	Y	R/W	Profibus Data Out, Channel link 7 PID map			2124	2524	3124	3524
293	PBUS_OUT_LINK8		W	Y	R/W	Profibus Data Out, Channel link 8 PID map			2125	2525	3125	3525
294	PBUS_OUT_LINK9		W	Y	R/W	Profibus Data Out, Channel link 9 PID map			2126	2526	3126	3526
295	PBUS_OUT_LINK10		W	Y	R/W	Profibus Data Out, Channel link 10 PID map			2127	2527	3127	3527
296	PBUS_OUT_LINK11		W	Y	R/W	Profibus Data Out, Channel link 11 PID map			2128	2528	3128	3528
297	PBUS_OUT_LINK12		W	Y	R/W	Profibus Data Out, Channel link 12 PID map			2129	2529	3129	3529



Parameter Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
298	PBUS_IN_LINK1		W	Y	R/W	Profibus Data In, Channel link 1 PID map			212A	252A	312A	352A
299	PBUS_IN_LINK2		W	Y	R/W	Profibus Data In, Channel link 2 PID map			212B	252B	312B	352B
300	PBUS_IN_LINK3		W	Y	R/W	Profibus Data In, Channel link 3 PID map			212C	252C	312C	352C
301	PBUS_IN_LINK4		W	Y	R/W	Profibus Data In, Channel link 4 PID map			212D	252D	312D	352D
302	PBUS_IN_LINK5		W	Y	R/W	Profibus Data In, Channel link 5 PID map			212E	252E	312E	352E
303	PBUS_IN_LINK6		W	Y	R/W	Profibus Data In, Channel link 6 PID map			212F	252F	312F	352F
304	PBUS_IN_LINK7		W	Y	R/W	Profibus Data In, Channel link 7 PID map			2130	2530	3130	3530
305	PBUS_IN_LINK8		W	Y	R/W	Profibus Data In, Channel link 8 PID map			2131	2531	3131	3531
306	PBUS_IN_LINK9		W	Y	R/W	Profibus Data In, Channel link 9 PID map			2132	2532	3132	3532
307	PBUS_IN_LINK10		W	Y	R/W	Profibus Data In, Channel link 10 PID map			2133	2533	3133	3533
308	PBUS_IN_LINK11		W	Y	R/W	Profibus Data In, Channel link 11 PID map			2134	2534	3134	3534
309	PBUS_IN_LINK12		W	Y	R/W	Profibus Data In, Channel link 12 PID map			2135	2535	3135	3535
310	PBUS_ACYC_MODE		W	Y	R/W	Profibus Acyclic Mode Type			2136	2536	3136	3536
311	VAR_RPDO_1_COM		W	N	R	RPDO1			2137	2537	3137	3537
312	VAR_RPDO_2_COM		W	N	R	RPDO2			2138	2538	3138	3538
313	VAR_RPDO_3_COM		W	N	R	RPDO3			2139	2539	3139	3539
314	VAR_RPDO_4_COM		W	N	R	RPDO4			213A	253A	313A	353A
315	VAR_RPDO_5_COM		W	N	R	RPDO5			213B	253B	313B	353B
316	VAR_RPDO_6_COM		W	N	R	RPDO6			213C	253C	313C	353C
317	VAR_RPDO_7_COM		W	N	R	RPDO7			213D	253D	313D	353D
318	VAR_RPDO_8_COM		W	N	R	RPDO8			213E	253E	313E	353E
319	VAR_RPDO_1_MAP1		W	N	R	RPDO1 Mapped Object 1 High Byte			213F	253F	313F	353F
320	VAR_RPDO_1_MAP2		W	N	R	RPDO1 Mapped Object 2 Low Byte			2140	2540	3140	3540
321	VAR_RPDO_1_MAP3		W	N	R	RPDO1 Mapped Object 3 High Byte			2141	2541	3141	3541
322	VAR_RPDO_1_MAP4		W	N	R	RPDO1 Mapped Object 4 Low Byte			2142	2542	3142	3542
323	VAR_RPDO_2_MAP1		W	N	R	RPDO2 Mapped Object 1 High Byte			2143	2543	3143	3543
324	VAR_RPDO_2_MAP2		W	N	R	RPDO2 Mapped Object 2 Low Byte			2144	2544	3144	3544
325	VAR_RPDO_2_MAP3		W	N	R	RPDO2 Mapped Object 3 High Byte			2145	2545	3145	3545
326	VAR_RPDO_2_MAP4		W	N	R	RPDO2 Mapped Object 4 Low Byte			2146	2546	3146	3546
327	VAR_RPDO_3_MAP1		W	N	R	RPDO3 Mapped Object 1 High Byte			2147	2547	3147	3547
328	VAR_RPDO_3_MAP2		W	N	R	RPDO3 Mapped Object 2 Low Byte			2148	2548	3148	3548
329	VAR_RPDO_3_MAP3		W	N	R	RPDO3 Mapped Object 3 High Byte			2149	2549	3149	3549
330	VAR_RPDO_3_MAP4		W	N	R	RPDO3 Mapped Object 4 Low Byte			214A	254A	314A	354A
331	VAR_RPDO_4_MAP1		W	N	R	RPDO4 Mapped Object 1 High Byte			214B	254B	314B	354B
332	VAR_RPDO_4_MAP2		W	N	R	RPDO4 Mapped Object 2 Low Byte			214C	254C	314C	354C
333	VAR_RPDO_4_MAP3		W	N	R	RPDO4 Mapped Object 3 High Byte			214D	254D	314D	354D
334	VAR_RPDO_4_MAP4		W	N	R	RPDO4 Mapped Object 4 Low Byte			214E	254E	314E	354E
335	VAR_RPDO_5_MAP1		W	N	R	RPDO5 Mapped Object 1 High Byte			214F	254F	314F	354F
336	VAR_RPDO_5_MAP2		W	N	R	RPDO5 Mapped Object 2 Low Byte			2150	2550	3150	3550
337	VAR_RPDO_5_MAP3		W	N	R	RPDO5 Mapped Object 3 High Byte			2151	2551	3151	3551
338	VAR_RPDO_5_MAP4		W	N	R	RPDO5 Mapped Object 4 Low Byte			2152	2552	3152	3552
339	VAR_RPDO_6_MAP1		W	N	R	RPDO6 Mapped Object 1 High Byte			2153	2553	3153	3553
340	VAR_RPDO_6_MAP2		W	N	R	RPDO6 Mapped Object 2 Low Byte			2154	2554	3154	3554
341	VAR_RPDO_6_MAP3		W	N	R	RPDO6 Mapped Object 3 High Byte			2155	2555	3155	3555
342	VAR_RPDO_6_MAP4		W	N	R	RPDO6 Mapped Object 4 Low Byte			2156	2556	3156	3556
343	VAR_RPDO_7_MAP1		W	N	R	RPDO7 Mapped Object 1 High Byte			2157	2557	3157	3557
344	VAR_RPDO_7_MAP2		W	N	R	RPDO7 Mapped Object 2 Low Byte			2158	2558	3158	3558
345	VAR_RPDO_7_MAP3		W	N	R	RPDO7 Mapped Object 3 High Byte			2159	2559	3159	3559
346	VAR_RPDO_7_MAP4		W	N	R	RPDO7 Mapped Object 4 Low Byte			215A	255A	315A	355A
347	VAR_RPDO_8_MAP1		W	N	R	RPDO8 Mapped Object 1 High Byte			215B	255B	315B	355B
348	VAR_RPDO_8_MAP2		W	N	R	RPDO8 Mapped Object 2 Low Byte			215C	255C	315C	355C
349	VAR_RPDO_8_MAP3		W	N	R	RPDO8 Mapped Object 3 High Byte			215D	255D	315D	355D
350	VAR_RPDO_8_MAP4		W	N	R	RPDO8 Mapped Object 4 Low Byte			215E	255E	315E	355E
351	VAR_TPDO_1_COM		W	N	R	TPDO1			215F	255F	315F	355F
352	VAR_TPDO_2_COM		W	N	R	TPDO2			2160	2560	3160	3560
353	VAR_TPDO_3_COM		W	N	R	TPDO3			2161	2561	3161	3561
354	VAR_TPDO_4_COM		W	N	R	TPDO4			2162	2562	3162	3562

Parameter Reference



Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
355	VAR_TPDO_5_COM		W	N	R	TPDO5			2163	2563	3163	3563
356	VAR_TPDO_6_COM		W	N	R	TPDO6			2164	2564	3164	3564
357	VAR_TPDO_7_COM		W	N	R	TPDO7			2165	2565	3165	3565
358	VAR_TPDO_8_COM		W	N	R	TPDO8			2166	2566	3166	3566
359	VAR_TPDO_1_MAP1		W	N	R	TPDO1 Mapped Object 1 High Byte			2167	2567	3167	3567
360	VAR_TPDO_1_MAP2		W	N	R	TPDO1 Mapped Object 2 Low Byte			2168	2568	3168	3568
361	VAR_TPDO_1_MAP3		W	N	R	TPDO1 Mapped Object 3 High Byte			2169	2569	3169	3569
362	VAR_TPDO_1_MAP4		W	N	R	TPDO1 Mapped Object 4 Low Byte			216A	256A	316A	356A
363	VAR_TPDO_2_MAP1		W	N	R	TPDO2 Mapped Object 1 High Byte			216B	256B	316B	356B
364	VAR_TPDO_2_MAP2		W	N	R	TPDO2 Mapped Object 2 Low Byte			216C	256C	316C	356C
365	VAR_TPDO_2_MAP3		W	N	R	TPDO2 Mapped Object 3 High Byte			216D	256D	316D	356D
366	VAR_TPDO_2_MAP4		W	N	R	TPDO2 Mapped Object 4 Low Byte			216E	256E	316E	356E
367	VAR_TPDO_3_MAP1		W	N	R	TPDO3 Mapped Object 1 High Byte			216F	256F	316F	356F
368	VAR_TPDO_3_MAP2		W	N	R	TPDO3 Mapped Object 2 Low Byte			2170	2570	3170	3570
369	VAR_TPDO_3_MAP3		W	N	R	TPDO3 Mapped Object 3 High Byte			2171	2571	3171	3571
370	VAR_TPDO_3_MAP4		W	N	R	TPDO3 Mapped Object 4 Low Byte			2172	2572	3172	3572
371	VAR_TPDO_4_MAP1		W	N	R	TPDO4 Mapped Object 1 High Byte			2173	2573	3173	3573
372	VAR_TPDO_4_MAP2		W	N	R	TPDO4 Mapped Object 2 Low Byte			2174	2574	3174	3574
373	VAR_TPDO_4_MAP3		W	N	R	TPDO4 Mapped Object 3 High Byte			2175	2575	3175	3575
374	VAR_TPDO_4_MAP4		W	N	R	TPDO4 Mapped Object 4 Low Byte			2176	2576	3176	3576
375	VAR_TPDO_5_MAP1		W	N	R	TPDO5 Mapped Object 1 High Byte			2177	2577	3177	3577
376	VAR_TPDO_5_MAP2		W	N	R	TPDO5 Mapped Object 2 Low Byte			2178	2578	3178	3578
377	VAR_TPDO_5_MAP3		W	N	R	TPDO5 Mapped Object 3 High Byte			2179	2579	3179	3579
378	VAR_TPDO_5_MAP4		W	N	R	TPDO5 Mapped Object 4 Low Byte			217A	257A	317A	357A
379	VAR_TPDO_6_MAP1		W	N	R	TPDO6 Mapped Object 1 High Byte			217B	257B	317B	357B
380	VAR_TPDO_6_MAP2		W	N	R	TPDO6 Mapped Object 2 Low Byte			217C	257C	317C	357C
381	VAR_TPDO_6_MAP3		W	N	R	TPDO6 Mapped Object 3 High Byte			217D	257D	317D	357D
382	VAR_TPDO_6_MAP4		W	N	R	TPDO6 Mapped Object 4 Low Byte			217E	257E	317E	357E
383	VAR_TPDO_7_MAP1		W	N	R	TPDO7 Mapped Object 1 High Byte			217F	257F	317F	357F
384	VAR_TPDO_7_MAP2		W	N	R	TPDO7 Mapped Object 2 Low Byte			2180	2580	3180	3580
385	VAR_TPDO_7_MAP3		W	N	R	TPDO7 Mapped Object 3 High Byte			2181	2581	3181	3581
386	VAR_TPDO_7_MAP4		W	N	R	TPDO7 Mapped Object 4 Low Byte			2182	2582	3182	3582
387	VAR_TPDO_8_MAP1		W	N	R	TPDO8 Mapped Object 1 High Byte			2183	2583	3183	3583
388	VAR_TPDO_8_MAP2		W	N	R	TPDO8 Mapped Object 2 Low Byte			2184	2584	3184	3584
389	VAR_TPDO_8_MAP3		W	N	R	TPDO8 Mapped Object 3 High Byte			2185	2585	3185	3585
390	VAR_TPDO_8_MAP4		W	N	R	TPDO8 Mapped Object 4 Low Byte			2186	2586	3186	3586
391	VAR_TPDO_1_COM_ET								2187	2587	3187	3587
392	VAR_TPDO_2_COM_ET								2188	2588	3188	3588
393	VAR_TPDO_3_COM_ET								2189	2589	3189	3589
394	VAR_TPDO_4_COM_ET								218A	258A	318A	358A
395	VAR_TPDO_5_COM_ET								218B	258B	318B	358B
396	VAR_TPDO_6_COM_ET								218C	258C	318C	358C
397	VAR_TPDO_7_COM_ET								218D	258D	318D	358D
398	VAR_TPDO_8_COM_ET								218E	258E	318E	358E
399	VAR_TPDO_1_COM_IT								218F	258F	318F	358F
400	VAR_TPDO_2_COM_IT								2190	2590	3190	3590
401	VAR_TPDO_3_COM_IT								2191	2591	3191	3591
402	VAR_TPDO_4_COM_IT								2192	2592	3192	3592
403	VAR_TPDO_5_COM_IT								2193	2593	3193	3593
404	VAR_TPDO_6_COM_IT								2194	2594	3194	3594
405	VAR_TPDO_7_COM_IT								2195	2595	3195	3595
406	VAR_TPDO_8_COM_IT								2196	2596	3196	3596
407	VAR_CAN_HEARTBEAT				R/W	CAN Heartbeat rate (0x1017)	Range: 0 - 65335 milliseconds		2197	2597	3197	3597
408	VAR_PBUS_STATUS				R	PROFIBUS Status			2198	2598	3198	3598
409	VAR_PBUS_MASTER_TIMEOUT_VAL				R/W	Timeout Value for PROFIBUS master			2199	2599	3199	3599



Parameter Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Units	RAM Register 32bit Integer Access Address 0x2000	RAM Register 32bit Float Access Address 0x2400	EPM Reg Copy 32bit Integer Access Address 0x3000	EPM Reg Copy 32bit Float Access Address 0x3400
410	VAR_PBUS_DATA_EXCHANGE_TIMEOUT				R/W	PROFIBUS Data Exchange Timeout	Range: 0 - 327680 milliseconds		219A	259A	319A	359A
411	VAR_PTC_RX				R	PTC Resistance in ohms			219B	259B	319B	359B
412	VAR_PBUS_FIRMWARE_REV					PROFIBUS Firmware Revision as a hex number	first word major, least word minor, 0x00010001 means rev 1.1		219C	259C	319C	359C
413	VAR_PBUS_TIMEOUT_ACTION_CFG					EPM PROFIBUS Timeout Action	Data Exchange Timeout: Bit0=1 fault, =0 no action; Master Monitor Timeout: Bit1=1 Fault, =0 no action; Module timeout (card not present) Bit2=1 fault, =0 no action		219D	259D	319D	359D

Lenze AC Tech Corporation

630 Douglas Street, Uxbridge MA 01569
Sales: 800-217-9100 • Service: 508-278-9100
www.lenze-actech.com

P94CAN01C



MotionView[®]
OnBoard

PositionServo DeviceNet Communications Module Communications Interface Reference Guide

About These Instructions

This documentation applies to DeviceNet communications for the PositionServo drive and should be used in conjunction with the PositionServo User Manual (S94PM01) and the PositionServo Programming Manual (PM94M01). These documents should be read in their entirety as they contain important technical data and describe the installation and operation of the drive.

Copyright ©2008 by AC Technology Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from AC Technology Corporation. The information and technical data in this manual are subject to change without notice. AC Tech makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. AC Tech assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView[®], PositionServo[®], and all related indicia are either registered trademarks or trademarks of Lenze AG in the United States and other countries.

DeviceNet[™], EtherNet/IP[™], CIP[™] and all related indicia are either registered trademarks or trademarks of the ODVA (Open DeviceNet Vendors Association).



1.	Safety Information.....	1
1.1	Warnings, Cautions & Notes.....	1
1.1.1	General	1
1.1.2	Application	1
1.1.3	Installation	1
1.1.4	Electrical Connection.....	2
1.1.5	Operation	2
2.	Introduction.....	3
2.1	Fieldbus Overview	3
2.2	Module Specification	3
2.3	Module Identification Label	3
3.	Installation	4
3.1	Mechanical Installation	4
3.2	DeviceNet Terminal Block.....	4
3.3	Electrical Installation.....	5
3.3.1	Cable Types	5
3.3.2	Network Limitations	5
3.3.3	Connections and Shielding	6
3.3.4	Network Termination.....	6
4.	Configuring Drive for DeviceNet Communication	8
4.1	Connect to the Drive with MotionView OnBoard.....	8
4.2	Set up the CAN network.....	9
4.2.1	Enable DeviceNet Communication	10
4.2.2	Set CAN Parameters.....	10
4.2.3	Set CANOpen Parameters.....	11
4.2.4	Set DeviceNet Parameters.....	12
4.3	Configuration Parameters	12
4.4	Drive-Specific Error Codes.....	13
5.	Polled I/O	14
5.1	Command Output Assembly.....	14
5.1.1	Byte 0 – Control Word	15
5.1.2	Byte 2 - Command Type.....	15
5.1.3	Byte 3 - Response Type	16
5.1.4	Bytes 4 through 7 - Data	16



Contents

5.2	Response Input Assembly	16
5.2.1	Byte 0 - Status Byte 1	17
5.2.2	Byte 1 - Data Scale Factor	17
5.2.3	Byte 2 - Status Byte 2	17
5.2.4	Byte 3 - Response Type	18
5.2.5	Bytes 4 through 7 - Data	18
6	Explicit Messaging	19
6.1	Objects 64h and 65h	19
6.2	Example Explicit Message.....	19
7.	Reference	20
7.1	Reference Documents.....	20
7.2	Common Terms	20
7.3	Parameter Quick Reference	21



1. Safety Information

1.1 Warnings, Cautions & Notes

1.1.1 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.

1.1.2 Application

Drive controllers are components that 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.

1.1.3 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, that can easily be damaged by inappropriate handling. Do not damage or destroy any electrical components since this might endanger your health!

When installing the drive ensure optimal airflow by observing all clearance distances in the drive's user manual. Do not expose the drive to excessive: vibration, temperature, humidity, sunlight, dust, pollutants, corrosive chemicals or other hazardous environments.



Safety Information

1.1.4 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 regulatory 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.

1.1.5 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.). The user is allowed to adapt the controller to his application as described in the documentation.



DANGER!

After the controller has been disconnected from the supply voltage, do not touch the live components and power connection, since capacitors could still be charged. Wait at least 60 seconds 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 drive. The system design must incorporate adequate protection to prevent personnel from accessing moving equipment while power is applied to the drive system.

Table 1: 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	



NOTE:

The complete list of variables can be found in the PositionServo Programming Manual (PM94M01).



2. Introduction

The following information is provided to explain how the PositionServo drive operates on a DeviceNet network; it is not intended to explain how DeviceNet itself works. Therefore, a working knowledge of DeviceNet is assumed, as well as familiarity with the operation of the PositionServo drive.

2.1 Fieldbus Overview

The DeviceNet Fieldbus is an internationally accepted communications protocol designed for commercial and industrial installations of factory automation and motion control applications. High data transfer rates combined with its efficient data formatting, permit the coordination and control of multi-node applications.

2.2 Module Specification

- Group 2 Server Device
- Supported baudrates: 125kbps, 250kbps, 500kbps
- Supported input/output polled data words: Polled, Bit Strobe, COS, Cyclic
- Explicit communication for parameter access

2.3 Module Identification Label

Figure 1 illustrates the label on the DeviceNet communications module. The PositionServo DeviceNet module is identifiable by:

- One label affixed to the side of the module.
- The TYPE identifier in the center of the label: E94ZADVN1.
- P23 Connector Designation

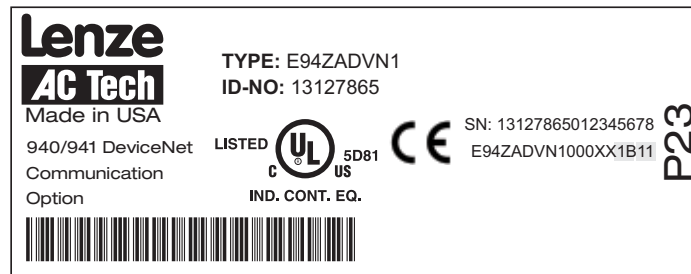


Figure 1: PositionServo DeviceNet Module Label



Installation

3. Installation

3.1 Mechanical Installation

1. Ensure that for reasons of safety, the AC supply and +24V DC backup supply have been disconnected before opening the bay cover plate.
2. Remove the two COMM module screws that secure Option Bay 1 and with the aid of a flat head screw driver, gently pry up the Option Bay 1 cover plate and remove.
3. Install the DeviceNet Module and replace screws as illustrated in Figure 2.

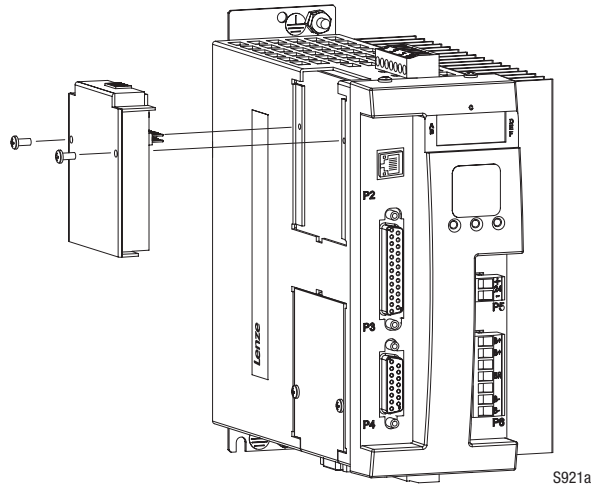


Figure 2: Installation of DeviceNet Communications Module

3.2 DeviceNet Terminal Block

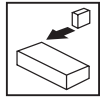
Table 2 and Figure 3 illustrate the pinout of the PositionServo DeviceNet Module connector. This connector provides 5-wire connection to the network.

Table 2: DeviceNet Interface Pin Assignments

Terminal	Name	Wire Color	Description
1	V-	Black	0V
2	CAN L	Blue	CAN Bus Low (Negative data line)
3	Shield	Bare	
4	CAN H	White	CAN Bus High (Positive data line)
5	V+	Red	11-25VDC power supply; current consumption 100mA @ 11VDC max



Figure 3: DeviceNet Interface Pinout



3.3 Electrical Installation

3.3.1 Cable Types

Due to the high data rates used on DeviceNet networks, it is paramount that correctly specified cable is used. The use of low quality cable will result in excess signal attenuation and data loss. Several types of cable are available for DeviceNet networks: flat cable, thicknet, mid cable and thinnet. Installation is typically done with thicknet for trunk cable and thinnet for drop cable. Thicknet has a 3” minimum bend radius. Thinnet is more flexible, with a 2” minimum bend radius, and as such is easier to install. Thinnet can be used for the entire installation. The type of cable used, the lengths of the overall network and the drop cables all affect the maximum baud rate.

Cable specifications and approved manufacturers are available from the official DeviceNet website at: <http://www.ovda.org>.

3.3.2 Network Limitations

There are several factors that must be taken into consideration when designing a DeviceNet network. For full details refer to the official “DeviceNet Planning and Installation Manual” available on the <http://www.ovda.org> website. However, here is an abbreviated checklist:

- DeviceNet networks are limited to a maximum of 64 nodes. Devices default to node 63 so leave node 63 open to avoid duplicate node addresses when adding devices.
- Maximum total network length is governed by the data rate and cable type used. Refer to Table 3.

Table 3: Network Length, Drop Cable Length and Baud Rate

Data Rate	MAXIMUM Network Length				Sum of all Drop Cable Lengths
	Flat Cable	Thicknet	Mid Cable	Thinnet	
125 kbps	420m	500m	300m	100m	156m
250 kbps	200m	250m	250m	100m	78m
500 kbps	75m	100m	100m	100m	39m

- Cumulative drop line does not exceed the network specified limit.
- Network drops/spurs must not exceed 6 meters (19’ 8.2”).
- Use fiber optic segments to:
 - Extend networks beyond normal cable limitations
 - Overcome different ground potential problems
 - Overcome very high electromagnetic interference
- Ground at only one location, preferably in the center of the network.



Installation

3.3.3 Connections and Shielding

- ODVA specifies to ground the DeviceNet network at one location only.
- The ground location should be done on the node that is closest to the physical center of the network to maximize the performance and minimize the effect of outside noise.
- The grounding connection method with regards to the network “V-” connections depends upon the cable type used (see cable data sheet or ODVA “DeviceNet Planning and Installation Manual” for further details).

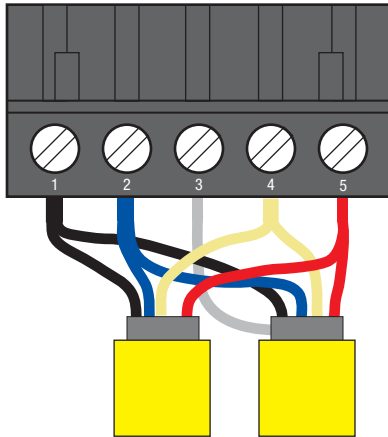


Figure 4a: Network Daisy Chain Connection

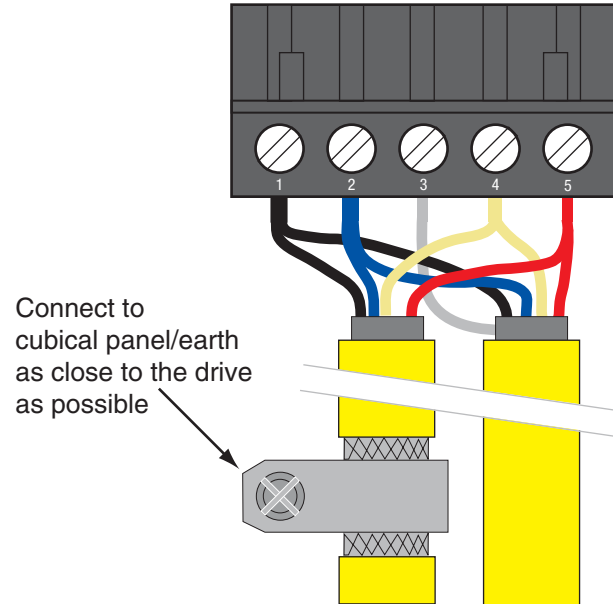


Figure 4b: Ground Connection of Network Center Node

3.3.4 Network Termination

In high speed fieldbus networks such as DeviceNet it is essential to install the specified termination resistors, i.e. one at both ends of a network segment. Failure to do so will result in signals being reflected back along the cable which will cause data corruption. The method of termination varies with the type of network cable available. If terminating using an open-style resistor on the drive connection, use a 120Ω 1/4W 1% resistor and fit as illustrated in Figure 5.

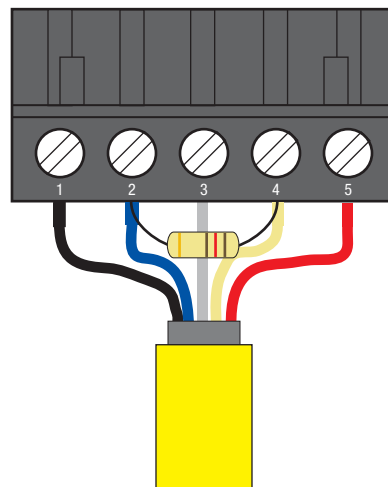


Figure 5: Network Termination on Drive Connector

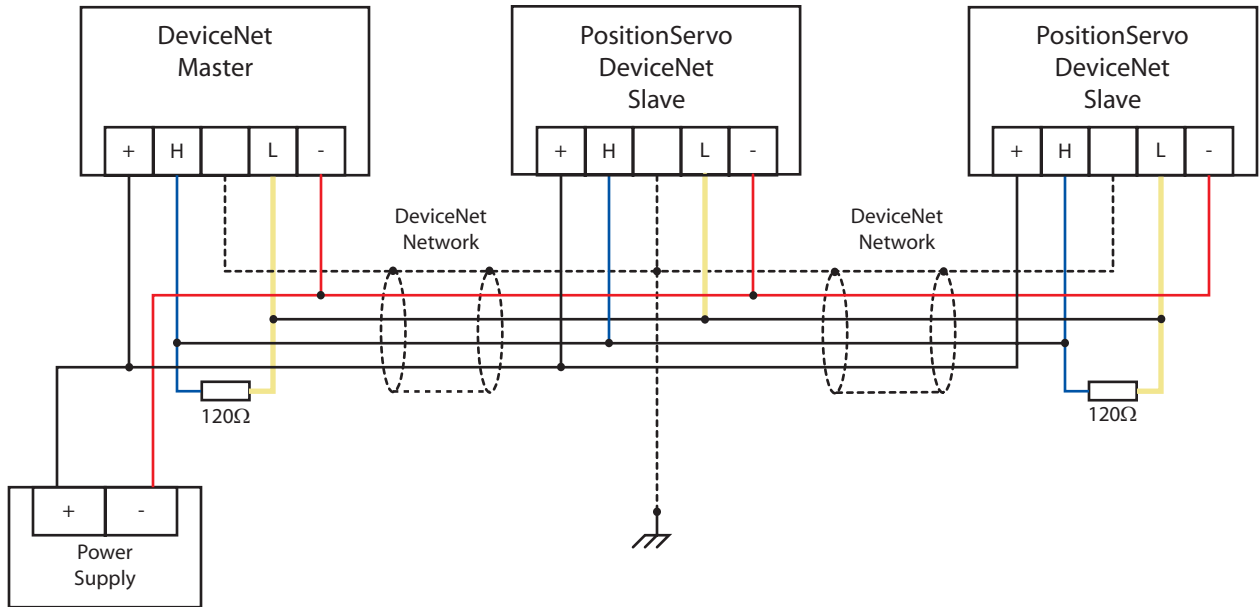
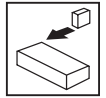


Figure 6: DeviceNet Network Wiring



Commissioning

4. Configuring Drive for DeviceNet Communication

4.1 Connect to the Drive with MotionView OnBoard

With the drive power disconnected, install the DeviceNet module and connect the network cable as instructed in the preceding sections. Ensure the drive Run/Enable terminal is disabled then apply the correct voltage to the drive (refer to drive's user manual for voltage supply details).

Refer to the PositionServo User Manual, section 6.2 for full detail on configuring & connecting a drive via MotionView OnBoard (MVOB) software. Contained herein is a brief description of launching MVOB and communicating with the drive.

1. Open your PC's web browser. Enter the drive's default IP address [192.168.124.120] in the browser's Address window.
2. The authentication screen may be displayed if the PC does not have Java RTE version 1.4 or higher. If so, to remedy this situation, download the latest Java RTE from <http://www.java.com>.
3. When MotionView has finished installing, a Java icon entitled [MotionView OnBoard] will appear on your desktop and the MVOB splash screen is displayed. Click [Run] to enter the MotionView program.
4. Once MotionView has launched, verify motor is safe to operate, click [YES, I have] then select [Connect] from the Main toolbar (top left). The Connection dialog box will appear.
5. Select [Discover] to find the drive(s) on the network available for connection.

[Discover] may fail to find the drive's IP address on a computer with both a wireless network card and a wired network card. If this happens, try one of the following remedies:

Disable the wireless network card and then use [Discover].

Type in the drive's IP address manually at the box [IP Address].

Then click [Connect]

6. Highlight the drive (or drives) to be connected and click [Connect] in the dialog box.

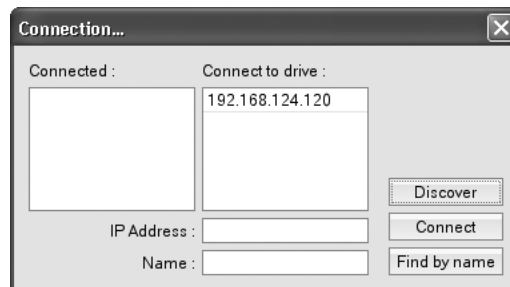


Figure 7 Connection Box with Discovered Drive

In the lower left of the MotionView display, the Message Window will contain the connection status message. The message "Successfully connected to drive B04402200450_192.168.124.120" indicates that the drive B04402200450 with IP address 192.168.124.120 is connected.

A connection needs to be setup only once per session or any time the communication settings are changed. If the work is saved to a project file then the connection does not need to be setup unless different communication settings are used.

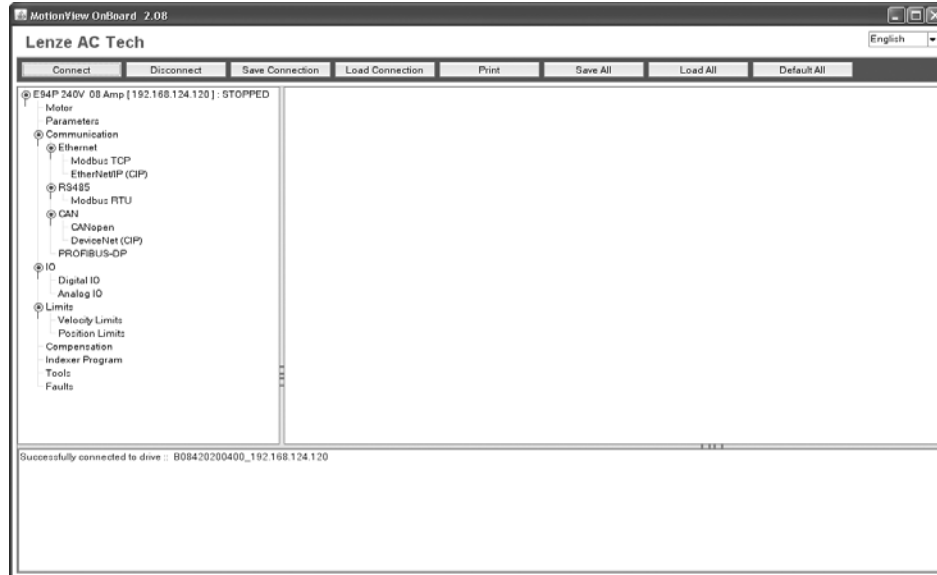


Figure 8: Successfully Connected

4.2 Set up the CAN network

To configure the PositionServo drive for DeviceNet communication, the drive must first be configured for a CAN network. Several parameters need to be set to enable the PositionServo to operate on a CAN network. These parameters are listed under the [Communication], [CAN] and [DeviceNet] folders in the MotionView OnBoard software. Alternatively these parameters can be reached from the drive's front display and keypad. CAN related parameters are explained herein:

- **CAN Control** Enabled/Disabled: Use this parameter to enable or disable CAN followed by reboot. This parameter takes effect after the drive has been re-booted (power cycled).
- **CAN baud rate** 10k- 1000k: Parameter takes effect after drive has been re-booted (power cycled). **Note:** DeviceNet baud rate: 125k, 250k or 500k bps only.
- **CAN address** 1-127: sets drive's CAN ID. This parameter takes effect after the drive has been re-booted (power cycled). **Note:** DeviceNet valid addresses = 0-63.
- **CAN Boot Up Mode** Pre-Operational, Operational or Pseudo Master modes are available after power up.
 - **Pre-Operational** default mode for CAN Open slave. Drive will await message from master to enter Operational mode
 - **Operational** drive will enter Operational mode immediately after power up without receiving activation message from master. This feature is useful in a master-less network.
 - **Pseudo Master** in this mode drive will send activation message (with specified delay, see below) for all CAN slaves waiting in Pre-Operational mode. This mode is useful when emulating master functionality and activating passive slaves. Only one drive can be configured as the pseudo master and only when there is no other master device.
- **CAN Boot up delay** If drive is configured for Pseudo Master mode it will send activation message with delay specified in this parameter. Delay is used to allow specified slaves to boot up and configure their hardware to listen to the Master messages.



Commissioning

4.2.1 Enable DeviceNet Communication

Click on the [Communications] folder in the Node Tree and click on the down arrow next [▼] to [Fieldbus Selection]. Select [DeviceNet] from the pull down menu.

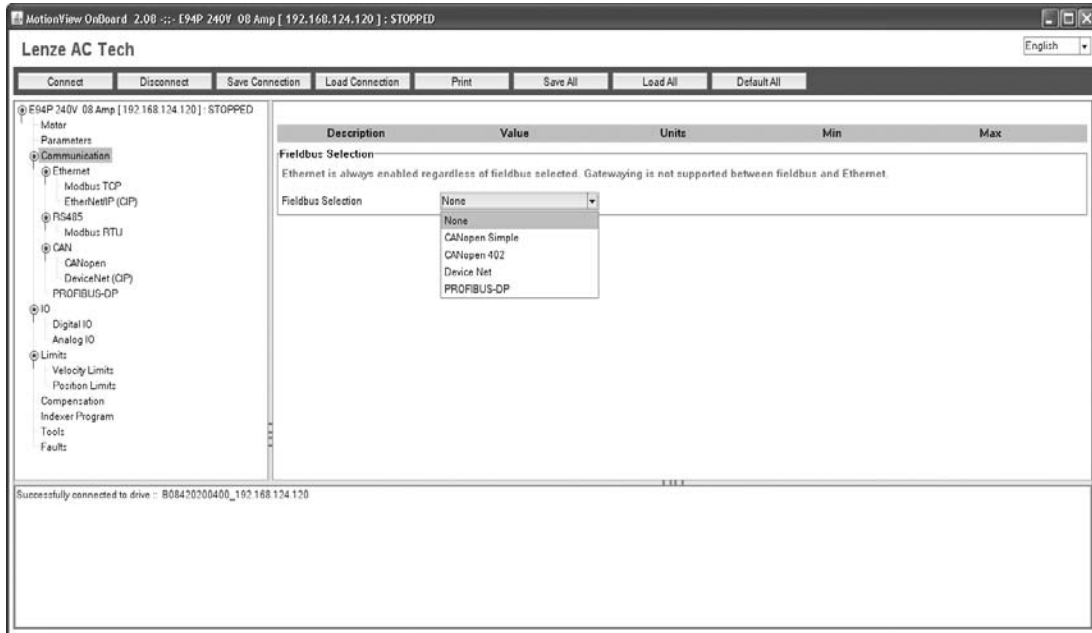


Figure 9: Enable DeviceNet Fieldbus Protocol

To activate any changes made the drive has to be reinitialized. Hence the warning within MotionView

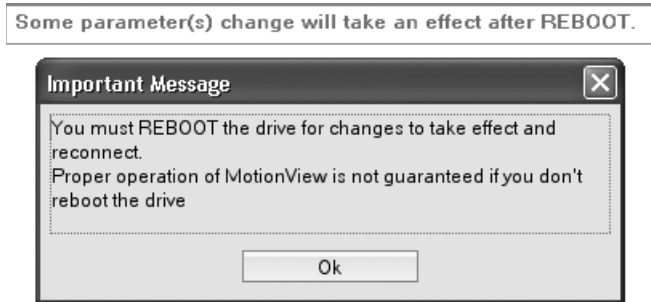


Figure 10: REBOOT Message

This can be done by cycling the power to the drive.

4.2.2 Set CAN Parameters

The [CAN] folder contains the configuration parameters for the CAN interface. To change a CAN parameter, use the pull-down menu to select a pre-defined value or click in the box adjacent to the parameter and enter a numeric value that is within the parameter's specified range. Table 4 lists the range and default value for each CAN parameter.

Table 4: CAN Parameters

Parameter	Range	Default Value
CAN Baud Rate	10k, 25k, 50k, 125k, 250k, 500k, 800k, 1000k	125k
CAN Address	1 - 127	63

Commissioning

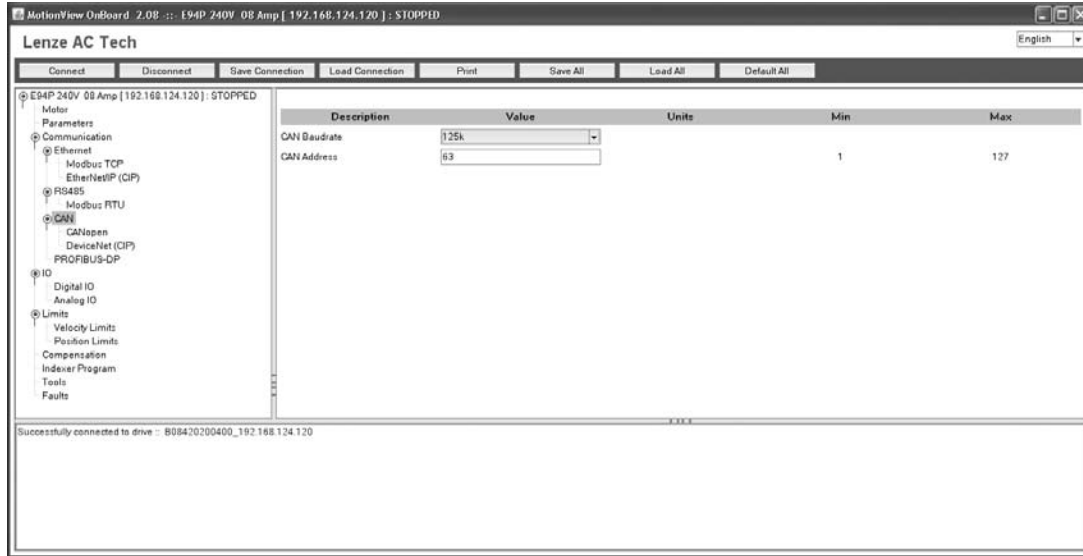


Figure 11: Set CAN Baud Rate & Address

4.2.3 Set CANOpen Parameters

The [CANOpen] folder contains the configuration parameters for the CANOpen Industrial Protocol. To change a CANOpen parameter, use the pull-down menu to select a pre-defined value or click in the box adjacent to the parameter and enter a numeric value that is within the parameter's specified range. Table 5 lists the range and default value for each CANOpen parameter.

Table 5: CANOpen Parameters

Parameter	Range	Default Value
CAN Bootup Mode	Pre-operational, Operational, Pseudo master mode	Operational
CAN Bootup Delay	0 - 5 seconds	5 sec
CAN Heart Beat Time (0x1017)	0 - 65535 milliseconds	2000 msec

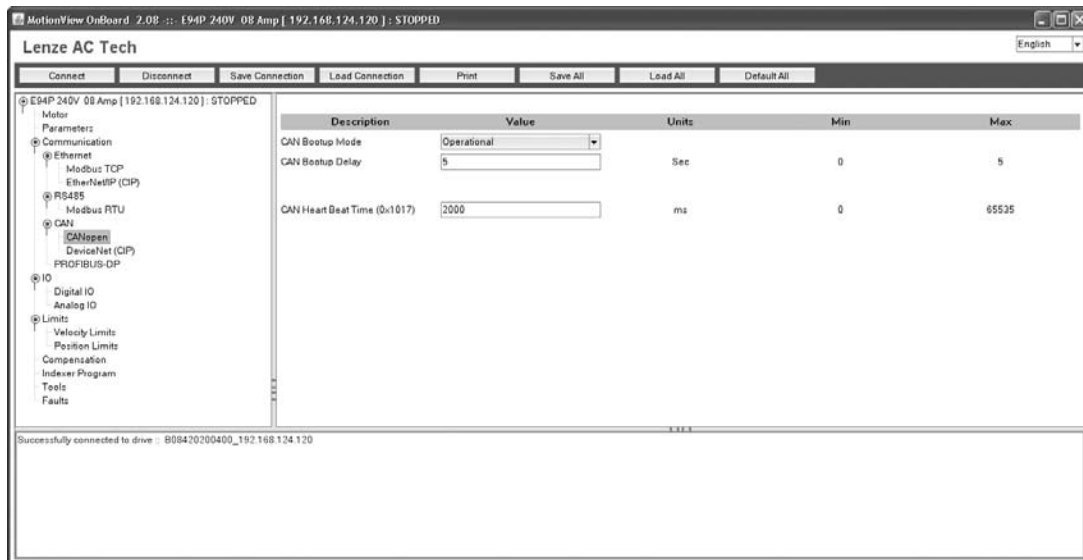


Figure 12: Set CANOpen Parameters



Commissioning

4.2.4 Set DeviceNet Parameters

The [DeviceNet (CIP)] folder contains the configuration parameters for the DeviceNet Industrial Protocol. To change a DeviceNet parameter, use the pull-down menu to select a pre-defined value or click in the box adjacent to the parameter and enter a numeric value that is within the parameter's specified range. Table 6 lists the range and default value for each DeviceNet parameter.

