USER MANUAL

Accessory 24M2A

MACRO Analog Output Servo Module

3Ax-603744-10x

February 14, 2015

DELTA TAU
Data Systems, Inc.

NEW IDEAS IN MOTION …

21314 Lassen St. Chatsworth, CA 91311 // Tel. (818) 998-2095 Fax. (818) 998-7807 // www.deltatau.com

Copyright Information

To report errors or inconsistencies, call or email:

Delta Tau Data Systems, Inc. Technical Support
Phone: (818) 717-5656
Fax: (818) 998-7807
Email: support@deltatau.com
Website: http://www.deltatau.com

Operating Conditions

All Delta Tau Data Systems, Inc. motion controller products, accessories, and amplifiers contain static sensitive components that can be damaged by incorrect handling. When installing or handling Delta Tau Data Systems, Inc. products, avoid contact with highly insulated materials. Only qualified personnel should be allowed to handle this equipment.

In the case of industrial applications, we expect our products to be protected from hazardous or conductive materials and/or environments that could cause harm to the controller by damaging components or causing electrical shorts. When our products are used in an industrial environment, install them into an industrial electrical cabinet or industrial PC to protect them from excessive or corrosive moisture, abnormal ambient temperatures, and conductive materials. If Delta Tau Data Systems, Inc. products are directly exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation.

Safety Instructions

Qualified personnel must transport, assemble, install, and maintain this equipment. Properly qualified personnel are persons who are familiar with the transport, assembly, installation, and operation of equipment. The qualified personnel must know and observe the following standards and regulations:

IEC364resp.CENELEC HD 384 or DIN VDE 0100
IEC report 664 or DIN VDE 0110
National regulations for safety and accident prevention or VBG 4

Incorrect handling of products can result in injury and damage to persons and machinery. Strictly adhere to the installation instructions. Electrical safety is provided through a low-resistance earth connection. It is vital to ensure that all system components are connected to earth ground.

This product contains components that are sensitive to static electricity and can be damaged by incorrect handling. Avoid contact with high insulating materials (artificial fabrics, plastic film, etc.). Place the product on a conductive surface. Discharge any possible static electricity build-up by touching an unpainted, metal, grounded surface before touching the equipment.

Keep all covers and cabinet doors shut during operation. Be aware that during operation, the product has electrically charged components and hot surfaces. Control and power cables can carry a high voltage, even when the motor is not rotating. Never disconnect or connect the product while the power source is energized to avoid electric arcing.

Warning

A Warning identifies hazards that could result in personal injury or death. It precedes the discussion of interest.

Caution

A Caution identifies hazards that could result in equipment damage. It precedes the discussion of interest.

Note

A Note identifies information critical to the understanding or use of the equipment. It follows the discussion of interest.

REVISION HISTORY				
REV.	**DESCRIPTION**	**DATE**	**CHG**	**APPVD**
1	CHANGED 7-SEGMENT DISPLAY DESCRIPTIONS REMOVED DUPLICATE SECTIONS FOR 7-SEGMENT DISPLAY AND CONNECTOR DESCRIPTIONS FORMATTING HEADER/FOOTER CORRECTIONS MOVED “FLAG AND LIMIT WIRING” TO “CONNECTIONS” SECTION	06/11/06	C.PERRY	A. SOTELO
2	E-POINT JUMPER DESCRIPTIONS REVISED	06/19/06	C.PERRY	A. SOTELO
3	REVISED MACRO FIBER OPTION CONNECTOR DESCRIPTIONS CHANGED MECHANICAL LAYOUT AND CONNECTION SCHEMATICS CHANGED 24v INPUT LOGIC SUPPLY CONNECTOR (j10) WIRING ADDED TWO SINGLE-ENDED WIRING METHODS FOR SINUSIOD FEEDBACK MODIFIED MACRO RING ASCII COMMANDS MACRO ASCII COMMUNICATION GLOBAL COMMANDS REVISED “SETTING UP DIGITAL QUADRATURE ENCODERS” REVISED “SET UP PROCEDURES FOR SSI ENCODERS” REVISED “SET UP PROCEDURES FOR RESOLVERS” REVISED “SET UP PROCEDURES FOR SINUSOIDAL ENCODERS” REVISED “SET UP PROCEDURES FOR PHASE SHIFT” REVISED “SET UP PROCEDURES FOR POWER-ON ABSOLUTE POSITION OF RESOLVER ADDED “MANUAL SETUP FOR MOTOR OPERATION” SECTION	08/21/08	C.PERRY	K. ZHAO
4	ADDED MI16, MI17 AND MI18 FUNCTIONALITY DESCRIPTION	01/05/10	C.PERRY	S.SATTARI
5	COMPLETE MANUAL REVISION	02/14/15	DCDP	R. NADDAF