Table 6: DeviceNet (CIP) Parameters

Parameter	Range	Default Value
DeviceNet Poll I/O Scaling**	10 ⁰ , 10 ¹ , 10 ² , 10 ³ , 10 ⁴	10 ¹ = 10

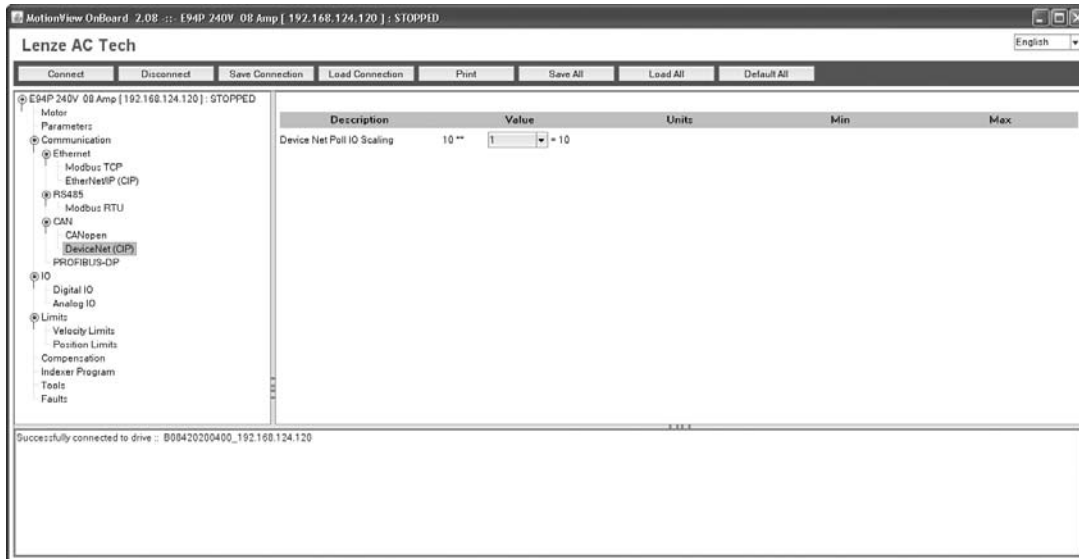


Figure 13: Set DeviceNet I/O Scaling

4.3 Configuration Parameters

All standard CAN parameters are reused by DeviceNet.

To enable DeviceNet functionality, set variable 238 (VAR_CAN_ENABLE_EPM) = 3.

Variable 276 (DNET_SCALE_POLL_IO) is a new parameter that determines the scale factor for the data of the Polled I/O messages. (Refer to section 5). Variable 276 can be set from 0-4 and the default value is 1. Variable 276 is accessible through the MotionView program.

All Polled I/O messages have Integer 32 bit data values that are scaled by the value of the 10^{DNET_SCALE_POLL_IO}.

Example:

For DNET_SCALE_POLL_IO = 1.

A. If the value of 14.7 needs to be sent, then it must send the integer value of 14.7 * (10^{DNET_SCALE_POLL_IO}) = 147

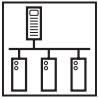
B. If the received value is 156, then it must be divided by (10^{DNET_SCALE_POLL_IO}) to find the real value. Actual Value = 156 / (10^{DNET_SCALE_POLL_IO}) = 15.6



4.4 Drive-Specific Error Codes

The description of the standard PositionServo Fault Codes can be found in the drive's user manual (S94PM01). The DFAULT parameter (PID 9) indicates the last drive fault. In addition, a new error code (43) is introduced to indicate a problem in the DeviceNet Polled I/O message format.

Refer to section 5.2.4. The polled I/O response type 0x14 is used for error response.



Cyclic Data Access

5. Polled I/O

“OUT data and “IN data” describe the direction of data transfer as seen by the DeviceNet master controller. For polled I/O messaging, there are two assemblies: the Command (Output) Assembly - Instance 1, and the Response (Input) Assembly - Instance 2.

The poll operation works as follows:

1. The DeviceNet Master (PLC) sends an I/O Command Poll Assembly initialized with the desired command and the desired response types.
2. The DeviceNet Slave (PositionServo drive) receives the Command I/O Poll Assembly.
3. The PositionServo Drive executes the command specified in the Command assembly.
4. The PositionServo Drive replies by sending the requested Response Assembly.



NOTE:

All Polled I/O messages have Integer 32 bit data values scaled by the value of the 10^DNET_SCALE_POLL_IO.



NOTE:

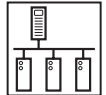
The Enable bit takes precedence over the rest of the command bits and the command type. If the drive is disabled (Enable bit = 0) no motion command can be executed and the move command will be ignored. The response assembly will be issued as requested.

5.1 Command Output Assembly

Table 7 lists the bit assignments of the Polled I/O Command also known as the Output Assembly. Byte 0 is the Control Byte, Byte 1 is not used, Byte 2 is the Command Type, Byte 3 is the Response Type and Bytes 4 through 7 contain the actual command data.

Table 7: Output Assembly

Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Enable	Reg Arm	Hard Stop	Smooth Stop	NA	Relative	NA	Start Motion
1	NA							
2	Command Axis = 001			Command Assembly Type				
3	Response Axis = 001			Response Assembly Type				
4	Data Low Byte							
5	Data Low Middle Byte							
6	Data High Middle Byte							
7	Data High Byte							



5.1.1 Byte 0 – Control Word

Table 8: Output Assembly - Control Word

Bit	Name	Description	Note
7	Enable	0 – disable drive 1 – enable drive	This bit controls PID 52 The Enable bit takes precedence over the rest of the command bits and the command type. If the drive is disabled (Enable bit = 0) no motion command can be executed and the move command will be ignored. The response assembly will be issued as requested.
6	Reg Arm	The change from 0 to 1 arms the registration input C1	It is equivalent to the REGISTRATION ON language statement. (Internally this is the RegistrationOn function).
5	Hard Stop	1 – The drive stops the motion quickly using the QDECEL and QACCEL values.	This bit controls PID 136 and sets it to 2
4	Smooth Stop	1 – The drive stops motion using the normal DECEL and ACCEL values.	This bit controls PID 136 and sets it to 1
3	NA	0 – default setting (bit not used)	
2	Relative	1 – Relative motion will be executed. 0 – Absolute motion will be executed	
1	NA (0)	0 – default setting (bit not used)	
0	Start Motion	Change from 0 to 1 starts a motion command equivalent to the MOVED or MOVEP language commands	This bit in combination with the Relative bit controls PID's 92 and 93.

5.1.2 Byte 2 - Command Type

Command Axis (bits 7 to 5)

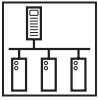
The values of these bits should always be 001 since the PositionServo has only 1 axis per drive.

Command Assembly Type

The Command Assembly Type ranges from 0x0 to 0x9 as listed in Table 9.

Table 9: Command Assembly Type

Type	Command	Description	Note
0x0	NOP		
0x1	Start Trapezoidal Move	Initiates trapezoidal motion	Should be used together with the Start Motion and Relative control bits
0x2	Set Target Reference	Sets the target reference PID 139 (IREF)	Target reference in RPS for velocity mode and in phase Amps (RMS) for current mode.
0x3	Set Acceleration	Sets the acceleration PID 181 (ACCEL)	
0x4	Set Deceleration	Sets the deceleration PID 182 (DECEL)	
0x5	Set Maximum Velocity	Sets the maximum profile velocity PID 180 (MAXVEL)	
0x6	Set Quick Deceleration	Sets the quick deceleration PID 183 (QDECEL)	
0x7	Set Velocity Profile	Sets the velocity for profiled velocity mode PID 185 (VEL)	Profiled velocity is the special operation of Position mode. To set profiled velocity mode set PID 138 to 1. To return back to normal positioning mode set PID 138 to 0.
0x8	Start S-curved Move	Initiates S-curve motion	Should be used together with the Start Motion and Relative control bits
0x9	Set User Variable V0	Sets PID 100 (V0)	



Cyclic Data Access

5.1.3 Byte 3 - Response Type

Response Axis (bits 7 to 5)

The values of these bits should always be 001 since the PositionServo has only 1 axis per drive.

Response Assembly Type (bits 4 to 0)

This field specifies the type of the response assembly that this command requests. For list of all available response assembly types refer to section 5.2.

5.1.4 Bytes 4 through 7 - Data

These 4 bytes contain the actual command data in little endian format. For the receiving node to reconstruct the 32-bit data value as it was originally transmitted, parameters are transmitted in four 8-bit bytes in the order of the lowest byte first and the highest byte last. Said another way, in little endian format, the least significant byte is stored at the lowest address.

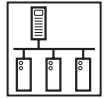
Endian Format	16-bit Value	32-bit Value	
	Byte Order	Word Order	Byte Order
Little	Low byte first High byte second	Low word first High word second	Low byte first Mid low byte second Mid high byte third High byte fourth

5.2 Response Input Assembly

Table 10 lists the Polled I/O Response also known as the Input Assembly.

Table 10: Input Assembly

Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Enable State	Reg Level	Home Level	NA (0)	General Fault	In Position	NA (0)	In Motion
1	Data Scale Factor							
2	NA (1)	NA (0)	NA (0)	NA (0)	NA (0)	Negative HW Limit	Positive HW Limit	NA (0)
3	Response Axis = 001			Response Assembly Type				
4	Data Low Byte							
5	Data Low Middle Byte							
6	Data High Middle Byte							
7	Data High Byte							



5.2.1 Byte 0 - Status Byte 1

Table 11: Status Word 1

Bit	Name	Description	Note
7	Enable State	0 – drive disabled 1 – drive enabled	Bit 0 of the drive status PID 54
6	Reg Level	1 – registration event has occurred	Bit 19 of the drive status PID 54
5	Home Level	1 – drive is homed	Bit 23 of the drive extended status PID 83
4	NA (0)	0 – default setting (bit not used)	
3	General Fault	0 – no fault 1 – fault has occurred	This bit is set when bit 3 of the drive status PID 54 is set or when a malformed DeviceNet Polled I/O command message is detected.
2	In Position	1 – motion complete and target position is within specified limits	Bit 5 of the drive status PID 54
1	NA (0)	0 – default setting (bit not used)	
0	In Motion	1 – If velocity is not 0 or the motion stack is not empty	This is the negated Bit 25 of the drive status PID 54.

5.2.2 Byte 1 - Data Scale Factor

The current value of the DNET_SCALE_POLL_IO (PID 276) parameter is returned in this byte. Refer to section 4.2 for further explanation about the role of this parameter.

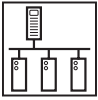
5.2.3 Byte 2 - Status Byte 2

Negative HW Limit

- 1 – If negative limit switch condition engaged.
Bit 29 of the drive status PID 54

Positive HW Limit

- 1 – If positive limit switch condition engaged.
Bit 28 of the drive status PID 54



Cyclic Data Access

5.2.4 Byte 3 - Response Type

Response Axis (bits 7 to 5)

The values of these bits should be always 001 since the PositionServo has only 1 axis per drive.

Response Assembly Type (bits 4 to 0)

The Response Assembly Type ranges from 0x0 to 0x14 as listed in Table 12.

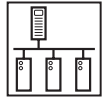
Table 12: Response Assembly Type

Type	Response	Description	Note
0x0	NOP		
0x1	Actual Position	Returns the Actual Position PID 215 (APOS)	
0x2	Commanded Position	Returns the Target Position PID 214 (TPOS)	
0x3	Actual Velocity	Returns the Actual Velocity PID 260 (VELOCITY_ACTUAL)	
0x4	Torque	Returns the Torque calculated as $Torque = PID\ 188\ (PHCUR) * PID\ 20\ (M_KT)$	
0x5	940 Status	Returns the 940 status PID 54 (STATUS)	
0x6	940 Extended Status	Returns the 940 status PID 83 (EXSTATUS)	
0x7	940 Position Error	Returns the 940 PID 216 (POSERROR)	
0x8	940 User Variable V1	Returns the 940 PID 101 (V1)	
0x14	Command/Response Error	This response is returned if any fault occurs or the last sent command is unsupported	In this case byte 6 of the response assembly has copy of the command message byte 2 and byte 7 of the response assembly has copy of the command message byte 3 Bytes 4 and 5 of the response assembly contain the 940 Fault Code. Fault Code (43) is returned when a malformed DeviceNet Polled I/O message is detected.

5.2.5 Bytes 4 through 7 - Data

These 4 bytes contain the actual response data in little endian format. For the receiving node to reconstruct the 32-bit data value as it was originally transmitted, parameters are transmitted in four 8-bit bytes in the order of the lowest byte first and the highest byte last. Said another way, in little endian format, the least significant byte is stored at the lowest address.

Endian Format	16-bit Value	32-bit Value	
	Byte Order	Word Order	Byte Order
Little	Low byte first High byte second	Low word first High word second	Low byte first Mid low byte second Mid high byte third High byte fourth



6 Explicit Messaging

6.1 Objects 64h and 65h

The PositionServo system objects 64h and 65h encapsulate all valid PositionServo variables (Property IDs or PIDs). Object 64h covers PIDs in range 1-255. Object 65h covers PIDs in range 256 and up. Each PositionServo PID is represented by an instance of a System940 object. For object 64h the instance number matches the PID's index. For object 65h instance calculated as:

$$\text{Object 65h Instance} = \text{PID} - 255, \text{ for PIDs with ID 256 and up}$$

6.2 Example Explicit Message



NOTE:

The complete list of variables (PIDs) can be found in the PositionServo Programming Manual (PM94M01).

Example of how to access PID # 275:

Since $\text{PID } 275 > 255$, it falls under Object 0x65 and the Instance ID is $(275 - 255) = 20$ (0x14 in hex).

All aspects of control and parameterization in the PositionServo drive are accomplished through the system variables. Some of the variables are parameters such as Current Limit or Target Position. Some of the variables are action properties, i.e. writing values to these variables will execute a particular process. As an example, writing to variable VAR_ENABLE (ID=52), a non-0 value will enable the drive. Writing the same variable with a 0 (zero) value will disable the drive. Another example could be writing the variable VAR_MOVED with a value of 10, which would execute relative motion for 10 user units.

Every variable in the PositionServo can be read/written as a 32-bit INTEGER or 32-bit REAL(float) value.

Conversion is done automatically. In addition each variable can be read from its RAM (current) copy or from non-volatile (EPM) storage. The value is initialized at the time of power up. To accommodate different access (RAM or EPM) and format (integer or float) types, attributes are implemented. For example to reach variable VAR_CURRENTLIMIT (ID=30) as FLOAT in RAM (run-time value) you would use InstanceID = 30 with attribute 2. For the same variable accessed in EPM (non-volatile copy) you would use attribute 3. Refer to Table 13.

Table 13: Objects 64(Hex) and 65(Hex)

Instance Attributes

Integer, RAM	0
Integer, EPM	1
Float, RAM	2
Float, EPM	3
String, RAM	4
String, EPM	5

Instance Services

- Get_Attribute_All
- Get_Attribute_Single

Instance

Instance = variable ID.

Refer to PositionServo Programming Manual (PM94M01) for variable ID list.

For example: Instance of VAR_CURRENTLIMIT is 30 since its ID=30.



Reference

7. Reference

7.1 Reference Documents

- DeviceNet Information: <http://www.odva.org>
- PositionServo Programming Manual (PM94M01): <http://www.lenze-actech.com>
- PositionServo User Manual (S94PM01): <http://www.lenze-actech.com>
- PositionServo CANOpen Communications Reference (P94CAN01): <http://www.lenze-actech.com>



NOTE:

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

7.2 Common Terms

CAN	Controller Area Network
CIP™	Common Industrial Protocol
COS	Change of State. Device produces data whenever its state changes or at a base heartbeat rate.
Cyclic	Device will produce data at a defined interval.
EDS	Electronic Data Sheet
Explicit Message	Contains module vendor information; sent via a high CAN identifier (600-7BF Hex)
I/O Message	Contains real time I/O information of a module; sent via a low CAN identifier (000 - 3FF Hex)
PIT	Production Inhibit Time
PID	Parameter Index Number; Parameter ID; Property ID (The identifier number for the PositionServo variable as listed in the PS Programming Manual PM94M01)
Polled	Device receives data from Master in a sequential order according to the number of nodes.
PMSCS	Predefined Master/Slave Connection Set
UCMM	UnConnected Message Manager: Device can exchange data in peer-to-peer mode



7.3 Parameter Quick Reference

Table 14 lists each parameter number and provides its function, default value and access rights.

Table 14: Parameter Quick Reference

Parameter	Function	Default Value	Access Rights	Cross Reference
PIDxxx	Network Protocol	Ethernet		4.2.1 Enable DeviceNet Communication
PID234	CAN Baud Rate	125k	R/W	4.2.2 Set CAN Parameters
PID235	CAN Address	63	R/W	4.2.2 Set CAN Parameters
PID236	CAN Bootup Mode	Operational	R/W	4.2.3 Set CANOpen Parameters
PID237	CAN Bootup Delay	5 s	R/W	4.2.3 Set CANOpen Parameters
PID238	CAN Enable	3 (for DeviceNet)	R/W	4.3 Configuration Parameters
PID276	DeviceNet Poll I/O Scaling	10 ¹ = 10	R/W	4.2.4 Set DeviceNet Parameters
PID407	CAN Heart Beat Time (0x1017)	2000 ms	R/W	4.2.3 Set CANOpen Parameters
PID311-318	CAN Receive PDO			NOTE: PIDs 311 - 406 are for REFERENCE ONLY. These variables are set through MotionView. Do NOT use directly. These variables are used by MotionView for non-volatile settings of CAN TPDO/RPDO.
PID319-350	CAN RPDO Mapping			
PID351-358	CAN Transmit PDO			
PID359-390	CAN TPDO Mapping			
PID391-398	CAN TPDO COM ET			
PID399-406	CAN TPDO COM IT			

AC Technology Corporation

630 Douglas Street Uxbridge, MA 01569

Telephone: (508) 278-9100 Facsimile: (508) 278-7873

www.lenze-actech.com

P94DVN01A



MotionView[®]
OnBoard

PositionServo ETHERNET/IP
Communications Protocol Reference Guide

About These Instructions

This documentation applies to EtherNet/IP communications for the PositionServo drive and should be used in conjunction with the PositionServo User Manual (S94H201) that shipped with the drive. These documents should be read in their entirety as they contain important technical data and describe the installation and operation of the drive.

Copyright ©2008 by Lenze AC Tech Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from Lenze AC Tech Corporation. The information and technical data in this manual are subject to change without notice. Lenze AC Tech makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. Lenze AC Tech assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView[®], PositionServo[®], and all related indicia are trademarks of Lenze AG in the United States and other countries.

CompoNet[™], DeviceNet[™], CIP[™], CIP Safety[™], CIP Sync[™], CIP Motion[™], DeviceNet Safety[™] and EtherNet/IP Safety[™] and all related indicia are trademarks of the ODVA (Open DeviceNet Vendors Association). EtherNet/IP[™] is a trademark used under license by ODVA.

RSLogix[™], RSLogix[™] 5000, CompactLogix, CompactLogix 5000, ControlLogix[®], MicroLogix[™], SoftLogix, Allen Bradley[®] and all related indicia are trademarks of Rockwell Automation[®] Corporation.



1.	Safety Information.....	1
1.1	Warnings, Cautions & Notes.....	1
1.2	Reference Documents.....	2
2.	Introduction.....	3
2.1	EtherNet/IP Overview	3
2.2	Ethernet TCP/IP Configuration	3
3.	Installation	5
3.1	Mechanical Installation	5
3.2	Electrical Installation.....	5
3.3	Grounding.....	5
3.4	Cabling	5
3.5	Maximum Network Length.....	5
3.6	Minimum Node to Node Cable Length	6
3.7	Network Topology.....	6
3.8	Example Networks.....	7
4	Configuring EtherNet/IP	9
4.1	Connect to the Drive with MotionView OnBoard.....	9
4.2	Configuring a Scanner or Bridge	13
4.3	Adding a Bridge or Scanner to the I/O Configuration.....	13
4.4	Adding the Adapter and Drive to the I/O Configuration.....	16
4.5	Saving the Configuration.....	19
5	I/O Messaging.....	20
5.1	Overview of I/O Messaging	20
5.2	I/O Assemblies.....	20
5.3	Using Assemblies for Control and Status/Data Monitoring	21
5.4	Using DataLinks.....	21
5.5	Assembly Object.....	22
5.6	Example Ladder Logic Program	24
6	Explicit Messages	27
6.1	Formatting Explicit Messages	27
6.2	Performing Explicit Messages	29
6.3	Explicit Message Example.....	30



Contents

7	Ethernet/IP Objects	34
7.1	Identity Object	34
7.2	PositionServo System Object.....	35
7.3	Assembly Object.....	36
7.4	TCP/IP Interface Object	36
7.5	Ethernet Link Object	37
8	Applications	38
8.1	Application Example 1 - Velocity Control	38
8.2	Application Example 2 - Indexing	41
8.3	Application Example 3 - Configuration Using Explicit Messages	47
8.4	Application Note - Detection of EtherNet/IP Exclusive Ownership Loss	54



1. Safety Information

1.1 Warnings, Cautions & Notes

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

Drive controllers are components that 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 (2004/108/EC).

The drive controllers meet the requirements of the Low Voltage Directive 2006/95/EC. 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, that can easily be damaged by inappropriate handling. Do not damage or destroy any electrical components since this might endanger your health!

When installing the drive ensure optimal airflow by observing all clearance distances in the drive's user manual. Do not expose the drive to excessive: vibration, temperature, humidity, sunlight, dust, pollutants, corrosive chemicals or other hazardous environments.

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).



Safety Information

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, do not touch the live components and power connection, since capacitors could still be charged. Wait at least 60 seconds 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 drive. The system design must incorporate adequate protection to prevent personnel from accessing moving equipment while power is applied to the drive system.

Table 1: 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

- EtherNet/IP Information: <http://www.odva.org>
- PositionServo Programming Manual (PM94H201): <http://www.lenzeamericas.com>
- PositionServo User Manual (S94H201): <http://www.lenzeamericas.com>



NOTE:

The complete list of variables can be found in the PositionServo Programming Manual (PM94H201).



2. Introduction

EtherNet/IP just like its close siblings DeviceNet and ControlNet, uses CIP (Common Industrial Protocol a.k.a. Control and Information Protocol) to exchange data between devices on an Ethernet network. AC Tech implementation of CIP follows the standard supported by the ODVA (governing organization) and supports the two main types of EtherNet/IP communication: Explicit Messaging and I/O Messaging.

The purpose of this document is to describe the EtherNet/IP implementation specifics for the PositionServo drive as well as provide the necessary information and examples for users and network programmers. This document assumes the reader is familiar with the general concept of CIP and has a basic knowledge of Ethernet TCP/IP communication principles.

2.1 EtherNet/IP Overview

EtherNet/IP implements network protocol using the seven layer Open Systems Interconnection (OSI) model as illustrated in Figure 1. Ethernet has an active infrastructure and as such EtherNet/IP can support an almost unlimited number of point-to-point nodes. The EtherNet/IP system requires just one connection for configuration and control. An EtherNet/IP system uses peer-to-peer communication and can be setup to operate in a master/slave or distributed control configuration.

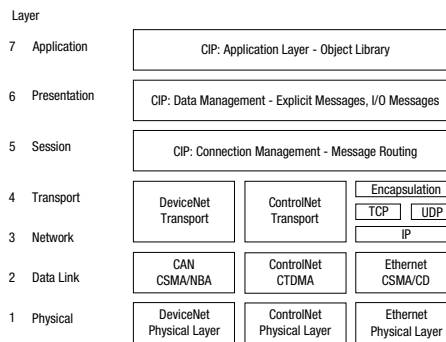


Figure 1: OSI Model

2.2 Ethernet TCP/IP Configuration

Typically, an EtherNet/IP network is made up of segments containing point-to-point connections in a star configuration as illustrated in Figure 2. At the center of this star topology is a bank of Ethernet 2 & 3 switches that can support a great number of point-to-point nodes.

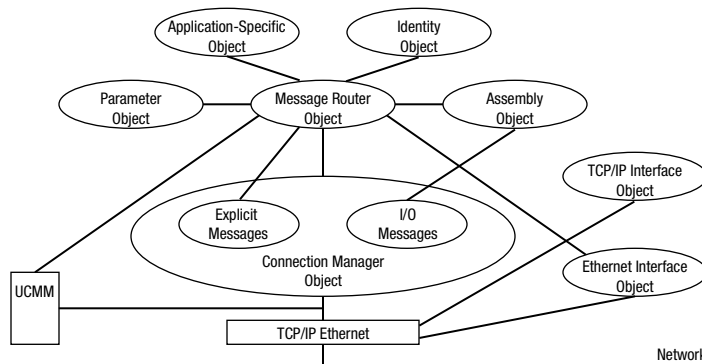


Figure 2: EtherNet/IP Star Configuration



Introduction

2.2.1 MultiCast Configuration

By default the PositionServo drive automatically generates the multicast address used for I/O messaging. The default multicast TTL (time to leave) value is 1 which means that the multicast I/O packets will be propagated over the local subnet only.

The user is allowed to explicitly set the drive's multicast address and TTL values but this feature should be used carefully. In the Communication folder of the MotionView program there is a menu for Ethernet IP settings. The TTL and Mcast Config attributes in the TCP/IP object are also implemented. Note that the Num Mcast value in the Mcast Config attribute must always be 1.

The user configurable PositionServo system variables for multicast are:

Variable ID Meaning

273	TTL
275	Multicast address (default 239.192.15.32)

2.2.2 IGMP Implementation

The IGMP v2 version of the IGMP (Internet Group Management Protocol) is used.

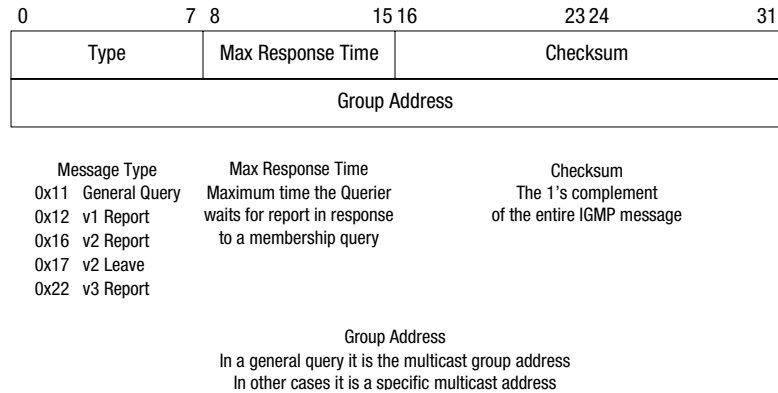


Figure 3: IGMP v2 Message Format

2.2.3 TCP/IP Sockets

The PositionServo supports up to 3 TCP/IP socket connections.

2.2.4 CIP Connections

The PositionServo supports up to 10 CIP connections.



3. Installation

Ethernet/IP communication is not supported by the RS232-based PositionServo drive even if the RS232-based drive has the Ethernet option module E94ZAETH1 installed. Ethernet/IP is also not supported by the MVCD PositionServo drives (part number ending in “X”). Ethernet/IP is supported by the MVOB equipped PositionServo drives (part number ending in “M” or “S”).

3.1 Mechanical Installation

No mechanical installation is necessary. The Ethernet Module is the standard interface on the PositionServo drive.

3.2 Electrical Installation

Table 2 and Figure 4 illustrate the pinout of the PositionServo Ethernet interface. The 8-pin connector provides a standard RJ45 UTP/STP (Unscreened/Screened Twisted Pair) connection to a 10Mbps or 100Mbps Ethernet system.

Table 2: Ethernet Interface Pin Designation

Terminal	Name	Description
1	TxB (+)	Transmit B (+)
2	TxA (-)	Transmit A (-)
3	RxB (+)	Receive B (+)
4	--	Not Used
5	--	Not Used
6	RxA(-)	Receive A (-)
7	--	Not Used
8	--	Not Used

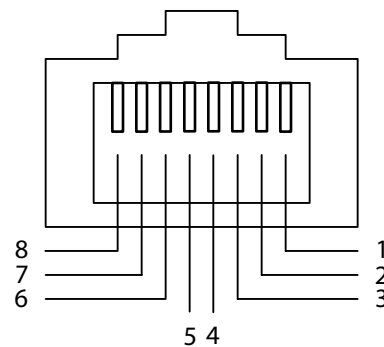


Figure 4: Ethernet Interface Pin Designation

3.3 Grounding

The PositionServo Ethernet interface is supplied with a grounding tag on the module that should be connected to the closest possible grounding point using the minimum length of cable. This will greatly improve the noise immunity of the module. If standard Ethernet UTP or STP cables are used, supplementary grounding is not required when connecting to the PositionServo Ethernet interface.

3.4 Cabling

To ensure long-term reliability it is recommended that any cables used to connect a system together are tested using a suitable Ethernet cable tester, this is of particular importance when cables are made up on site. It is recommended that a minimum specification of CAT5e is installed on new installations, as this gives a good cost performance ratio. If you are using existing cabling this may limit the maximum data rate depending on the cable ratings. In noisy environments the use of STP or fiber optic cable will offer additional noise immunity.

3.5 Maximum Network Length

The main restriction imposed on Ethernet cabling is the length of a single section of cable as detailed in Table 3. If distances greater than this are required it may be possible to extend the network with additional switches or by using a fiber optic converter. Cabling issues are the single biggest cause of network downtime. Ensure cabling is correctly routed, wiring is correct, connectors are properly fitted and any switches or routers used are rated for industrial use. Office grade Ethernet equipment does not offer the same degree of noise immunity as equipment intended for industrial use.



Installation

Table 3: Maximum Network Length

Type of Cable	Data Rate (bits/sec)	Maximum Trunk Length (m)
Copper - UTP/STP CAT 5	10M	100
Copper - UTP/STP CAT 5	100M	100
Fiber Optic - Multi-mode	10M	2000
Fiber Optic - Multi-mode	100M	3000
Fiber Optic - Single-mode	10M	no standard
Fiber Optic - Single-mode	100M	up to 100000



NOTE:

The distances specified are absolute recommended maximums for reliable transmission of data. The distances for the fiber optic sections will be dependent on the equipment used on the network. The use of wireless networking products is not recommended for control systems, as performance may be affected by many external influences.

3.6 Minimum Node to Node Cable Length

There is no minimum length of cable recommended in the Ethernet standards for UTP or STP. For consistency across fieldbus modules, a minimum network device-to-device distance equal to 1 meter of cable is recommended. This minimum length helps to ensure good bend radii on cables and avoids unnecessary strain on connectors.

3.7 Network Topology

Given its universal connectivity, an ethernet network may contain varied connection devices including hubs, switches and routers. Mixing commercial and industrial ethernet networks is possible but care should be taken to ensure clean data transmission. A large, high performance industrial Ethernet network is best served by managed switches that permit data control and monitoring capability.

3.7.1 Hubs

A hub provides a basic connection between network devices. Each device is connected to one port on the hub. Any data sent by a device is then sent to all ports (floods) on the hub. The use of hubs is not recommended for use within control systems due to the increased possibility of collisions. Collisions can cause delays in data transmission and are best avoided, in severe cases a single node can prevent other nodes on the same hub (or collision domain) from accessing the network. If using hubs or repeaters you must ensure that the path variability value and propagation equivalent values are checked. This is beyond the scope of this manual.

3.7.2 Switches

Switches offer a better solution to hubs because after initially learning the addresses of connected devices the switch will only send data to the port that has the addressed device connected to it. This prevents excessive traffic. Some managed switches allow the switching of data to be controlled and monitored which may be of particular importance on large or high performance systems. The word “switch” is sometimes used interchangeably with the terms scanner, matrix and bridge.

3.7.3 Routers

A router is used to communicate between two physical networks (or subnets) and provides some degree of security by allowing only defined connections between the two networks. A typical use would be connecting the office and manufacturing networks or connecting a network to an I.S.P (Internet Service Provider). A router is sometimes known as a gateway as it provides a “gateway” between two networks.



3.7.4 Firewalls

A firewall allows separate networks to be connected together similar to a router, however the firewall offers more security features and control. Typical features include address translation, port filtering, protocol filtering, URL filtering, port mapping, service attack prevention, monitoring and virus scanning. A firewall is the preferred method of allowing traffic from a manufacturing network to the business network.

3.7.5 VPN (Virtual Private Network)

A VPN is a method of using a non-secure or public network that allows devices to be connected together as if they were connected on a private network. A typical example would be the connection of two remote offices such as London and New York. Each office would require a high speed Internet connection and a Firewall (or VPN device). In order to configure the VPN, encryption keys are exchanged so that both offices can communicate. The data is then sent across the Internet (or shared network) in an encrypted form, giving the illusion of a single connected network (speed limitations may apply).

3.8 Example Networks

3.8.1 Single PC to Single PositionServo Drive

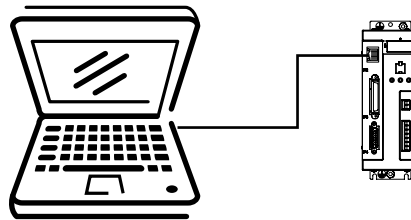


Figure 5: PC to PositionServo Drive

3.8.2 Single PC to Multiple PositionServo Drives and Single Switch

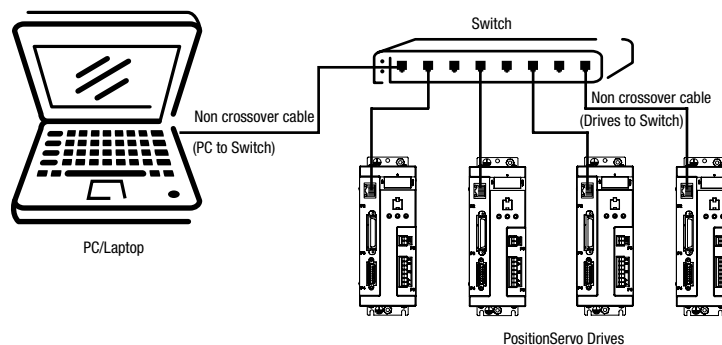


Figure 6: PC to Multiple PositionServo Drives



Installation

3.8.3 Single PC to Multiple PositionServo Drives and Multiple Switches

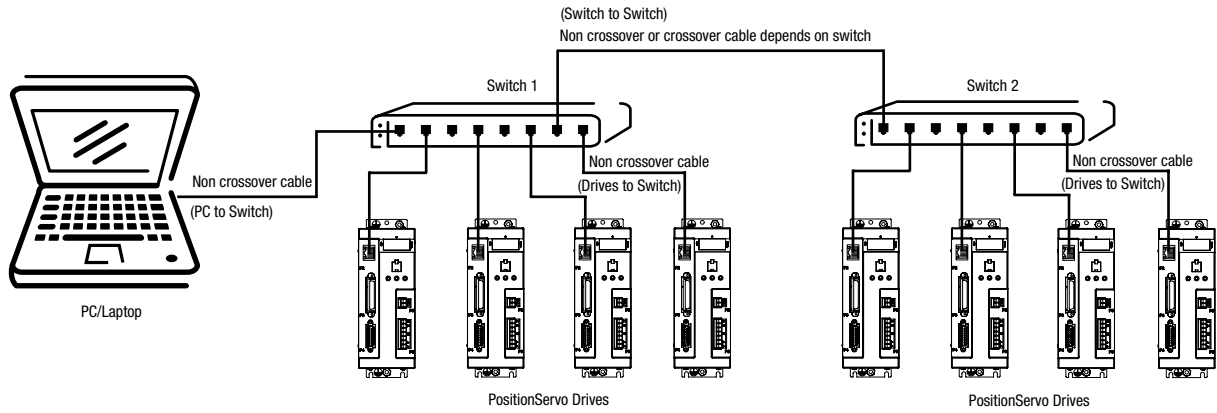


Figure 7: PC to Multiple PositionServo Drives and Multiple Switches



4 Configuring EtherNet/IP

To setup an Ethernet/IP network, the ethernet port on each device that will be part of the network must be configured. For the example illustrated in sections 4 through 6 of this manual, the devices on the network include an Allen-Bradley 1769-L32E CompactLogix controller, a PC and a PositionServo drive.

4.1 Connect to the Drive with MotionView OnBoard

Ensure the drive Run/Enable terminal is disabled then apply the correct voltage to the drive (refer to drive's user manual for voltage supply details).

Refer to the PositionServo User Manual, section 6.2 for full detail on configuring & connecting a drive via MotionView OnBoard (MVOB) software. Contained herein is a brief description of launching MVOB and communicating with the drive.

1. Open your PC's web browser. Enter the drive's default IP address [192.168.124.120] in the browser's Address window.
2. The authentication screen may be displayed if the PC does not have Java RTE version 1.4 or higher. If so, to remedy this situation, download the latest Java RTE from <http://www.java.com>.
3. When MotionView has finished installing, a Java icon entitled [MotionView OnBoard] will appear on your desktop and the MVOB splash screen is displayed. Click [Run] to enter the MotionView program.
4. Once MotionView has launched, verify motor is safe to operate, click [YES, I have] then select [Connect] from the Main toolbar (top left). The Connection dialog box will appear.
5. Select [Discover] to find the drive(s) on the network available for connection.

[Discover] may fail to find the drive's IP address on a computer with both a wireless network card and a wired network card. If this happens, try one of the following remedies:

Disable the wireless network card and then use [Discover].

Type in the drive's IP address manually at the box [IP Address].

Then click [Connect]

6. Highlight the drive (or drives) to be connected and click [Connect] in the dialog box.

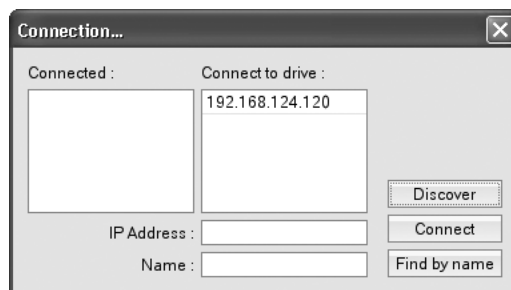


Figure 8 Connection Box with Discovered Drive

In the lower left of the MotionView display, the Message Window will contain the connection status message. The message "Successfully connected to drive B04402200450_192.168.124.120" indicates that the drive B04402200450 with IP address 192.168.124.120 is connected.



Commissioning

A connection needs to be setup only once per session or any time the communication settings are changed. If the work is saved to a project file then the connection does not need to be setup unless different communication settings are used.

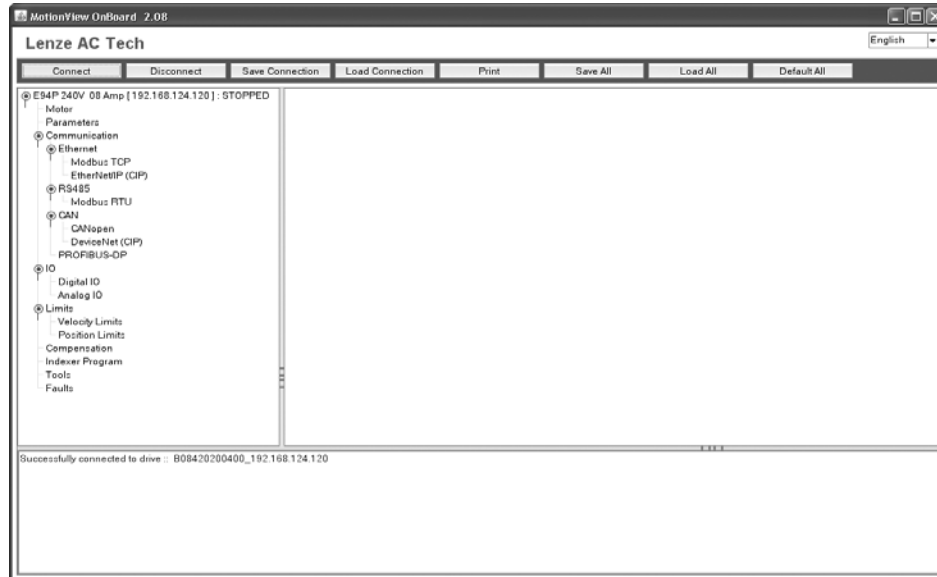


Figure 9: Successfully Connected

Click on the [Communications] folder in the Node Tree. If the [Fieldbus Selection] is other than [None], click the down arrow [▼] next to [Fieldbus Selection] and select [None]. To activate any changes made the drive has to be reinitialized. Hence the warning within MotionView. This can be done by cycling the power to the drive.

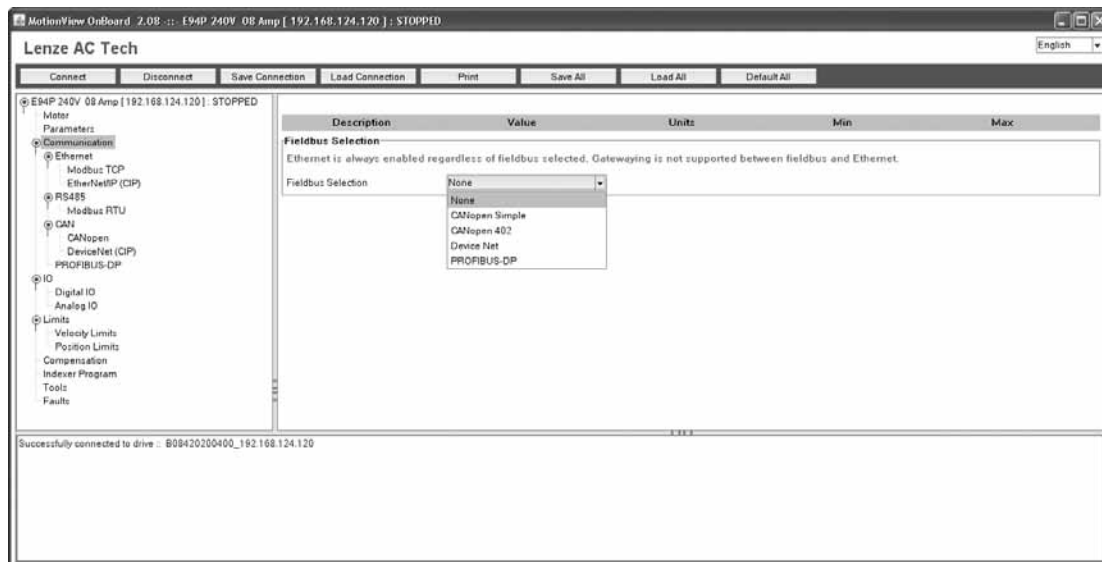


Figure 10: Communications Folder



Some parameter(s) change will take an effect after REBOOT.



Figure 11: REBOOT Message

Ethernet Hardware Settings

The Ethernet folder displays the IP Address, Subnet Mask and Default Gateway for the drive selected in the Node Tree. The TCP Reply Delay can be set in 1 millisecond increments from 0 to 15ms. To obtain the IP address via DHCP, check the box adjacent to [Obtain IP address using DHCP].

Table 4: Ethernet Hardware Setup

Parameter	Range	Default Value
Obtain IP Address using DHCP	Yes or No	No
IP Address	###.###.###.### Automatically Generated	192.168.124.120
Subnet Mask	###.###.###.###	255.255.255.0
Default Gateway	###.###.###.###	192.168.124.1
TCP Reply Delay	0 - 15 milliseconds	2 ms

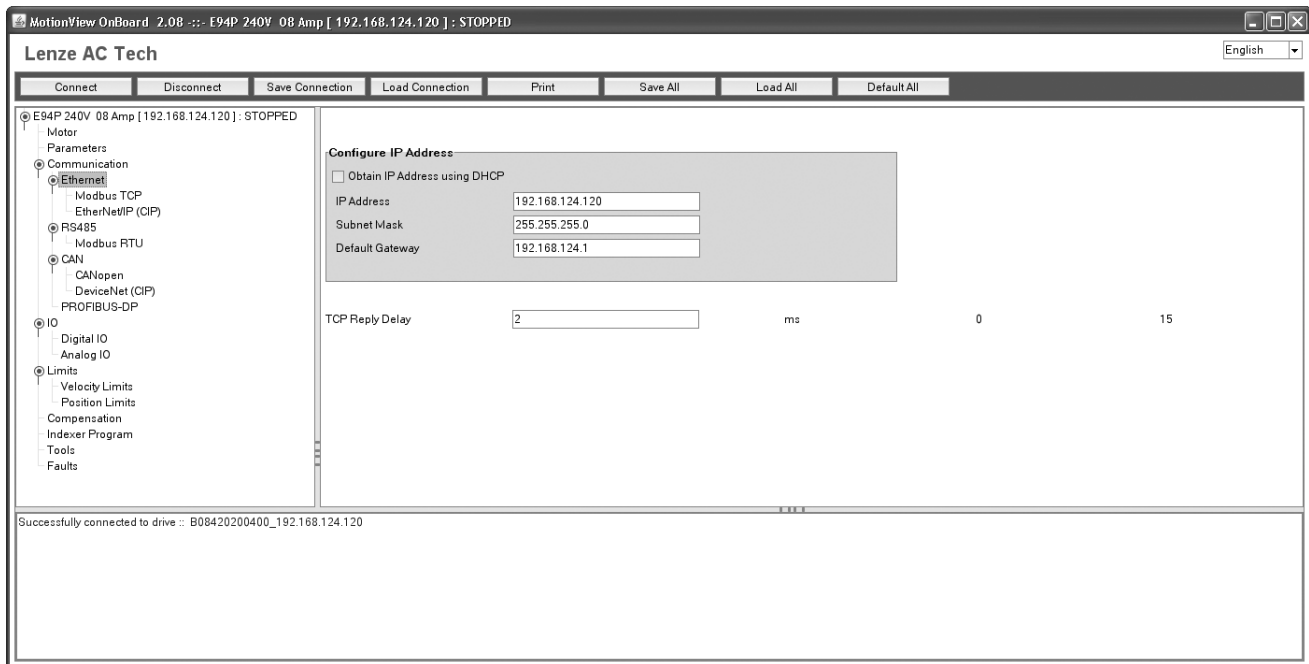


Figure 12: Ethernet



NOTE:

Every time the IP address is reconfigured, the drive must be rebooted so that the change can take effect.



Commissioning

EtherNet/IP Parameters

Defined by the Ethernet hardware settings, the EtherNet/IP folder contains the configuration parameters for the EtherNet/IP (Industrial Protocol). To change an EtherNet/IP parameter, use the pull-down menu to select a pre-defined value or click in the box adjacent to the parameter and enter a numeric value that is within the parameter's specified range. Table 5 lists the range and default value for each EtherNet/IP parameter. In general, there is no need to change parameters for multicast operations. Consult your IT administrator for these settings as their configuration is very network-specific.

Table 5: EtherNet/IP Parameters

Parameter	Range	Default Value
Multicast Control	Automatically Generated; Explicitly Set	Automatically Generated
Multicast TTL	0 - 255	1
Multicast Address	###.###.###.###	239.192.15.32
Input Assembly Links A-D	--	--
Enable	Enable/Disable	Enable
Parameter ID Number	Index number of the assigned User Variable V0-V31	In Link A = 100, In Link B = 101, In Link C = 102, In Link D = 103
Units	RAM Float (4 Bytes); RAM Integer (4 Bytes)	RAM Float (4 Bytes)
Output Assembly Links A-D	--	--
Enable	Enable/Disable	Enable
Parameter ID Number	Index number of the assigned User Variable V0-V31	Out Link A = 104, Out Link B = 105, Out Link C = 106, Out Link D = 107
Units	RAM Float (4 Bytes); RAM Integer (4 Bytes)	RAM Float (4 Bytes)

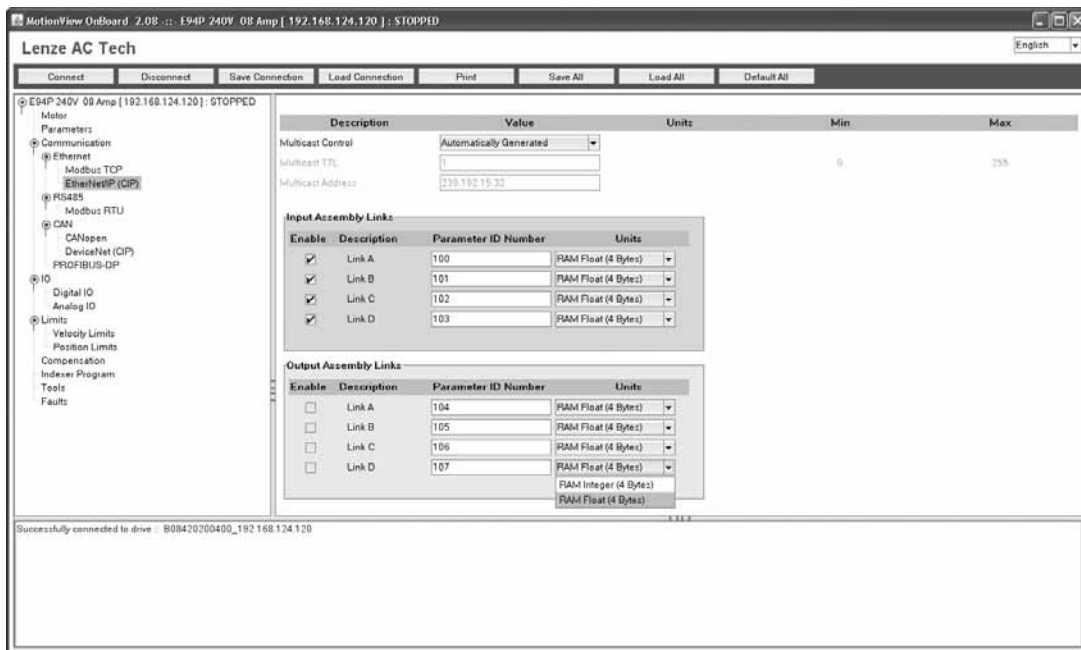


Figure 13: EtherNet/IP Parameters



4.2 Configuring a Scanner or Bridge

To configure a simple network like the network illustrated in Figure 14, follow the steps in paragraphs 4.3 through 4.5. This example uses an Allen-Bradley 1769-L32E CompactLogix controller to communicate with PositionServo drives using implicit I/O messaging over an ethernet network. The controller has a scanner (bridge) that needs to be configured. The I/O assembly object instances will be used for status, input and output data and to map them in the controller memory. Section 5.6 illustrates how to write a simple Ladder program to use the I/O messaging for control and status information.

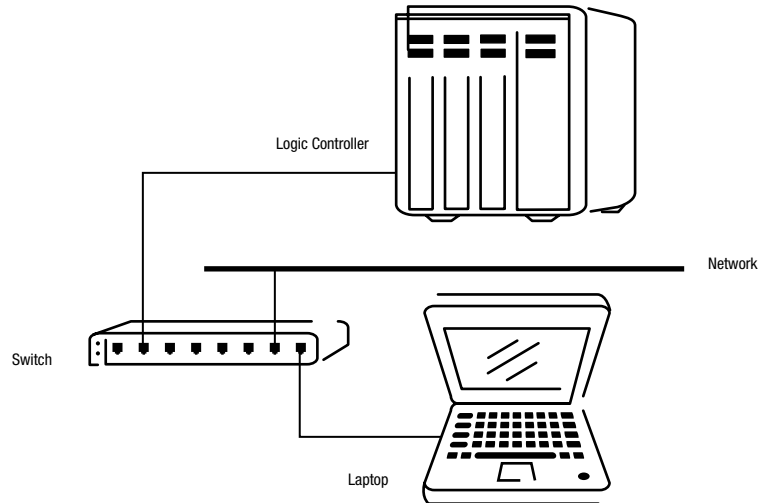


Figure 14: Example Network

4.3 Adding a Bridge or Scanner to the I/O Configuration

To establish communications over an EtherNet/IP network, add the controller and its scanner or bridge to the I/O configuration.

1. Start RSLogix 5000

The RSLogix 5000 window opens as illustrated in Figure 15. For the CompactLogix L32E controller, the I/O configuration already includes a local Ethernet port.

If a SoftLogic controller or ControlLogix controller is used then an Ethernet port scanner needs to be added as illustrated in Figure 15.



Commissioning

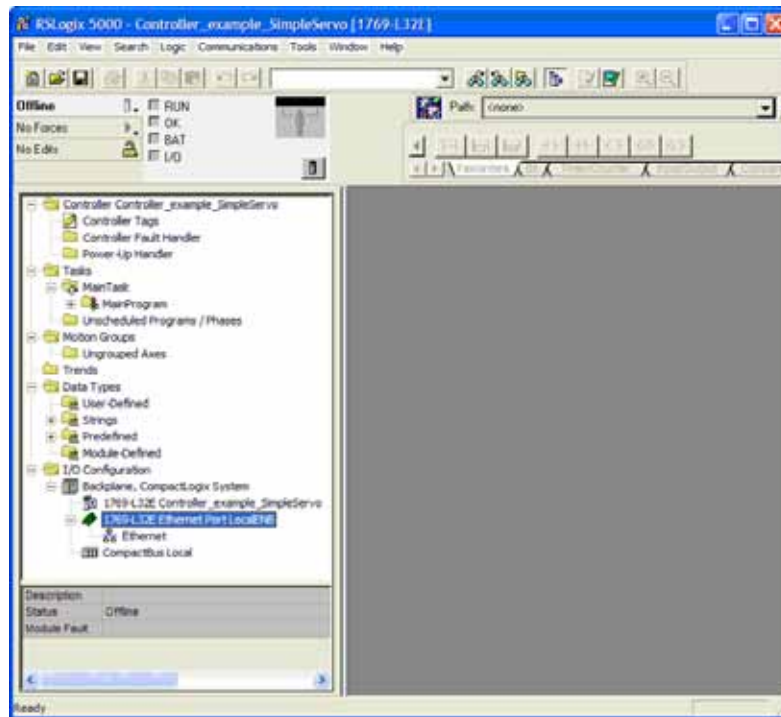


Figure 15: RSLogix 5000 Window (CompactLogix L32E)

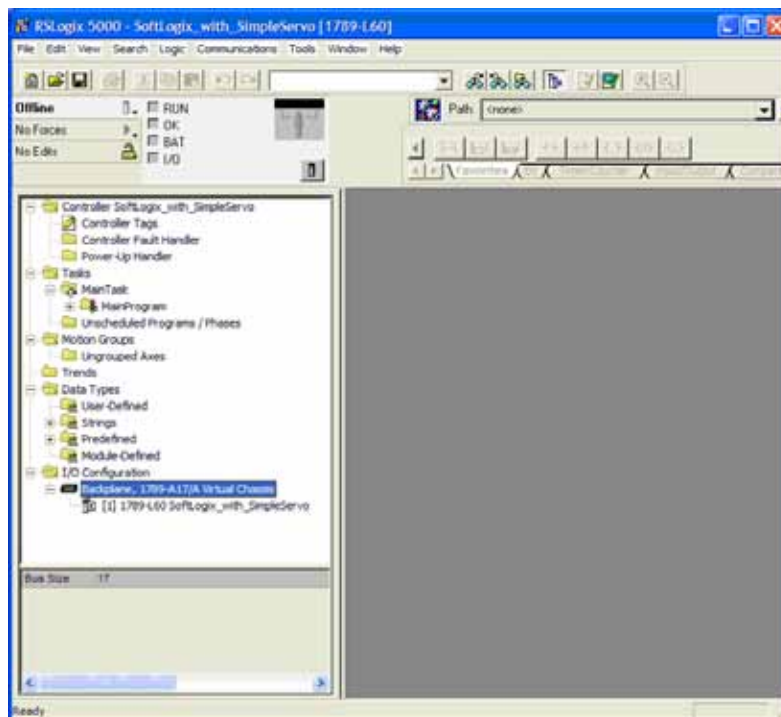


Figure 16: RSLogix 5000 Window (SoftLogix 5800)



2. For CompactLogix and SoftLogix only:

Right click on [Backplane, 1789-A17/A Virtual Chassis] to choose the Ethernet adapter.

Select [New module] and the “Select Module” dialog box will open.

Under the “By Category” tab, click the [+] icon to expand the [Communications] folder

Select the EtherNet/IP scanner or bridge used by your controller. (This example uses the SoftLogix5800 EtherNet/IP)

Then select the major revision of your controller’s firmware in the Major Revision box.

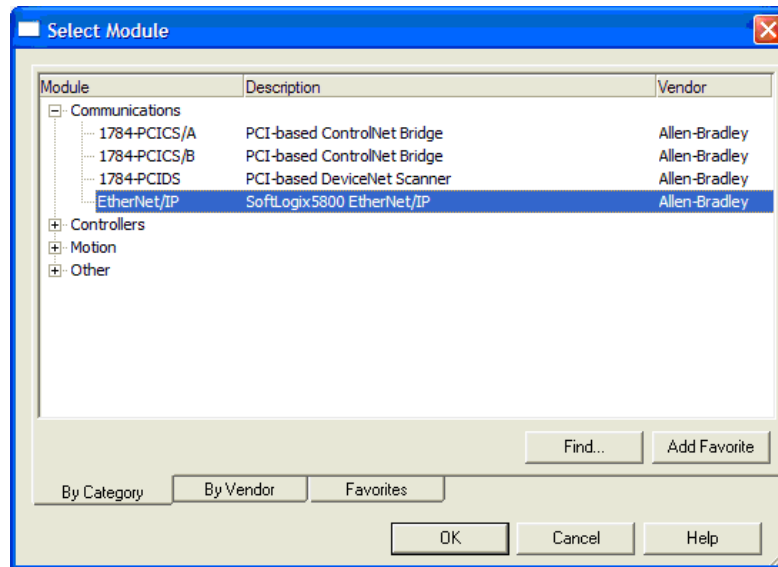


Figure 17: Ethernet Adapter selection (SoftLogix 5800)

3. Click [OK].

The Module Properties dialog box opens. For the CompactLogix controller, right click on [1769-L32E EthernetPort LocalENB] in I/O folder and then select “Properties”.

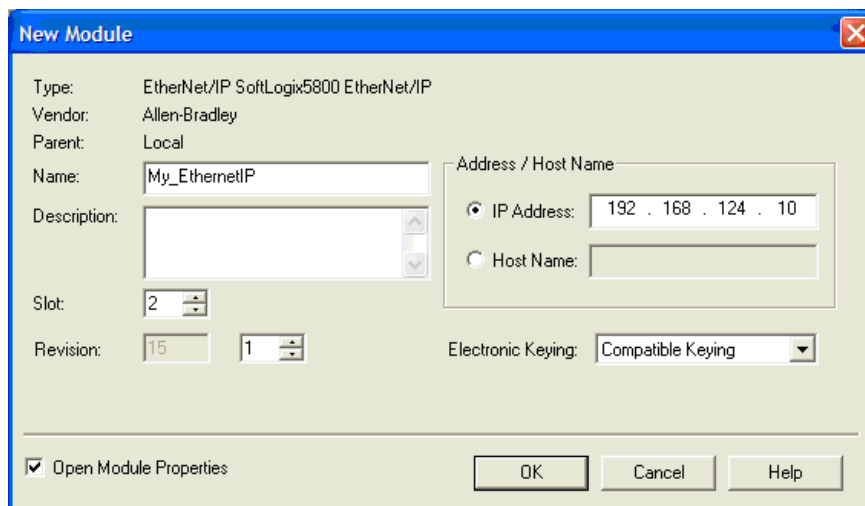


Figure 18: Ethernet Scanner Properties Setup (SoftLogix 5800)



Commissioning

- Set the “New Module” properties using the information in Table 6

Table 6: “New Module” Fields

Box	Type
Name	A name to identify the scanner or bridge.
Slot	The slot # of the EtherNet/IP scanner or bridge in the rack.
Revision	The minor revision of the firmware in the scanner. (You have already set the major revision in the Select Module Type dialog box)
IP Address	The IP address of the EtherNet/IP scanner or bridge.
Electronic Keying	Compatible Module. This setting for Electronic Keying ensures the physical module is consistent with the software configuration before the controller and scanner or bridge make a connection. Therefore, ensure that you have set the correct revision in this dialog box. Refer to the online Help if the controller and scanner have problems making a connection and you want to change this setting.

- Click [OK] to finish.

The scanner (or bridge) is now configured for the EtherNet/IP network. Its name is now listed in the I/O Configuration folder.

4.4 Adding the Adapter and Drive to the I/O Configuration

To transmit data between the scanner (or bridge) and the adapter, the PositionServo drive must be added as a slave device of the scanner.

- In the Control Organizer pane, right-click on the scanner or bridge and select [New Module]. For this example, right-click on the [1769-L32E Ethernet Port LocalENB for CompactLogix]. If using the SoftLogix controller right-click on [1789-L60 SoftLogix_With_SimpleServo].
- The “Select Module” dialog box will open as illustrated in Figure 19.

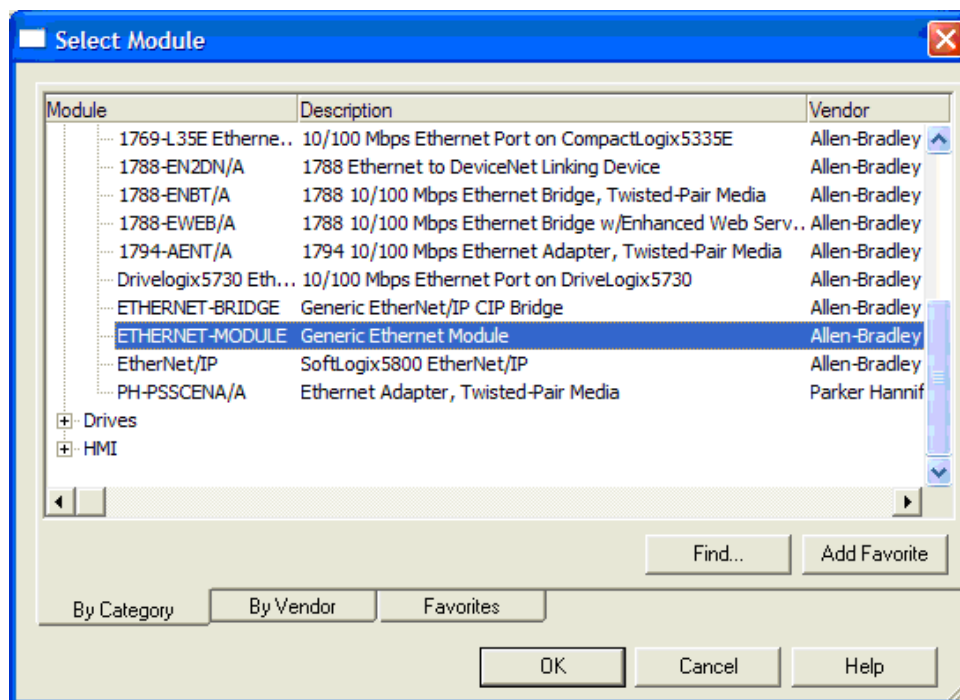


Figure 19: Select Module Dialog Box



- Select [ETHERNET-MODULE] to configure, and then click [OK]. The Module Properties dialog box will open as shown in Figure 20.

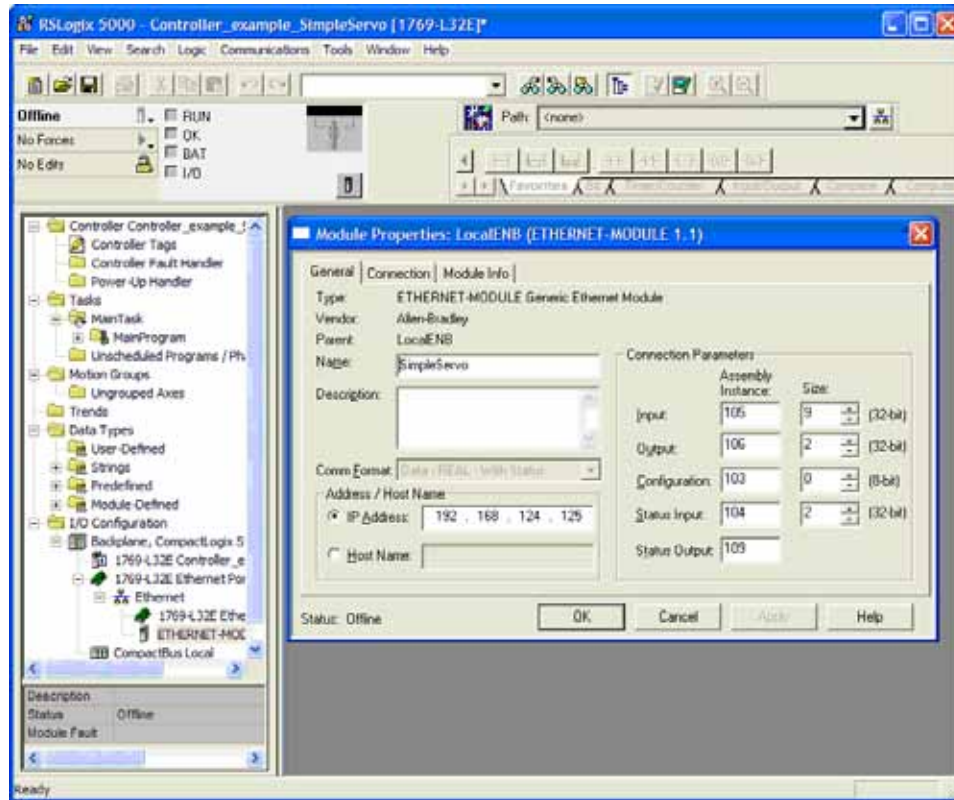


Figure 20: Module Properties Dialog Box

- In the General tab, edit the adapter information as specified in Table 7.

Table 7: Adapter Properties

Box	Type
Name	A name to identify the adapter and drive.
Comm. Format	Data – REAL – With Status This will treat data as real (float) numbers and send status information that is packed as 32 bit words
IP Address	The IP address of the drive controller you intend to connect to

- Click on the [Connection] tab and edit the connection parameters as specified in Table 8.

Table 8: Connection Parameters

Box	Assembly Instance	Size
Input	105 (This value is required)	9 (This value is the exact size of the assembly)
Output	106 108	2 (If using 106) up to 6 (This value is the exact size of the assembly)
Configuration	103 (This value is required)	0
Status Input	104 (This value is required)	2 (This value is required)
Status Output	109 (This value is required)	



Commissioning

6. Click [Next >] to display the next page.
7. In the Requested Packet Interval (RPI) box, set the value to 5.0 milliseconds or greater. This value determines the maximum interval that a controller should use to move data to or from the adapter. To conserve bandwidth, use higher values for communicating with low priority devices.
8. Click [Finish].

The new node (“SimpleServo” in this example) now appears under the scanner or bridge (“1769-L32E...” in this example) in the I/O Configuration folder as shown in Figure 21. To view the module-defined data types and tags that have been automatically created, double-click on the [Data Types] folder and then double-click on the [Module-Defined] folder. After the configuration is saved and downloaded, these tags allow the user to access the Input and Output data of the adapter via the controller’s ladder logic.

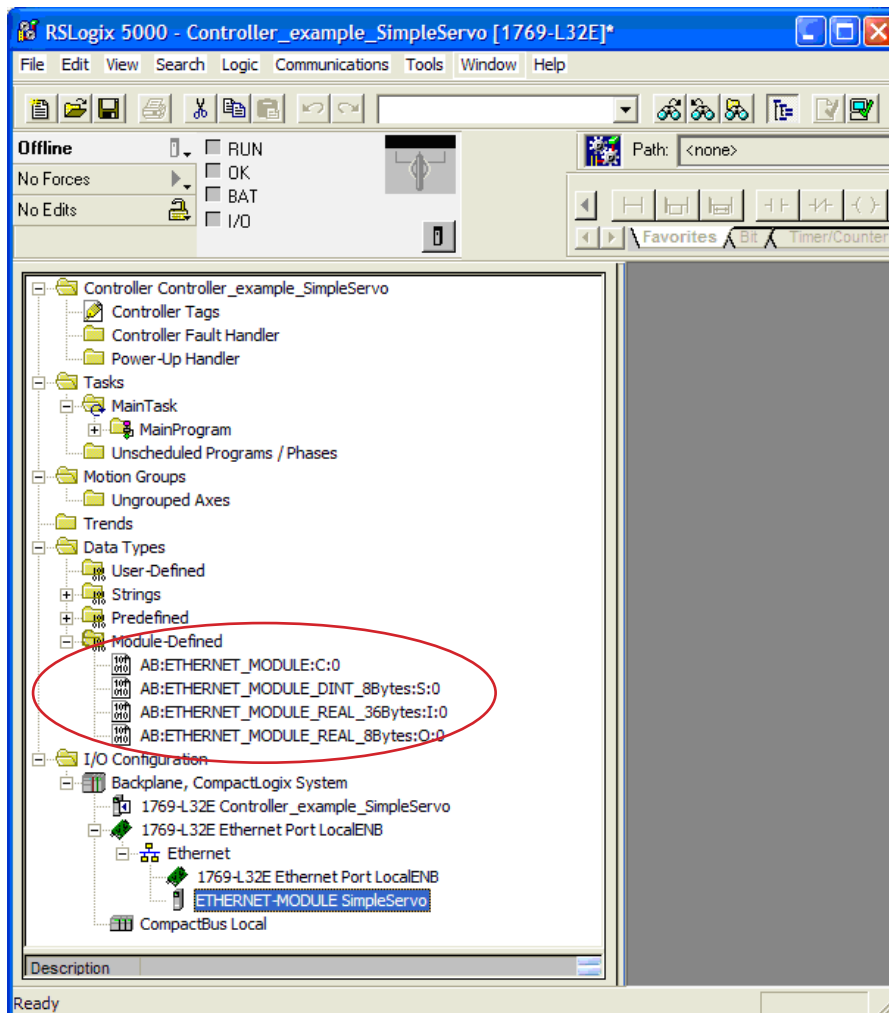


Figure 21: Module-Defined Tags



4.5 Saving the Configuration

After adding the scanner (or bridge) and the adapter to the I/O configuration, the configuration must be downloaded to the controller. The configuration should also be saved to a file on your computer.

1. On the top toolbar, click [Communications] then select [Download] from the pull down menu. The Download dialog box will open.



NOTE:

If a message box reports that RSLogix is unable to go online, then select 'Communications Who Active' to try and find your controller in the 'Who Active' dialog box. If the controller is not shown, then the Ethernet/IP driver needs to be added or configured in RSLinx. Refer to the RSLinx online help.

2. Click [Download] to download the configuration to the controller. When the download is successfully completed, RSLogix enters online mode and the I/O OK box in the upper-left part of the screen is green.
3. On the top toolbar, click [File] then select [Save] from the pull down menu. If this is the first time the project is saved, then the [Save As] dialog box will open. Navigate to a folder, type a file name and then click [Save] to save the configuration to a file on your computer.



Cyclic Data Access

5 I/O Messaging

5.1 Overview of I/O Messaging

Typically I/O messaging is used for the data exchange between a scanner and an adapter device in a cyclic manner. Therefore it is used for data that needs to be updated periodically. A good example is the reference set point value for velocity or torque. Imagine a PLC that calculates the needed speed based on a specific condition of the system (conveyor for example) and periodically sends the new demanded speed over the interface to the drive.

Other examples include fetching data from the drive for monitoring purposes or perhaps controlling functions like current velocity, current and acceleration. In this case data flows from the drive to the PLC. Since these quantities are variables that change over time, they need to be updated (received) periodically.

Table 9: PositionServo implementation of Slave I/O Messaging

Feature	Implement Slave Message?
Bit strobe	N
Cyclic	Y
Change of State	N
Polling (released as Application trigger)	Y
Input only connection	Y
Exclusive owner	Y
Listen only connection	Y

5.2 I/O Assemblies

5.2.1 General Information

PositionServo Ethernet/IP implementation supports the I/O assembly object class 0x04. PositionServo assemblies are static. There are several Input and Output pre-defined assemblies (assembly object instances) that can be used for data exchange. The terms Input and Output refer to the point of view of the scanner. Output data is produced by the scanner and consumed by the adapter. Input data is produced by the adapter and consumed by the scanner. The PositionServo is always an adapter device. An example of a scanner is a PLC or CLC (Continuous Loop Controller).

Depending on the assembly number the memory map of the data can have a different size and meaning. Please refer to the detailed assembly contents later in this chapter.

5.2.2 Important Note for Input Assemblies

Input assemblies (adapter to scanner) are mapped to the adapter memory from byte 0. There is no preceding 4 byte header like that found in most Allen-Bradley equipment. The PositionServo does not use preceding header functionality for real time status. So the start address in the assembly memory map is the actual start of the 1st assembly data item. The user should supply the actual assembly length when mapping the input assembly to the controller memory.

5.2.3 Important Note for Output Assemblies

Output assemblies (scanner to adapter) are assumed to have the preceding 4 byte header. When mapping the assembly this header will automatically be added to the data stream by most AB PLC/CLC equipment. If you use equipment other than AB for the scanner, configure it to send the 4 byte header preceding the actual assembly data. The data in the header should be set to 0.



5.3 Using Assemblies for Control and Status/Data Monitoring

Output assemblies are commonly used for controlling the enable/disable state of the drive and for supplying the velocity or torque reference.

Input assemblies are commonly used to monitor the drive status and run-time quantities such as current velocity, current, actual position and position error.

The recommended configurations for I/O assemblies are:

Output assembly - instance #106

Input assembly - instance #104 and #105 (used for status read)

5.4 Using DataLinks

A DataLink is a mechanism used by the PositionServo drive to transfer data to and from the PLC controller.

DataLinks are the data pointers used in some of the input and output assemblies. There are 4 input DataLinks and 4 output DataLinks. DataLinks are configured through their corresponding control variable. The lower word (LSW) of the control variable contains any valid system variable ID. The high word (MSW) contains the control and information bits (as shown in Table 12). When the DataLink is used in an assembly, the value of the parameter with the ID whose ID is in DataLink's LSW will be transferred by the assembly. Since any PositionServo variable can be accessed as a Real or 32-bit integer, there is a control bit in the Datalink's control word to configure the presentation format of the variable's value.

Datalink's control word is a regular system variable and can be accessed by an explicit message or from the user's program. Tables 10 and 11 provide the variable ID's for the input and output datalinks.

Table 10: Input DataLinks (for IN assembly)

Link	Variable ID
LinkA_in	261
LinkB_in	262
LinkC_in	263
LinkD_in	264

Table 11: Output DataLinks (for OUT assembly)

Link	Variable ID
LinkA_out	265
LinkB_out	266
LinkC_out	267
LinkD_out	268

Table 12: DataLink's Control Word

MSW (bits 16-31)			LSW (bits 0-15)
Bit 31	Bit 30	Bits 29-16	Bits 15-0
Enable	Format	Reserved (Set to 0)	ID



Cyclic Data Access

Enable: Enable the transfer data

Format: Data presentation format

0 = U32 (32 bit integer)

1 = F32 (Real)

ID: variable ID, link uses data from/for

Example:

A DataLink needs to be configured to the transfer data of the Phase Current in REAL format. The ID of VAR_PHCUR (Phase current) is 188 (dec). The VAR ID List is published in the PositionServo Programming Manual (PM94H201). As shown in Table 13, the control word structure for this example DataLink is C00000BC (hex) or 322125660 (decimal).

Table 13: Example Control Word Structure

MSW (bits 31-16)			LSW (bits 15-0)
Bit 31	Bit 30	Bits 29-16	Bits 15-0
1	1	0 ... 0	BC (h), 188 (d)

5.5 Assembly Object

An Assembly Object is the “assembly” or mapping of data from different instances of various classes into a single attribute. With assembly mapping, the I/O data is produced in one block. An assembly object can be used to configure a device using one block of data instead of setting the individual device parameters.

In a device (PositionServo drive) an input assembly collects data from the device and puts it on the network for the master (controlling) device to consume. An output assembly consumes data sent out by the controlling device from the network and writes that data to the output application (driving a motor).

Table 14: Assembly Object Instances:

Assembly Object Instances		
Assembly #	Name	
	103	Dummy Assembly required for implicit configuration. Contains no data.
IN	104	PositionServo Status and IO Assembly
IN	105	PositionServo universal monitor
OUT	106	PositionServo basic control
OUT	108	PositionServo extended control
OUT	109	PositionServo “keep alive”

Table 15: Assembly #104 PositionServo Status and I/O Assembly

32-bit WORD	Variable ID	Type	Name
0	--	U32	STATUS2 WORD (refer to Table 18)
1	65	U32	INPUTS

Table 16: Assembly #109 PositionServo “keep alive”

32-bit WORD	Variable ID	Type	Name
0	272	U32	CIP heartbeat

Assembly #109 is used for indication of the communication activity. It can also be used with AB PLC controllers as Status Assembly when setup as the data scanner.

Cyclic Data Access



NOTE:

The Variable ID is the PositionServo variable's index number. Refer to the PositionServo Programming Manual (PM94H201)

STATUS2 WORD format

The STATUS2 WORD includes bits from the PositionServo STATUS (#53) and EXSTATUS (#54) system variables as shown in Table 17.

Table 17: STATUS2 Word

Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Motion Suspended	Registration Captured	I Limit	Motion Q Empty	Motion Q Full	In Position	Fault	Enable
1		Homed	Homing Active	Negative LS Engaged	Positive LS Engaged	Interface Control Disabled	Motion Completed	Reserved
2	0							
3	0							

Table 18: Mapping STATUS2 bits to STATUS (PID 54) and EXSTATUS (PID 83)

STATUS2 bits	Function	STATUS / EXSTATUS bits
0	Drive Enabled	STATUS.0
1	Drive Fault	STATUS.3
2	Motion completed and position within specified limits	STATUS.5
3	Motion stack full	STATUS.7
4	Motion stack empty	STATUS.8
5	Current limit reached	STATUS.13
6	Registration position obtained	STATUS.19
7	Motion suspended	STATUS.22
8	Reserved	STATUS.24
9	Motion completed (velocity =0 and stack empty)	STATUS.25
10	Interface control is disabled	STATUS.27
11	Positive limit switch engaged	STATUS.28
12	Negative limit switch engaged	STATUS.29
13	Homing in progress	EXSTATUS.21
14	Homed	EXSTATUS.22

Table 19: Assembly #105: PositionServo Universal Monitor

32-bit Word	Variable ID	Type	Name
0	260	F32	Actual Velocity
1	215	F32	Actual Position
2	216	F32	Position error
3	188	F32	Phase current
4	213	F32	Registered position
5	mapped	F32	DataLink_A_in
6	mapped	F32	DataLink_B_in
7	mapped	F32	DataLink_C_in
8	mapped	F32	DataLink_D_in



NOTE:

See section 5.4, Using DataLinks, for details on DataLinks A-D



Cyclic Data Access

Table 20: Assembly #106 PositionServo Basic Control

32-bit Word	Variable ID	Type	Name
0	52	F32	DRIVE ENABLE: Non 0 = enabled, 0 = disabled
1	139	F32	REFERENCE: Velocity mode = velocity in RPS; Current mode = current in phase A(rms)

Table 21: Assembly #108: PositionServo Extended Control.

32-bit Word	Variable ID	Type	Name
0	52	F32	DRIVE ENABLE: Non 0 = enabled, 0 = disabled
1	139	F32	REFERENCE: Velocity mode = velocity in RPS; Current mode = current in phase A(rms)
2	mapped	F32	DataLink_A_out
3	mapped	F32	DataLink_B_out
4	mapped	F32	DataLink_C_out
5	mapped	F32	DataLink_D_out



NOTE:

The Variable ID is the PositionServo variable's index number. Refer to the PositionServo Programming Manual (PM94H201)



NOTE:

Refer to section 5.4, Using DataLinks, for details on DataLinks A-D

5.6 Example Ladder Logic Program

The example Ladder Logic program illustrated in this section works with the CompactLogix controller and PositionServo drives.

5.6.1 Function of Example Program

This example program allows the user to:

1. Obtain status information from the drive.
2. Use the Logic Command to control the drive (for example enable/disable).
3. Send a Reference to the drive and receive Feedback from the drive.

5.6.2 RSLogix 5000 Configuration

Controller Tags

When the adapter and drive are added to the I/O configuration (Section 4.4 Fig. 21), the RSLogix 5000 controller automatically creates the controller tags for these devices. This example program uses the controller tags shown in Figure 22.

	+	-	SimpleServo:I	{..}	{..}	AB:ETHERNET_MODULE_REAL_36Bytes:I:0
	+	-	SimpleServo:O	{..}	{..}	AB:ETHERNET_MODULE_REAL_8Bytes:O:0
	+	-	SimpleServo:S	{..}	{..}	AB:ETHERNET_MODULE_DINT_8Bytes:S:0

Figure 22: Controller Tags

Cyclic Data Access



Click on the [+] icon next to the tag name to expand the tags and reveal the output and input configuration. The output tag for this example program requires two REAL data words as shown in Figure 22. The input tag for this example requires nine REAL data words (Figure 23) and input status tag requires two 32-bit words

<input type="checkbox"/>	SimpleServo:I	{ . . }	{ . . }		AB:ETHERNET_MODULE_REAL_36Bytes:I:0
<input type="checkbox"/>	SimpleServo:I.Data	{ . . }	{ . . }	Float	REAL[9]
	SimpleServo:I.Data[0]	0.0		Float	REAL
	SimpleServo:I.Data[1]	0.0		Float	REAL
	SimpleServo:I.Data[2]	0.0		Float	REAL
	SimpleServo:I.Data[3]	0.0		Float	REAL
	SimpleServo:I.Data[4]	0.0		Float	REAL
	SimpleServo:I.Data[5]	0.0		Float	REAL
	SimpleServo:I.Data[6]	0.0		Float	REAL
	SimpleServo:I.Data[7]	0.0		Float	REAL
	SimpleServo:I.Data[8]	0.0		Float	REAL
<input type="checkbox"/>	SimpleServo:O	{ . . }	{ . . }		AB:ETHERNET_MODULE_REAL_8Bytes:O:0
<input type="checkbox"/>	SimpleServo:O.Data	{ . . }	{ . . }	Float	REAL[2]
	SimpleServo:O.Data[0]	0.0		Float	REAL
	SimpleServo:O.Data[1]	0.0		Float	REAL
<input type="checkbox"/>	SimpleServo:S	{ . . }	{ . . }		AB:ETHERNET_MODULE_DINT_8Bytes:S:0
<input checked="" type="checkbox"/>	SimpleServo:S.Data	{ . . }	{ . . }	Decimal	DINT[2]
	SimpleServo:S.Data[0]	0		Decimal	DINT
	SimpleServo:S.Data[1]	0		Decimal	DINT

Figure 23: Input, Output and Status Tags for Ladder Logic Program

Program Tags

In addition to the controller tags that are automatically created, the following program tags must be created for this example program.

	Name	Value	△	Force Mask ←	Style ←	Data Type	Description
	CMD_DriveEnable				Decimal	BOOL	Enable command
<input checked="" type="checkbox"/>	Simple Servo:C	{ . . }		{ . . }		AB:ETHERNET_MODULE:C:0	
<input checked="" type="checkbox"/>	SimpleServo:I	{ . . }		{ . . }		AB:ETHERNET_MODULE_REAL_36Bytes:I:0	
<input checked="" type="checkbox"/>	SimpleServo:O	{ . . }		{ . . }		AB:ETHERNET_MODULE_REAL_8Bytes:O:0	
<input checked="" type="checkbox"/>	SimpleServo:S	{ . . }		{ . . }		AB:ETHERNET_MODULE_DINT_8Bytes:S:0	
	VelocityReferenceRPS	0.0			Float	REAL	Drive velocity reference
	StatusFlag_DriveEnabled	0			Decimal	BOOL	
	StatusFlag_DriveFaut	0			Decimal	BOOL	
	ActualVelocity	0.0			Float	REAL	Feedback velocity from the...
	CMD_ReferenceVelocity	0.0			Float	REAL	Velocity set point reference
	DriveEnable	0.0			Float	REAL	Drive enable control variable...
<input checked="" type="checkbox"/>	PLCOUT_DriveEnabled	0			Decimal	BOOL	
	PLCOUT_DriveFault	0			Decimal	BOOL	

Figure 24: Program Tags for Example Program

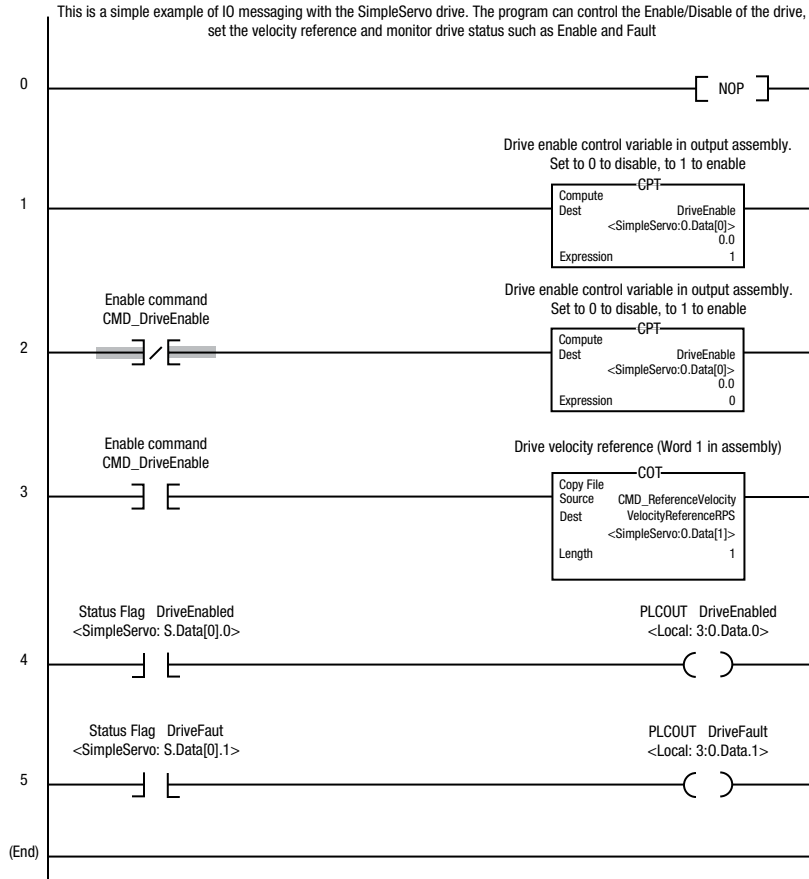


Cyclic Data Access

This example uses the I/O assemblies mapped as shown previously Figures 20 and 21.

Main Routine - Ladder Diagram
 CompactLogix1: Main Task: Main Program
 Total number of rungs in routine: 6

Page 1
 mm/dd/yyyy hh:mm:ss PM
 CompactLogix1.ACD



RSLogix 5000

Figure 25: Example Ladder Logic Program



6 Explicit Messages

Explicit Messaging is used to transfer data that does not require continuous updates. With Explicit Messaging, you can configure and monitor a slave device's parameters on the EtherNet/IP network. This section provides information and examples that explain how to use Explicit Messaging to monitor and configure a PositionServo drive.



STOP!

Risk of injury to personnel and/or damage to equipment exists. The examples in this publication are intended solely for purposes of example. Lenze AC Tech Corporation does not assume responsibility or liability (to include intellectual property liability) for actual use of the examples shown in this publication.



STOP!

Risk of equipment damage exists. If Explicit Messages are programmed to frequently write parameter data to Non-Volatile Storage (NVS), the NVS will quickly exceed its life cycle and cause the drive to malfunction. Do not create a program that frequently uses Explicit Messages to write parameter data to NVS. DataLinks do not write to NVS and should be used for frequently changed parameters.

6.1 Formatting Explicit Messages

Explicit Messages for the ControlLogix Controller:

CompactLogix accommodates both downloading Explicit Message Requests and uploading Explicit Message Responses. The controller can accommodate one request or response for each transaction block. Each transaction block must be formatted as shown in the Figure 26.

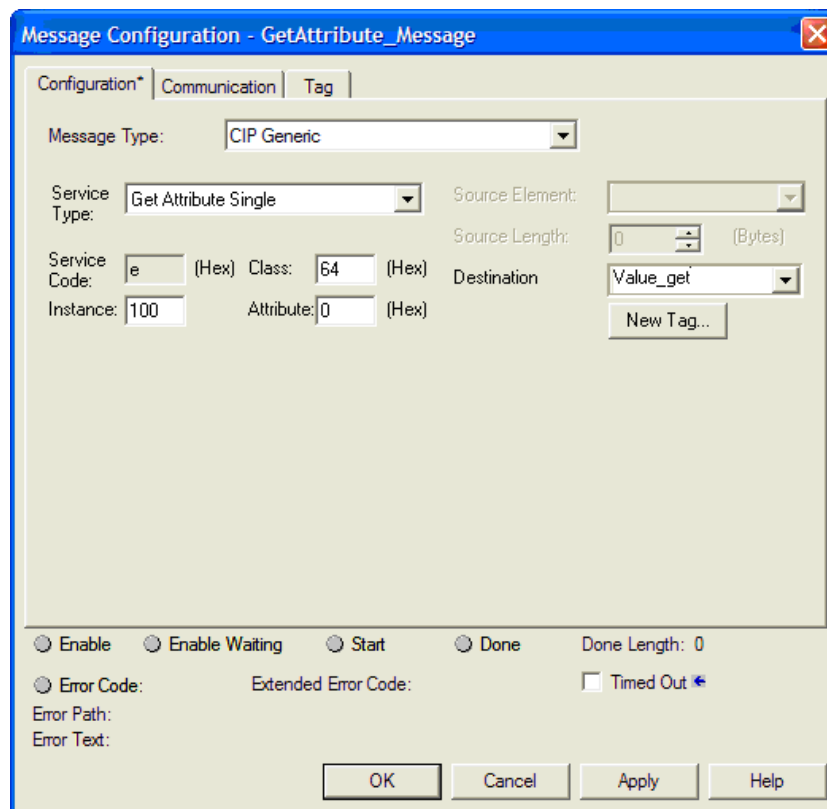


Figure 26: Explicit Message Configuration Dialog Box



Acyclic Data Access

**NOTE:**

To display the Message Configuration dialog box in RSLogix 5000, add a message instruction (MSG), create a new tag for the message (properties: Base tag type, MESSAGE data type, controller scope), and click the Configure button.

Table 22: Configuration Dialog Fields for Explicit Message in RSLogix 5000

Box	Description
Message Type	The message type is usually CIP Generic.
Service Type	The service type indicates the service (for example, Get Attribute Single or Set Attribute Single) to be performed. Available services depend on the class and instance in use.
Service Code	The service code is the code for the requested EtherNet/IP service. This value changes based on the Service Type that has been selected. In most cases, this is a read-only box. If you select "Custom" in the Service Type box, then you need to specify a service code in this box (for example, E(h) for a Get Attributes Single service or 10(h) for a Set Attributes Single service)
Class	The class is an EtherNet/IP class. Refer to Section 7, EtherNet/IP Objects, for available classes. The most frequently used classes for the PositionServo are: 4 (Assembly object) and 64(h) PositionServo System variables class.
Instance	The instance is an instance (or object) of an EtherNet/IP class. Refer to Section 7, EtherNet/IP Objects, for available instances for each class.
Attribute	The attribute is a class or instance attribute. Refer to Section 7, EtherNet/IP Objects, for available attributes for each class or instance.
Source Element	This box contains the name of the tag for any service data to be sent from the PLC to the drive.
Source Length	This box contains the number of bytes of service data to be sent in the message.
Destination	This box contains the name of the tag that will receive service response data from the adapter and drive.
Path	The path is the route that the message will follow. Tip: Click Browse to find the path or type in the name of an adapter that you previously mapped.
Name	The name for the message.



6.2 Performing Explicit Messages

There are five basic events in the Explicit Messaging process as defined herein and illustrated in Figure 27. The details of each step will vary depending on the controller (ControlLogix, PLC, or SLC). Refer to the documentation for your controller.



NOTE:

There must be a request message and a response message for all Explicit Messages, whether you are reading or writing data.

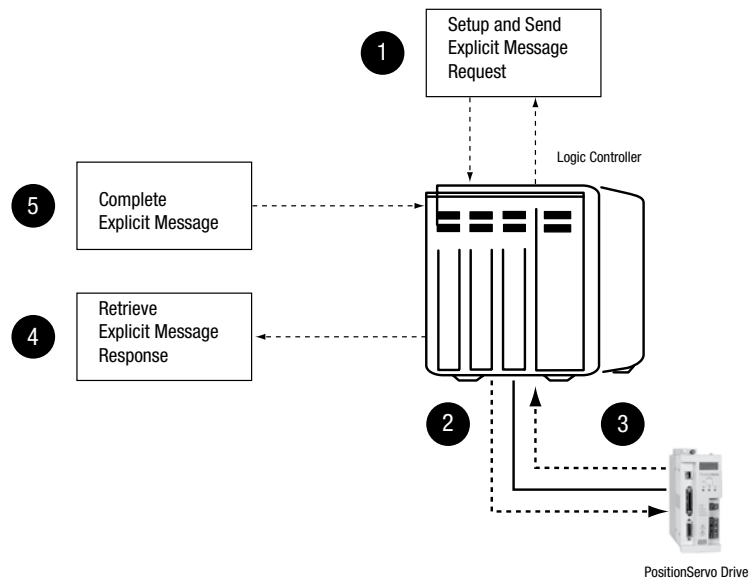


Figure 27: Explicit Messaging Process

Event in Explicit Messaging Process:

1. Format the required data and setup the ladder logic program to send an Explicit Message request to the scanner or bridge module (This is a download).
2. The scanner (or bridge) module transmits the Explicit Message Request to the slave device over the EtherNet/IP network.
3. The slave device transmits the Explicit Message Response back to the scanner. The data is stored in the scanner buffer.
4. The controller retrieves the Explicit Message Response from the scanner's buffer (This is an upload).
5. The Explicit Message is complete.