Table of Contents

INTRODUCTION

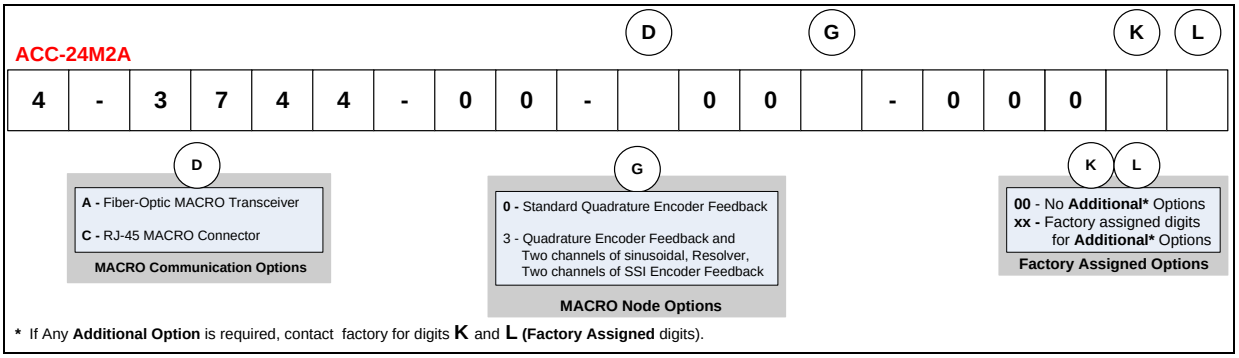
The ACC-24M2A is a two (2) axis servo peripheral designed to work with Turbo PMAC2 Ultralite, Power PMAC EtherLite, or UMAC MACRO controllers to remotely interface to two (2) channels of analog style amplifiers. This device produces a ± 10 V_{dc} control signal to control analog amplifiers.

The ACC-24M2A can process the following feedback types:

- Quadrature
- 1 Vpp Sinusoidal
- Resolver
- SSI

SPECIFICATIONS

Part Number



ACC-24M2A Options

ACC-24M2A may be ordered equipped with the following options:

Options Included	Part Number
2-axis MACRO Analog Servo Peripheral With Fiber-optic MACRO connectors *(Opt-A Included)*	4-3744-00-A000-00000
2-axis MACRO Analog Servo Peripheral With RJ-45 isolated electrical MACRO connectors *(Opt-C Included)*	4-3744-00-C000-00000
2-axis MACRO Analog Servo Peripheral With Fiber-optic MACRO connectors *(Opt-A Included)* Two channels of sinusoidal, Resolver *(Opt-3 Included)* Two channels of SSI Encoder Feedback	4-3744-00-A003-00000
2-axis MACRO Analog Servo Peripheral With RJ-45 isolated electrical MACRO connectors *(Opt-C Included)* Two channels of sinusoidal, Resolver *(Opt-3 Included)* Two channels of SSI Encoder Feedback	4-3744-00-C003-00000