The scanner module may be integrated with the controller as it is in the CompactLogix controller.



Acyclic Data Access

6.3 Explicit Message Example

To format and execute a [Get Attribute Single] or [Set Attribute Single] Explicit Message using a CompactLogix controller, use this example program.

Message Formats

When formatting an example message, refer to Formatting Explicit Messages in this chapter for an explanation of the content of each box. Also, to format and execute these example messages use the Controller tags displayed in Figure 28.

Scope		CompactLogix1		Show...			STRING, ALARM, AXIS_CONSUMED, AXIS_GENERIC, AXIS_GENERIC_DRIVE, AXIS_SERVO, AXIS_SEF		
	Name	Alias For	Base Tag	Data Type	Style	Description			
▶	+ Simple Servo:C			AB:ETHERNET_MOD...					
	+ SimpleServo:S			AB:ETHERNET_MOD...					
	+ SimpleServo:I			AB:ETHERNET_MOD...					
	+ SimpleServo:O			AB:ETHERNET_MOD...					
	CMD_GetValue			BOOL	Decimal				
	CMD_SetValue			BOOL	Decimal				
	+ GetAttribute_Message			MESSAGE					
	+ SetAttribute_Message			MESSAGE					
	Value_Get			REAL	Float	Value received from...			
	Value_Set			REAL	Float	Value to send to drive.			

Figure 28: Controller Tags for Explicit Message Example

Ladder Logic Rungs

The ladder logic rungs for the examples in this chapter can be appended after rung 5 in the ladder logic program illustrated in Figure 25 or the program can be constructed as a stand alone program.



6.3.1 Example of Get Attribute Single Message (Rung 1 of Figure 31)

Figure 29 illustrates the configuration of the message to read the value from the drive to the PLC controller memory. In this example, the PLC reads instance #100 (User Variable V0) from the PositionServo system variables class 64(h) and stores it in the controller tag Value_Get.

In the [Message Configuration] menu set the parameters for this example as listed in Table 23 and illustrated in Figure 29.

Table 23: Message Configuration Parameters for Get Attribute Single

Step 1	Step 2	Step 3
Click on the [Tag] tab and set:	Click on the [Communication] tab and set:	Click on the [Configuration] tab and set:
Tag Name: GetAttribute_Message	Path: SimpleServo	Message Type: CIP Generic
Description: leave blank	Communication Method: CIP	Service Type: Get Attribute Single
Type: Base	Connected: Put a check in this box	Service Code: e
Alias For: leave blank		Instance: 100
Data Type: MESSAGE		Class: 64
Scope: CompactLogix1		Attribute: 2
Style: leave blank		Source Element: leave blank
		Source Length: 0
		Destination: Value_Get

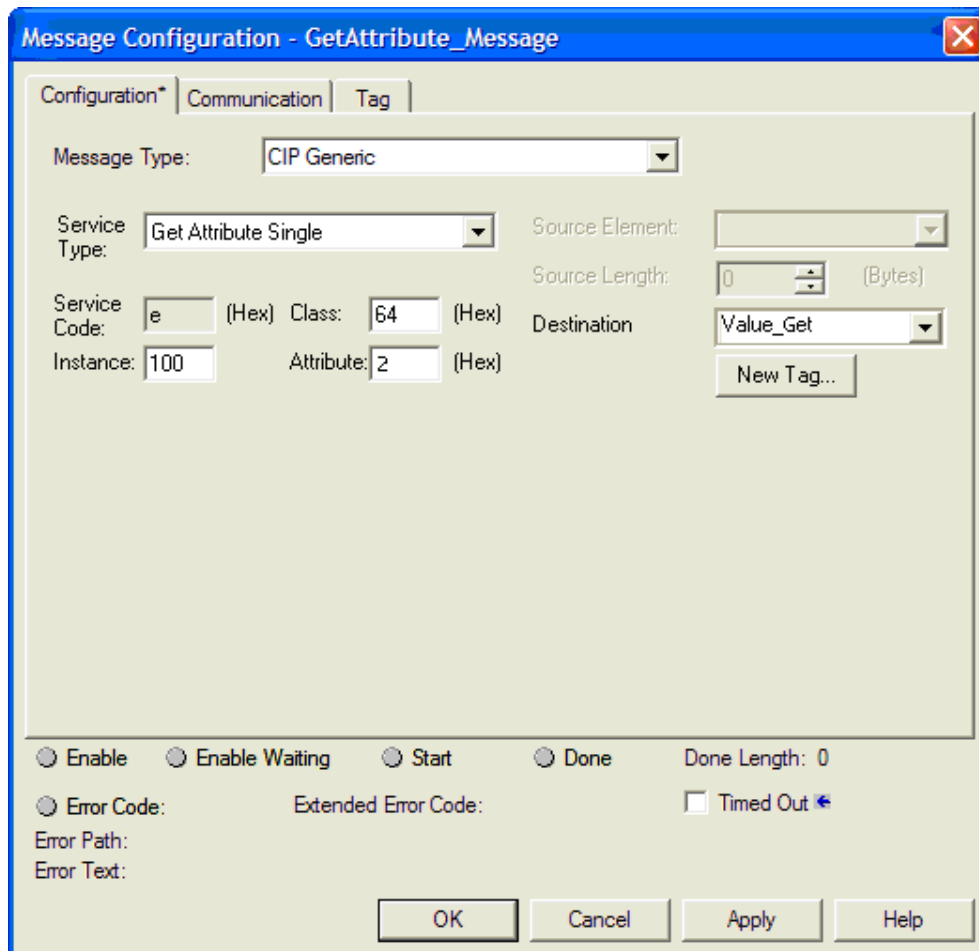


Figure 29: Get Attribute Single



Acyclic Data Access

6.3.2 Example of Set Attribute Single Message (Rung 2 of Figure 31)

Figure 30 illustrates the configuration of the message to write the value from the PLC controller memory to the drive. In this example, the PLC writes instance #100 (User Variable V0) from the controller tag Value_Set.

Table 24: Message Configuration Parameters for Set Attribute Single

Step 1	Step 2	Step 3
Click on the [Tag] tab and set:	Click on the [Communication] tab and set:	Click on the [Configuration] tab and set:
Tag Name: SetAttribute_Message	Path: SimpleServo	Message Type: CIP Generic
Description: leave blank	Communication Method: CIP	Service Type: Set Attribute Single
Type: Base	Connected: Put a check in this box	Service Code: 10
Alias For: leave blank		Instance: 100
Data Type: MESSAGE		Class: 64
Scope: CompactLogix1		Attribute: 2
Style: leave blank		Source Element: Value_Set
		Source Length: 4
		Destination: leave blank

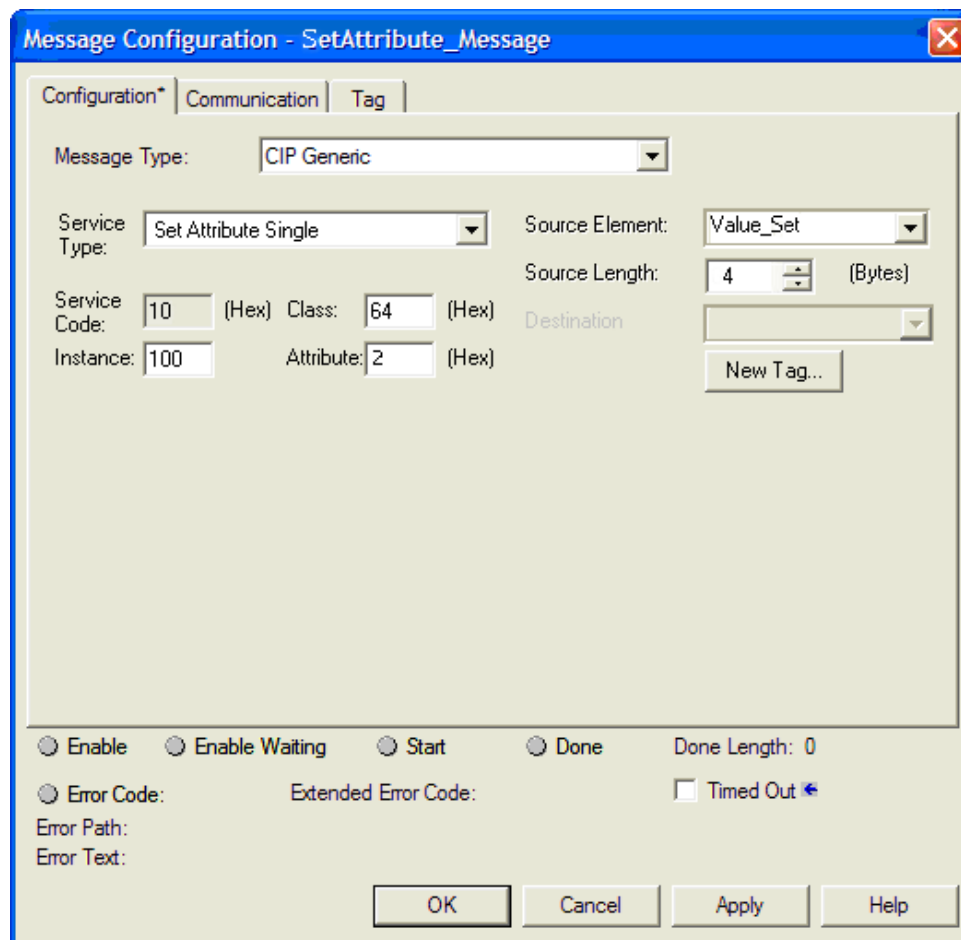


Figure 30: SetAttribute_Single

Acyclic Data Access

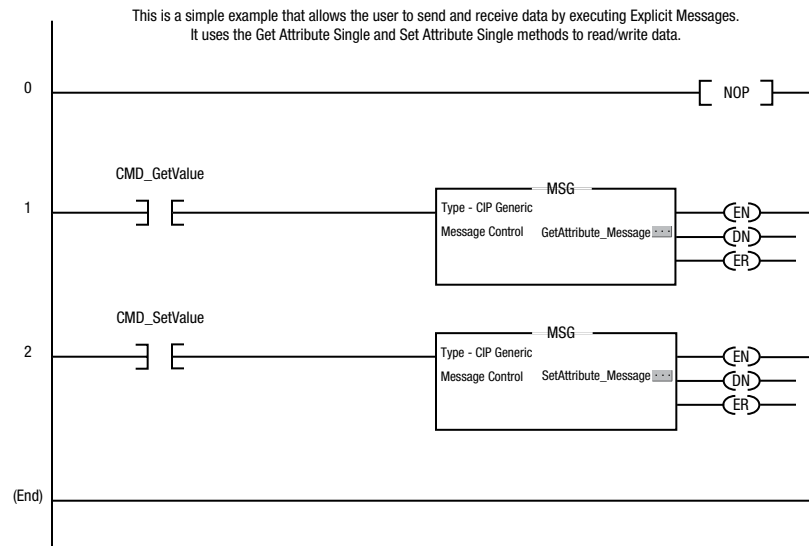


Main Routine - Ladder Diagram

CompactLogix1: Main Task: Main Program
Total number of rungs in routine: 3

Page 1

mm/dd/yyyy hh:mm:ss PM
CompactLogixExplicitMessaging.ACD



RSLogix 5000

Figure 31: Example Ladder Logic Explicit Message Program



Acyclic Data Access

7 Ethernet/IP Objects

Section 7 contains information about the Ethernet/IP objects that can be accessed using Explicit Messages. For information on the format of Explicit Messages and example ladder logic programs, refer to section 6.

Table 25: Ethernet/IP Objects

Object	Class Code	
	Hex	Dec
Identity	0x01	1
Assembly	0x04	4
System940	0x64	100
TCP/IP Interface Object	0xF5	245
Ethernet Link Object	0xF6	246

The CIP family of protocols has a library of commonly defined objects currently divided into 46 object classes. Class 1 = DeviceNet, Class 2 = EtherNet/IP and Class 3 = ControlNet. Refer to the EtherNet/IP specification for more information about Ethernet/IP objects. Information about the EtherNet/IP specification is available on the ODVA web site (<http://www.odva.org>).

7.1 Identity Object

The Identity Object defines the device. A device typically does not change its identity so all attributes are normally read only. The identity object's data can be queried from a target node without knowing what that device is before a message is sent. From this data, the EDS file of the device can be identified.

Class Code	0x01	
Class Attributes:		
	Revision	1
	Max Instance	1
	Number Instance	
	Max ID# class attribute	
	Max ID# instance attribute	
Class Services:		
	Get_Attribute_Single()	
	Instance 1	
	Instance Attributes	
	Vendor ID	
	Device Type	
	Product Code	
	Revision	
	Major	
	Minor	
	Product Name	EnetIP 940
	Instance Services	
	Get_Attribute_All	
	Get_Attribute_Single	
	Reset	



7.2 PositionServo System Object

The PositionServo system object encapsulates all valid PositionServo variables. Each PositionServo variable is represented by an instance of a System940 object. The instance number therefore matches the variable's index. A complete list of PositionServo variables with their corresponding indices is in the PositionServo Programming Manual (PM94H201).

All aspects of control and parameterization in the PositionServo drive are accomplished through the system variables. Some of the variables are parameters such as Current Limit or Target Position. Some of the variables are action properties, i.e. writing values to these variables will execute a particular process. As an example, writing to variable VAR_ENABLE (ID=52), a non-0 value will enable the drive. Writing the same variable with a 0 value will disable the drive. Another example could be writing the variable VAR_MOVED with a value of 10, which would execute relative motion for 10 user units.

Every variable in the PositionServo can be read/written as a 32-bit INTEGER or 32-bit REAL(float) value. Conversion is done automatically. In addition each variable can be read from its RAM (current) copy or from non-volatile (EPM) storage. The value is initialized at the time of power up.

To accommodate different access (RAM or EPM) and format (integer or float) types, attributes are implemented. For example to reach variable VAR_CURRENTLIMIT (ID=30) as FLOAT in RAM (run-time value) you would use InstanceID = 30 with attribute 2. For the same variable accessed in EPM (non-volatile copy) you would use attribute 3.

Class Code		0x64	
Class Attributes:			
Class Services:			
	Instance Attributes		
		Integer, RAM	0
		Integer, EPM	1
		Float, RAM	2
		Float, EPM	3
		String, RAM	4
		String, EPM	5
	Instance Services		
		Get_Attribute_All	
		Get_Attribute_Single	
	Instance	Instance = variable ID. Refer to PositionServo Programming Manual (PM94H201) for variable ID list. For example: Instance of VAR_CURRENTLIMIT is 30 since its ID=30	



NOTE:

For Attributes 4 and 5, the PositionServo uses a 4 byte header in the data to denote the number of bytes in the ASCII string AFTER the 4 byte header. When performing an explicit write to either Attributes 4 or 5 the user must set the length of the message equal to the number of ASCII bytes for the data +4.



NOTE:

Attributes 1, 3 and 5 are WRITE ONLY to the EPM. Attempts to read attributes 1, 3 or 5 result in the data pulled from attributes 0, 2 and 4 respectively.



Acyclic Data Access

7.3 Assembly Object

An Assembly Object is the “assembly” or mapping of data from different instances of various classes into a single attribute. With assembly mapping, the I/O data is produced in one block. An assembly object can be used to configure a device using one block of data instead of setting the individual device parameters.

Class Code		0x04	
Class Attributes:			
		Revision	2
		Max Instance	
Class Services:			
		Get_Attribute_Single()	
	Instances:		
		Refer to Table 14	
	Instance Attributes		
			3
	Instance Services		
		Get_Attribute_All	
		Get_Attribute_Single	

7.4 TCP/IP Interface Object

The TCP/IP Interface object is the connection object that allows for I/O and Explicit messages to be sent from the device on the network to the other devices.

Class Code		0xF5	
Class Attributes:			
		Revision	1
		Max Instance	
		Number Instance	
Class Services:			
		Get_Attribute_All	
		Get_Attribute_Single	
	Instance Attributes		
		Status	
		Configuration Capability	6
		Configuration Control	
		Physical Link -> Path	0x20 0xF6 0x24 0x01
		Interface Configuration	
		Host Name	
		TTL	
		Mcast Config	
	Instance Services		
		Get_Attribute_Single	
		Set_Attribute_Single	



7.5 Ethernet Link Object

The Ethernet Link object is the network link object that defines the CIP as Ethernet, DeviceNet or ControlNet.

Class Code		0xF6	
Class Attributes:			
		Revision	1
		Max Instance	1
		Number Instance	
Class Services:			
		Get_Attribute_All()	
		Get_Attribute_Single()	
	Instance Attributes		
		Interface Speed	10
		Interface Flags	
		Physical Address	
	Instance Services		
		Get_Attribute_Single	
		Get_Attribute_All	
		Set_Attribute_Single	
	Instance Specific Service		
		Get_and_Clear()	
	Instance		
		none	



8 Applications

8.1 Application Example 1 - Velocity Control

This application illustrates how to control velocity using an Allen-Bradley PLC and an AC Tech PositionServo drive.

Objective:

This example shows how to use I/O messaging (I/O scan) to control the PositionServo drive in velocity mode using Ethernet/IP communication protocol.

Equipment:

1. PositionServo drive (firmware revision 3.4 or later)
2. Allen-Bradley PLC SoftLogix (CompactLogix, ControlLogix can be used with modifications to the I/O mapping specific to these models)
3. Ethernet hub or switch.

Description:

This example implements 4 simple presets of velocity drive with the velocity window comparator driven by the actual velocity read from the drive over the Ethernet/IP interface.

PositionServo drives support I/O messaging. I/O messaging is convenient when data must be updated cyclically i.e. within a certain update time interval (rate) like a speed or torque reference. The PositionServo drive has an Assembly Object (class 0x04) with several instances for Input and Output. Refer to Section 7.2, Assembly Object, for details on the implementation of assembly objects with the PositionServo. This example uses:

- Assembly #104,105 (input) and #106 (output)
- Location 0 of #106 is used to control the drive's ENABLE/DISABLE state
- Location 1 of #106 is used to set the velocity reference (IREF)
- Location 0 of #105 is used to monitor the actual shaft velocity



Requirements:

A PositionServo drive must be configured before this example can be executed. The PositionServo drive can be configured in 2 ways: by using MotionView software or by running a short user's program. Note that setup can also be performed using Explicit messages (refer to section 8.3). The configuration file (for use with MotionView) and the user's program are both provided on the same CD this example resides.

Table 26: PositionServo Drive Setup

MotionView			User's Program Statement
	Parameter	Value	
1	Units	1	VAR_UNITS=1
2	Drive mode	Velocity	VAR_DRIVEMODE=1
3	Enable Accel/Decel limits	Enable	VAR_ENABLEACCELDECEL=1
4	Accel	100	VAR_ACCEL_LIMIT=100
5	Decel	100	VAR_DECEL_LIMIT=100
6	Reference	Internal	VAR_REFERENCE=1
7	Enable switch function	Inhibit	VAR_ENABLE_SWITCH_TYPE=0

Running the example:

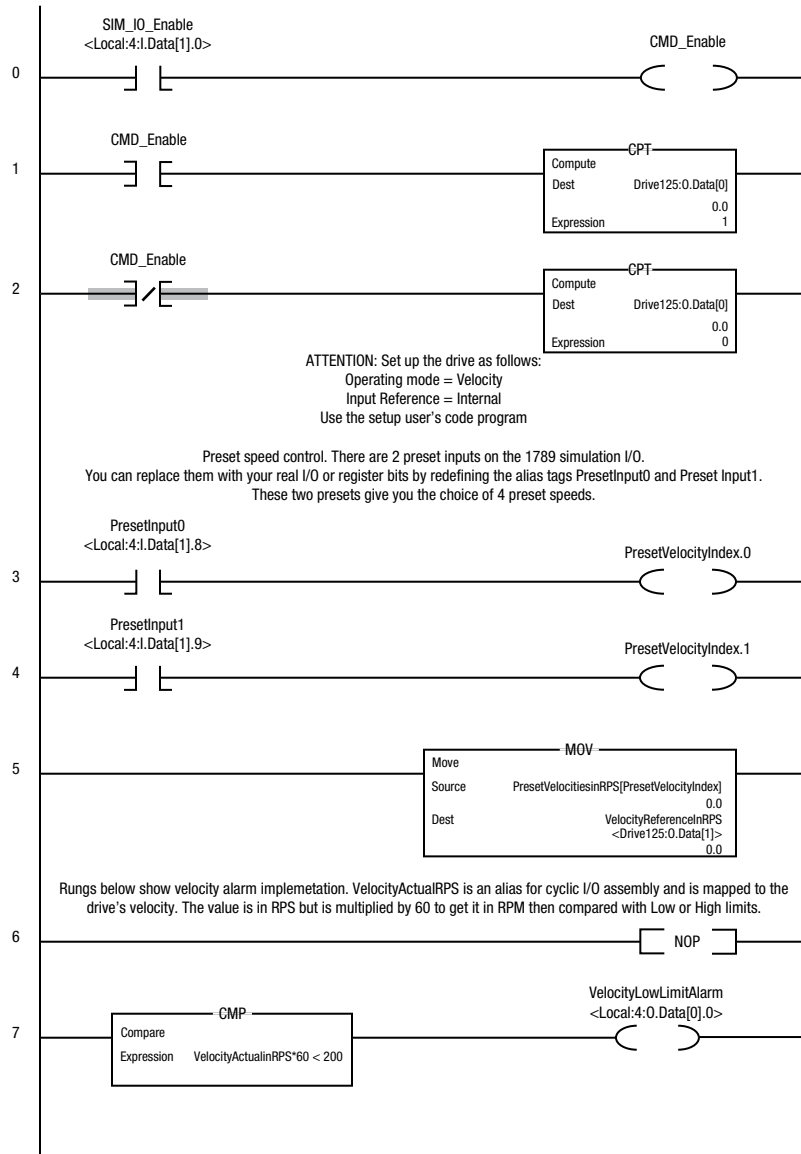
1. Configure PositionServo drive as shown in Table 26.
2. Make sure the A3_IN input of the PositionServo drive is energized so the drive is not hardware-inhibited to run.
3. Open the project file "SoftLogixVelocityControl.ACD".
4. Add the 1789-SIM software module to your chassis.
5. Edit the I/O module 1789-MODULE SIM_I032 properties to change the controller slot number to match your chassis.
6. Edit the Ethernet/IP adapter address to match your computer IP address.
7. Edit the Ethernet adapter Drive125 to match your PositionServo IP address.
8. Load the project to your controller and set the PLC to RUN.
9. Operate using the Simulated module I/O to enable the drive and then set the different presets using the SIM-I032 Soft IO simulator.



Applications

Main Routine - Ladder Diagram
 SoftLogixMSGgen: Main Task: Main Program
 Total number of rungs in routine: 9

Page 1
 mm/dd/yyyy hh:mm:ss AM
 SoftLogixVelocityControl.ACD



RSLogix 5000

Figure 32: Velocity Control Ladder Program



8.2 Application Example 2 - Indexing

This application illustrates how to index using an Allen-Bradley PLC and an AC Tech PositionServo drive.

Objective:

This example shows how to use explicit messages to configure indexing parameters and issue indexing commands. I/O messaging (I/O scan) is used to monitor real time data such as status, velocity target and actual position etc.

Equipment:

1. PositionServo drive (firmware revision 3.4 or later)
2. Allen-Bradley PLC SoftLogix (CompactLogix and ControlLogix can be used with modifications to the I/O mapping specific to these models)
3. Ethernet hub or switch

Description:

This example implements a simple indexer. Two inputs of the PLC are used to select the current index and one input is used to start the indexing. All parameters are setup for the desired index using explicit messages. The Start command also uses an explicit message.

PositionServo drives support both explicit and I/O messaging features. In this example, explicit messages are used for configuring and starting the indexing and I/O messaging is used for monitoring the drive status and changing other data like velocity or position.

The PositionServo drive has an Assembly Object (class 0x04) with several instances for Input and Output. Refer to Section 7.2, Assembly Object, for details on the implementation of assembly objects with the PositionServo. This example uses:

- Assembly #104,105 (input) and #106 (output) are used in this example
- Location 0 of #106 is used to control the drive's ENABLE/DISABLE state
- Location 1 of #106 is used to set the velocity reference (IREF)
- Location 0 of #105 is used to monitor the actual shaft velocity



Applications

Requirements:

A PositionServo drive must be configured before this example can be executed. The PositionServo drive can be configured in 2 ways: by using MotionView software or by running a short user's program. Note that setup can also be performed using Explicit messages (refer to section 8.3). The configuration file (for use with MotionView) and the user's program are both provided on the same CD this example resides.

Table 27: PositionServo Drive Setup

MotionView			User's Program Statement
	Parameter	Value	
1	Units	1	VAR_UNITS=1
2	Drive mode	Position	VAR_DRIVEMODE=2
3	Reference	Internal	VAR_REFERENCE=1
7	Enable switch function	Inhibit	VAR_ENABLE_SWITCH_TYPE=0

Running the example:

1. Configure the PositionServo drive as shown in Table 27.
2. Make sure the A3_IN input of the PositionServo drive is energized so the drive is not hardware-inhibited to run.
3. Open the project file "SoftLogixVelocityControl.ACD"
4. Add the 1789-SIM software module to your chassis.
5. Edit the I/O module 1789-MODULE SIM_I032 properties to change controller slot number to match your chassis.
6. Edit the Ethernet/IP adapter address to match your computer IP address
7. Edit the Ethernet adapter Drive125 to match your PositionServo IP address
8. Load the project to your controller and set the PLC to RUN.
9. Operate using the Simulated module I/O to enable the drive and then set the different presets using the SIM-I032 Soft IO simulator.



Example Details:

1. The simulated software I/O module 1789-SIM is used to control the application. You can substitute your I/O with one from your target PLC or create BOOL type tags and use them instead of the I/O to control the application.
2. The I/O are assigned per Table 28.

Table 28: I/O Assignments

I/O #	Assignment
0	Drive enable/disable
1	Set motion profile (set accel, decel, max velocity)
2	Executes index
8	Index input 0
9	Index input 1

3. To execute the index, (upon I/O2 engaged) the PLC sends the value of the motion distance from the internal tag to the PositionServo system variable #93 (MOVED) using an explicit message. Writing the value to the variable causes the drive to start to index (relative motion).
4. Before the first move can be executed, I/O1 must be activated to set the move parameters such as accel, decel and profile max. velocity. The logic is written in such a way (rung 7) so that the MOVE command will be ignored until all messages for setting up the parameters are executed. This will prevent situations where the index command is issued while setup is in progress.
5. The value of the index is taken from the PLC tag pointed to by I/O8 and 9. The PLC tag allows 4 different indexes.

Feel free to modify this example for the number of indexes needed. The index pointer is in the MotionIndex Selector tag. It holds the index number that will index the motion values array tag, MotionIndexes[]. You can modify the dimension of this array to hold required number of indexes.

Finally variable #93 can be changed to variable #92 and this will execute Absolute position motion. The indexes stored in MotionIndexes[] will then represent the actual position value. Also, variables #177 and #178 can be used instead of variables #93 and #92 (respectively) to create S-curved motion.



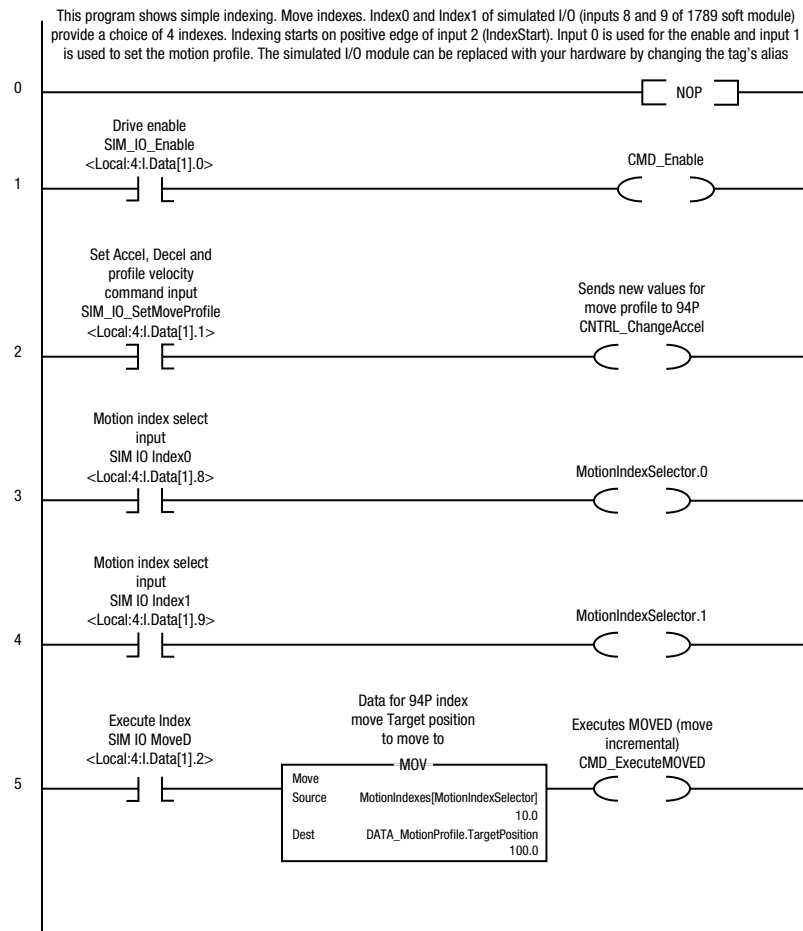
Applications

Main Routine - Ladder Diagram

SoftLogixMSGgen: Main Task: Main Program
Total number of rungs in routine: 11

Page 1

mm/dd/yyyy hh:mm:ss AM
SoftLogixIndexing.ACD



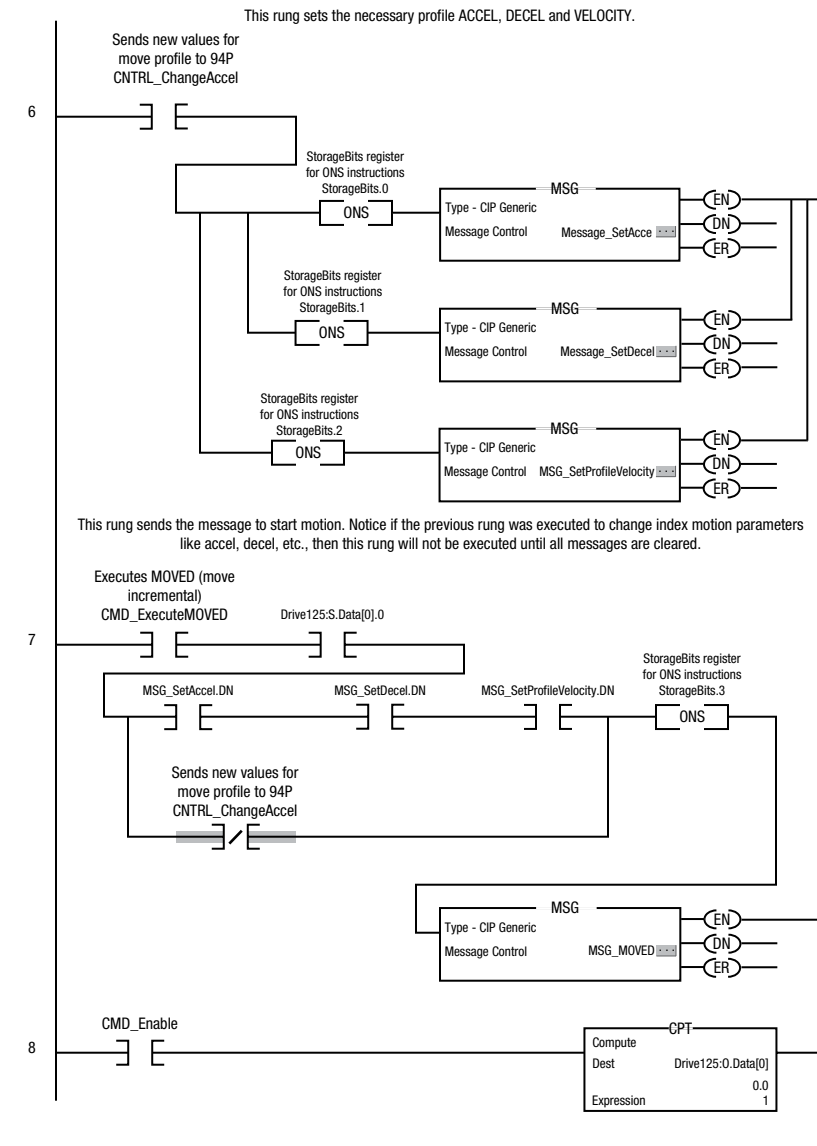
RSLogix 5000

Figure 33a: Indexing Ladder Diagram



Main Routine - Ladder Diagram
 SoftLogixMSGgen: Main Task: Main Program
 Total number of rungs in routine: 11

Page 2
 mm/dd/yyyy hh:mm:ss AM
 SoftLogixIndexing.ACD



RSLogix 5000

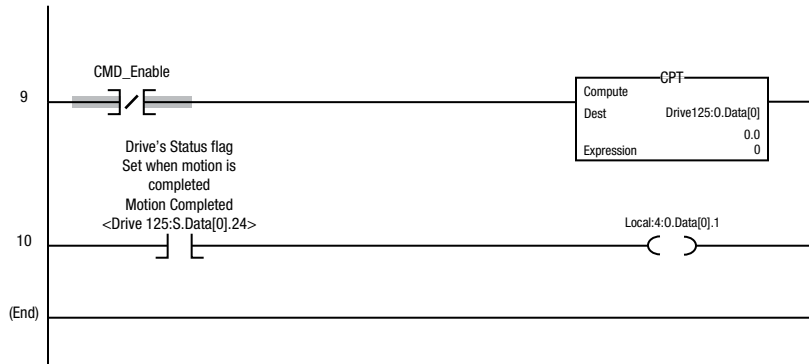
Figure 33b: Indexing Ladder Diagram, Page 2



Applications

Main Routine - Ladder Diagram
SoftLogixMSGgen: Main Task: Main Program
Total number of rungs in routine: 11

Page 3
mm/dd/yyyy hh:mm:ss AM
SoftLogixIndexing.ACD



RSLogix 5000

Figure 33c: Indexing Ladder Diagram, Page 3



8.3 Application Example 3 - Configuration Using Explicit Messages

This application illustrates how to configure a PositionServo drive using explicit messages.

Objective:

This example shows how to configure a PositionServo drive by sending a list of explicit messages.

Equipment:

1. PositionServo drive (firmware revision 3.4 or later)
2. Allen-Bradley PLC SoftLogix (CompactLogix and ControlLogix can be used with modifications to the I/O mapping specific to these models)
3. Ethernet hub or switch

Description:

This example implements a state machine which sends a list of predefined messages to the PositionServo drive. Since the sending a message process inside the PLC takes few steps, the message is sent successfully when the MSG instruction returns the status. The operation of the rest of the logic in PLC program usually depends upon result of the MSG operation. In a situation where multiple messages need to be sent, redundant status handling logic multiplies and complicates the overall PLC program. This example illustrates a practical example of the usage of multiple configuration messages. The velocity control program from Application Example 1 (section 8.1) will be used.

Recall that the PositionServo drive needs to be configured ahead of time to be used in Example 1. The drive's mode, accel, decel and other parameters need to be configured by using MotionView software or by writing a short user's program. We will modify the PLC program in such a way that the configuration is performed directly from the PLC.

To do this create a MSG instruction and configure it with the PositionServo drive address, Class 0x64 (PositionServo variables class) and the CIP table operation SetAttributeSingle. Create two arrays – one holding instances of the class (i.e system variables IDs) and the other array will hold VALUES that the drive's variables will be set to.

Then execute the MSG instruction as many times as there are number of variables to set, each time setting the MSG instruction INSTANCE with the value from the list. The 'value to send' (actual data) will be copied from the second array to the designated tag that the MSG instruction Source Data field is pointed too.

RSLogix 5000 File:

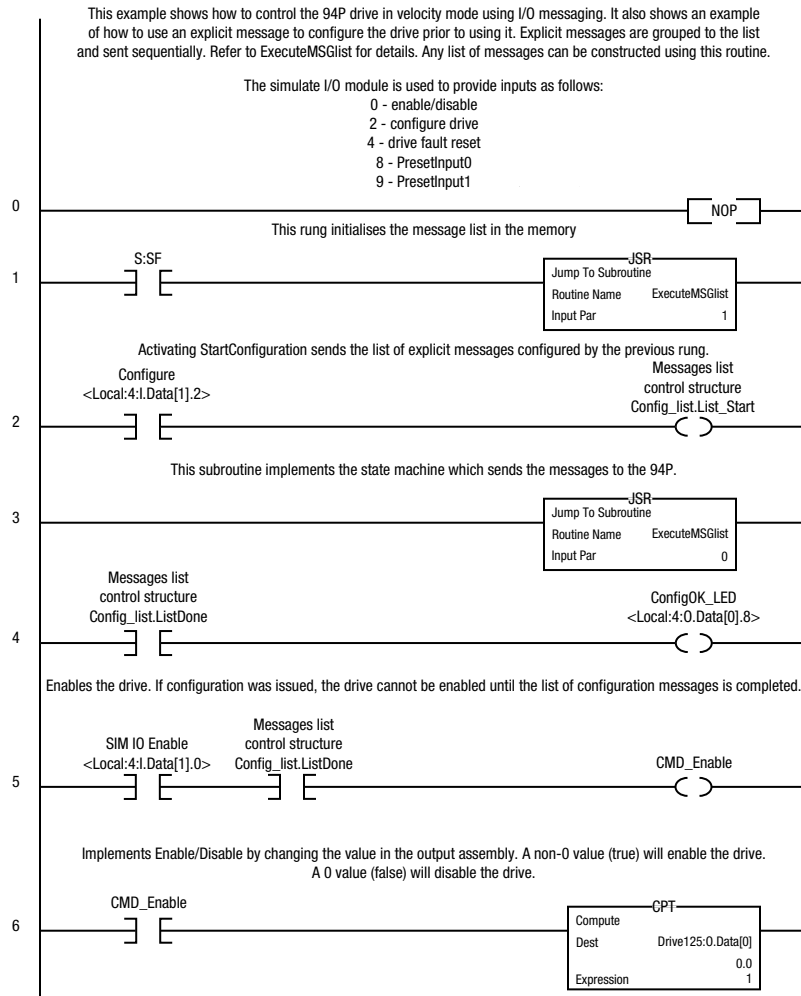
For this Application Example open the project file "SoftLogixConfigurationMessages.ACD" (on CD).



Applications

Main Routine - Ladder Diagram
 SoftLogixMSGgen: Main Task: Main Program
 Total number of rungs in routine: 17

Page 1
 mm/dd/yyyy hh:mm:ss PM
 SoftLogixConfigurationMessages.ACD



RSLogix 5000

Figure 34a: Explicit Messages Ladder Diagram Page 1

Applications

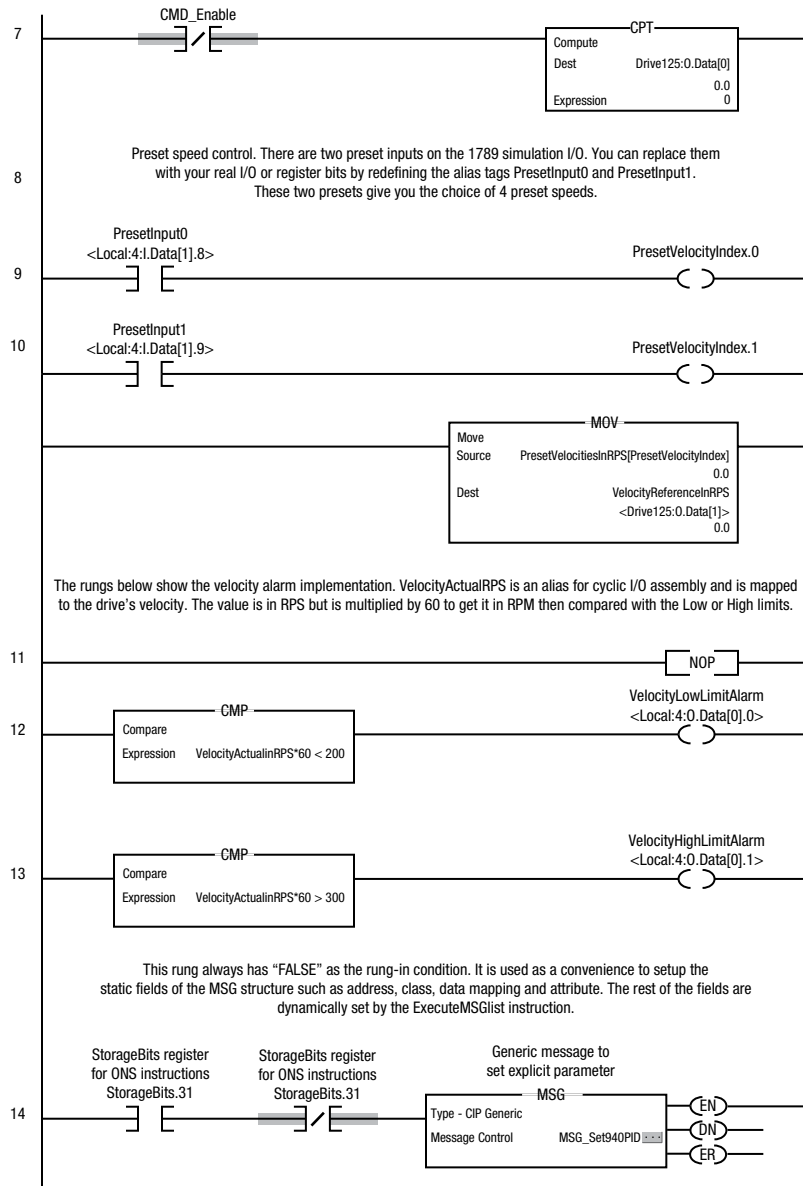


Main Routine - Ladder Diagram

SoftLogixMSGgen: Main Task: Main Program
Total number of rungs in routine: 17

Page 2

mm/dd/yyyy hh:mm:ss AM
SoftLogixConfigurationMessages.ACD



RSLogix 5000

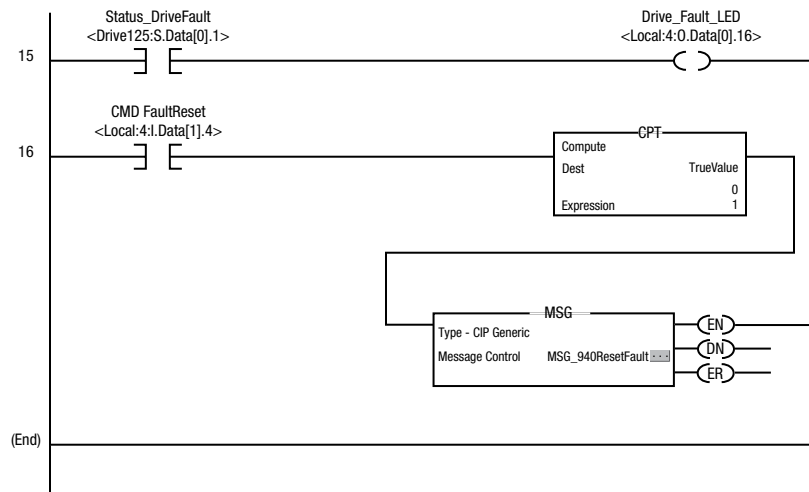
Figure 34b: Explicit Messages Ladder Diagram Page 2



Applications

Main Routine - Ladder Diagram
SoftLogixMSGgen: Main Task: Main Program
Total number of rungs in routine: 17

Page 3
mm/dd/yyyy hh:mm:ss AM
SoftLogixConfigurationMessages.ACD



RSLogix 5000

Figure 34c: Explicit Messages Ladder Diagram Page 3



Data type Name: Msg_List_Control

Description:
Messages list control structure

Size: 184 byte(s)

Name	Data Type	Style	Description
State	DINT	Decimal	
ListLength	DINT	Decimal	
MsgIndex	DINT	Decimal	
ListDone	BOOL	Decimal	
ListErr	BOOL	Decimal	
List_instance	DINT[20]	Decimal	
List_Value	REAL[20]	Float	
List_CurrentValue	REAL	Float	
List_latch	BOOL	Decimal	
List_Start	BOOL	Decimal	

RSLogix 5000

Figure 35: User's Defined Type for Message List Control

Rung 1

At the first scanner call to subroutine, ExecuteMSGlist with argument = 1 initialises the list of explicit messages. Refer to the routine in this section. The list of messages is kept in tag Config_List of the user's defined type (Figure 34a). The tag has 2 arrays inside: List_instance[] and List_value[]. These arrays will hold pairs of instances-value when the MSG instruction fields are configured for the next send.

Rung 2,3

When I/O2 is triggered, ExecuteMSGlist gets called with argument = 0. This starts the state machine (if transmission is not already in progress) and transmits messages from the list one by one. Calling the subroutine does not stop the scanning but merely starts the transmission of the next message from the list if the previous one has already been sent until the list is not over. Transmission of the message happens in the background without waiting until message TX is completed.



Applications

Rung 4

Indicates on the I/O that the full configuration list is done. This status interlocks rung 5 to prevent the ENABLE command before configuration is completed.

Rung 5-13,15

Same as in original example.

Rung 14

This rung is a “Convenience” entry. This rung never becomes true. It is needed to configure the MSG instruction field with constants (drive address, message type etc.). The instance field is changed by the ExecuteMSGlist routine according to the value in the list.

User’s defined type for messages list control:

ExecuteMSGlist routine source code

```
//This is sub program sends sequence of messages
//This routine is useful for configuration sequences.
SBR(InitList);

IF InitList THEN
//Initialisation state. List of messages initialized

//0 Drive mode - velocity
    Config_list.List_instance[0] :=34;
    Config_list.List_Value[0] :=1;
//1 Reference source - internal
    Config_list.List_instance[1] :=37;
    Config_list.List_Value[1] :=1;
//2 Enable switch type - Inhibit only
    Config_list.List_instance[2] :=29;
    Config_list.List_Value[2] :=0;
//3 Enable Accel/Decel
    Config_list.List_instance[3] :=75;
    Config_list.List_Value[3] :=1;
//4 Set accel limit for velocity mode
    Config_list.List_instance[4] :=76;
    Config_list.List_Value[4] :=1000;
//5 Set decel limit for velocity mode
    Config_list.List_instance[5] :=77;
    Config_list.List_Value[5] :=1000;

//--Total number of messages = 6
    Config_list.ListLength:=6;

//reset state to IDLE (==0)
    Config_list.State:=0;

ELSE
```




```
CASE Config_list.State of

0:          //it is idle state , just return
  If Config_list.List_Start AND NOT Config_list.List_latch then
    Config_list.State:=1;
  END_IF;
  Config_list.List_latch:=Config_list.List_Start;

1:  //State Start
    Config_list.MsgIndex :=0;  //reset index
    Config_list.ListDone:=0;  //reset DONE flag
    Config_list.ListErr:=0;   //reset error flag
    MSG_Set940PID.Instance:=Config_list.List_instance[Config_list.MsgIndex];
    //this is normally should be done in state2
    Config_list.List_CurrentValue:=Config_list.List_Value[Config_list.MsgIndex];
//but redundant here so we don't have to wait for next loop
    MSG(MSG_Set940PID);      //send first message
    Config_list.State:=2;    //advance current state to TX

2:  //State TX
    if MSG_Set940PID.DN Then //message sent so advance index and check if it is not over yet
      Config_list.MsgIndex:=Config_list.MsgIndex+1; //advance index in the list
      if Config_list.MsgIndex <= (Config_list.ListLength-1) then
        MSG_Set940PID.Instance:=Config_list.List_instance[Config_list.MsgIndex];
        Config_list.List_CurrentValue:=Config_list.List_Value[Config_list.MsgIndex];
        MSG(MSG_Set940PID);
      else
        Config_list.ListDone:=1;
        Config_list.State:=0;
      end_if;

    elsif MSG_Set940PID.ER then
      Config_list.State:=0;
      Config_list.ListErr:=1;
    END_IF;
END_CASE;

END_IF;
```



Applications

8.4 Application Note - Detection of EtherNet/IP Exclusive Ownership Loss

The PositionServo provides bits in the extended status register to detect a loss of exclusive ownership over EtherNet/IP. The user can use these bits in a logical program to detect this condition and take action as necessary for their application, such as to prevent a runaway condition in velocity or torque mode. Here is an example of PositionServo indexer logic to perform this:

```
PROGRAM_START:
IF (!(VAR_EXSTATUS & (0x1000000))) ; checks if Exclusive ownership is lost
DISABLE ; in case of EIP exclusive ownership loss, disable drive
ENDIF

GOTO PROGRAM_START

END
```

Lenze AC Tech Corporation
630 Douglas Street • Uxbridge, MA 01569 • USA
Sales: 800 217-9100 • Service: 508 278-9100
www.lenzeamericas.com

P94ETH01D



MotionView[®]
OnBoard

PositionServo PROFIBUS-DP Communication Module Communications Interface Reference Guide

About These Instructions

This documentation applies to the optional PROFIBUS DP communications module for the PositionServo drive and should be used in conjunction with the PositionServo User Manual (Document S94PM01) that shipped with the drive. These documents should be read in their entirety as they contain important technical data and describe the installation and operation of the drive.

© 2008 AC Technology Corporation

No part of this documentation may be copied or made available to third parties without the explicit written approval of AC Technology Corporation. All information given in this documentation has been carefully selected and tested for compliance with the hardware and software described. Nevertheless, discrepancies cannot be ruled out. AC Tech does not accept any responsibility nor liability for damages that may occur. Any necessary corrections will be implemented in subsequent editions.

Contents



1	Safety Information	1
1.1	Warnings, Cautions and Notes	1
1.1.1	General.....	1
1.1.2	Application.....	1
1.1.3	Installation.....	1
1.1.4	Electrical Connection	2
1.1.5	Operation.....	2
2	Introduction	3
2.1	Fieldbus Overview	3
2.2	Module Specification	3
2.3	Module Identification Label	3
3	Installation.....	4
3.1	Mechanical Installation	4
3.2	PROFIBUS DP Connector.....	5
3.3	Electrical Installation.....	6
3.3.1	Cable Types.....	6
3.3.2	Network Limitations.....	6
3.3.3	Connections and Shielding.....	7
3.3.4	Network Termination	7
4	Commissioning.....	8
4.1	Overview	8
4.2	Configuring the Network Master	8
4.2.1	Master Support Files.....	8
4.2.2	PROFIBUS-DP Master Setup Procedure.....	8
4.3	Configuring the PositionServo PROFIBUS DP Module.....	9
4.3.1	Connecting	9
4.3.2	Connect to the Drive with MotionView OnBoard.....	9
4.3.3	Setting the Network Protocol.....	10
4.3.4	PROFIBUS-DP Node Settings.....	11
4.3.5	Node Address	12
4.3.6	Baud / Data Rate	12
4.3.7	Data Mapping	12
4.3.8	Re-Initialising.....	13
4.3.9	Non-Module Parameter Settings	13
5.	Cyclic Data Access	14
5.1	What is Cyclic Data?	14
5.2	Channel Data Sizes.....	14
5.3	Mapping Cyclic Data.....	15
5.3.1	Data IN (Din) Channels.....	15
5.3.2	Data OUT (Dout) Channels.....	16



Contents

6.	Acyclic Parameter Access	17
6.1	What is Acyclic Data?	17
6.2	Setting the Acyclic Mode.....	17
6.2.1	Acyclic Modes	17
6.2.2	Acyclic Mode 1	18
6.2.3	Acyclic Mode 2	18
6.3	Modes 1 & 2 – 8BAD Format	18
6.3.1	8BAD - Function Code (Byte 0).....	19
6.3.2	8BAD – Access Control and Status (Byte 1).....	19
6.3.3	8BAD – PID Index (Bytes 2 and 3)	20
6.3.4	8BAD – Data (Bytes 4 to 7)	20
6.4	Acyclic Parameter Access Examples	20
6.4.1	Example 1: Read Velocity Accel Limit.....	20
6.4.2	Example 2: Write to Velocity Accel Limit.....	21
7	Drive Control and Status	22
7.1	Overview	22
7.2	Control BITs.....	22
7.2.1	Software Enable/Disable	22
7.2.2	Drive Reset (Cold Boot)	22
7.2.3	Suspend Motion.....	22
7.2.4	Stop Motion	23
7.3	Status Word.....	23
7.3.1	Status Flags Register	23
7.3.2	Extended Status Bits	24
8	Advanced Features	25
8.1	Module Firmware.....	25
8.2	Node Address Lock.....	25
8.3	PROFIBUS Status	25
8.4	PROFIBUS DP Timeout Action.....	26
8.4.1	Module Timeout Action	26
8.4.2	Master Monitor Timeout Action	27
8.4.3	Data Exchange Timeout Action	27
8.5	Sync and Freeze	28
8.5.1	Sync and Freeze Overview	28
8.5.2	Sync and Freeze Status	28
9	Diagnostics.....	29
9.1	Faults	29
9.2	Troubleshooting.....	29
10	Parameter Quick Reference	30



1 Safety Information

1.1 Warnings, Cautions and Notes

1.1.1 General

Some parts of Lenze controllers (frequency inverters, servo inverters, DC controllers) can be live, moving and rotating. 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.

1.1.2 Application

Drive controllers are components 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 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 drive as directed) is only allowed when there is compliance to 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. In the case of radio interference, special measures may be necessary for drive controllers.

1.1.3 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! When installing the drive ensure optimal airflow by observing all clearance distances in the drive's user manual. Do not expose the drive to excessive: vibration, temperature, humidity, sunlight, dust, pollutants, corrosive chemicals or other hazardous environments.



Safety Information

1.1.4 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 in accordance with the appropriate regulations (e.g. cable cross-sections, fuses, PE connection). Additional information can be obtained from the regulatory documentation.

The regulatory 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.

1.1.5 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, do not touch the live components and power connection until the capacitors have discharged. Please observe the corresponding notes on the controller.
- Do not continuously cycle input power to the controller more than once every three minutes.
- Close all protective covers and doors during operation.



WARNING!

Network control permits automatic starting and stopping 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.

Table 1: Pictographs used in these instructions

Pictograph	Signal word	Meaning	Consequences 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 for persons	Death or injury
	STOP!	Possible damage to equipment	Damage to drive system or its surroundings
	NOTE	Useful tip: If observed, it will make using the drive easier	



2 Introduction

The following information is provided to explain how the PositionServo drive operates on a PROFIBUS network; it is not intended to explain how PROFIBUS itself works. Therefore, a working knowledge of PROFIBUS is assumed, as well as familiarity with the operation of the PositionServo drive.

2.1 Fieldbus Overview

The PROFIBUS DP fieldbus is an internationally accepted communications protocol designed for commercial and industrial installations of factory automation and motion control applications. High data transfer rates combined with its efficient data formatting, permit the coordination and control of multi-node applications.

2.2 Module Specification

- Auto detection of data rates
- Supported baudrates: 12Mbps, 6Mbps, 3Mbps, 1.5Mbps, 500kbps, 187.5kbps, 93.75kbps, 45.45kbps, 19.2kbps, 9.6kbps.
- Scalable amount of input and output process data channels (maximum of 12 in either direction).
- Parameter access data channel

2.3 Module Identification Label

Figure 1 illustrates the labels on the PositionServo PROFIBUS DP communications module. The PositionServo PROFIBUS DP module is identifiable by:

- One label affixed to the side of the module.
- The TYPE identifier in the center of the label: E94ZAPFB1.
- The port (interface) identifier, P24, on the right hand side of label.

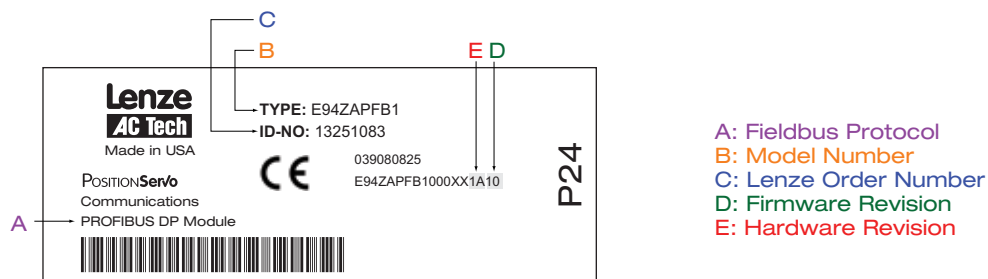


Figure 1: PositionServo PROFIBUS DP Module Label



Installation

3 Installation

3.1 Mechanical Installation

1. Ensure that for reasons of safety, the AC supply, DC supply and +24V DC backup supply have been disconnected before opening the bay cover plate.
2. Remove the two COMM module screws that secure Option Bay 1. With the aid of a flat head screw driver, gently pry up the Option Bay 1 cover plate and remove.
3. Fit the 6 way pin header into the module before fitting the module into the drive.
4. Install the PROFIBUS DP Module into the drive.
5. Replace the two COMM module screws that secure Option Bay 1. Using a phillips-head screwdriver, replace the GROUND screw as illustrated in Figure 2.

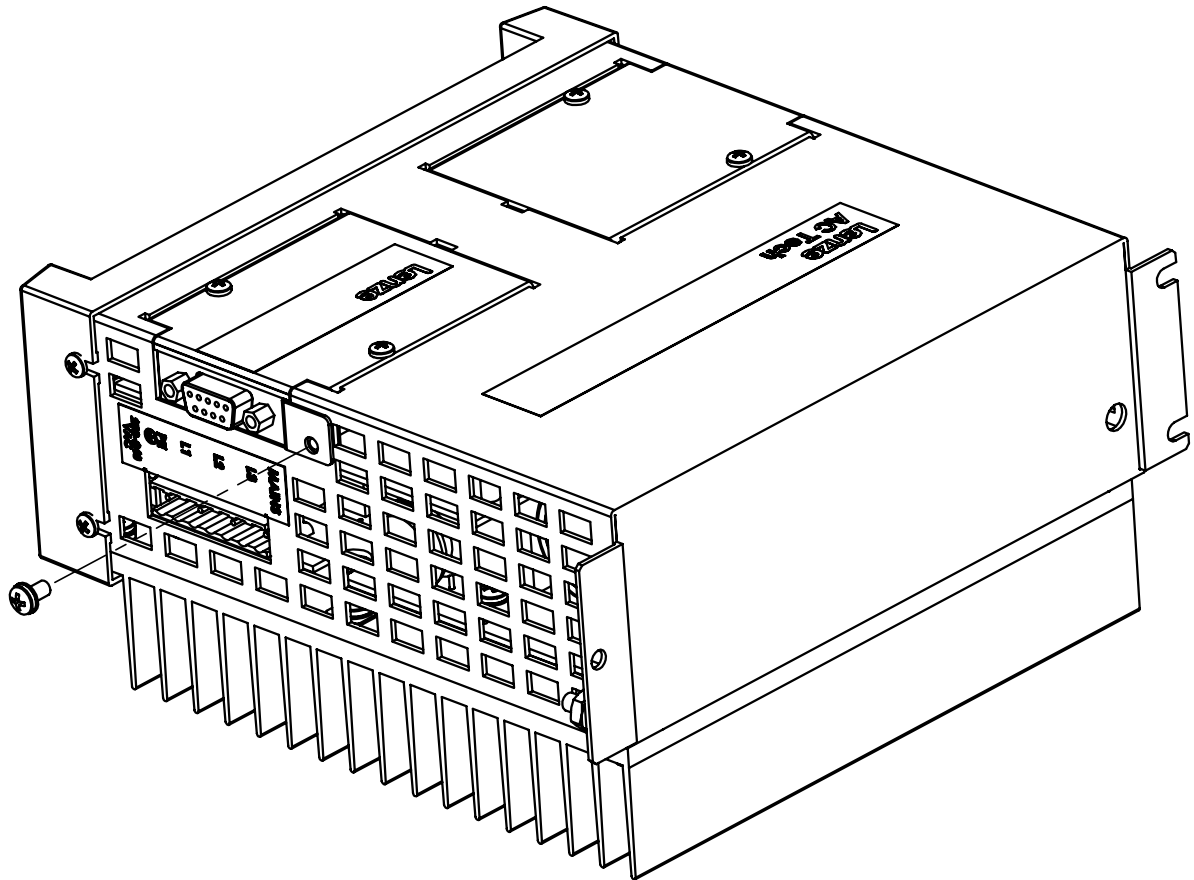


Figure 2: Installing the PROFIBUS-DP Communications Module



3.2 PROFIBUS DP Connector

Table 2 identifies the terminals and describes the function of each. Figure 3 illustrates the PROFIBUS DP DB-9 connector.

Table 2: PROFIBUS DP D-Type Connections

Pin Number	Function	Description
1	Shield	Cable Shield Connection
2	N/C	No Connection
3	RxD / TxD-P	Data Line B (Red)
4	N/C	No Connection
5	DGND	Data Ground
6	+5V	5V Output Supply
7	N/C	No Connection
8	RxD / TxD-N	Data Line A (Green)
9	N/C	No Connection

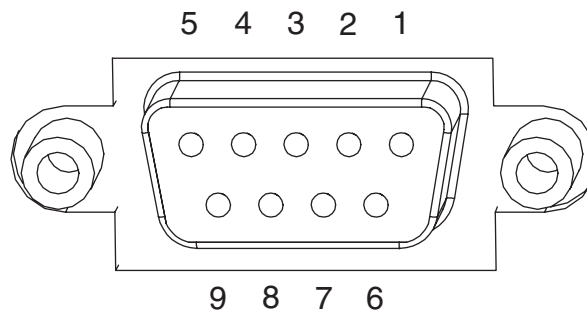


Figure 3: PROFIBUS DP Connector



NOTE

The PositionServo PROFIBUS DP module is equipped with a D sub-type connector. Always ensure that any connectors used on the network are fully approved for use with PROFIBUS DP. Some available connector types have built in termination that allows the network to be isolated, which can be very useful when fault finding. For data rates above 1.5Mbps, use a connector fitted with integrated inductors.



Installation

3.3 Electrical Installation

3.3.1 Cable Types

Due to the high data rates used on PROFIBUS DP networks it is paramount that correctly specified quality cable is used. The use of low quality cable will result in excess signal attenuation and data loss. Cable specifications and approved manufacturers are available from the official PROFIBUS website at: <http://www.profibus.com>

3.3.2 Network Limitations

There are several limiting factors that must be taken into consideration when designing a PROFIBUS DP network, for full details refer to the official “Installation Guidelines for PROFIBUS DP/FMS” which is available from <http://www.profibus.com>. However, here is a simple checklist:

- PROFIBUS DP networks are limited to a maximum of 125 nodes.
- Only 32 nodes may be connected on a single network segment.
- A network may be built up from one or several segments with the use of network repeaters.
- Maximum total network length is governed by the data rate used. Refer to Table 3.
- Minimum of 1 meter of cable between nodes.
- Use fiber optic segments to:
 - Extend networks beyond normal cable limitations.
 - Overcome different ground potential problems.
 - Overcome very high electromagnetic interference.
- Spurs or T connections are only acceptable by the PROFIBUS DP specification when operating at data rates of 1.5Mbps or less, however it is strongly advised not to use spurs as extreme care must be taken during the network design phase to avoid problems.

Table 3: Standard “Type A” Cable Network Length Specifications

Baud Rate	Maximum Segment Length	Recommended Maximum Total Network Length
9.6kbps	1200 meters	6000 meters
19.2kbps	1200 meters	6000 meters
45.45kbps	1200 meters	6000 meters
93.75kbps	1000 meters	5000 meters
187.5kbps	1000 meters	5000 meters
500kbps	400 meters	2000 meters
1.5Mbps	200 meters	1000 meters
3Mbps	100 meters	500 meters
6Mbps	100 meters	500 meters
12Mbps	100 meters	500 meters



NOTE

The recommended maximum network length is achievable with the use of repeaters. Due to signal propagation delay within the repeaters it is recommended that no more than 4 repeaters be used between any two network nodes



3.3.3 Connections and Shielding

The PositionServo PROFIBUS DP module is equipped with a D sub-type connector.

Always ensure that any connectors used on the network are fully approved for use with PROFIBUS DP. Some available connector types have built in termination that allows the network to be isolated, which can be very useful when finding. For data rates above 1.5Mbps, use a connector fitted with integrated inductors.

3.3.4 Network Termination

In high speed fieldbus networks such as PROFIBUS DP it is essential to install the specified termination resistors, i.e. one at both ends of a network segment. Failure to do so will result in signals being reflected back along the cable which will cause data corruption.

PROFIBUS-DP uses active (powered) termination. Therefore it is strongly recommended that "stand alone" active termination units are used to maintain the integrity of the network. If the PositionServo is used to provide network termination, in the event of a power loss to the drive, network termination will also be lost.

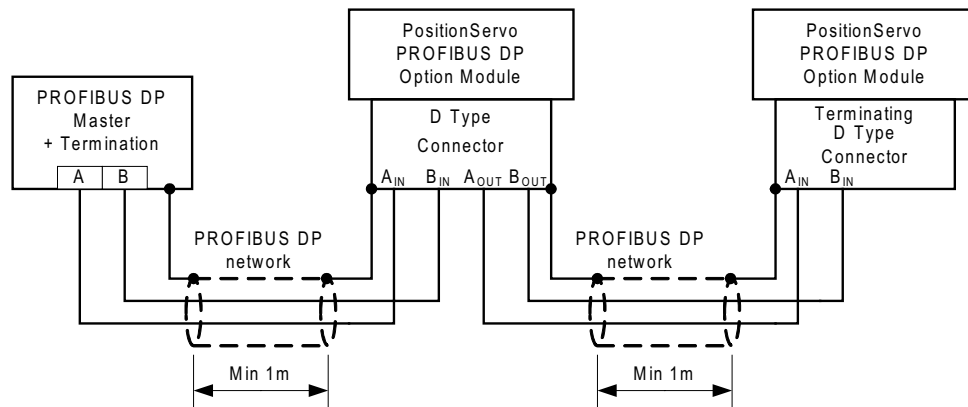


Figure 4a: Network without Active Termination

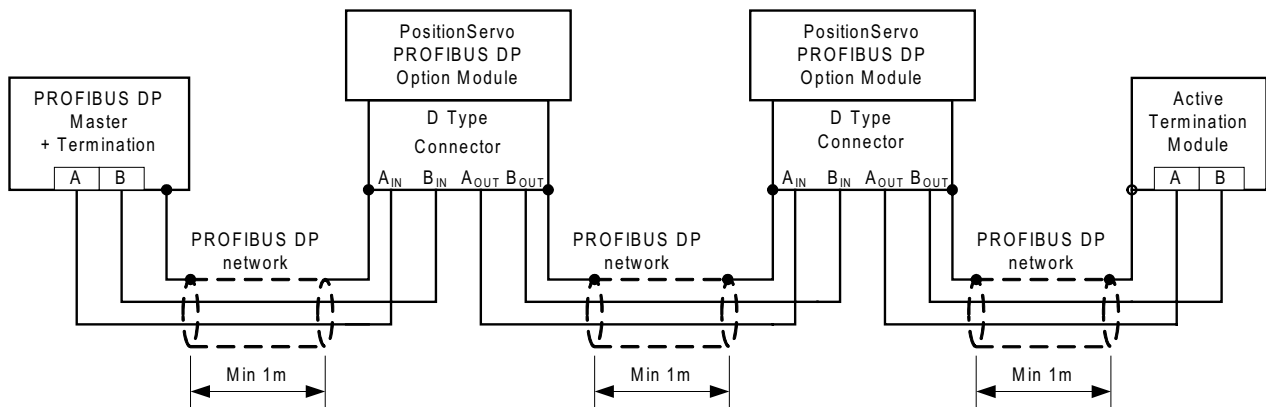


Figure 4b: Network with Active Termination



Commissioning

4 Commissioning

4.1 Overview

It is assumed that the user has familiarised themselves with how to set parameters using MotionView software. Refer to the PositionServo with MVOB User Manual (S94PM01) for more details.

The details that follow provide a step-by-step guide to quickly and easily set-up a PositionServo drive to communicate on a PROFIBUS DP fieldbus network, in a basic format. There are many more features and settings available for the PROFIBUS DP option module, for details on these refer to the fuller description in the sections that follow.

4.2 Configuring the Network Master

4.2.1 Master Support Files

Most PROFIBUS-DP master configuration software utilises GSD files to configure the network profile and communications with the relevant devices. GSD files are text files that contain information about the device timings, features supported and available data formats for the PROFIBUS-DP device. Device icon files are also supplied for use with the PROFIBUS-DP configuration software.



NOTE

Many manufacturers offer language-specific GSD files for their PROFIBUS-DP devices. In this case the term and file suffix “GSD” is used for their primary/default language choice and additional files may be available for alternative languages and will be named differently. For example, for manufacturers where English is not the primary language it may be possible to obtain GSD and GSE files where the GSD file is written in the native/home language and the GSE file will be written in English etc.

The PositionServo GSD files are available on the CD ROM that ships with the drive and on the Lenze-AC Tech website.

4.2.2 PROFIBUS-DP Master Setup Procedure

The method for configuring master devices differs greatly between manufacturers. Provided herein is a very basic, generic guide to setting up a network master.

1. Launch the Master configuration software.
2. Install/Import the required GSD support file(s) using the wizard tool if provided.
3. Setup master PROFIBUS DP port with required criteria such as node address and baudrate etc.
4. Add or “drag and drop” the required slave devices from the GSD library to the PROFIBUS DP network which is typically depicted on screen.
5. Configure the slave node address, ensuring that each node has a unique and individual address.
6. Configure each slave's I/O data size. (This is typically done by dragging and dropping the required amount of modules from the GSD file library or picking the modules from a list).

NOTE: Although there are only 4 modules listed in the GSD file, these can be used several times to create the required amount of data.

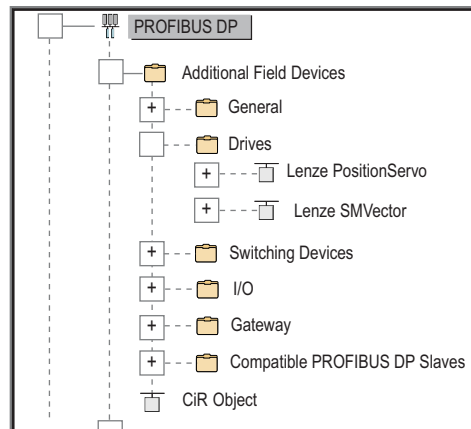


Figure 5: PROFIBUS DP Master Setup

7. Save the configuration and download to the master.

4.3 Configuring the PositionServo PROFIBUS DP Module

4.3.1 Connecting

With the drive power disconnected, install the PROFIBUS DP module and connect the network cable as instructed in the preceding sections. Ensure the drive Run/Enable terminal is disabled then apply the correct voltage to the drive (refer to drive's user manual for voltage supply details).

4.3.2 Connect to the Drive with MotionView OnBoard

Refer to the PositionServo User Manual, section 6.2 for full details on configuring and connecting a drive via MotionView OnBoard (MVOB) software. Contained herein is a brief description of launching MVOB and communicating with the drive.

1. Open the PC's web browser. Enter the drive's default IP address [192.168.124.120] in the browser's Address window.
2. The authentication screen may be displayed if the PC does not have Java RTE version 1.4 or higher. If so, to remedy this situation, download the latest Java RTE from <http://www.java.com>.
3. When MotionView has finished installing, a Java icon entitled [MotionView OnBoard] will appear on your desktop and the MVOB splash screen is displayed. Click [Run] to enter the MotionView program.
4. Once MotionView has launched, verify motor is safe to operate, click [YES, I have] then select [Connect] from the Main toolbar (top left). The Connection dialog box will appear.
5. Select [Discover] to find the drive(s) on the network available for connection.

[Discover] may fail to find the drive's IP address on a computer with both a wireless network card and a wired network card (or a PC with more than one network connection). If this happens, try one of the following remedies:

Disable the wireless network card and then use [Discover].

Type in the drive's IP address manually at the box [IP Address].

Then click [Connect]



Commissioning

6. Highlight the drive (or drives) to be connected and click [Connect] in the dialog box.

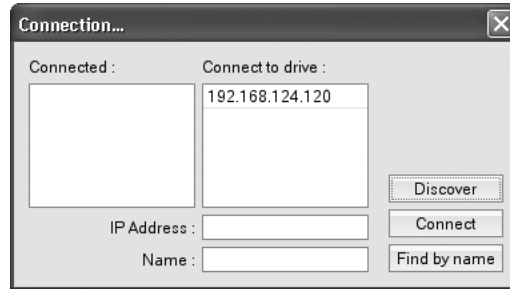


Figure 6: Connection Box with Discovered Drive

In the lower left of the MotionView display, the Message Window will contain the connection status message. The message “Successfully connected to drive B04402200450_192.168.124.120” indicates that the drive B04402200450 with IP address 192.168.124.120 is connected.

4.3.3 Setting the Network Protocol

In the left-hand node tree of MotionView OnBoard, click on the [Communications] folder. Using the drop down menu, select [PROFIBUS-DP] as the required fieldbus selection.

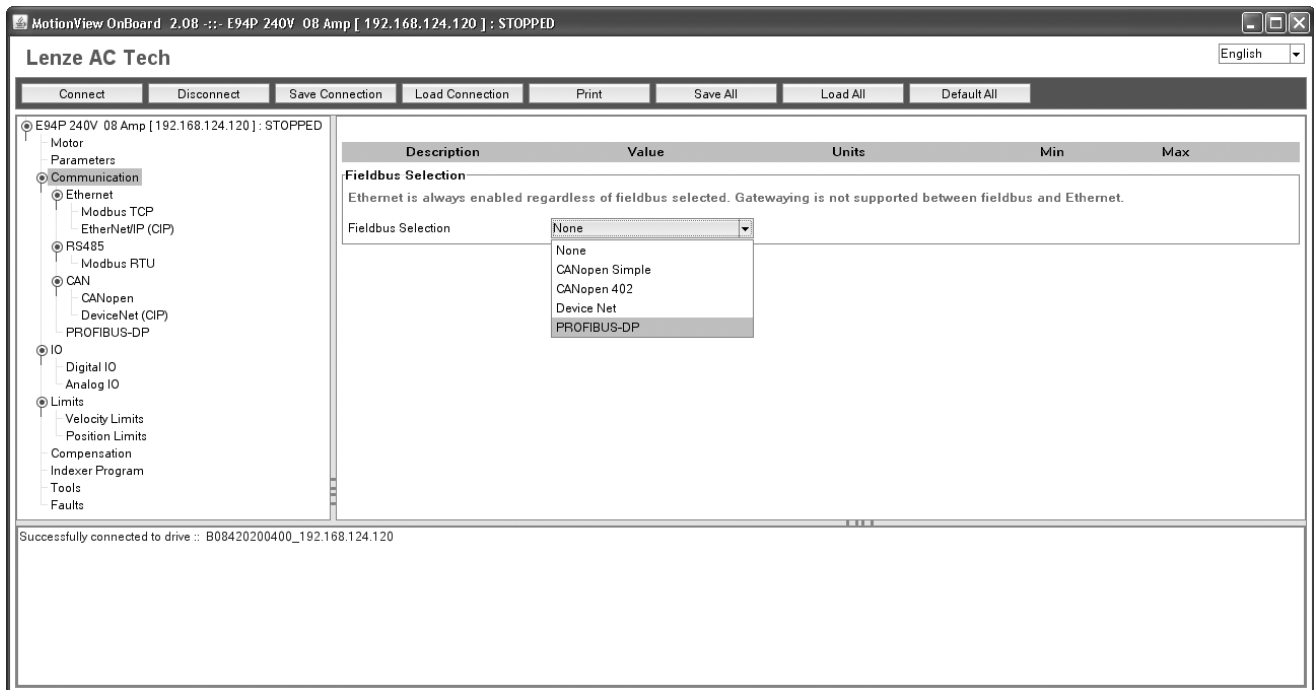


Figure 7: Fieldbus Selection

The Important Message box (to REBOOT) is displayed because the Communication setting has been changed (from None to PROFIBUS DP in this example). Click [OK] to dismiss the dialog box. Reboot the drive.

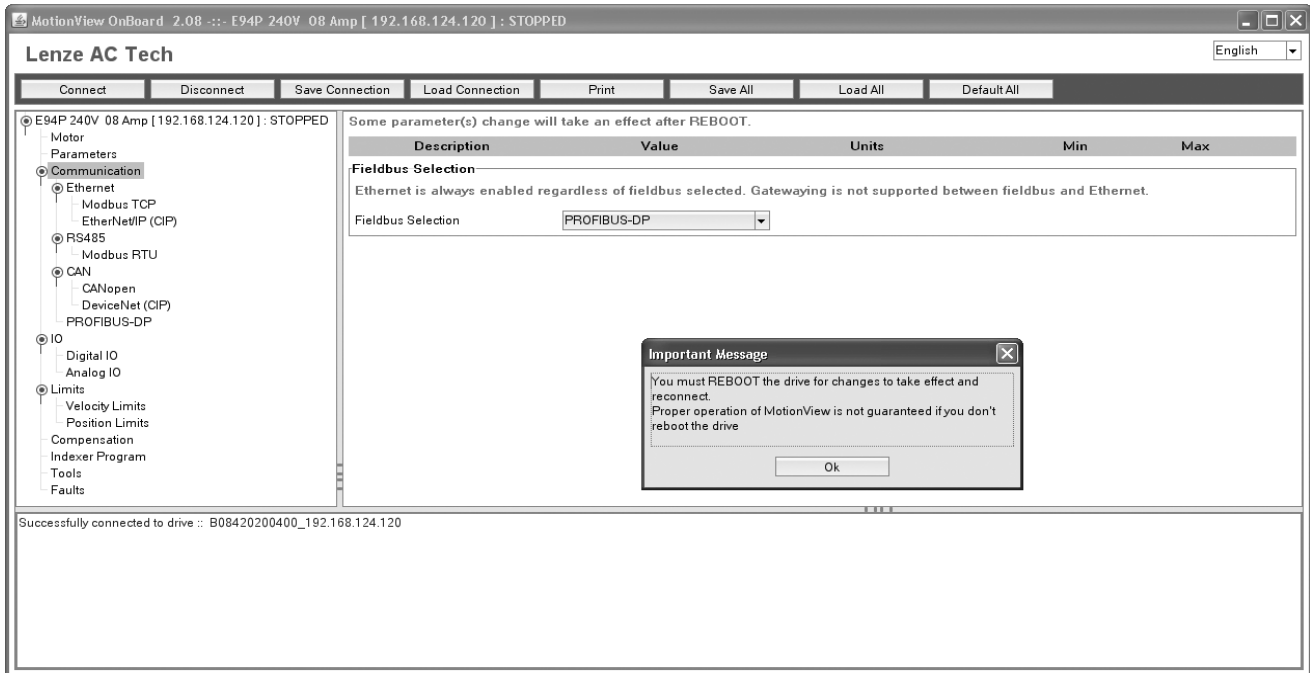


Figure 8: REBOOT Message

4.3.4 PROFIBUS-DP Node Settings

To access the PositionServo PROFIBUS-DP node settings, click on the [PROFIBUS-DP] folder icon.

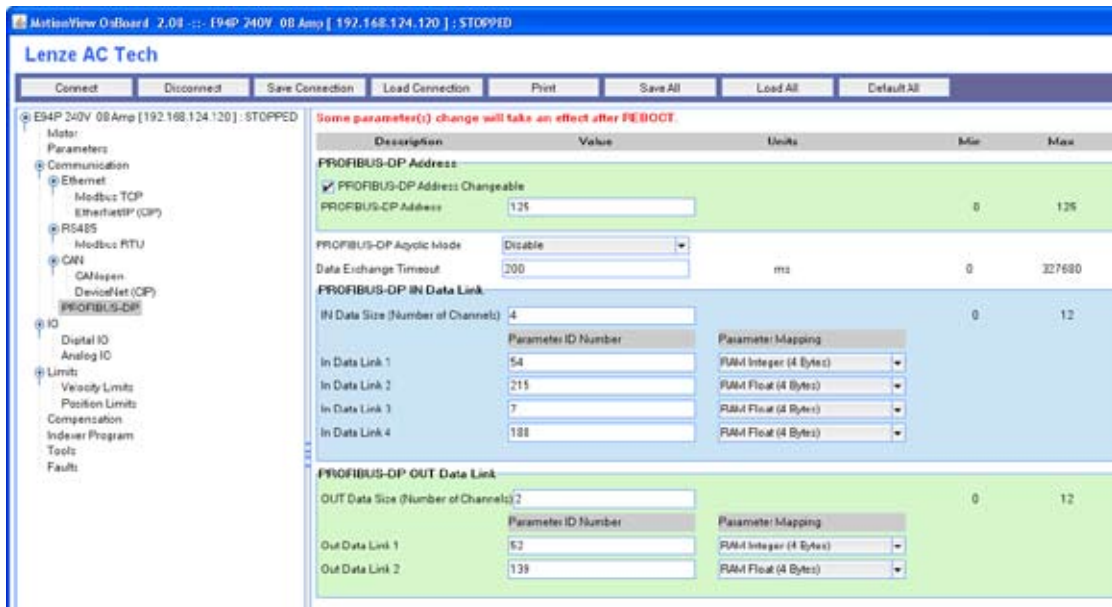


Figure 9: PROFIBUS DP Settings



Commissioning

4.3.5 Node Address

Description	Value
PROFIBUS-DP Address	
<input checked="" type="checkbox"/> PROFIBUS-DP Address Changeable	
PROFIBUS-DP Address	125

Figure 10: PROFIBUS DP Node Address

PID283 - Node Address			
Default:	126	Range:	0 - 126
Access:	RW	Type:	Integer

Set PID283 to the required value. The default address is 126. The permissible address range is: 0 – 125.

Each node on the network must have an individual address, if two or more nodes have duplicate addresses this may prevent the network from functioning correctly. Node 126 is a special node address intended for “New” nodes only where by node configuration is performed via a network master device.

4.3.6 Baud / Data Rate

The PositionServo PROFIBUS-DP module automatically detects and synchronises to the data rate of the network to which it has been connected.

4.3.7 Data Mapping

- The PositionServo PROFIBUS-DP module has support for up to 12 cyclic data channels in both directions.
- Cyclic data configuration is described in full in section 5.
- The default mapping for PositionServo PROFIBUS-DP is 4 Data IN links and 2 Data OUT links, the configuration is shown in Table 4.

Table 4: Default Mapped Cyclic Data

Data OUT Link	Mapped Function	Data Format		Data IN Link	Mapped Function	Data Format
1	52 – VAR_ENABLE	RAM Integer		1	54 – VAR_STATUS	RAM Integer
2	139 – VAR_IREF	RAM Float		2	215 – VAR_APOS	RAM Float
				3	7 – VAR_VELOCITY_ACTUAL	RAM Float
				4	188 – VAR_PHCUR	RAM Float

NOTE

- The data size of each IN and OUT Link / Channel is 4 Bytes per link.
- The terms “OUT data” and “IN data” describe the direction of data transfer as seen by the PROFIBUS-DP network master controller.



4.3.8 Re-Initialising

To activate any changes made the drive has to be reinitialized. Hence the warning within MotionView

Some parameter(s) change will take an effect after REBOOT.

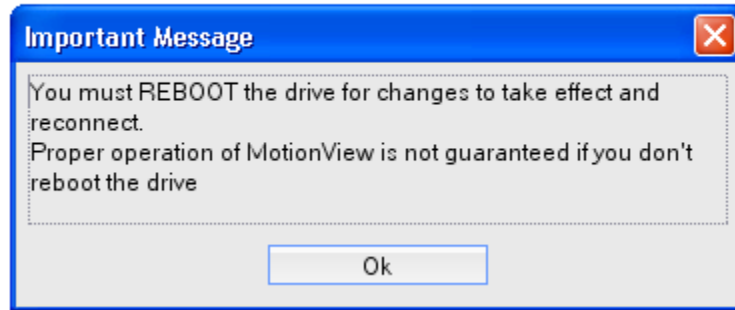


Figure 11: REBOOT Message

This can be done by cycling the power to the drive.

4.3.9 Non-Module Parameter Settings

In addition to configuring the PROFIBUS-DP option module and depending upon the application there may be several drive based parameters that will need to be set using MotionView or an Indexer program or via the PROFIBUS parameter access channel. Such as:

- PID34 – Drive Mode (VAR_DRIVEMODE)
- PID37 – Reference (VAR_REFERENCE)
- PID29 – Enable switch funtion (VAR_ENABLE_SWITCH_TYPE)



Cyclic Data Access

5. Cyclic Data Access

5.1 What is Cyclic Data?

- Cyclic / Process / Polled data is the name given to the method used to transfer routine process data between the network master and slave nodes.
- Cyclic data transfer must be configured during network setup.
- The terms “OUT data” and “IN data” describe the direction of data transfer as seen by the PROFIBUS DP network master controller.
- The cyclic data source and destinations are configured and controlled by the PositionServo PROFIBUS DP module's mapping capabilities.

5.2 Channel Data Sizes

- During network setup, it is necessary to program the network master with the amount of IN and OUT cyclic data used for each slave device that it is associated with. This process is simplified with the use of GSD support files (refer to paragraph 4.2.2, PROFIBUS-DP Master Setup Procedure, for details).
- The amount of cyclic data configured in each PositionServo PROFIBUS-DP module must be equal to the amount configured in the network master. Failure to do this may result in lost data and/or network master configuration errors.

The image shows two configuration panels. The top panel, titled 'PROFIBUS-DP IN Data Link', has a light blue background and contains a text input field for 'IN Data Size (Number of Channels)' with the value '4'. The bottom panel, titled 'PROFIBUS-DP OUT Data Link', has a light green background and contains a text input field for 'OUT Data Size (Number of Channels)' with the value '2'.

Figure 12: Set Channel Data Size

- Each cyclic channels utilises 4 Bytes of data.
- The IN and OUT data sizes can also be set by using PID285 and PID284 respectively.

PID285 - IN Data Size			
Default:	4	Range:	0 - 12
Access:	RW	Type:	Integer

PID284 - OUT Data Size			
Default:	2	Range:	0 - 12
Access:	RW	Type:	Integer



5.3 Mapping Cyclic Data

5.3.1 Data IN (Din) Channels

- The PROFIBUS-DP module has 12 cyclic IN channels each of which utilises 4 Bytes of data.
- The amount of IN channels activated and mappable is set by PID285.
- IN data mapping can be set via the MVOB [Communications] [PROFIBUS DP] folder:

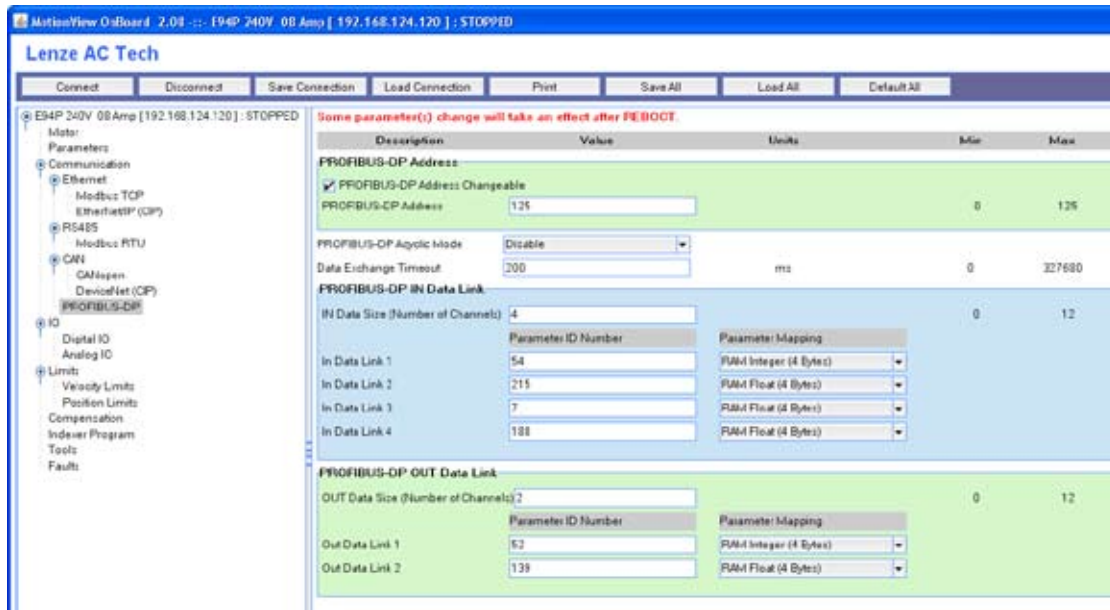


Figure 13: PROFIBUS DP Communications folder

- Or PIDs 298 - 309 can be used to directly edit the IN mapping details.

PID298 to PID309 - Din Mapping Channels			
Default:	Various	Range:	0 - 999
Access:	RW	Type:	Integer

Table 5 lists the default mapping source data for IN data being sent from the drive to the network master.

Table 5 – IN Data (Din) Mappings

PID	Din Channel	Default	Format	Function
PID298	Channel 1 mapping	54	Integer	VAR_STATUS
PID299	Channel 2 mapping	215	Float	VAR_APOS
PID300	Channel 3 mapping	7	Float	VAR_VELOCITY_ACTUAL
PID301	Channel 4 mapping	188	Float	VAR_PHCUR
PID302	Channel 5 mapping	116	Integer	VAR_V16
PID303	Channel 6 mapping	117	Float	VAR_V17
PID304	Channel 7 mapping	118	Integer	VAR_V18
PID305	Channel 8 mapping	119	Float	VAR_V19
PID306	Channel 9 mapping	120	Integer	VAR_V20
PID307	Channel 10 mapping	121	Float	VAR_V21
PID308	Channel 11 mapping	122	Integer	VAR_V22
PID309	Channel 12 mapping	123	Float	VAR_V23



Cyclic Data Access

5.3.2 Data OUT (Dout) Channels

- The PROFIBUS-DP module has 12 cyclic OUT channels each of which utilises 4 Bytes of data.
- The amount of OUT channels activated and mappable is set by PID284.
- OUT data mapping can be set via the MVOB [Communications] [PROFIBUS DP] folder:

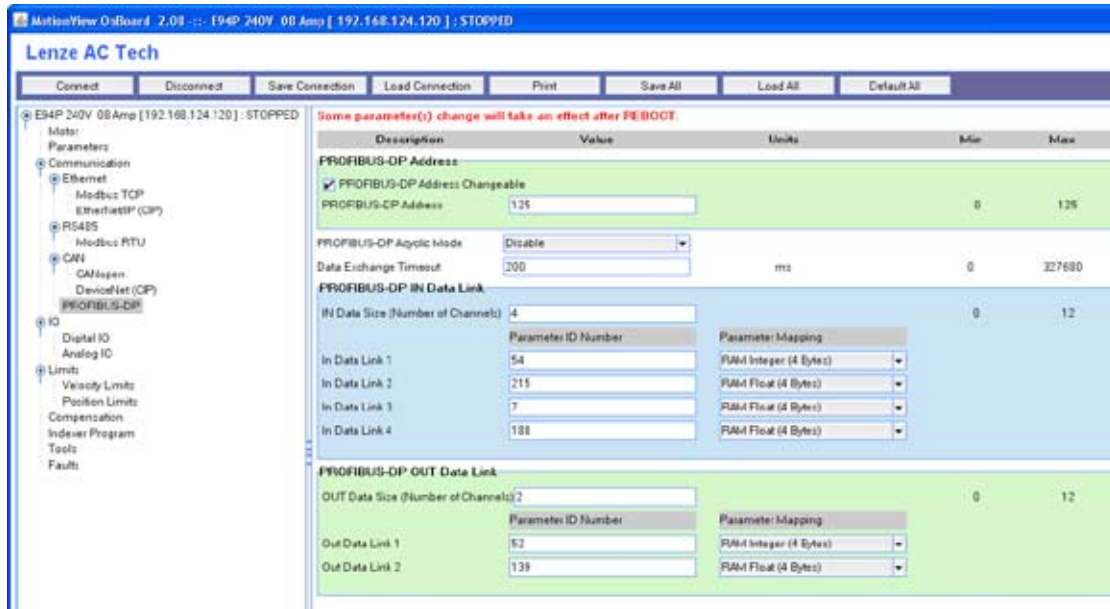


Figure 14: PROFIBUS DP Communications folder

- Or PIDs 286 - 297 can be used to directly edit the OUT mapping details.

PID286 to PID297 - Dout Mapping Channels			
Default:	Various	Range:	0 - 999
Access:	RW	Type:	Integer

- Table 6 lists the default mapping destinations for OUT going data being sent from the network master.

Table 6 – OUT Data (Dout) Mappings

PID	Dout Channel	Default	Format	Function
PID286	Channel 1 mapping	52	Integer	VAR_ENABLE
PID287	Channel 2 mapping	139	Float	VAR_IREF
PID288	Channel 3 mapping	102	Integer	VAR_V2
PID289	Channel 4 mapping	103	Float	VAR_V3
PID290	Channel 5 mapping	104	Integer	VAR_V4
PID291	Channel 6 mapping	105	Float	VAR_V5
PID292	Channel 7 mapping	106	Integer	VAR_V6
PID293	Channel 8 mapping	107	Float	VAR_V7
PID294	Channel 9 mapping	108	Integer	VAR_V8
PID295	Channel 10 mapping	109	Float	VAR_V9
PID296	Channel 11 mapping	110	Integer	VAR_V10
PID297	Channel 12 mapping	111	Float	VAR_V11

Acyclic Parameter Access



6. Acyclic Parameter Access

6.1 What is Acyclic Data?

- Acyclic / non-cyclic / Service access provides a method for the network master to access any drive or module parameter.
- This kind of parameter access is typically used for monitoring or low priority non-scheduled parameter access.
- The PositionServo PROFIBUS-DP module supports several different methods of doing this.

6.2 Setting the Acyclic Mode

6.2.1 Acyclic Modes

The Acyclic mode can be set in the MVOB [Communications] [PROFIBUS DP] folder by clicking on the down arrow [▼] next to [PROFIBUS DP Acyclic Mode] and selecting the mode from the pull-down menu.

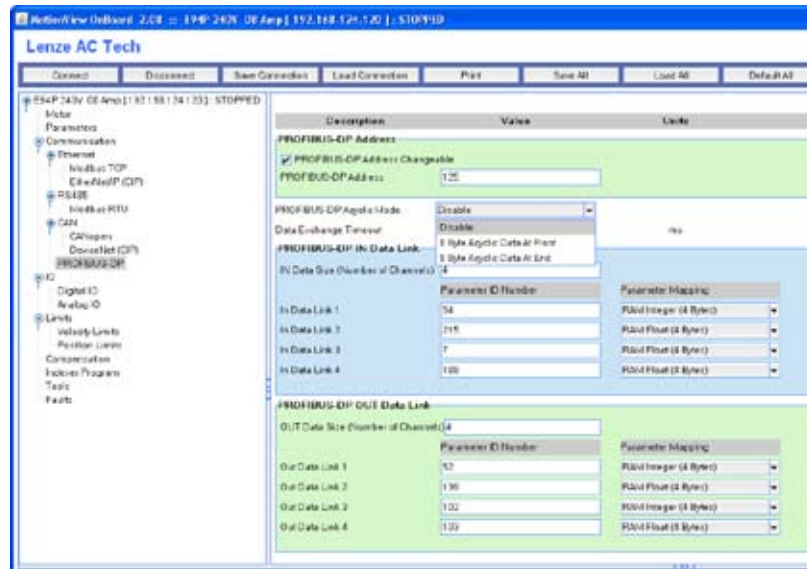


Figure 15: Select PROFIBUS Acyclic Mode

Or PID310 can be used to select the required Acyclic mode. Refer to section 6.3 for details on type of acyclic mode. The acronym “8BAD” indicates “8 Byte Acyclic Data”.

PID310 - Acyclic Mode			
Default:	0	Range:	0 - 2
Access:	RW	Type:	Integer

Table 7: Acyclic Modes

PID310	Acyclic Mode	Description
0	Disabled	No acyclic parameter access
1	8BAD-F	8 Byte Acyclic Data at Front
2	8BAD-E	8 Byte Acyclic Data at End



Acyclic Parameter Access

6.2.2 Acyclic Mode 1

PID310 = 1 (Mode 1 – 8BAD-F)

Setting this mode configures the PROFIBUS-DP module to expect 8 additional cyclic bytes at the FRONT of all normal process cyclic data.

6.2.3 Acyclic Mode 2

PID310 = 2 (Mode 2 – 8BAD-E)

Setting this mode configures the PROFIBUS-DP module to expect 8 additional cyclic bytes at the END of all normal process cyclic data.



NOTE

Enabling an 8BAD mode adds to the total amount of IN and OUT cyclic data and is reflected in the total channel data size. Care should also be taken in selecting the correct module from the GSD file when configuring the network master. Changes made to PID310 will only take effect after re-initialising the drive.

6.3 Modes 1 & 2 – 8BAD Format

The 8BAD format of acyclic parameter data access is a simple method that utilises 8 bytes of cyclic data which can be placed either before the regular cyclic data or after depending on the user's preference or application requirements. 8BAD is comprised of 8 bytes of data.

Table 8: 8BAD Format

Byte	Description
0	Function Code
1	Access Control & Status
2	MSB
3	LSB
4	MSB
5	
6	
7	LSB

Acyclic Parameter Access



6.3.1 8BAD - Function Code (Byte 0)

Table 9 lists the Function Code, Byte 0, of the 8BAD format.

Table 9: Function Code

Byte	Bit	Description
0	0	0 – Idle
	1	1 – Read RAM integer
		2 – Read RAM float
	2	3 – Read EPM integer
		4 – Read EPM float
	3	5 – Write RAM integer
		6 – Write RAM float
	4	7 – Write EPM integer
8 – Write EPM float.		
7	0 - No Fault	
	1 - Fault, Access Failure. Refer to Access & Control Status	

6.3.2 8BAD – Access Control and Status (Byte 1)

The purpose of the Access Control and Status Byte is to provide transfer control and diagnostic information. The Status bits provide diagnostics on the message currently being processed.

Table 10: Access Control & Status

Byte	Bit	Description
1	0	0 = No fault, Write ACK
		1 = Invalid function
		2 = Parameter does not exist
		3 = Read only parameter
	1	4 = Value not in range
		5 = Access failure
	2	6 = Write operation failure
		15 = ACT unknown exception
4	1 = Valid response to the request message - bit set by module to indicate the data in message is valid or acknowledgment for write access.	
	0 = if bit 7 of Byte 0 is set and the exception number is higher than 0	
	5	1 = Module is processing the master's request. Any data being sent to the master at this time is invalid 0 = valid reply
	6	reserved
7	Toggle bit. (Handshake) Master toggles this bit to indicate a new message. The old command (if not finished) is cancelled.	



NOTE

- Bits 0 to 6 are set by the module. Bit 7 is set by the master, it is initialized to be zero. The module matches the state of bit 7 in its response message.
- Bit 7 of the Access Control & Status byte will cause the message to be executed when it changes its state. Each time this bit changes state it indicates a new request is being made. This bit must be set by the Master/PLC once all other bytes have been set in the acyclic data portion of the message. Otherwise a partially assembled message will be processed by the drive causing an unexpected result.
- The drive will copy the state of Bit 7 from the message sent by the master to Bit 7 in the response.



Acyclic Parameter Access

6.3.3 8BAD – PID Index (Bytes 2 and 3)

This is the drive parameter index number to be Read or Written to from the master. For the reply message from the drive this will contain the drive parameter index number that message corresponds to.

6.3.4 8BAD – Data (Bytes 4 to 7)

The actual PID data is present in these 4 bytes. (DWORD)

- On read command bit 5 of the Access Control & Status byte indicates whether the data is valid or not.
- On write command bit 5 of the Access Control & Status byte indicates whether the write operation is completed or not.

6.4 Acyclic Parameter Access Examples

Only the acyclic parameter information is configured for the these examples.

6.4.1 Example 1: Read Velocity Accel Limit

Example 1: Read Velocity Accel Limit, PID76 / VAR_ACCEL_LIMIT(= 1000, default value). This first example provides a valid send/receive transmission and an invalid send/receive transmission for a read operation from PID76.

Valid Transmission:

SEND: message consisting of:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x01	0x00 or 0x80	0x00	0x4C	0x00	0x00	0x00	0x00
Read	Toggle	PID 76		Data			

RECEIVE: response consisting of:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x01	0x10 or 0x90	0x00	0x4C	0x00	0x00	0x03	0xE8
Read	Valid response	PID 76		Data = 1000			

Invalid Transmission:

SEND: message consisting of non-existing PID:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x01	0x00 or 0x80	0x03	0x08	0x00	0x00	0x00	0x00
Read	Toggle	PID 776		Data			

RECEIVE: response consisting of:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x01	0x12 or 0x92	0x00	0x4C	0x00	0x00	0x00	0x00
Read	Valid response, PID does not exist	PID 776		Data = 0			

Acyclic Parameter Access



6.4.2 Example 2: Write to Velocity Accel Limit

Example 2: Write to Velocity Accel Limit, PID76 / VAR_ACCEL_LIMIT with a value of 1500. This second example provides a valid send/receive transmission and an invalid send/receive transmission for a write operation to PID76.

Valid Transmission:

SEND: message consisting of:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x05	0x00 or 0x80	0x00	0x4C	0x00	0x00	0x00	0x00
Write	Toggle	PID 76		Data = 1500			

RECEIVE: response consisting of:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x05	0x10 or 0x90	0x00	0x4C	0x00	0x00	0x05	0xDC
Write	Valid response	PID 76		Data = 1500			

Invalid Transmission:

SEND: message trying to write to a read-only PID:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x01	0x00 or 0x80	0x03	0x4A	0x00	0x00	0x05	0xDC
Write	Toggle	PID 74		Data			

RECEIVE: response consisting of:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x01	0x14 or 0x94	0x00	0x4A	0x00	0x00	0x05	0xDC
Write	Valid response, Read-only parameter	PID 74		Data = 1500			



Drive Control and Status

7 Drive Control and Status

7.1 Overview

The control and status words provide a means for the digital control and monitoring of the drive using a single data word. Each control bit has a particular function and provides a method of controlling the output functions of the drive, such as run and direction. Each bit in the status word provides feedback about the drive's state of health and operational condition.

7.2 Control BITS

There are several control bit available within PositionServo that can be written to through cyclic or acyclic communications. Some of the most commonly used ones are listed as follows, for a complete list of drive control functions see the Programming Manual:-

7.2.1 Software Enable/Disable

PID52 - Enable			
Default:	N/A	Range:	0 - 1
Access:	WO	Type:	Integer

This is the VAR_ENABLE function.

0 - disable

1 – enable

This function is the default mapping for cyclic Data Out Channel 1.

7.2.2 Drive Reset (Cold Boot)

PID53 - Reset			
Default:	N/A	Range:	0 - 1
Access:	WO	Type:	Integer

This is the VAR_RESET function.

0 - no action

1 - reset drive

7.2.3 Suspend Motion

PID91 - Suspend Motion			
Default:	0	Range:	0 - 1
Access:	RW	Type:	Integer

This is the VAR_SUSPEND_MOTION function.

0 - motion enabled

1 - motion disabled



7.2.4 Stop Motion

PID136 - Stop Motion			
Default:	N/A	Range:	0 - 1
Access:	WO	Type:	Integer

This is the VAR_STOP_MOTION function.

0 - no action

1 - stops motion

7.3 Status Word

There are several status words and individual status bits / flags available within PositionServo that can be read from through cyclic or acyclic communications

7.3.1 Status Flags Register

PID54 - DSTATUS			
Default:	N/A	Range:	
Access:	RO	Type:	Integer

This is the VAR_DSTATUS function.

Table 11: DSTATUS Register

Bit in register	Description
0	Set when drive enabled
1	Set if DSP subsystem at any fault
2	Set if drive has a valid program
3	Set if byte-code or system or DSP at any fault
4	Set if drive has a valid source code
5	Set if motion completed and target position is within specified limits
6	Set when scope is triggered and data collected
7	Set if motion stack is full
8	Set if motion stack is empty
9	Set if byte-code halted
10	Set if byte-code is running
11	Set if byte-code is set to run in step mode
12	Set if byte-code is reached the end of program
13	Set if current limit is reached
14	Set if byte-code at fault
15	Set if no valid motor selected
16	Set if byte-code at arithmetic fault
17	Set if byte-code at user fault
18	Set if DSP initialization completed
19	Set if registration has been triggered
20	Set if registration variable was updated from DSP after last trigger
21	Set if motion module at fault
22	Set if motion suspended
23	Set if program requested to suspend motion



Drive Control and Status

Bit in register	Description
24	Set if system waits completion of motion
25	Set if motion command completed and motion Queue is empty
26	Set if byte-code task requested reset
27	If set interface control is disabled. This flag is set/clear by ICONTROL ON/OFF statement.
28	Set if positive limit switch reached
29	Set if negative limit switch reached
30	Events disabled. All events disabled when this flag is set. After executing EVENTS ON all events previously enabled by EVENT EventName ON statements become enabled again

7.3.2 Extended Status Bits

PID84 - DEXSTATUS			
Default:	N/A	Range:	
Access:	RO	Type:	Integer

This is the VAR_EXSTATUS function

Table 12: Encoding for Extended Status Bits

Bit #	Function	Comment
0	Reserved	
1	Velocity in specified window	Velocity in limits as per parameter #59: VAR_VLIMIT_SPEEDWND
2-4	Reserved	
5	Velocity at 0 (zero)	Velocity 0: Zero defined by parameter #58: VAR_VLIMIT_ZEROSPEED
6,7	Reserved	
8	Bus voltage below under-voltage limit	Utilized to indicate drive is operating from +24V keep alive and a valid DC bus voltage level is not present.
9,10	Reserved	
11	Regen circuit is on	Drive regeneration circuit is active. Drive will be dissipating power through the braking resistor (if fitted).
12-20	Reserved	
21	Set if homing operation in progress	Drive executing Pre-defined homing function (refer to section 2.15, PS Programming Manual, PM94M01).
22	Set if system homed	Drive completed Pre-defined homing function (refer to section 2.15, PS Programming Manual, PM94M01).
23	If set then last fault will remain on the display until re-enabled.	User can set this bit to retain fault code on the display until re-enabled. It is useful if there is a fault handler routine. When the fault handler is exited, the fault number on the display will be replaced by current status (usually DiS if bit #24 is not set). Setting bit #24 retains diagnostics on the display.
24	Set if EIP IO exclusive owner connection is established. Cleared if closed.	Checks if drive is controlled by EthernetIP master. Use bit #25 and bit #26 to process "lost of connection" condition (if needed) in the user's program
25	Set if EIP IO exclusive owner connection times out. Cleared if exc. owner conn exists.	Checks if connection with Ethernet/IP master is lost. Use bit #26 and bit #25 to process "lost of connection" condition (if needed) in the user's program
26-30	Reserved	



8 Advanced Features

8.1 Module Firmware

PID412 - Module Firmware			
Default:	N/A	Range:	0 - 0xFFFFFFFF
Access:	RO	Type:	Integer

Displays the module firmware revision as a hexadecimal number that is divided into two bytes. Example: 0x104 = 0x01, 0x04 = version 1.04

8.2 Node Address Lock

Some PROFIBUS-DP masters have the capability to set the node address remotely. While this can be a useful feature during commissioning and or network fault recovery, it is not always desirable.

Enabling the Node Address Lock will prevent the accidental changing of the node address by preventing the master from writing to it.



Figure 16: Node Address Lock

8.3 PROFIBUS Status

PID408 - Node Status Word			
Default:	N/A	Range:	0 - 10
Access:	RO	Type:	Integer

This is the VAR_PBUS_STATUS function.

Table 13: Node Status Word

Bit	Function	Description
0	Module Present	0 - Module not detected 1 - Module detected
1	Master Monitor Timeout	0 - No timeout 1 - Timeout occurred
2	Data Exchange Timeout	0 - No timeout 1 - Timeout occurred
3	Reserved	
4	Reserved	
5	Reserved	
6	Reserved	
7	Reserved	



Advanced Features

Bit	Function	Description
8	Reserved	
9	Clear Out Data	Sync and Freeze Status Refer to section 8.5 for details.
10	Unfreeze	
11	Freeze	
12	Unsync	
13	Sync	
14	Reserved	
15	Reserved	
16 - 31	Reserved	

8.4 PROFIBUS DP Timeout Action

The Module Timeout, Master Monitor Timeout and Data Exchange Timeout settings are used to configure the drive's response when a network or module error occurs. All 3 timeout functions can be set from one single parameter, alternatively they can be set individually and easily from within the MotionView PROFIBUS configuration folder.

PID413 - Timeout Actions			
Default:	N/A	Range:	0 - 0xFFFFF
Access:	RO	Type:	Integer

This is the VAR_TIMEOUT_ACTION_CFG function.

Table 14: Timeout Configuration

BITs	Function	Description
0	Module Timeout	Used to activate the module – drive timeout action
1	Reserved	
2	Master Monitor Timeout	Used to activate the master monitoring timeout action
3	Reserved	
4	Data Exchange Timeout	Used to activate the data exchange timeout action
5-31	Reserved	

8.4.1 Module Timeout Action

This parameter controls the action to be taken in the event of a Module-to-Drive time out. If enabled and a timeout occurs the drive will trip with fault “F_46”.

PROFIBUS DP Timeout Action	
Module Timeout Action	No Action ▼
Master Monitor Timeout Action	No Action
Data Exchange Timeout Action	Fault



8.4.2 Master Monitor Timeout Action

PROFIBUS DP Timeout Action	
Module Timeout Action	No Action ▼
Master Monitor Timeout Action	No Action ▼
Data Exchange Timeout Action	No Action
	Fault

This parameter controls the action to be taken in the event of a Master Monitoring Timeout. The timeout period is set by the network master during the parameterization phase. If enabled and a timeout occurs the drive will trip with fault “F_47”.

PID409 - Master Timeout Value			
Default:	N/A	Range:	0 - 65535
Access:	RO	Type:	Integer

This is the VAR_PBUS_MASTER_TIMEOUT_VAL function.

This parameter displays the Monitoring / Watchdog Time (in seconds) set by the network master during the parameterization phase.

8.4.3 Data Exchange Timeout Action

Data Exchange Timeout	200
PROFIBUS DP Timeout Action	
Module Timeout Action	No Action ▼
Master Monitor Timeout Action	No Action ▼
Data Exchange Timeout Action	No Action ▼
	No Action
	Fault

Data Exchange Time-out provides an independent method for the module to ensure that communication with the master is still present. This parameter sets the time out limit so if no data is received for the time period set, the module will react as per the timeout action selection.

If enabled and a timeout occurs the drive will trip with fault “F_48”



Advanced Features

8.5 Sync and Freeze

8.5.1 Sync and Freeze Overview

The network master can put cyclic data into groups which allows multiple cyclic channels to be suspended and updated using the SYNC and FREEZE commands.

The SYNC Command:

- Controls data to the drive. (Dout)
- The SYNC command will cause a single transfer of the previously grouped data and stop any more data from being received by the drive.
- The SYNC command may be repeated while in this state to allow another single transfer of data to the drive.
- Issuing an UNSYNC command will revert the drive to a continuous cyclic update of the received data.

The FREEZE Command:

- Controls data from the drive. (Din)
- The FREEZE command will cause a single update of the previously grouped Din data. In the next data cycle, the drive transfers “frozen” data to the master.
- The Din data will not be updated until the next FREEZE command is received (next “snapshot” taken) or the FREEZE mode is cancelled by an UNFREEZE command.
- Issuing an UNFREEZE command will revert the drive to a continuous cyclic update of the transmitted data.

8.5.2 Sync and Freeze Status

All devices that support SYNC and FREEZE provide an 8-bit status word. The SYNC and FREEZE status word is available as part of the PROFIBUS status word. The function of the 8-BITS is described in Table 15.

Table 15: SYNC and FREEZE Status

PID408 Value	Description
Bit 8	Reserved
Bit 9	Clear Out Data
Bit 10	Unfreeze
Bit 11	Freeze
Bit 12	Unsync
Bit 13	Sync
Bit 14	Reserved
Bit 15	Reserved



9 Diagnosics

9.1 Faults

In addition to the normal drive fault codes, the additional codes listed in Table 16 may be generated by the option module during a fault condition

Table 16: Fault Codes

Fault Code	Definition	Remedy
F046	Module timeout	Module to drive communications time out. Check the connection between drive and option module Refer to section 8.4.1 Module Timeout Action
F047	Master Monitoring timeout	Check network connection, cabling and termination. Refer to section 8.4.2 Master Monitor for details
F048	Data Exchange Timeout	Check network connection, cabling and termination. Refer to section 8.4.3 Data Exchange for details

9.2 Troubleshooting

Table 17: Troubleshooting

Symptom	Possible Cause	Remedy
No communications from the option module	Module is not initialised	Check the drive to module connection. Check PID408 Status Word BIT-0.
	Incorrect PROFIBUS-DP settings	Check the configuration using MotionView
	Improper wiring	Check wiring between the PROFIBUS-DP network and communication module. Ensure that the terminal block is properly seated. Check connection between module and drive.
One of the PROFIBUS-DP monitoring messages timed out and its time-out reaction is set to FAULT.		Identify the time-out message and modify appropriate time-out time or reaction to the time-out settings.
Module does not enter the Data Exchange State	Data size configuration mismatch between the Master and the Drive	Check the configuration sizes for Dout and Din Data. Refer to Parameters PID284 and PID285.



Parameter Reference

10 Parameter Quick Reference

Table 18 lists each parameter number and provides its function, default value and access rights.

Table 18: Parameter Quick Reference

Parameter	Function	Default Value	Access Rights	Cross Reference
PID283	Node Address	126	RW	4.3.5 Node Address
PID284	OUT Data Size	2	RW	5.2 Channel Data Sizes
PID285	IN Data Size	4	RW	5.2 Channel Data Sizes
PID298-309	Data IN Mapping		RW	5.3.1 Data IN (Din) Channels
PID286-297	Data OUT Mapping		RW	5.3.2 Data OUT (Dout) Channels
PID310	Acyclic Mode	0	RW	6.2.1 Acyclic Modes
PID408	Node Status Word	N/A	RO	8.3 PROFIBUS Status
PID409	Master Monitor Timeout	N/A	RO	8.4.2 Master Monitor Timeout Action
PID413	Timeout Actions		RW	8.4 PROFIBUS DP Timeout Actions

AC Technology Corporation

630 Douglas Street, Uxbridge MA 01569
Sales: 800-217-9100 * Service: 508-278-9100
www.lenze-actech.com

P94PFB01A



Modbus RTU & Modbus TCP/IP Communication Communications Interface Reference Guide

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, S94PM01) that shipped with the drive. These documents should be read in their entirety as they contain important technical data and describe the installation and operation of the drive and the applicable option module.

Copyright ©2005 by Lenze AC Tech Corporation.

All rights reserved. No part of this manual may be reproduced or transmitted in any form without written permission from Lenze AC Tech Corporation. The information and technical data in this manual are subject to change without notice. Lenze AC Tech Corporation makes no warranty of any kind with respect to this material, including, but not limited to, the implied warranties of its merchantability and fitness for a given purpose. Lenze AC Tech Corporation assumes no responsibility for any errors that may appear in this manual and makes no commitment to update or to keep current the information in this manual.

MotionView[®], PositionServo[®], and all related indicia are trademarks of Lenze AG.

Modbus[®] is a registered trademark of 'Schneider Automation'.



1	Safety Information.....	5
1.1	Warnings, Cautions & Notes.....	5
1.2	Reference Documents.....	6
2	Introduction.....	7
2.1	Fieldbus Overview	7
2.2	EIA-485 Module.....	7
2.2.1	Specification	7
2.2.2	Module Identification Label.....	7
2.3	Ethernet Port	8
3	Installation	9
3.1	Mechanical Installation	9
3.2	Connectors	10
3.2.1	EIA-485 Module	10
3.2.2	Ethernet Port.....	10
3.3	Electrical Installation.....	11
3.3.1	Cable Types	11
3.3.2	Network Limitations: EIA-485.....	11
3.3.3	Network Limitations: Ethernet	11
3.3.4	Connections and Shielding: EIA-485.....	12
3.3.5	Connections and Shielding: Ethernet	13
3.3.6	Network Termination: EIA-485	13
3.3.7	Network Termination: Ethernet.....	13
3.3.8	Network Schematic: EIA-485.....	14
3.3.9	Network Schematic: Ethernet	14
4	Commissioning	15
4.1	Overview	15
4.2	Configuring the Network Master/Client.....	15
4.3	Configuring the PositionServo Slave/Server.....	17
4.3.1	Connecting.....	17
4.3.2	Connect to the Drive with MotionView OnBoard	17
4.3.3	Modbus RTU Slave Node Settings.....	18
4.3.4	Modbus TCP/IP Server Node Settings	19
4.3.5	Re-Initializing	21
4.3.6	Non-Communication Based Parameter Settings	21
4.4	Drive Monitoring	22



Contents

4.5	Controlling the Drive	22
4.6	Changing Drive Parameters	22
4.7	EIA-485 (RS485) Parameters	22
4.8	Ethernet Parameters	23
4.9	Negative Number Transmission	23
5	Modbus Implementation	24
5.1	Supported Function Codes	24
5.2	Data Format, Size and Memory Area	24
5.3	Register Numbering	25
5.4	Endian Format	26
5.5	Registers Access	26
	5.5.1 Register Reading	26
	5.5.2 Register Writing	26
5.6	No Response Conditions	26
5.7	Exception Responses	27
5.8	Modbus Message Frame	27
	5.8.1 PDU Function Code	27
	5.8.2 PDU Data	27
	5.8.3 ADU for Modbus RTU	28
	5.8.4 ADU for Modbus TCP	28
6	Reference	29
6.1	PID List with Modbus Values	29



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 VDE 0110 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 (2004/108/EC).

The drive controllers meet the requirements of the Low Voltage Directive 2006/95/EC. 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.



Safety Information

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 servicing the drive. Observe all 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.

Table 1: 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
Refer to: <http://www.modbus.org/tech.php>
- Modbus Over Serial Line Specification & Implementation Guide V1.0.
Refer to: <http://www.modbus.org>
- PositionServo Programming Manual: PM94P01, PM94M01(MVOB)
Refer to: <http://www.lenze-actech.com>



NOTE:

The complete list of variables can be found in the PositionServo Programming Manual (PM94P01, PM94M01).

2 Introduction

The following information is provided to explain how the PositionServo drive operates on a Modbus network; it is not intended to explain how Modbus itself works. Therefore, a working knowledge of Modbus is assumed, as well as familiarity with the operation of the PositionServo drive.

2.1 Fieldbus 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 which the PositionServo drive is always a slave node. While the Modbus RTU protocol does not specify the physical layer, the most commonly used is 2-wire EIA-485 (RS485). The PositionServo requires the use of an EIA-485 option module (E94ZARS41) to be able to connect to such a network and communicate via Modbus RTU.

Modbus TCP/IP uses an Ethernet physical layer and as such peer-to-peer and client-server communication techniques are possible. However, the PositionServo drive is always a server (slave node).

2.2 EIA-485 Module

2.2.1 Specification

- Supported baudrates: 115200bps, 57600bps, 38400bps, 19200bps, 9600bps
- Parity modes supported: Even, Odd, None
- Stop bits supported: 2, 1.5, 1
- EIA-485, 2-wire (half duplex)
- Network impedance loading of 1 unit (EIA-485 specification stipulates max of 32 units per network segment)

2.2.2 Module Identification Label

Figure 1 illustrates the labels on the PositionServo EIA-485 (RS485) option module. The PositionServo EIA-485 module is identifiable by:

- One label affixed to the side of the module.
- The TYPE identifier in the center of the label: E94ZARS41
- The port (interface) identifier, P21, on the right hand side of the label.

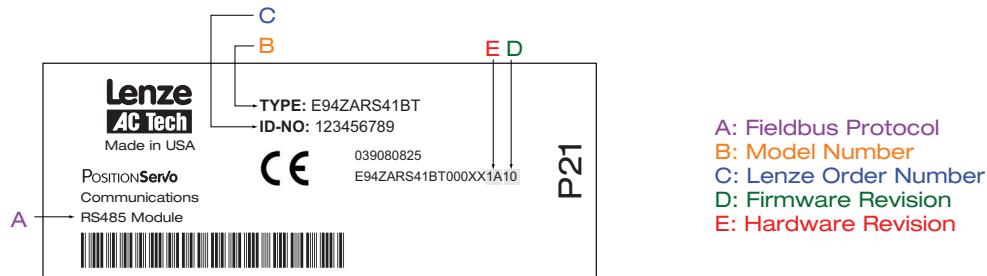


Figure 1: PositionServo EIA-485 (RS485) Module Label



Introduction

2.3 Ethernet Port

- Supported baudrates: 100Mbps and 10Mbps
- Supports two simultaneous Modbus TCP/IP connections on port 502
- Complies with IEEE 802.3
- Standard screened RJ45 connector with integrated status LEDs
- On open connections with no activity for more than 75 seconds, the PositionServo Drive sends a TCP keep-alive message every 75 seconds to check the connection status.



NOTE:

The PositionServo does **not** support auto negotiation/cross over. Therefore, unless the connecting device supports auto negotiation/cross over, a crossover cable will be required for one-to-one connection.

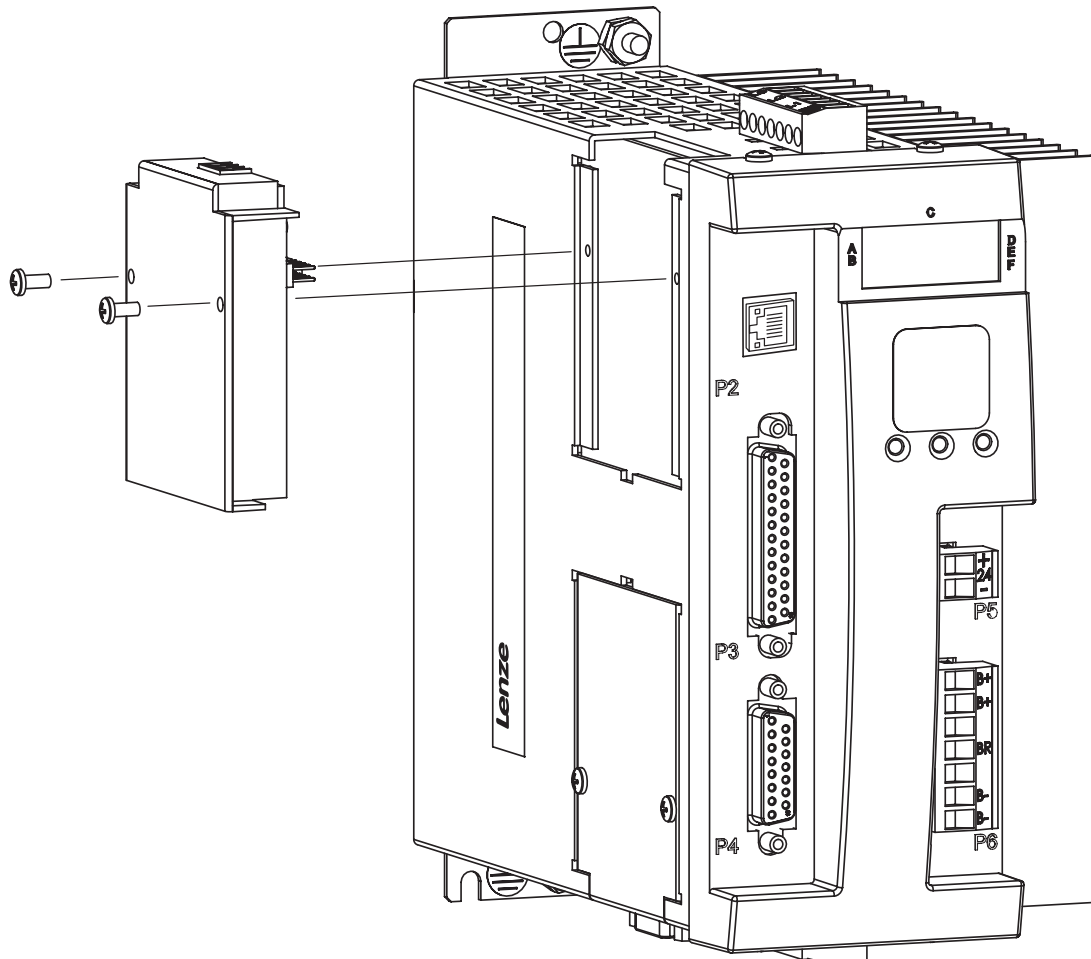


3 Installation

Section 3.1 is only applicable to Modbus RTU communication with the EIA-485 (RS485) option module, E94ZARS41. Modbus TCP/IP communication uses the P2 Ethernet port on the front of the PositionServo.

3.1 Mechanical Installation

1. Ensure that for reasons of safety, the AC supply, DC supply and +24VDC backup supply have been disconnected before opening the option bay cover.
2. Remove the two COMM module screws that secure Option Bay 1. With a flat head screwdriver, lift the Option Bay 1 cover plate and remove.
3. Fit the 20-pin header into the module before fitting the module into the drive.
4. Install the EIA-485 (RS485) COMM Module (E94ZARS41) in Option Bay 1.
5. Replace the two COMM module screws (max torque: 0.3Nm/3lb-in) to secure Option Bay 1 in place.



S921

Figure 2: Installation of EIA-485 (RS485) Communications Module



Installation

3.2 Connectors

3.2.1 EIA-485 Module

Table 2 and Figure 3 illustrate the pinout of the PositionServo EIA-485 (RS485) Option Module E94ZARS41. The 3-pin connector provides 2-wire plus isolated ground connection to the network.

Table 2: EIA-485 (RS485) Interface Pin Designation

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



Figure 3: EIA-485 (RS485) Interface Pin Designation

3.2.2 Ethernet Port

Port P2 on the front of the PositionServo is an RJ45 Standard Ethernet connector that is used to communicate with a host via Ethernet TCP/IP.

Table 3: P2 Pin Assignments (Communications)

Pin	Name	Function	RJ45 Connector
1	+ TX	Transmit Port (+) Data Terminal	
2	- TX	Transmit Port (-) Data Terminal	
3	+ RX	Receive Port (+) Data Terminal	
4	N.C.		
5	N.C.		
6	- RX	Receive Port (-) Data Terminal	
7	N.C.		
8	N.C.		

The status LEDs integrated in the RJ45 connector indicate link activity and baudrate. The green LED indicates baudrate and blinks steadily when the drive is running at the network speed (10/100Mbps). The yellow LED indicates link activity and flashes when the drive is communicating (transmitting/receiving) with the network.



3.3 Electrical Installation

3.3.1 Cable Types

Due to the high data rates used on Modbus networks it is paramount that correctly specified quality cable is used. The use of low quality cable will result in excess signal attenuation and data loss.

For EIA-485 it is recommended to use a good quality shielded twisted pair cable with characteristic impedance of 120Ω .

For Ethernet it is recommended that a minimum specification of CAT5e UTP cable (unscreened) is used. However, for environments that high levels of electrical noise STP (screened) cable is recommended.

3.3.2 Network Limitations: EIA-485

There are several limiting factors that must be taken into consideration when designing a Modbus RTU network, however, here is a simple checklist:

- Modbus RTU networks are limited to a maximum of 247 nodes.
- Only 32 nodes (based on each node having a load impedance of 1 unit) may be connected on a single network segment. Certain Modbus EIA-485 masters may only be able to support a fewer number of nodes (i.e., 8, 16). Refer to the documentation for the Modbus master in use.
- A network may be built up from one or more segments with the use of network repeaters.
- Maximum network segment length is 1200 meters for baudrates up to and including 19200bps. Certain Modbus masters may be limited to shorter runs. Refer to the documentation for the Modbus master in use.
- Minimum of 1 meter of cable between nodes.
- Use fiber optic segments to:
 - Extend networks beyond normal cable limitations.
 - Overcome different ground potential problems.
 - Overcome very high electromagnetic interference.
- EIA-485 is a linear daisy chain topology. Both ends of the network segment must be terminated by a $120\Omega \pm 1\%$ resistor.

3.3.3 Network Limitations: Ethernet

There are several limiting factors that must be taken into consideration when designing a Modbus TCP/IP network, however, here is a simple checklist:

- Modbus TCP/IP networks are limited to a maximum of 255 nodes per subnet (based on a Class C addressing system).
- Hubs are not recommended for general use as they contribute in creating network data collisions (ports on a hub do not route data direct to other ports but instead all ports are open to receive data from every port) and as such will cause additional delays in transmissions while the re-attempts are carried out.
- Switches are the recommended solution for connecting a multi-node network as they route network data direct from port to port (collisions may occur during network start-up or when a device is connected and the correct port routing is established) and therefore reduce the possibility of collisions.
- “Office grade” Ethernet equipment does not generally offer the same level of noise immunity or robustness as “industrial grade” Ethernet equipment.



Installation

- Maximum cable length for UTP/STP CAT5e cable is typically 100m. For other categories consult the cable data sheet.
- Use fiber optic segments to:
 - Extend networks beyond normal cable limitations.
 - Overcome different ground potential problems.
 - Overcome very high electromagnetic interference.
- Spurs or T connections are not permitted on an Ethernet cable. To create additional connections an Ethernet switch must be used.
- The use of wireless networking products for industrial applications is becoming more acceptable, but extreme care must be taken during the design phase and consultation with an industrial wireless provider is strongly recommended.

3.3.4 Connections and Shielding: EIA-485

To ensure good system noise immunity all network cables should be correctly grounded:

- Minimum recommendation of grounding is that the network cable is grounded once in every cubical.
- Ideally the network cable should be grounded on or as near to each drive as possible.
- For wiring of cable to the connector plug the unscreened cable cores should be kept as short as possible; recommended maximum of 20mm.

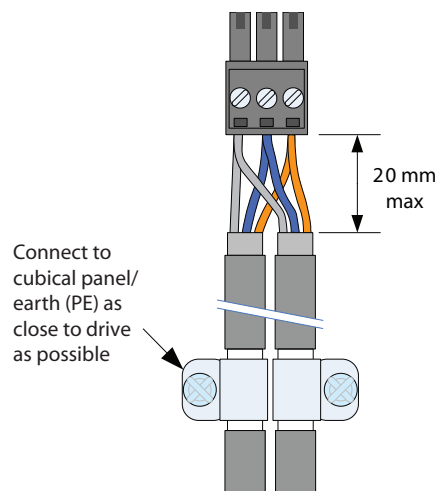


Figure 4: EIA-485 Connection



3.3.5 Connections and Shielding: Ethernet

The use of pre-fabricated cables is recommended as this reduces the chances of connections mistakes and poor quality connections.

If cable connections are assembled on site then it is strongly recommended that these cables are tested with a suitable Ethernet cable tester

STP cables are the preferred solution as they provide a screen/shield surrounding the inner cores and have an integrated screened surround on the RJ45 connector for quick and easy connection.



Figure 5: CAT5e STP Cable

Images ©2000-2009 Belkin International, Inc

3.3.6 Network Termination: EIA-485

In high speed (EIA-485) networks (typically 19.2kbps or higher) it is essential to install the specified termination resistors, i.e. one at both ends of a network segment. Failure to do so will result in signals being reflected back along the cable which will cause data corruption. A 120Ω 1% $\frac{1}{4}W$ resistor should be fitted to both ends of a network segment across the TxA and TxB lines.

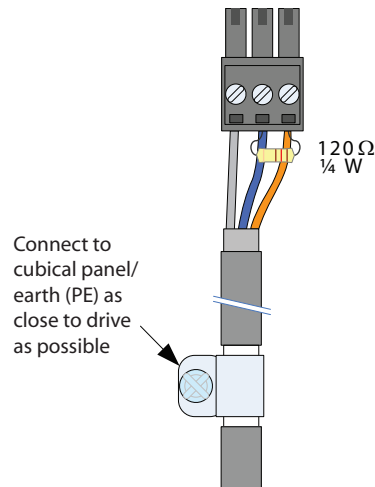


Figure 6: EIA-485 (RS485) Network Termination

3.3.7 Network Termination: Ethernet

Ethernet network cable termination is not required as it is integrated into the circuitry of each device's RJ45 port.



Installation

3.3.8 Network Schematic: EIA-485

Figure 7 illustrates the connection of the cables for a PositionServo drive in a Modbus master/slave network.

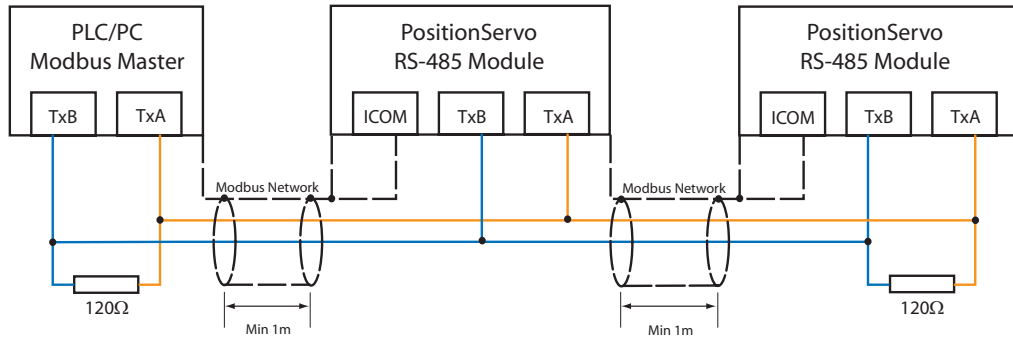


Figure 7: 120Ω (1%) Termination in EIA-485 (RS485) Network

3.3.9 Network Schematic: Ethernet

Figure 8 illustrates a one-to-one ethernet connection. Figure 9 illustrates a multi node ethernet connection.

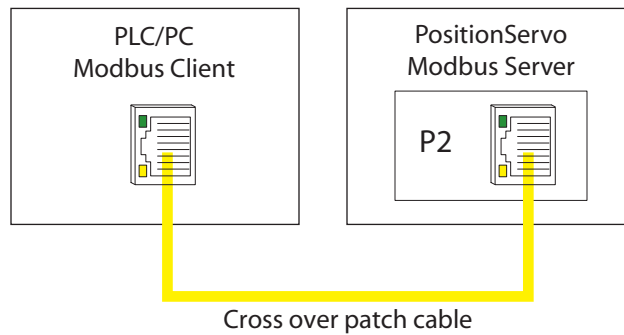


Figure 8: One-to-One Connection

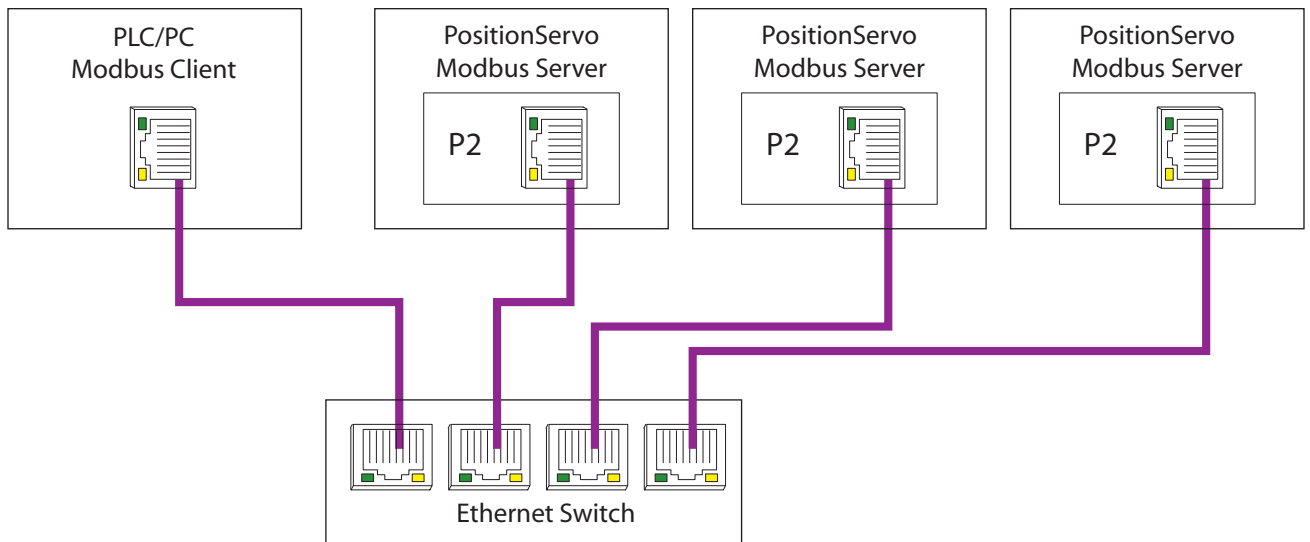


Figure 9: Multi Node Connection



4 Commissioning

4.1 Overview

It is assumed that the user has familiarised themselves with how to set parameters using MotionView software. Refer to the PositionServo with MVOB User Manual (S94PM01) for more details. The details that follow provide a step-by-step guide to quickly and easily set-up a PositionServo drive to communicate on a Modbus network

4.2 Configuring the Network Master/Client

The method for configuring master/client devices differs greatly between manufacturers. Provided herein is a very basic, generic guide to setting up a network master/client. Consult the master/client manufacturer for configuration assistance if required.

1. Launch the Master/client configuration software.
2. Setup the Master/client Modbus port as required. Refer to Table 4.

Table 4: Modbus RTU and Modbus TCP/IP Settings

Modbus RTU Master settings	Modbus TCP Client settings
Node address	IP Address or DHCP enabled
Baud rate	Subnet mask set as required, i.e. 255.255.255.0
Data bits = 8 (for Modbus RTU)	Gateway set as required
Parity	Service port = 502
Stop bits	Baud rate set as required, i.e. 100Mbps
Flow control	

3. Add generic Modbus slave/server node to the master/client
4. Set the slave/server node address.
5. Assign the Modbus slave/service registers as required.



NOTE:

In true Modbus, 3X and 4X Registers are numbered starting at 1. This is known as 'one based' addressing. However, when transmitted to a slave over the serial link, the actual address transmitted is one less.

Some Modbus masters will allow for the first register number to be 0. This is known as 'zero based' addressing. If this is the case, the Modbus register numbers listed in this manual must be offset by -1 to properly program a master using 'zero-based' addressing.

Refer to Section 6 for a list of the PositionServo Modbus registers.



Commissioning

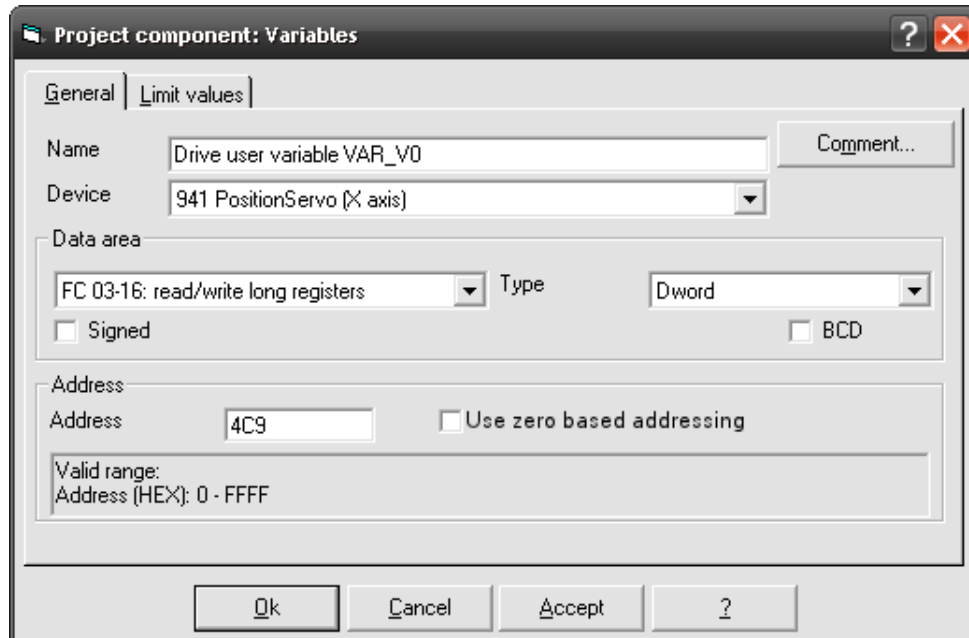


Figure 10: Example Modbus Register Assignment

6. Repeat steps 3 to 5 for each required slave/server node

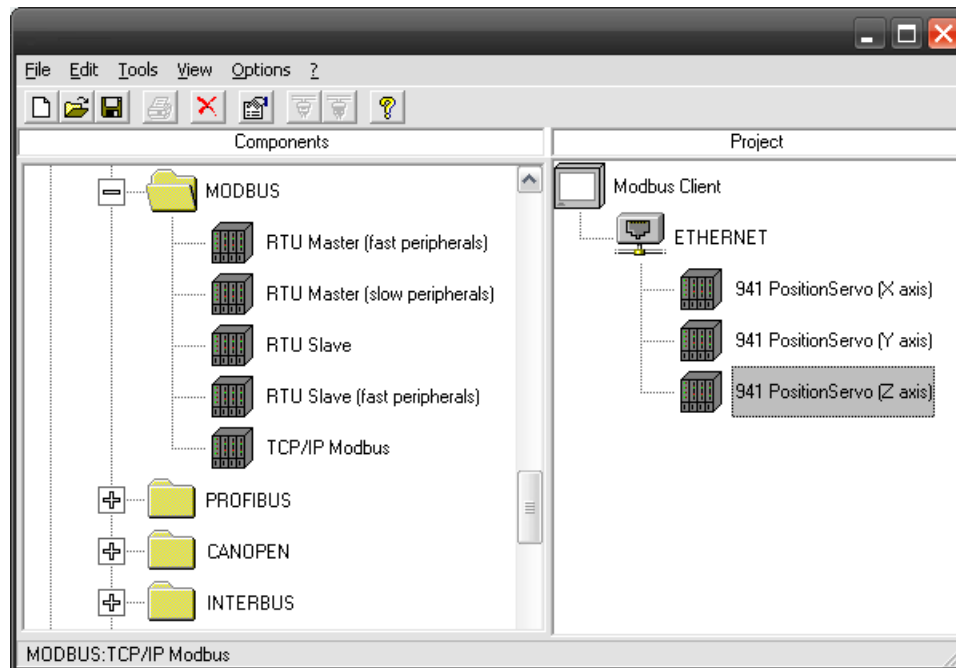


Figure 11: Example Modbus Master/Client Configuration

7. Save the configuration and download to the master/client



4.3 Configuring the PositionServo Slave/Server

4.3.1 Connecting

With the drive power disconnected, install the EIA-485 (RS485) module and connect the network cable as instructed in the preceding sections. Ensure the drive Run/Enable terminal is disabled then apply the correct voltage to the drive (refer to drive's user manual for voltage supply details).

4.3.2 Connect to the Drive with MotionView OnBoard

Refer to the PositionServo User Manual, section 6.2 for full details on configuring and connecting a drive via MotionView OnBoard (MVOB) software. Contained herein is a brief description of launching MVOB and communicating with the drive.

1. Open the PC's web browser. Enter the drive's default IP address [192.168.124.120] in the browser's Address window.
2. The authentication screen may be displayed if the PC does not have Java RTE version 1.4 or higher. If so, to remedy this situation, download the latest Java RTE from <http://www.java.com>.
3. When MotionView has finished installing, a Java icon entitled [MotionView OnBoard] will appear on your desktop and the MVOB splash screen is displayed. Click [Run] to enter the MotionView program.
4. Once MotionView has launched, verify motor is safe to operate, click [YES, I have] then select [Connect] from the Main toolbar (top left). The Connection dialog box will appear.
5. Select [Discover] to find the drive(s) on the network available for connection.

[Discover] may fail to find the drive's IP address on a computer with both a wireless network card and a wired network card (or a PC with more than one network connection). If this happens, try one of the following remedies:

Disable the wireless network card and then use [Discover].

Type in the drive's IP address manually at the box [IP Address].

Then click [Connect]

6. Highlight the drive (or drives) to be connected and click [Connect] in the dialog box.

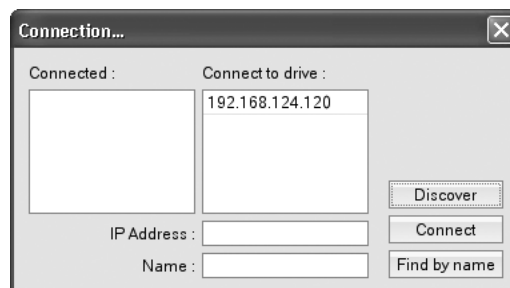


Figure 12: Connection Box with Discovered Drive

In the lower left of the MotionView display, the Message Window will contain the connection status message. The message "Successfully connected to drive B04402200450_192.168.124.120" indicates that the drive B04402200450 with IP address 192.168.124.120 is connected.



Commissioning

4.3.3 Modbus RTU Slave Node Settings

If using the EIA-485 (RS485) module, open MotionView and click on the [Communication] folder. Then select the [RS485] folder to set/change the RS485 parameters: Configuration, Baud Rate, Parity, Stop Bits and Address.

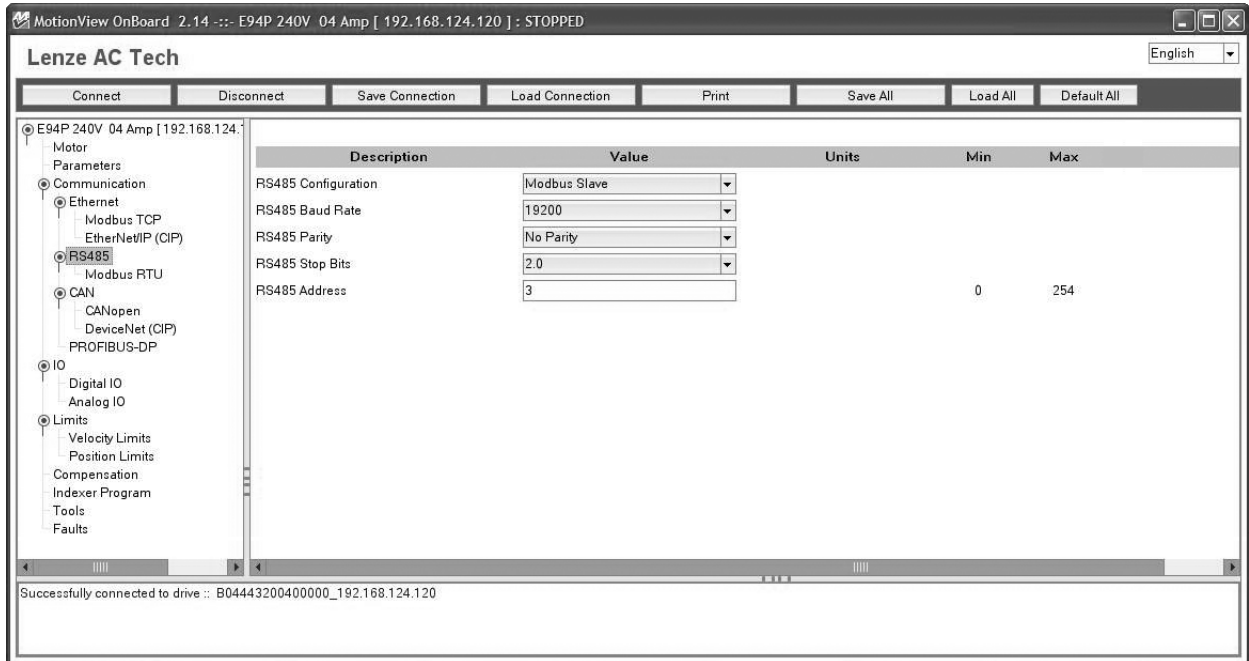


Figure 14: RS-485 Folder

Configuration: 'Modbus slave' = the modbus slave protocol is enabled on the RS485 port.
UPPP = the RS485 uses UPPP (Point-to-Point Protocol).

Baud Rate: 115200bps, 57600bps, 38400bps, 19200bps, 9600bps

Parity: Even, Odd, None

Stop Bits: 2, 1.5, 1

Address: 1-247

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 RS485 default configuration is: UPPP, 19200bps, No Parity, 2 Stop Bits and Address = 1.

TIP - Avoid using address 1. Most Modbus devices ship with a default address of 1. As duplicate addressing on a Modbus network is not allowed, this can lead to conflicts when replacing and commissioning nodes. To avoid this it is recommended that you do not set the slave address to 1.

Modbus RTU Folder - Modbus Reply Delay

Modbus Reply Delay is 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 slower to respond than others and an increase of the 'Modbus reply delay' value may be required to successfully work with these devices.

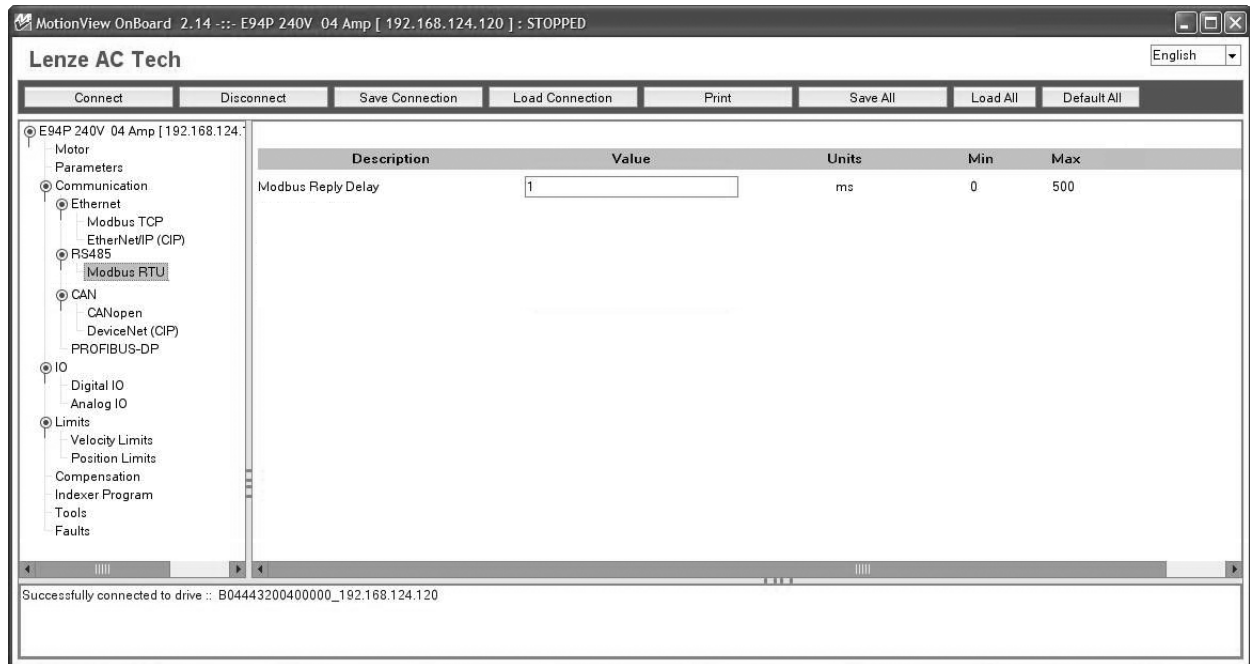


Figure 15: Modbus RTU Folder

4.3.4 Modbus TCP/IP Server Node Settings

The IP address of the PositionServo drive is composed of four sub-octets that are separated by three dots. Each sub-octet can be configured with a number between 1 and 254. As shipped from the factory the default IP address of a drive is:

192.168.124.120.

There are two methods of changing the current IP address. An address can be assigned to the drive automatically (dynamic IP address) when the drive is connected to a DHCP (Dynamic Host Configuration Protocol) enabled server, or the drive can have an IP address assigned to it manually by the user (static IP address).

4.3.4.1 Obtaining the PositionServo's Current Ethernet Settings

The current Ethernet setting and IP address of the PositionServo drive can be obtained from the drive display and keypad. Press the recessed 'mode' button (↵) on the display and use the "UP" and "DOWN" buttons (▲ ▼) to access parameters IP_1, IP_2, IP_3 and IP_4. Each of these parameters contain one sub-octet of the full IP address, for example in the case of the drive default (factory set) address parameters:

IP_1 = 192

IP_2 = 168

IP_3 = 124

IP_4 = 120

By accessing these four parameters the full IP address on the drive can be obtained.

If parameters IP_1, IP_2, IP_3 and IP_4 all contain '----' rather than a numerical values it means that the drive has DHCP enabled and the DHCP server is yet to assign the drive its dynamic IP address. As soon as an IP address is assigned by the server the address assigned will be display by the drive in the above parameters. See section on obtaining IP addresses through DHCP.



Commissioning

4.3.4.2 Configuring the IP Address Manually (Static Address)

When connecting directly from PositionServo drive to the PC without a DHCP server or when connecting to a private network (where all devices have static IP addresses) the IP address of the PositionServo drive will need to be assigned manually.

To assign the address manually, the drive must have its DHCP mode disabled. This can be done using the drive keypad and display. Press the recessed 'mode' button (↵) on the display and use the "UP" and "DOWN" buttons (▲ ▼) to access parameter 'DHCP'. Check this parameter is set to a value of '0'. If the DHCP parameter is set to '1' then use the 'mode' (↵) and down (▼) arrows to set to '0' and then cycle power to the drive in order for this change to take effect. When DHCP is disabled and power cycled to the drive, it will revert back to its previous static IP address.

It is most common for the PositionServo drive IP address to be left at its default value (192.168.124.120) and to configure the PC Ethernet port to communicate on this subnet. If more than one drive needs to be connected to the PC at any one time then the IP_4 parameter can be accessed via the keypad and changed to provide a unique IP address on the network for each drive. Note that IP_4 is the only octet that can be changed (IP_1, IP_2, and IP_3 are read-only) and that power must be cycled to the drive for any changes to take effect.

If the PositionServo drive(s) needs to be configured for a specific subnet with different values to default (for IP_1, IP_2, and IP_3, and IP_4) then this needs to be performed with the MotionView configuration tool. First establish communications using the default drive address or with an address that was established by changing IP_4 parameters via the drive keypad. Follow the rest of these instructions in order to establish communications and launch MotionView using this address. Once within the MotionView software a full IP address can be assigned.

From the Node tree within MotionView select the [Communications] folder and then the [Ethernet] sub-folder as shown in Figure 16. The settings reflect those that will appear in the software parameter view window.

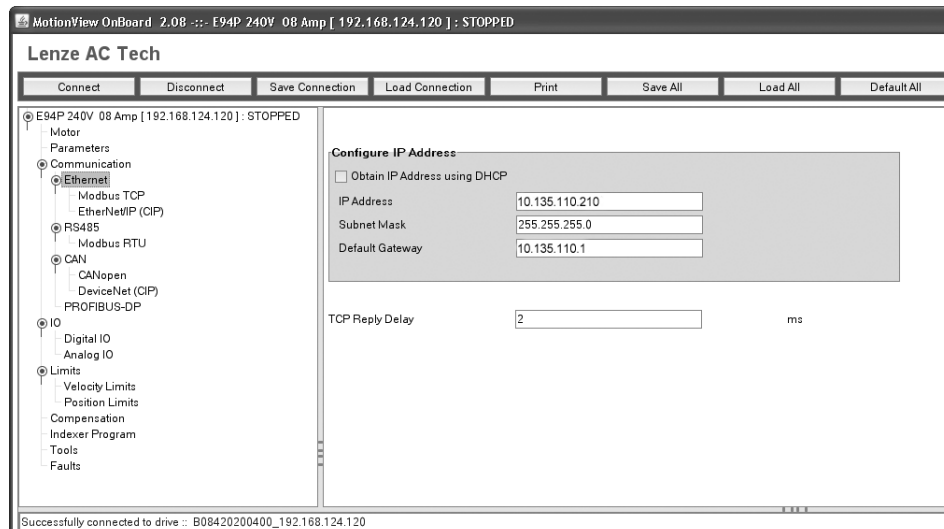


Figure 16: Ethernet Folder

The IP address, subnet mask, and default gateway address can all be edited in this screen. If the text in any of these boxes turns red once it has been entered then this means that the values or format used is invalid and the values will not be applied.

To enable DHCP, click the box adjacent to [Obtain IP Address using DHCP] to place a check mark in this box .



To disable DHCP, click the box again. Power must be cycled for any changes to [Configure IP Address] to take effect. On changing any ethernet parameter value, the dialog box in Figure 17 will appear. Click [OK] and cycle power for changes to take effect.

4.3.4.3 Configuring the IP Address Automatically (Dynamic Address)

When connecting a PositionServo drive onto a network domain with a DHCP enabled server (where all devices have dynamic IP addresses assigned by the server) the IP address of the PositionServo drive can also be assigned automatically by the server.

To have the address assigned automatically the drive must have its DHCP mode enabled. This can be done by using the drive keypad and display. Press the 'mode' button on the display and use the "UP" and "DOWN" buttons to access parameter 'DHCP'. Check this parameter is set to a value of '1'. If the DHCP parameter is set to '0' then use the 'mode' and up arrow to set to '1' and then cycle power to the drive in order for this change to take effect.

When the PositionServo drive is waiting for an IP address to be assigned to it by the server it will display '----' in each of the four octet parameters (IP_1, IP_2, IP_3, and IP_4) on its display. Once the address is assigned by the server it will appear in these parameters. If this parameters continue to display '----' then it is likely that a connection between the drive and server has not been established, or the server is not DHCP enabled.

DHCP can be enabled through the MotionView software for convenience should the operator wish to configure the drive using a manual (static) IP address and switch over to an automatic (dynamic) address once configuration is complete.

4.3.5 Re-Initializing

To activate any changes made the drive has to be reinitialized. Hence the warning within MotionView

Some parameter(s) change will take an effect after REBOOT.

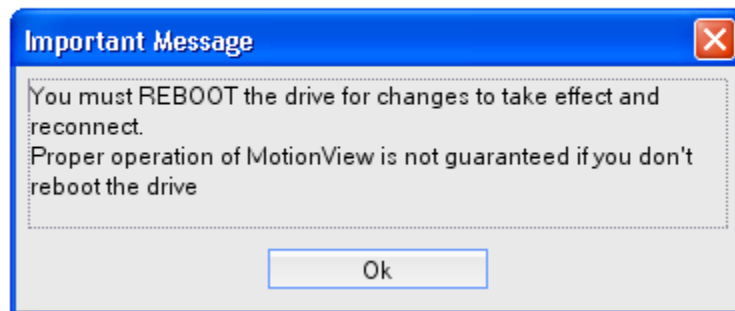


Figure 17: REBOOT Message

This can be done by cycling the power to the drive.

4.3.6 Non-Communication Based Parameter Settings

In addition to configuring the Modbus settings and depending upon the application there may be several drive based parameters that will need to be set using MotionView or an Indexer program or via the Modbus parameter access channel. Such as:

- PID34 – Drive Mode (VAR_DRIVEMODE)
- PID37 – Reference (VAR_REFERENCE)
- PID29 – Enable switch function (VAR_ENABLE_SWITCH_TYPE)



Commissioning

4.4 Drive Monitoring

The master/client can read the drive parameters as long as Modbus communications are enabled.



NOTE:

The complete list of variables can be found in the PositionServo Programming Manual (PM94P01, PM94M01).

4.5 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 list in the PositionServo Programming Manual for the functionality of the drive's variables.

4.6 Changing Drive Parameters

To change drive parameters, simply write to the appropriate register as listed in the PositionServo Programming Manual (PM94P01 or PM94M01).

4.7 EIA-485 (RS485) Parameters

Drive variables #172-176 are EIA-485 (RS485) communication programming parameters specifically for configuration of the EIA-485 interface.

Table 5: EIA-485 (RS485) Variables - Excerpted from PS Variable List

PID	Variable Name	Type	Format	EPM	Access	Description	Units
172	VAR_SERIAL_ADDRESS		W	Y	R/W	RS485 drive ID. Range: 0 - 254	
173	VAR_MODBUS_BAUDRATE		W	Y	R/W	Baud rate for Modbus operations: 2 - 9600 5 - 57600 3 - 19200 6 - 115200 4 - 38400	
174	VAR_MODBUS_DELAY		W	Y	R/W	Modbus reply delay in mS Range: 0 - 1000	mS
175	VAR_RS485_CONFIG		W	Y	R/W	RS485 configuration: 0 - normal IP over PPP 1 - ModBus	
176	VAR_PPP_BAUDRATE NOTE: Does NOT apply to MVOB.		W	Y	R/W	RS232/485 (normal mode) baud rate: 2 - 9600 5 - 57600 3 - 19200 6 - 115200 4 - 38400	



4.8 Ethernet Parameters

Drive variables #67-70 are Ethernet communication programming parameters specifically for configuration of the ethernet interface.

Table 6: Ethernet Variables - Excerpted from PS Variable List

PID	Variable Name	Type	Format	EPM	Access	Description	Units
67	VAR_IP_ADDRESS		W	Y	R/W	Ethernet IP address. IP address changes at next boot up. 32 bit value	
68	VAR_IP_MASK		W	Y	R/W	Ethernet IP NetMask. Mask changes at next boot up. 32 bit value	
69	VAR_IP_GATEWAY		W	Y	R/W	Ethernet Gateway IP address. Address changes at next boot up. 32 bit value	
70	VAR_IP_DHCP		W	Y	R/W	Use DHCP: 0, 1 0 - manual; 1 - use DHCP service	

4.9 Negative Number Transmission

Drive variables 51, 60, 79, 81 and 90 are signed integer values and could be negative. These registers are sent over the modbus communications in signed internal units.

Table 7: Signed Integer Variables - Excerpted from PS Variable List

PID	Variable Name	Type	Format	EPM	Access	Description	Units
51	VAR_VREG_WINDOW	vel	W	Y	R/W	Gains scaling coefficient Range: -16 to +4	
60	VAR_VLIMIT_ATSPEED		F	Y	R/W	Target Velocity for At Speed window Range: -10000 - +10000	Rpm
79	VAR_M2SRATIO_MASTER		W	Y	R/W	Master to system ratio. Master counts range: -32767 - +32767 Value will be applied upon write to PID #80. Write to this PID followed by writing to PID#80 to apply new ratio pair	
81	VAR_S2PRATIO_SECOND		W	Y	R/W	Secondary encoder to prime encoder ratio. Second counts range: -32767 - +32767 Value will be applied upon write to PID #82. Write to this PID followed by writing to PID#82 to apply new ratio pair	
90	VAR_AIN1_OFFSET			Y	R/W	Analog input #1 offset. Applied when used as current/velocity reference Range: -10,000 to +10,000	mV



Protocol Implementation

5 Modbus Implementation

5.1 Supported Function Codes

The Modbus function codes supported by the PositionServo drive are:

- 03 – Read Holding Register
- 16 – Preset (write) Multiple Registers

5.2 Data Format, Size and Memory Area

Modbus registers are limited by protocol definition to a length of 16-bits per register. The user must use two consecutive 16-bit registers to read/write one 32-bit register.

All PositionServo drive parameters are 32-bit in size but can be accessed in 3 different formats:

- IEEE Floating Point (FLOAT or REAL)
- 32-bit integer (DWORD or DINT)
- 16-bit integer (WORD or INT) where by the true 32-bit value consumes two consecutive 16-bit registers

Furthermore, PositionServo parameters exist in each of the 3 formats in both RAM (volatile) and EPM (non-volatile) areas. Therefore the memory addresses are divided into six ranges according to their format and memory type as shown in Table 8.

Table 8: Memory Address Ranges

Memory Area Offset	0	512	1024	1556	2068	2304
Type	RAM	RAM	EPM	EPM	RAM	EPM
Format	32-bit INT	Float	32-bit INT	Float	16-bit INT	16-bit INT

The Modbus register address of a drive parameter can be calculated as follows:

$$\text{Modbus}_{\text{Register}} = (2 \times \text{PID}_{\text{Number}}) + \text{Memory}_{\text{Offset}} + \text{Modbus}_{\text{Offset}}$$

Where:

PIDNumber = PositionServo **P**arameter **I**ndex Number. Refer to section xxxx for a full list.

MemoryOffset = Memory offset as per table 4 above

ModbusOffset = 0 for zero based addressing
1 for traditional Modbus addressing

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):

$$\text{<register address>} = 0 + 2 * \text{<variable index>} + 1;$$

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

$$\text{<register address>} = 512 + 2 * \text{<variable index>} + 1;$$

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

$$\text{<register address>} = 1024 + 2 * \text{<variable index>} + 1;$$



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 only 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.

Refer to section 6 for a complete list of Modbus registers for each variable.

5.3 Register Numbering

Modbus registers start at 1 which historically coincided with many older slave devices that often have parameters starting at address 1. However, the true data addressed within a Modbus telegram starts at address 0. This means that registers are offset by 1 compared to the true data address transmitted on the network, e.g.

Holding register 40001 is actually accessed as 0000 in the message telegram address field

The conversion from Modbus register number to the Modbus data address field is performed automatically by the Modbus Master/Client. The PositionServo adheres to the Modbus Standard in that its registers start at 1.



NOTE:

Some Modbus masters will allow for the first register number to be 0. This is known as 'zero based' addressing. If this is the case, the Modbus register numbers listed in this manual must be offset by -1 to properly program a master using 'zero-based' addressing.

- Using a master that supports traditional register addressing to access PositionServo parameter 100 (user variable VAR_VO) as a 16-bit value would use Modbus register 42405
- Using a master that has zero based addressing enabled would use Modbus register 42404



Protocol Implementation

5.4 Endian Format

Modbus uses “big-endian” representation of the register data. This means that when a numerical value that is larger than a single byte is transmitted, the MOST significant byte (MSB) is sent first, e.g.

- 16-bit integer value 0x1234 = 2 bytes of 0x12 and 0x34
- 32-bit integer value 0x12345678 = 4 bytes of 0x12, 0x34, 0x56 and 0x78

5.5 Registers Access

- Care should be taken when accessing registers from multiple sources such as multiple clients or the drive Indexer program as data could be over written or out of sequence
- Writing to the EPM area of memory simultaneously writes to the RAM area too
- Writing to the EPM area of memory should be done conservatively as the EEPROM (EPM) has a typical life expectancy of 1 million writes

5.5.1 Register Reading

Use the function code “03 (0x03) Read 4X Holding Registers” to read an adjoining block of holding registers in a remote device.



NOTE:

Do NOT attempt to read any write-only variables. Attempting to read a write-only variable can result in erroneous data.

5.5.2 Register Writing

No discrete coil access (function code 1) 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.



NOTE:

Do NOT attempt to write to any read-only variables. Attempting to write to a read-only variable can result in drive fault (F41).

5.6 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)



5.7 Exception Responses

If an invalid message is received, the drive will respond with a Modbus Exception as per the “Modbus application Protocol specification V1.1”, i.e. the exception function code = the request function code + 0x80 (an exception code is provided to indicate the reason of the error).

Table 9: Exception Codes

Code	V1.1 Specification	Description
0x01	Illegal Function	function not supported by PositionServo
0x02	Illegal data address	requested address is not a valid register address
0x03	Illegal Data Value	set value not valid for specific variable

5.8 Modbus Message Frame

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

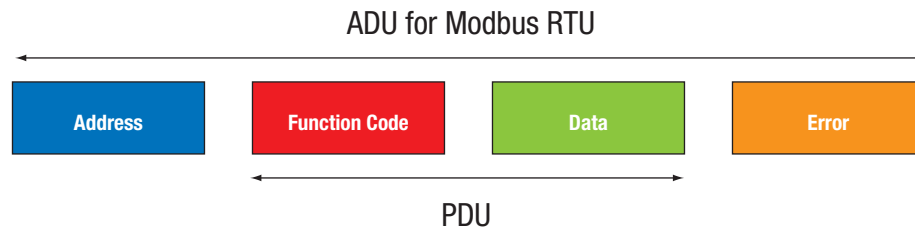


Figure 18: Modbus RTU Frame EIA-485

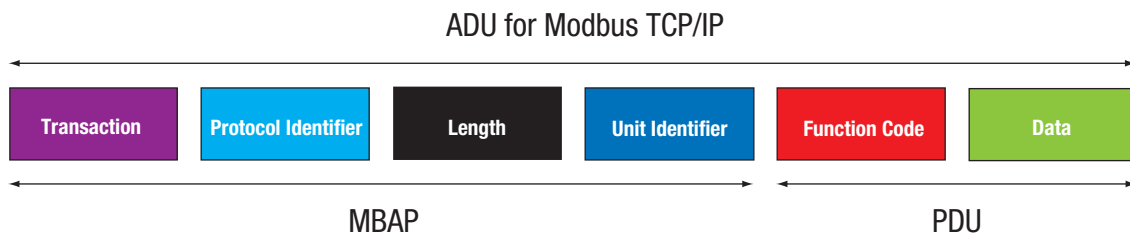


Figure 19: Modbus TCP/IP Request/Response

5.8.1 PDU Function Code

- Size = 1 byte
- The function code indicates what kind of action to perform.
- The function code (depending upon the function) is normally followed by a data field that contains request and response parameters.

5.8.2 PDU Data

The data field of messages sent from a master/client to slave/server device contains additional information that the slave/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 slave/server does not require any additional information. The function code alone specifies the action.



Protocol Implementation

If no error occurs related to the Modbus function requested in a properly received Modbus ADU, the data field of a response from a slave/server to a master/client 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.

5.8.3 ADU for Modbus RTU

The ADU for Modbus RTU consists of the Address field, Error Check and the common Modbus PDU.

Address Field

As described in the previous section the valid slave nodes addresses are in the range of 0 – 247 decimal. The individual slave devices are assigned addresses in the range of 1 to 247. A master addresses a slave by placing the slave address in the address field of the message. When the slave returns its response, it places its own address in the response address field to let the master know which slave is responding.

Error Check Field

Error checking field is the result of a Cyclical Redundancy Checking (CRC) calculation that is performed on the message contents.

The CRC field checks the contents of the entire message. It is applied regardless of any parity checking method used for the individual characters of the message.

The CRC field contains a 16-bit value implemented as two 8-bit bytes.

The CRC field is appended to the message as the last field in the message. When this is done, the low-order byte of the field is appended first, followed by the high-order byte. The CRC high-order byte is the last byte to be sent in the message.

The CRC value is calculated by the sending device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal then it results in an error.

5.8.4 ADU for Modbus TCP

The ADU for Modbus TCP consists of the MBAP Header and the common Modbus PDU.

The MBAP header is 7 bytes long.

The actual IP addressing and message error checking are performed by the Ethernet physical layer, refer to the ISO 7-layer model and the Modbus-IDA website for further details.

Table 10: MBAP Header

Field	Length (Bytes)	Description	Client	Server
Transaction Identifier	2	Identification of a Modbus Request/Response transaction	Initialized by the client	Recopied by server from received request
Protocol Identifier	2	0 = Modbus protocol	Initialized by the client	Recopied by server from received request
Length	2	Number of following bytes	Initialized by the client (request)	Initialized by the server (response)
Unit Identifier	1	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by server from received request

Table Copyright © 2005-2009 Modbus-IDA from the official Modbus Messaging Implementation Guide V1.0b



6 Reference

6.1 PID List with Modbus Values

This is a condensed PID List to show the corresponding Modbus 4X Registers for PIDs 1-256. Modbus RTU can not access beyond PID256. For the complete variable list refer to the PositionServo Programming Manual (PM94P01 or PM94M01).

These variables can be accessed from the user's program or any supported communications interface protocol. From the user program, any variable can be accessed by either its variable name or by its index value (using the syntax: @<VARINDEX> , where <VARINDEX> is the variable index from the PID List. From the communications interface any variable can be accessed by its index value.

The column "**Type**" indicates the type of variable:

mtr Motor: denotes a motor value
 mtn Motion: writing to an "mtn" variable could cause the start of motion ⚠
 vel Velocity: denotes a velocity or velocity scaling value

The column "**Format**" provides the native format of the variable:

W 32 bit integer
 F float (real)

When setting a variable via an external device the value can be addressed as floating or integer. The value will automatically adjusted to fit it's given form.

The column "**EPM**" shows if a variable has a non-volatile storage space in the EPM memory:

Y Variable has non-volatile storage Space in EPM
 N Variable does not exist in EPM memory

The user's program uses a RAM (volatile) 'copy' of the variables stored on the EPM. At power up all RAM copies of the variables are initialized with the EPM values. The EPM's values are not affected by changing the variables in the user's program. When the user's program reads a variable it always reads from the RAM (volatile) copy of the variable. Communications Interface functions can change both the volatile and non-volatile copy of the variable. If the host interface requests a change to the EPM (non-volatile) value, this change is done both in the user program's RAM memory as well as in the EPM. Interface functions have the choice of reading from the RAM (volatile) or from the EPM (non-volatile) copy of the variable.

The column "**Access**" lists the user's access rights to a variable:

R read only
 W write only
 R/W read/write

Writing to an R (read-only) variable or reading from a W (write-only) variable will not work.

The column "**Units**" shows units of the variable. Units unique to this manual that are used for motion are:

UU user units
 EC encoder counts
 S seconds
 PPS pulses per sample. Sample time is 512µs - servo loop rate
 PPSS pulses per sample per sample. Sample time is 512µs - servo loop rate



Reference



NOTE:

In true Modbus, 3X and 4X Registers are numbered starting at 1. This is known as 'one based' addressing. However, when transmitted to a slave over the serial link, the actual address transmitted is one less.

Some Modbus masters will allow for the first register number to be 0. This is known as 'zero based' addressing. If this is the case, the Modbus register numbers listed in this manual must be offset by -1 to properly program a master using 'zero-based' addressing.

Index	Name	Type	Format	EPM	Access	Description	Range	Unit	RAM	RAM	EPM	EPM	RAM	EPM
									Registers 32bit Integer Access 4X Register #	Register 32bit Float Access 4X Register #	Reg Copy 32bit Integer Access 4X Register #	Reg Copy 32bit Float Access 4X Register #	Register 16bit Signed Integer 4X Register #	Register 16bit Signed Integer 4X Register #
1	VAR_IDSTRING			N	R	Drive's identification string			3	515	1027	1539	2050	2306
2	VAR_NAME			Y	R/W	Drive's symbolic name			5	517	1029	1541	2051	2307
3	VAR_SERIAL_NUMBER				R	Drive's serial number			7	519	1031	1543	2052	2308
4	VAR_MEM_INDEX				R/W	Position in RAM file	(0 - 32767)		9	521	1033	1545	2053	2309
5	VAR_MEM_VALUE				R/W	Value to be read or written to the RAM file			11	523	1035	1547	2054	2310
6	VAR_MEM_INDEX_INCREMENT				R/W	Holds value the MEM_INDEX will modify once the R/W operation is complete			13	525	1037	1549	2055	2311
7	VAR_VELOCITY_ACTUAL				R	Actual measured motor velocity NOTE: Only applicable to MVOB drives with Firmware 4.00 and higher.			15	527	1039	1551	2056	2312
8	VAR_RSVD_2								17	529	1041	1553	2057	2313
9	VAR_DEFAULT				R	Drive Default Settings			19	531	1043	1555	2058	2314
10	VAR_M_ID	mtr		Y	R/W*	Motor ID			21	533	1045	1557	2059	2315
11	VAR_M_MODEL	mtr		Y	R/W*	Motor model			23	535	1047	1559	2060	2316
12	VAR_M_VENDOR	mtr		Y	R/W*	Motor vendor			25	537	1049	1561	2061	2317
13	VAR_M_ESET	mtr		Y	R/W*	Motor Feedback Resolver: 'Positive for CW' 0 - none 1 - Positive for CW	0 - 1		27	539	1051	1563	2062	2318
14	VAR_M_HALLCODE	mtr		Y	R/W*	Hallcode index	0 - 5		29	541	1053	1565	2063	2319
15	VAR_M_HOFFSET	mtr		Y	R/W*	Reserved			31	543	1055	1567	2064	2320
16	VAR_M_ZOFFSET	mtr		Y	R/W*	Resolver Offset	0 - 360		33	545	1057	1569	2065	2321
17	VAR_M_ICTRL	mtr		Y	R/W*	Reserved			35	547	1059	1571	2066	2322
18	VAR_M_JM	mtr		Y	R/W*	Motor moment of inertia Jm	0 - 0.1	Kgm2	37	549	1061	1573	2067	2323
19	VAR_M_KE	mtr		Y	R/W*	Motor voltage or back EMF constant Ke	1 - 500	V/Krpm	39	551	1063	1575	2068	2324
20	VAR_M_KT	mtr		Y	R/W*	Motor torque or force constant Kt	0.01 - 10	Nm/A	41	553	1065	1577	2069	2325
21	VAR_M_LS	mtr		Y	R/W*	Motor phase-to-phase inductance Lm	0.1 - 500	mH	43	555	1067	1579	2070	2326
22	VAR_M_RS	mtr		Y	R/W*	Motor phase-to-phase resistance Rm	0.01 - 500	[Ohm]	45	557	1069	1581	2071	2327
23	VAR_M_MAXCURRENT	mtr		Y	R/W*	Motor's max current(RMS)	0.5 - 50	[A]mp	47	559	1071	1583	2072	2328
24	VAR_M_MAXVELOCITY	mtr		Y	R/W*	Motor's max velocity	500 - 20000	RPM	49	561	1073	1585	2073	2329
25	VAR_M_NPOLES	mtr		Y	R/W*	Motor's poles number	2 - 200		51	563	1075	1587	2074	2330
26	VAR_M_ENCODER	mtr		Y	R/W*	Encoder resolution	256 - 65536 * 12/Npoles	PPR	53	565	1077	1589	2075	2331
27	VAR_M_TERMVOLTAGE	mtr		Y	R/W*	Nominal Motor's terminal voltage	50 - 800	[V]olt	55	567	1079	1591	2076	2332
28	VAR_M_FEEDBACK	mtr		Y	R/W*	Feedback type 1 - Encoder; 2 - Resolver	1 - 2		57	569	1081	1593	2077	2333
29	VAR_ENABLE_SWITCH_TYPE		W	Y	R/W	Enable switch function 0 - inhibit only; 1 - Run	0 - 1	Bit	59	571	1083	1595	2078	2334
30	VAR_CURRENTLIMIT		F	Y	R/W	Current limit		[A]mp	61	573	1085	1597	2079	2335
31	VAR_PEAKCURRENTLIMIT16		F	Y	R/W	Peak current limit for 16kHz operation		[A]mp	63	575	1087	1599	2080	2336
32	VAR_PEAKCURRENTLIMIT		F	Y	R/W	Peak current limit for 8kHz operation		[A]mp	65	577	1089	1601	2081	2337
33	VAR_PWMFREQUENCY		W	Y	R/W	PWM frequency selection 0 - 16kHz; 1 - 8kHz	0 - 1		67	579	1091	1603	2082	2338

* These are all R/W variables but they only become active after variable 247 is set.

Reference










Index	Name	Type	Format	EPM	Access	Description	Range	Unit	RAM Registers 32bit Integer Access 4X Register #	RAM Register 32bit Float Access 4X Register #	EPM Reg Copy 32bit Integer Access 4X Register #	EPM Reg Copy 32bit Float Access 4X Register #	RAM Register 16bit Signed Integer 4X Register #	EPM INT16 4X Register #
34	VAR_DRIVEMODE		W	Y	R/W	Drive mode 0 - torque; 1 - velocity ; 2 - position	0 - 2		69	581	1093	1605	2083	2339
35	VAR_CURRENT_SCALE		F	Y	R/W	Analog input #1 current reference scale	Model Dependent	A/V	71	583	1095	1607	2084	2340
36	VAR_VELOCITY_SCALE	vel	F	Y	R/W	Analog input #1 velocity reference scale	-10000 to +10000	RPM/V	73	585	1097	1609	2085	2341
37	VAR_REFERENCE		W	Y	R/W	Reference selection 1 - internal source; 0 - external	1 - 0		75	587	1099	1611	2086	2342
38	VAR_STEPINPUTTYPE		W	Y	R/W	Selects how position reference inputs operating 0 - Quadrature inputs (A/B) 1 - Step & Direction	0 - 1		77	589	1101	1613	2087	2343
39	VAR_MOTORTHERMALPROTECT		W	Y	R/W	Motor thermal protection function 0 - disabled; 1 - enabled	0 - 1		79	591	1103	1615	2088	2344
40	VAR_MOTORPTCRESISTANCE		F	Y	R/W	Motor thermal protection PTC cut-off resistance		[Ohm]	81	593	1105	1617	2089	2345
41	VAR_SECONDCODER		W	Y	R/W	Second encoder 0 - disabled; 1 - enabled	0 - 1		83	595	1107	1619	2090	2346
42	VAR_REGENDUTY		W	Y	R/W	Regen circuit PWM duty cycle in %	1-100%	%	85	597	1109	1621	2091	2347
43	VAR_ENCODERREPEATSRC		W	Y	R/W	Selects source for repeat buffers 0 - Model 940 - Encoder Port P4 Model 941 - 2nd Encoder Option Bay 1 - Model 940 - 2nd Encoder Option Bay Model 941 - Resolver Port P4	0 - 1		87	599	1111	1623	2092	2348
44	VAR_VP_GAIN	vel	W	Y	R/W	Velocity loop Proportional gain	0 - 32767		89	601	1113	1625	2093	2349
45	VAR_VI_GAIN	vel	W	Y	R/W	Velocity loop Integral gain	0 - 32767		91	603	1115	1627	2094	2350
46	VAR_PP_GAIN		W	Y	R/W	Position loop Proportional gain	0 - 32767		93	605	1117	1629	2095	2351
47	VAR_PI_GAIN		W	Y	R/W	Position loop Integral gain	0 - 16383		95	607	1119	1631	2096	2352
48	VAR_PD_GAIN		W	Y	R/W	Position loop Differential gain	0 - 32767		97	609	1121	1633	2097	2353
49	VAR_PL_LIMIT		W	Y	R/W	Position loop integral gain limit	0 - 20000		99	611	1123	1635	2098	2354
50	VAR_SEI_GAIN					Not Used			101	613	1125	1637	2099	2355
51	VAR_VREG_WINDOW	vel	W	Y	R/W	Gains scaling coefficient	-16 to +4		103	615	1127	1639	2100	2356
52	VAR_ENABLE		W	N	W	Software Enable/Disable 0 - disable; 1 - enable	0 - 1		105	617	1129	1641	2101	2357
53	VAR_RESET		W	N	W	Drive's reset (Disables drive, Stops running program if any, reset active fault) 0 - no action; 1 - reset drive	0 - 1		107	619	1131	1643	2102	2358
54	VAR_STATUS		W	N	R	Drive's status register			109	621	1133	1645	2103	2359
55	VAR_BCF_SIZE		W	Y	R	User's program Byte-code size		Bytes	111	623	1135	1647	2104	2360
56	VAR_AUTOBOOT		W	Y	R/W	User's program autostart flag 0 - program started manually (MV or interface); 1 - program started automatically after drive booted	0 - 1		113	625	1137	1649	2105	2361
57	VAR_GROUPID		W	Y	R/W	Network group ID	1 - 32767		115	627	1139	1651	2106	2362
58	VAR_VLIMIT_ZEROSPEED		F	Y	R/W	Zero Speed window	0 - 100	Rpm	117	629	1141	1653	2107	2363
59	VAR_VLIMIT_SPEEDWND		F	Y	R/W	At Speed window	10 - 10000	Rpm	119	631	1143	1655	2108	2364
60	VAR_VLIMIT_ATSPEED		F	Y	R/W	Target Velocity for At Speed window	-10000 - +10000	Rpm	121	633	1145	1657	2109	2365
61	VAR_PLIMIT_POSEERROR		W	Y	R/W	Position error	1 - 32767	EC	123	635	1147	1659	2110	2366
62	VAR_PLIMIT_ERRORTIME		F	Y	R/W	Position error time (time which position error has to remain to set-off position error fault)	0.25 - 8000	mS	125	637	1149	1661	2111	2367
63	VAR_PLIMIT_SEPOSEERROR		W	Y	R/W	Second encoder Position error	1 - 32767	EC	127	639	1151	1663	2112	2368
64	VAR_PLIMIT_SEERRORTIME		F	Y	R/W	Second encoder Position error time (time which position error has to remain to set-off position error fault)	0.25 - 8000	mS	129	641	1153	1665	2113	2369
65	VAR_INPUTS		W	N	R	Digital inputs states. A1 occupies Bit 0, A2- Bit 1 ... C4 - bit 11.			131	643	1155	1667	2114	2370



Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Unit	RAM	RAM	EPM	EPM	RAM	EPM
									Registers 32bit Integer Access 4X Register #	Register 32bit Float Access 4X Register #	Reg Copy 32bit Integer Access 4X Register #	Reg Copy 32bit Float Access 4X Register #	Register 16bit Signed Integer 4X Register #	Register 4X Register #
66	VAR_OUTPUT		W	N	R/W	Digital outputs states. Writing to this variables sets/resets digital outputs except outputs which have been assigned special function. Output 1 Bit 0 Output 2 Bit 1 Output 3 Bit 2 Output 4 Bit 3	Output 1 - Output 4		133	645	1157	1669	2115	2371
67	VAR_IP_ADDRESS		W	Y	R/W	Ethernet IP address. IP address changes at next boot up. 32 bit value			135	647	1159	1671	2116	2372
68	VAR_IP_MASK		W	Y	R/W	Ethernet IP NetMask. Mask changes at next boot up. 32 bit value			137	649	1161	1673	2117	2373
69	VAR_IP_GATEWAY		W	Y	R/W	Ethernet Gateway IP address. Address changes at next boot up. 32 bit value			139	651	1163	1675	2118	2374
70	VAR_IP_DHCP		W	Y	R/W	Use DHCP 0 - manual; 1 - use DHCP service	0 - 1		141	653	1165	1677	2119	2375
71	VAR_AIN1		F	N	R	Analog Input AIN1 current value		[V]olt	143	655	1167	1679	2120	2376
72	VAR_AIN2		F	N	R	Analog Input AIN2 current value		[V]olt	145	657	1169	1681	2121	2377
73	VAR_BUSVOLTAGE		F	N	R	Bus voltage		[V]olt	147	659	1171	1683	2122	2378
74	VAR_HTEMP		F	N	R	Heatsink temperature 0 - for temperatures < 40C and actual heat sink temperature for temperatures >40 C	0 - actual heat sink temperature	[c]	149	661	1173	1685	2123	2379
75	VAR_ENABLE_ACCELDECEL	vel		Y	R/W	Enable Accel/Decel function for velocity mode 0 - disable; 1 - enable	0 - 1		151	663	1175	1687	2124	2380
76	VAR_ACCEL_LIMIT	vel	F	Y	R/W	Accel value for velocity mode	0.1 - 5000000	Rpm*Sec	153	665	1177	1689	2125	2381
77	VAR_DECEL_LIMIT	vel	F	Y	R/W	Decel value for velocity mode	0.1 - 5000000	Rpm*Sec	155	667	1179	1691	2126	2382
78	VAR_FAULT_RESET		W	Y	R/W	Reset fault configuration 0 - on activation of Enable/Inhibit input (A3) 1 - on deactivation of Enable/Inhibit input (A3)	0 - 1		157	669	1181	1693	2127	2383
79	VAR_M2SRATIO_MASTER		W	Y	R/W	Master to system ratio Master counts range: -32767 - +32767	-32767 - +32767		159	671	1183	1695	2128	2384
80	VAR_M2SRATIO_SYSTEM		W	Y	R/W	Master to system ratio System counts range: 1 - 32767	1 - 32767		161	673	1185	1697	2129	2385
81	VAR_S2PRATIO_SECOND		W	Y	R/W	Secondary encoder to prime encoder ratio Second counts range: -32767 - +32767	-32767 - +32767		163	675	1187	1699	2130	2386
82	VAR_S2PRATIO_PRIME		W	Y	R/W	Secondary encoder to prime encoder ratio Prime counts range: 1 - 32767	1 - 32767		165	677	1189	1701	2131	2387
83	VAR_EXSTATUS		W	N	R	Extended status. Lower word copy of DSP status flags.			167	679	1191	1703	2132	2388
84	VAR_HLS_MODE		W	Y	R/W	Hardware limit switches 0 - not used; 1 - stop and fault; 2 - fault	0 - 2		169	681	1193	1705	2133	2389
85	VAR_AOUT_FUNCTION		W	Y	R/W	Analog output function range: 0 - 8 0 - Not assigned 1 - Phase Current (RMS) 2 - Phase Current (Peak Value) 3 - Motor Velocity 4 - Phase Current R 5 - Phase Current S 6 - Phase Current T 7 - Iq current 8 - Id current	0 - 8		171	683	1195	1707	2134	2390
86	VAR_AOUT_VELSCALE		F	Y	R/W	Analog output scale for velocity quantities.	0 - 10	mV/Rpm	173	685	1197	1709	2135	2391
87	VAR_AOUT_CURSCALE		F	Y	R/W	Analog output scale for current related quantities.	0 - 10	V/A	175	687	1199	1711	2136	2392
88	VAR_AOUT		F	N	W	Analog output value.(Used if VAR #85 is set to 0)	0 - 10	V	177	689	1201	1713	2137	2393
89	VAR_AIN1_DEADBAND		F	Y	R/W	Analog input #1 dead-band. Applied when used as current or velocity reference.	0 - 100	mV	179	691	1203	1715	2138	2394
90	VAR_AIN1_OFFSET			Y	R/W	Analog input #1 offset. Applied when used as current/velocity reference	-10,000 to +10,000	mV	181	693	1205	1717	2139	2395



Index	Name	Type	Format	EPM	Access	Description	Range	Unit	RAM Registers 32bit Integer Access 4X Register #	RAM Register 32bit Float Access 4X Register #	EPM Reg Copy 32bit Integer Access 4X Register #	EPM Reg Copy 32bit Float Access 4X Register #	RAM Register 16bit Signed Integer 4X Register #	EPM INT16 4X Register #
91	VAR_SUSPEND_MOTION		W	N	R/W	Suspend motion. Suspends motion produced by trajectory generator. Current move will be completed before motion is suspended. 0 - motion suspended; 1 - motion resumed	0 - 1		183	695	1207	1719	2140	2396
92	VAR_MOVEP		W	N	W	Target position for absolute move. Writing value executes Move to position as per MOVEP statement using current values of acceleration, deceleration and max velocity.			185	697	1209	1721	2141	2397
93	VAR_MOVED		W	N	W	Incremental position. Writing value <0> executes Incremental move as per MOVED statement using current values of acceleration, deceleration and max velocity			187	699	1211	1723	2142	2398
94	VAR_MDV_DISTANCE		F	N	W	Distance for MDV move			189	701	1213	1725	2143	2399
95	VAR_MDV_VELOCITY		F	N	W	Velocity for MDV move. Writing to this variable executes MDV move with Distance value last written to variable #94			191	703	1215	1727	2144	2400
96	VAR_MOVE_PW1		W	N	W	Writing value executes Move in positive direction while input true (active). Value specifies input # 0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	0 - 11		193	705	1217	1729	2145	2401
97	VAR_MOVE_PW0		W	N	W	Writing value executes Move in positive direction while input false (not active). Value specifies input # 0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	0 - 11		195	707	1219	1731	2146	2402
98	VAR_MOVE_NW1		F	N	W	Writing value executes Move negative direction while input true (active). Value specifies input # 0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	0 - 11		197	709	1221	1733	2147	2403
99	VAR_MOVE_NW0		F	N	W	Writing value executes Move negative direction while input false (not active). Value specifies input # 0 - 3 : A1 -A4 4 - 7 : B1 - B4 8 - 11 : C1 - C4	0 - 11		199	711	1223	1735	2148	2404
100	VAR_V0		F	Y	R/W	User variable			201	713	1225	1737	2149	2405
101	VAR_V1		F	Y	R/W	User variable			203	715	1227	1739	2150	2406
102	VAR_V2		F	Y	R/W	User variable			205	717	1229	1741	2151	2407
103	VAR_V3		F	Y	R/W	User variable			207	719	1231	1743	2152	2408
104	VAR_V4		F	Y	R/W	User variable			209	721	1233	1745	2153	2409
105	VAR_V5		F	Y	R/W	User variable			211	723	1235	1747	2154	2410
106	VAR_V6		F	Y	R/W	User variable			213	725	1237	1749	2155	2411
107	VAR_V7		F	Y	R/W	User variable			215	727	1239	1751	2156	2412
108	VAR_V8		F	Y	R/W	User variable			217	729	1241	1753	2157	2413
109	VAR_V9		F	Y	R/W	User variable			219	731	1243	1755	2158	2414
110	VAR_V10		F	Y	R/W	User variable			221	733	1245	1757	2159	2415
111	VAR_V11		F	Y	R/W	User variable			223	735	1247	1759	2160	2416
112	VAR_V12		F	Y	R/W	User variable			225	737	1249	1761	2161	2417
113	VAR_V13		F	Y	R/W	User variable			227	739	1251	1763	2162	2418
114	VAR_V14		F	Y	R/W	User variable			229	741	1253	1765	2163	2419
115	VAR_V15		F	Y	R/W	User variable			231	743	1255	1767	2164	2420
116	VAR_V16		F	Y	R/W	User variable			233	745	1257	1769	2165	2421
117	VAR_V17		F	Y	R/W	User variable			235	747	1259	1771	2166	2422
118	VAR_V18		F	Y	R/W	User variable			237	749	1261	1773	2167	2423
119	VAR_V19		F	Y	R/W	User variable			239	751	1263	1775	2168	2424
120	VAR_V20		F	Y	R/W	User variable			241	753	1265	1777	2169	2425



Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Unit	RAM	RAM	EPM	EPM	RAM	EPM
									Registers 32bit Integer Access 4X Register #	Register 32bit Float Access 4X Register #	Reg Copy 32bit Integer Access 4X Register #	Reg Copy 32bit Float Access 4X Register #	Register 16bit Signed Integer 4X Register #	Register #
121	VAR_V21		F	Y	R/W	User variable			243	755	1267	1779	2170	2426
122	VAR_V22		F	Y	R/W	User variable			245	757	1269	1781	2171	2427
123	VAR_V23		F	Y	R/W	User variable			247	759	1271	1783	2172	2428
124	VAR_V24		F	Y	R/W	User variable			249	761	1273	1785	2173	2429
125	VAR_V25		F	Y	R/W	User variable			251	763	1275	1787	2174	2430
126	VAR_V26		F	Y	R/W	User variable			253	765	1277	1789	2175	2431
127	VAR_V27		F	Y	R/W	User variable			255	767	1279	1791	2176	2432
128	VAR_V28		F	Y	R/W	User variable			257	769	1281	1793	2177	2433
129	VAR_V29		F	Y	R/W	User variable			259	771	1283	1795	2178	2434
130	VAR_V30		F	Y	R/W	User variable			261	773	1285	1797	2179	2435
131	VAR_V31		F	Y	R/W	User variable			263	775	1287	1799	2180	2436
132	VAR_MOVEDR_DISTANCE		F	N	W	Registered move distance. Incremental motion as per MOVEDR statement		UU	265	777	1289	1801	2181	2437
133	VAR_MOVEDR_DISPLACEMENT		F	N	W	Registered move displacement. Writing to this variable executes the move MOVEDR using value set by #132		UU	267	779	1291	1803	2182	2438
134	VAR_MOVEPR_DISTANCE		F	N	W	Registered move distance. Absolute motion as per MOVEPR statement		UU	269	781	1293	1805	2183	2439
135	VAR_MOVEPR_DISPLACEMENT		F	N	W	Registered move displacement. Writing to this variable makes the move MOVEPR using value set by #134		UU	271	783	1295	1807	2184	2440
136	VAR_STOP_MOTION			W	N	W	Stops motion 0 - no action; 1 - stops motion	0 - 1	273	785	1297	1809	2185	2441
137	VAR_START_PROGRAM			W	N	W	Starts user program 0 - no action; 1 - starts program	0 - 1	275	787	1299	1811	2186	2442
138	VAR_VEL_MODE_ON			W	N	W	Turns on Profile Velocity for Internal Position Mode 0 - normal operation; 1 - velocity mode on	0 - 1	277	789	1301	1813	2187	2443
139	VAR_IREF		F	N	W	Reference: Internal Torque or Velocity Mode		RPS Amps	279	791	1303	1815	2188	2444
140	VAR_NV0		F	N	R/W	User defined Network variable			281	793	1305	1817	2189	2445
141	VAR_NV1		F	N	R/W	User defined Network variable			283	795	1307	1819	2190	2446
142	VAR_NV2		F	N	R/W	User defined Network variable			285	797	1309	1821	2191	2447
143	VAR_NV3		F	N	R/W	User defined Network variable			287	799	1311	1823	2192	2448
144	VAR_NV4		F	N	R/W	User defined Network variable			289	801	1313	1825	2193	2449
145	VAR_NV5		F	N	R/W	User defined Network variable			291	803	1315	1827	2194	2450
146	VAR_NV6		F	N	R/W	User defined Network variable			293	805	1317	1829	2195	2451
147	VAR_NV7		F	N	R/W	User defined Network variable			295	807	1319	1831	2196	2452
148	VAR_NV8		F	N	R/W	User defined Network variable			297	809	1321	1833	2197	2453
149	VAR_NV9		F	N	R/W	User defined Network variable			299	811	1323	1835	2198	2454
150	VAR_NV10		F	N	R/W	User defined Network variable			301	813	1325	1837	2199	2455
151	VAR_NV11		F	N	R/W	User defined Network variable			303	815	1327	1839	2200	2456
152	VAR_NV12		F	N	R/W	User defined Network variable			305	817	1329	1841	2201	2457
153	VAR_NV13		F	N	R/W	User defined Network variable			307	819	1331	1843	2202	2458
154	VAR_NV14		F	N	R/W	User defined Network variable			309	821	1333	1845	2203	2459
155	VAR_NV15		F	N	R/W	User defined Network variable			311	823	1335	1847	2204	2460
156	VAR_NV16		F	N	R/W	User defined Network variable			313	825	1337	1849	2205	2461
157	VAR_NV17		F	N	R/W	User defined Network variable			315	827	1339	1851	2206	2462
158	VAR_NV18		F	N	R/W	User defined Network variable			317	829	1341	1853	2207	2463
159	VAR_NV19		F	N	R/W	User defined Network variable			319	831	1343	1855	2208	2464
160	VAR_NV20		F	N	R/W	User defined Network variable			321	833	1345	1857	2209	2465
161	VAR_NV21		F	N	R/W	User defined Network variable			323	835	1347	1859	2210	2466

Reference



Index	Name	Type	Format	EPM	Access	Description	Range	Unit	RAM	RAM	EPM	EPM	RAM	EPM
									Registers 32bit Integer Access 4X Register #	Register 32bit Float Access 4X Register #	Reg Copy 32bit Integer Access 4X Register #	Reg Copy 32bit Float Access 4X Register #	Register 16bit Signed Integer 4X Register #	Register 4X Register #
162	VAR_NV22		F	N	R/W	User defined Network variable			325	837	1349	1861	2211	2467
163	VAR_NV23		F	N	R/W	User defined Network variable			327	839	1351	1863	2212	2468
164	VAR_NV24		F	N	R/W	User defined Network variable			329	841	1353	1865	2213	2469
165	VAR_NV25		F	N	R/W	User defined Network variable			331	843	1355	1867	2214	2470
166	VAR_NV26		F	N	R/W	User defined Network variable			333	845	1357	1869	2215	2471
167	VAR_NV27		F	N	R/W	User defined Network variable			335	847	1359	1871	2216	2472
168	VAR_NV28		F	N	R/W	User defined Network variable			337	849	1361	1873	2217	2473
169	VAR_NV29		F	N	R/W	User defined Network variable			339	851	1363	1875	2218	2474
170	VAR_NV30		F	N	R/W	User defined Network variable			341	853	1365	1877	2219	2475
171	VAR_NV31		F	N	R/W	User defined Network variable			343	855	1367	1879	2220	2476
172	VAR_SERIAL_ADDRESS		W	Y	R/W	RS485 drive ID	0 - 254		345	857	1369	1881	2221	2477
173	VAR_MODBUS_BAUDRATE		W	Y	R/W	Baud rate for ModBus operations 2 - 9600; 3 - 19200 4 - 38400; 5 - 57600; 6 - 115200	2 - 6		347	859	1371	1883	2222	2478
174	VAR_MODBUS_DELAY		W	Y	R/W	ModBus reply delay in mS	0 - 1000		349	861	1373	1885	2223	2479
175	VAR_RS485_CONFIG		W	Y	R/W	Rs485 configuration 0 - normal IP over PPP; 1 - ModBus1 19200	0 - 1		351	863	1375	1887	2224	2480
176	VAR_PPP_BAUDRATE		W	Y	R/W	RS232/485 (normal mode) baud rate 2 - 9600; 3 - 19200 4 - 38400; 5 - 57600; 6 - 115200	2 - 6		353	865	1377	1889	2225	2481
177	VAR_MOVEPS		F	N	W	Same as variable #92 but using S-curve acceleration/deceleration			355	867	1379	1891	2226	2482
178	VAR_MOVEDS		F	N	W	Same as variable #93 but using S-curve acceleration/deceleration			357	869	1381	1893	2227	2483
179	VAR_MDVS_VELOCITY			N	W	Velocity for MDV move using S-curve accel/deceleration. Writing to this variable executes MDV move with Distance value last written to variable #94 (unless motion is suspended by #91).			359	871	1383	1895	2228	2484
180	VAR_MAXVEL		F	N	R/W	Max velocity for motion profile			361	873	1385	1897	2229	2485
181	VAR_ACCEL		F	N	R/W	Accel value for indexing		UU/S2	363	875	1387	1899	2230	2486
182	VAR_DECEL		F	N	R/W	Decel value for indexing		UU/S2	365	877	1389	1901	2231	2487
183	VAR_QDECEL		F	N	R/W	Quick decel value		UU/S2	367	879	1391	1903	2232	2488
184	VAR_INPOSLIM		W	N	R/W	Sets window for "In Position" Limits		UU	369	881	1393	1905	2233	2489
185	VAR_VEL		F	N	R/W	Velocity reference for "Profiled" velocity		UU/S	371	883	1395	1907	2234	2490
186	VAR_UNITS		F	Y	R/W	User units			373	885	1397	1909	2235	2491
187	VAR_MECCOUNTER		W	N	R/W	A/B inputs reference counter value		Count	375	887	1399	1911	2236	2492
188	VAR_PHCUR		F	N	R	Phase current		A	377	889	1401	1913	2237	2493
189	VAR_POS_PULSES		W	N	R/W	Target position in encoder pulses		EC	379	891	1403	1915	2238	2494
190	VAR_APOS_PULSES		W	N	R/W	Actual position in encoder pulses		EC	381	893	1405	1917	2239	2495
191	VAR_POSEERROR_PULSES		W	N	R	Position error in encoder pulses		EC	383	895	1407	1919	2240	2496
192	VAR_CURRENT_VEL_PPS		F	N	R	Set-point (target) velocity in PPS		PPS	385	897	1409	1921	2241	2497
193	VAR_CURRENT_ACCEL_PPSS		F	N	R	Set-point (target) acceleration (demanded value)		PPSS	387	899	1411	1923	2242	2498
194	VAR_IN0_DEBOUNCE		W	Y	R/W	Input A1 de-bounce time in mS	0 - 1000	mS	389	901	1413	1925	2243	2499
195	VAR_IN1_DEBOUNCE		W	Y	R/W	Input A2 de-bounce time in mS	0 - 1000	mS	391	903	1415	1927	2244	2500
196	VAR_IN2_DEBOUNCE		W	Y	R/W	Input A3 de-bounce time in mS	0 - 1000	mS	393	905	1417	1929	2245	2501
197	VAR_IN3_DEBOUNCE		W	Y	R/W	Input A4 de-bounce time in mS	0 - 1000	mS	395	907	1419	1931	2246	2502
198	VAR_IN4_DEBOUNCE		W	Y	R/W	Input B1 de-bounce time in mS	0 - 1000	mS	397	909	1421	1933	2247	2503
199	VAR_IN5_DEBOUNCE		W	Y	R/W	Input B2 de-bounce time in mS	0 - 1000	mS	399	911	1423	1935	2248	2504
200	VAR_IN6_DEBOUNCE		W	Y	R/W	Input B3 de-bounce time in mS	0 - 1000	mS	401	913	1425	1937	2249	2505
201	VAR_IN7_DEBOUNCE		W	Y	R/W	Input B4 de-bounce time in mS	0 - 1000	mS	403	915	1427	1939	2250	2506
202	VAR_IN8_DEBOUNCE		W	Y	R/W	Input C1 de-bounce time in mS	0 - 1000	mS	405	917	1429	1941	2251	2507



Reference

Index	Name	Type	Format	EPM	Access	Description	Range	Unit	RAM Registers 32bit Integer Access 4X Register #	RAM Register 32bit Float Access 4X Register #	EPM Reg Copy 32bit Integer Access 4X Register #	EPM Reg Copy 32bit Float Access 4X Register #	RAM Register 16bit Signed Integer 4X Register #	EPM INT16 4X Register #
203	VAR_IN9_DEBOUNCE		W	Y	R/W	Input C2 de-bounce time in mS	0 - 1000	mS	407	919	1431	1943	2252	2508
204	VAR_IN10_DEBOUNCE		W	Y	R/W	Input C3 de-bounce time in mS	0 - 1000	mS	409	921	1433	1945	2253	2509
205	VAR_IN11_DEBOUNCE		W	Y	R/W	Input C4 de-bounce time in mS	0 - 1000	mS	411	923	1435	1947	2254	2510
206	VAR_OUT1_FUNCTION		W	Y	R/W	Programmable Output 1 Function 0 - Not Assigned 1 - Zero Speed 2 - In Speed Window 3 - Current Limit 4 - Run time fault 5 - Ready 6 - Brake 7 - In position	0 - 7		413	925	1437	1949	2255	2511
207	VAR_OUT2_FUNCTION		W	Y	R/W	Programmable Output 2 Function	0 - 7		415	927	1439	1951	2256	2512
208	VAR_OUT3_FUNCTION		W	Y	R/W	Programmable Output 3 Function	0 - 7		417	929	1441	1953	2257	2513
209	VAR_OUT4_FUNCTION		W	Y	R/W	Programmable Output 4 Function	0 - 7		419	931	1443	1955	2258	2514
210	VAR_HALLCODE		W	N	R	Current hall code Bit 0 - Hall 1 Bit 1 - Hall 2 Bit 2 - Hall 3			421	933	1445	1957	2259	2515
211	VAR_ENCODER		W	N	R	Primary encoder current value		EC	423	935	1447	1959	2260	2516
212	VAR_RPOS_PULSES		W	N	R	Registration position		EC	425	937	1449	1961	2261	2517
213	VAR_RPOS		F	N	R	Registration position		UU	427	939	1451	1963	2262	2518
214	VAR_POS		F	N	R/W	Target position		UU	429	941	1453	1965	2263	2519
215	VAR_APOS		F	N	R/W	Actual position		UU	431	943	1455	1967	2264	2520
216	VAR_POSEERROR		W	N	R	Position error		EC	433	945	1457	1969	2265	2521
217	VAR_CURRENT_VEL		F	N	R	Set-point (target) velocity (demanded value)		UU/S	435	947	1459	1971	2266	2522
218	VAR_CURRENT_ACCEL		F	N	R	Set-point (target) acceleration (demanded value)		UU/S2	437	949	1461	1973	2267	2523
219	VAR_TPOS_ADVANCE		W	N	W	Target position advance. Every write to this variable adds value to the Target position summing point. Value gets added once per write. This variable useful when loop is driven by Master encoder signals and trying to correct phase. Value is in encoder counts		EC	439	951	1463	1975	2268	2524
220	VAR_IOINDEX		W	N	R/W	Same as INDEX variable in user's program.			441	953	1465	1977	2269	2525
221	VAR_PSLIMIT_PULSES		W	Y	R/W	Positive Software limit switch value in Encoder counts		EC	443	955	1467	1979	2270	2526
222	VAR_NSLIMIT_PULSES		W	Y	R/W	Negative Software limit switch value in Encoder counts		EC	445	957	1469	1981	2271	2527
223	VAR_SLS_MODE		W	Y	R/W	Soft limit switch action code: 0 - no action 1 - Fault 2 - Stop and fault (When loop is driven by trajectory generator only. With all other sources same action as 1)"	0 - 2		447	959	1471	1983	2272	2528
224	VAR_PSLIMIT		F	Y	R/W	Same as var 221 but value in User Units		UU	449	961	1473	1985	2273	2529
225	VAR_NSLIMIT		F	Y	R/W	Same as var 222 but value in User Units		UU	451	963	1475	1987	2274	2530
226	VAR_SE_APOS_PULSES		W	N	R	2nd encoder actual position in encoder counts		EC	453	965	1477	1989	2275	2531
227	VAR_SE_POSEERROR_PULSES		W	N	R	2nd encoder position error in encoder counts		EC	455	967	1479	1991	2276	2532
228	VAR_MODBUS_PARITY		W	Y	R/W	Parity for Modbus Control: 0 - No Parity; 1 - Odd Parity; 2 - Even Parity	0 - 2		457	969	1481	1993	2277	2533
229	VAR_MODBUS_STOPBITS		W	Y	R/W	Number of Stopbits for Modbus Control 0 - 1.0; 1 - 1.5; 2 - 2.0	0 - 2		459	971	1483	1995	2278	2534
230	VAR_M_NOMINALVEL		F	Y	R/W	Induction Motor Nominal Velocity	500 - 20000	RPM	461	973	1485	1997	2279	2535
231	VAR_M_COSPHI		F	Y	R/W	Induction Motor Cosine Phi	0 - 1.0		463	975	1487	1999	2280	2536
232	VAR_M_BASEFREQUENCY		F	Y	R/W	Induction Motor Base Frequency	0 - 400Hz	Hz	465	977	1489	2001	2281	2537
233	VAR_M_SERIES					Induction Motor Series			467	979	1491	2003	2282	2538

Reference



Index	Name	Type	Format	EPM	Access	Description	Range	Unit	RAM Registers 32bit Integer Access 4X Register #	RAM Register 32bit Float Access 4X Register #	EPM Reg Copy 32bit Integer Access 4X Register #	EPM Reg Copy 32bit Float Access 4X Register #	RAM Register 16bit Signed Integer 4X Register #	EPM INT16 4X Register #
234	VAR_CAN_BAUD_EPM		W	Y	R/W	CAN Bus Parameter: Baud Rate: 1 - 8 1 - 10k; 2 - 20k; 3 - 50k; 4 - 125k 5 - 250k; 6 - 500k; 7 - 800k; 8 - 1000k	1 - 8		469	981	1493	2005	2283	2539
235	VAR_CAN_ADDR_EPM		W	Y	R/W	CAN Bus Parameter: Address	1-127		471	983	1495	2007	2284	2540
236	VAR_CAN_OPERMODE_EPM		W	Y	R/W	CAN Bus Parameter: Boot-up Mode (Operational State Control) 0 - enters into pre-operational state 1 - enters into operational state 2 - pseudo NMT: sends NMT Start Node command after delay (set by variable 237)	0 - 2		473	985	1497	2009	2285	2541
237	VAR_CAN_OPERDELAY_EPM		W	Y	R/W	CAN Bus Parameter: pseudo NMT mode delay time in seconds	Refer to variable 236	sec	475	987	1499	2011	2286	2542
238	VAR_CAN_ENABLE_EPM		W	Y	R/W	CAN Bus Parameter: Mode Control 0 - Disable CAN interface 1 - Enable CAN interface in DS301 mode 2 - Enable CAN interface in DS402 mode 3 - Enable DeviceNet 4 - Enable PROFIBUS DP	0 - 4		477	989	1501	2013	2287	2543
239	VAR_HOME_ACCEL		F	Y	R/W	Homing Mode: ACCEL rate	0 - 10000000.0	UU/sec2	479	991	1503	2015	2288	2544
240	VAR_HOME_OFFSET		F	Y	R/W	Homing Mode: Home Position Offset	-32767 to +32767	UU	481	993	1505	2017	2289	2545
241	VAR_HOME_OFFSET_PULSES		W	Y	R/W	Homing Mode: Home Position Offset in encoder counts	+/- 2147418112	EC	483	995	1507	2019	2290	2546
242	VAR_HOME_FAST_VEL		F	Y	R/W	Homing Mode: Fast Velocity	-10000 to +10000	UU/sec	485	997	1509	2021	2291	2547
243	VAR_HOME_SLOW_VEL		F	Y	R/W	Homing Mode: Slow Velocity	-10000 to +10000	UU/sec	487	999	1511	2023	2292	2548
244	VAR_HOME_METHOD		W	Y	R/W	Homing Mode: Homing Method	1 - 35		489	1001	1513	2025	2293	2549
245	VAR_START_HOMING		W	N	W	Homing Mode: Start Homing 0 - No action; 1 - Start Homing	0 - 1		491	1003	1515	2027	2294	2550
246	VAR_HOME_SWITCH_INPUT		W	Y	R/W	Homing Mode: Switch Input Assignment: 0-3: A1-A4 4-7: B1-B4 8-11: C1-C4	0-11		493	1005	1517	2029	2295	2551
247	VAR_M_VALIDATE_MOTOR		W	N	W	Makes Drive accept Motor's parameters 0 - No action 1 - Validate Motor Data	0 - 1		495	1007	1519	2031	2296	2552
248	VAR_M_I2T		F	Y	R/W	Motor			497	1009	1521	2033	2297	2553
249	VAR_M_EABSOLUTE		F	Y	R/W	Motor			499	1011	1523	2035	2298	2554
250	VAR_M_ABSWAP		F	Y	R/W	Motor Encoder Feedback: B leads A 0 - No Action 1 - B leads A for forward checked (active)	0 - 1		501	1013	1525	2037	2299	2555
251	VAR_M_HALLS_INVERTED		F	Y	R/W	Motor Encoder Feedback: Halls 0 - No Action 1 - Inverted Halls Box checked (active)	0 - 1		503	1015	1527	2039	2300	2556
252	RESERVED					Do NOT use			505	1017	1529	2041	2301	2557
253	RESERVED					Do NOT use			507	1019	1531	2043	2302	2558
254	RESERVED					Do NOT use			509	1021	1533	2045	2303	2559
255	RESERVED					Do NOT use			511	1023	1535	2047	2304	2560
256	RESERVED					Do NOT use			513	1025	1537	2049	2305	2561

This is a condensed PID List to show the corresponding Modbus 4X Registers for PIDs 1-256. Modbus RTU can not access beyond PID256. For the complete variable list refer to the PositionServo Programming Manual (PM94P01 or PM94M01).

Lenze AC Tech Corporation

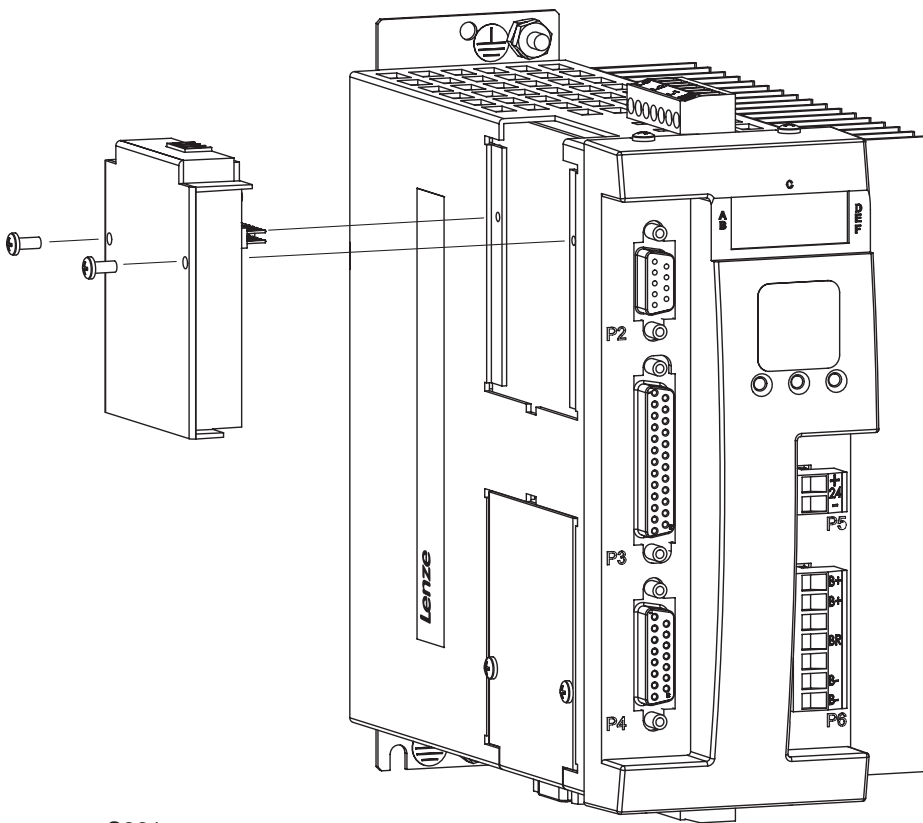
630 Douglas Street • Uxbridge, MA 01569 • USA

Sales: 800 217 9100 • Service: 508 278-9100

www.lenze-actech.com

P94MOD01D

Lenze



S921

RS-485 module Modbus

S94MOD01A

Table of Contents

1	Preface and General Information	1
2	Safety Information	2
3	Technical data	3
3.1	Related documents	3
3.2	General Modbus protocol description	3
3.3	Configuration	4
3.4	Supported Modbus features.	4
4	Parameter setting.	5
4.1	Register Memory	5
4.1.1	Read Only or Input Registers	5
4.1.2	Read/Write or Holding Registers	5
4.2	Discrete Memory	8
4.2.1	Discrete Inputs (Read Only Bits)	8
4.2.2	Coils (Read/Write Bits)	8



1 Preface and General Information

1.1 How to use these Operating Instructions

- These Operating Instructions are intended for safety-relevant operation on and with the module. They contain safety information which must be observed.
- All personnel working on and with the module must have these Operating Instructions available and observe the information and notes relevant for them.
- These instructions are only valid in combination with the Operating Instructions of the corresponding controller. They must always be complete and in a perfectly readable state.



2 Safety Information

2.1 Persons responsible for safety

Operator

- An operator is any natural or legal person who uses the drive system or on behalf of whom the drive system is used.
- The operator or his safety personnel is obliged to ensure
 - the compliance with all relevant regulations, instructions, and legislation.
 - that only skilled personnel works on and with the drive system.
 - that the personnel has the Operating Instructions available for all corresponding works.
 - that all unqualified personnel are prohibited from working on and with the drive system.

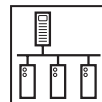
Qualified personnel

Qualified personnel are persons who - because of their education, experience, instructions, and knowledge about corresponding standards and regulations, rules for the prevention of accidents, and operating conditions - are authorized by the person responsible for the safety of the plant to perform the required actions and who are able to recognize potential hazards.

(Definition for qualified personnel to VDE 105 or IEC 364)

2.2 General safety information

- These safety notes do claim to be complete. In case of questions and problems please contact your Lenze representative.
- At the time of delivery the drive system meets the state of the art and ensures basically safe operation.
- The indications given in these Operating Instructions refer to the stated hardware and software versions of the controller.
- The controller is hazardous if:
 - unqualified personnel works on and with the controller.
 - the controller is used inappropriately.
- Ensure by appropriate measures that neither personal injury nor damage to property may occur in the event of failure of the drive system.
- The drive system must only be operated when no faults occur.
- Retrofitting, modifications, or redesigns are basically prohibited. Lenze must be contacted in all cases.



3 Technical data

3.1 Related documents

- MODBUS Application Protocol Specification V1.1
It can be found at: <http://www.modbus.org/default.htm>
- MODBUS over Serial Line Specification & Implementation guide V1.0

3.2 General Modbus protocol description

The MODBUS protocol defines a simple protocol data unit (PDU) independent of the underlying communication layers.

There are some additional data unit (ADU) fields introduced by the network layer

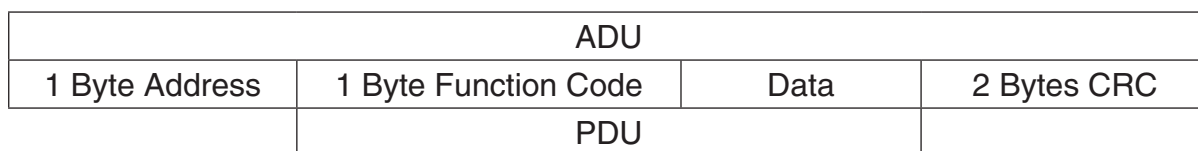


Figure 3: RS485 Network MODBUS frame

The client (also named 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 Client to a Server device the function code field tells the server what kind of action to perform.

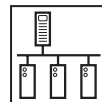
Sub-function codes are added to some function codes to define multiple actions.

The data field of messages sent from a client 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 items to be handled, and the count of actual data bytes in the field.

The data field may be nonexistent (of zero length) in certain kinds request, 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 client 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 client 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 client, 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.



3.3 Configuration

Drive 94 supports Modbus communication protocol through its RS485 optional card.

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 works in PPP mode.

Modbus baud rate – the RS485 baud rate in modbus mode. Note that modbus requires 2 stop bits and no parity.

Modbus reply delay – The delay introduced after receiving 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 might be required to successfully work with these devices.

Each 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.

Drive 94 is Modbus slave devices in a single master multiple (or one) slaves network. (Sometimes Slave devices are named servers because they wait for requests.)

Drive 94 communicates on the other end with Modbus Master.

Most Terminals can be configured as Masters with a Modbus Generic driver.

3.4 Supported Modbus features

Drive 94 uses the Modbus protocol delay of 3.5 characters time between packets to determine the end of packet.

The following ModBus functions are supported:

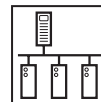
- 01 (0x01) Read Coils
- 02 (0x02) Read Discrete Inputs
- 03 (0x03) Read Holding Registers
- 04 (0x04) Read Input Registers
- 05 (0x05) Write Single Coil
- 06 (0x06) Write Single Register
- 15 (0x0F) Write Multiple Coils
- 16 (0x10) Write Multiple Registers

When the drive receives ModBus function it executes it and replies with Normal message, Exception Message or No message as described by the Modbus Application protocol Specification V1.1.

Modbus broadcast address is supported by the drive 94 also. In this case after receiving correct Modbus request the drive executes it but does not send any reply back.

The DWORD (double word) and FLOAT numbers are send with the LOW WORD FIRST convention by the 94 drive.

Note that some Terminals may have to be configured appropriately in order to accept such a WORD order.



4 Parameter setting

4.1 Register Memory

4.1.1 Read Only or Input Registers

The following read only ModBus registers (drive variables) accessible only through the ModBus function

- 03 (0x03) Read Holding Registers

4.1.2 Read/Write or Holding Registers

The following read/write ModBus registers (drive variables) accessible only through the ModBus functions

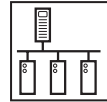
- 04 (0x04) Read Input Registers
- 06 (0x06) Write Single Register
- 16 (0x10) Write Multiple Registers

For all ModBus registers that are denoted in description field as 'W: EPROM & memory' exist another register with address+200 (203 for CurrentLimit, for example) to write only runtime memory location of that variable, without affecting EEPROM value. Read command returns the same result for both addresses.

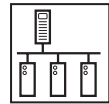
Parameter changes in EEPROM and memory using first set of addresses are permanent, however parameter changes in memory only using second set of addresses are valid only for one session. Session ends by powering the drive down.

Remember, when changing double word or float parameters, the change is applied to the drive only after second word is written.

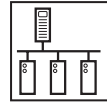
ModBus Register 16 bit address	Type	Controller Variable	Description
1	16 bit word	DriveMode	See Drive 94 manual W: EPROM & memory.
2	16 bit word	PwmIndex	See Drive 94 manual W: EPROM & memory.
3	32 bit float	CurrentLimit	See Drive 94 manual W: EPROM & memory. Low word starts first
5	32 bit float	PeakCurrentLimit	See Drive 94 manual W: EPROM & memory. Low word starts first
7	32 bit float	LowPeakCurrentLimit	See Drive 94 manual W: EPROM & memory. Low word starts first
9	16 bit word	InvertAnalogInput	See Drive 94 manual W: EPROM & memory.
10	32 bit float	CurrentAnalogInput	See Drive 94 manual W: EPROM & memory. Low word starts first



ModBus Register 16 bit address	Type	Controller Variable	Description
12	32 bit float	VelocityAnalogInput	See Drive 94 manual W: EPROM & memory. Low word starts first
14	16 bit word	AccDecFlag	See Drive 94 manual W: EPROM & memory.
15	32 bit float	AccelLimit	See Drive 94 manual W: EPROM & memory. Low word starts first
17	32 bit float	DecelLimit	See Drive 94 manual W: EPROM & memory. Low word starts first
19	16 bit int	GearRatioTop	See Drive 94 manual W: EPROM & memory.
20	16 bit word	GearRatioBottom	See Drive 94 manual W: EPROM & memory.
21	16 bit word	StepInputType	See Drive 94 manual W: EPROM & memory.
22	16 bit word	FaultReset	See Drive 94 manual W: EPROM & memory.
23	16 bit word	FeedbackLoss	See Drive 94 manual W: EPROM & memory.
24	16 bit word	ReferenceType	See Drive 94 manual W: EPROM & memory.
25	16 bit word	MotorTempSensorRes	See Drive 94 manual W: EPROM & memory.
26	16 bit word	MotorTempSensorEnable	See Drive 94 manual W: EPROM & memory.
27	16 bit word	RegenDutyCycle	See Drive 94 manual W: EPROM & memory.
28	16 bit word	EncoderRepeatSource	See Drive 94 manual W: EPROM & memory.
29	32 bit float	TorqueVelocityLimit	See Drive 94 manual W: EPROM & memory. Low word starts first
31	16 bit int	SeGearRatioTop	See Drive 94 manual W: EPROM & memory.
32	16 bit word	SeGearRatioBottom	See Drive 94 manual W: EPROM & memory.
33	16 bit word	SecondEncoderEnable	See Drive 94 manual W: EPROM & memory.
34	32 bit float	SeIGain	See Drive 94 manual W: EPROM & memory. Low word starts first
36	16 bit word	SePosErr	See Drive 94 manual W: EPROM & memory.



ModBus Register 16 bit address	Type	Controller Variable	Description
37	32 bit float	SeMaxErrTime	See Drive 94 manual W: EPROM & memory. Low word starts first
39	16 bit word	AnalogOutput	See Drive 94 manual W: EPROM & memory.
40	32 bit float	CurrentScale	See Drive 94 manual W: EPROM & memory. Low word starts first
42	32 bit float	VelocityScale	See Drive 94 manual W: EPROM & memory. Low word starts first
44	16 bit word	AnalogBand	See Drive 94 manual W: EPROM & memory.
45	16 bit word	AnalogOffset	See Drive 94 manual W: EPROM & memory.
46	16 bit word	ZeroSpeed	See Drive 94 manual W: EPROM & memory.
47	16 bit word	SpeedWindow	See Drive 94 manual W: EPROM & memory.
48	16 bit word	AtSpeed	See Drive 94 manual W: EPROM & memory.
49	16 bit word	PosErr	See Drive 94 manual W: EPROM & memory.
50	32 bit float	MaxErrTime	See Drive 94 manual W: EPROM & memory. Low word starts first
52	32 bit float	VelPGain	See Drive 94 manual W: EPROM & memory.
54	32 bit float	VellGain	See Drive 94 manual W: EPROM & memory.
56	32 bit float	PosPGain	See Drive 94 manual W: EPROM & memory.
58	32 bit float	PoslGain	See Drive 94 manual W: EPROM & memory. Low word starts first
60	32 bit float	DGain	See Drive 94 manual W: EPROM & memory.
62	32 bit float	ILimit	See Drive 94 manual W: EPROM & memory.
63	16 bit word	VelocityRegWnd	See Drive 94 manual W: EPROM & memory.
65	16 bit word	In2Func	See Drive 94 manual W: EPROM & memory.
66	16 bit word	In2Polarity	See Drive 94 manual W: EPROM & memory.
67	16 bit word	InBounceDelay for input 1	See Drive 94 manual W: EPROM & memory.



ModBus Register 16 bit address	Type	Controller Variable	Description
68	16 bit word	InBounceDelay for input 2	See Drive 94 manual W: EPROM & memory.
69	16 bit word	OutFunc for output 1	See Drive 94 manual W: EPROM & memory.
70	16 bit word	OutFunc for output 2	See Drive 94 manual W: EPROM & memory.
71	16 bit word	OutPolarity for output 1	See Drive 94 manual W: EPROM & memory.
72	16 bit word	OutPolarity for output 2	See Drive 94 manual W: EPROM & memory.
73	16 bit word	ControlWord	2 – Quick Stop 3 – Enable Drive 4 – Disable Drive 5 – Fault Reset 6 - Continue W: Memory only
75	32 bit dword	TargetVelocity	This register could provide the Target Velocity when the drive is in normal velocity mode. W: Memory only. Low word starts first
77	16 bit word	TargetTorque	This register could provide the Target Torque when the drive is in normal velocity mode. W: Memory only. Low word starts first

4.2 Discrete Memory

4.2.1 Discrete Inputs (Read Only Bits)

The following read only ModBus bit registers (drive bit variables) accessible only through the ModBus function

- 02 (0x02) Read Discrete Inputs

4.2.2 Coils (Read/Write Bits)

The following read/write ModBus bit registers (drive bit variables) accessible only through the ModBus functions

- 01 (0x01) Read Coils
- 05 (0x05) Write Single Coil
- 15 (0x0F) Write Multiple Coils

Modbus Register 16 bit address	Type	Description
1 (0x0001)	BIT	Digital OUT1
2 (0x0002)	BIT	Digital OUT2

Lenze

AC Technology Corporation • 630 Douglas Street • Uxbridge, MA 01569 • USA
☎ +1 (508) 278-9100