Environmental Specifications

Description	Unit	Specifications
Operating Temperature	°C	+0 to 45°C
Rated Storage Temperature	°C	-25 to +70
Humidity	%	10% to 90% non-condensing
Shock		Call Factory
Vibration		Call Factory
Operating Altitude	Feet (Meters)	To 3300 feet (1000meters)
Air Flow Clearances	in (mm)	1" (2.54mm) above and below unit for air flow

Electrical Specifications

Main Input Power	Nominal Input Voltage (Vdc)	24 V_{dc}
Output Power	DAC Output (V_{dc})	+/- 10 V_{dc}
	DAC Output (A)	0.045A
	Flag Output (V_{dc})	12-24V_{dc} Standard, 5 V_{dc} w/ RP38 Installed
	Flag Input (V_{dc})	12-24V_{dc} Standard, 5 V_{dc} w/ RP38 Installed

Note

Installing a 1 KΩ resistor pack at RP38 will make the flags 5 V_{dc}.

Physical Specifications

	Width	Height	Depth
Overall Dimensions	2.00in./50.8mm	9.75in./ 247.7mm	6.50in./ 165.1mm
Mounting Dimensions	1.25in./31.75mm	9.375in./ 238.13mm	

Weight: 2.3 lbs / 1.0 kg
See the "Layout" section of this manual for drawings of the physical layout.

RECEIVING AND UNPACKING

Unpacking Guidelines

Delta Tau products are thoroughly tested at the factory and carefully packaged for shipment. When the ACC-24M2A is received, do the following immediately:

1. Inspect the condition of the shipping container and report any damage immediately to the commercial carrier that delivered the drive.
2. Remove the device from the shipping container and remove all packing materials. Check all shipping material for connector kits, documentation, diskettes, CD ROM, or other small pieces of equipment. Be aware that some connector kits and other equipment pieces may be quite small and can be discarded accidentally if care is not used when unpacking the equipment. The container and packing materials can be retained for future shipment.
3. Electronic components in this device are design-hardened to reduce static sensitivity. However, use proper procedures when handling the equipment.
4. If ACC-24M2A is to be stored for several weeks before use, be sure that it is stored in a location that conforms to published storage humidity and temperature specifications stated in this manual.

Use of Equipment

The following guidelines describe the restrictions for proper use of ACC-24M2A:

- The components built into electrical equipment or machines can be used only as integral components of such equipment.
- ACC-24M2A must not be operated on power supply networks without a ground or with an asymmetrical ground.
- ACC-24M2A may be operated only in a closed switchgear cabinet, taking into account the ambient conditions defined in the environmental specifications.

Delta Tau guarantees the conformance of ACC-24M2A with the standards for industrial areas stated in this manual only if Delta Tau components (cables, controllers, etc.) are used.

MOUNTING

Installation Guidelines

This product should be installed in an area that is protected from direct sunlight, corrosives, harmful gases or liquids, dust, metallic particles, and other contaminants. Exposure to these can reduce the operating life and degrade the performance.

A couple other factors to evaluate carefully when selecting a location for installation:

- Allow for at least 1 inch (2.54mm) top and bottom clearance to permit airflow. At least 0.4 inches (10mm) clearance is required between each side.
- Temperature, humidity and vibration specifications should also be considered.

ACC-24M2A can be mounted with a 3-hole panel mount, two U-shape notches on the bottom and one pear-shaped hole on top. Mounting is also identical to this on all peripheral devices.

If multiple MACRO devices are used, they can be mounted side-by-side, leaving at least a 0.4 inch clearance between them. It is important that the airflow is not obstructed by the placement of conduit tracks or other devices in the enclosure.

ACC-24M2A should be mounted to an unpainted, electrically-conductive panel in order to allow for reduced electrical noise interference. The back panel should be machined to accept the mounting bolt pattern of the accessory. Make sure that all metal chips are cleaned up before the device is mounted so that there is no risk of getting metal chips inside the device.

ACC-24M2A is mounted to the back panel with three M4 screws and internal-tooth lock washers. The teeth of the washers must break through the device's anodizing in order to provide an electrically-conductive path in as many places as possible.

Caution

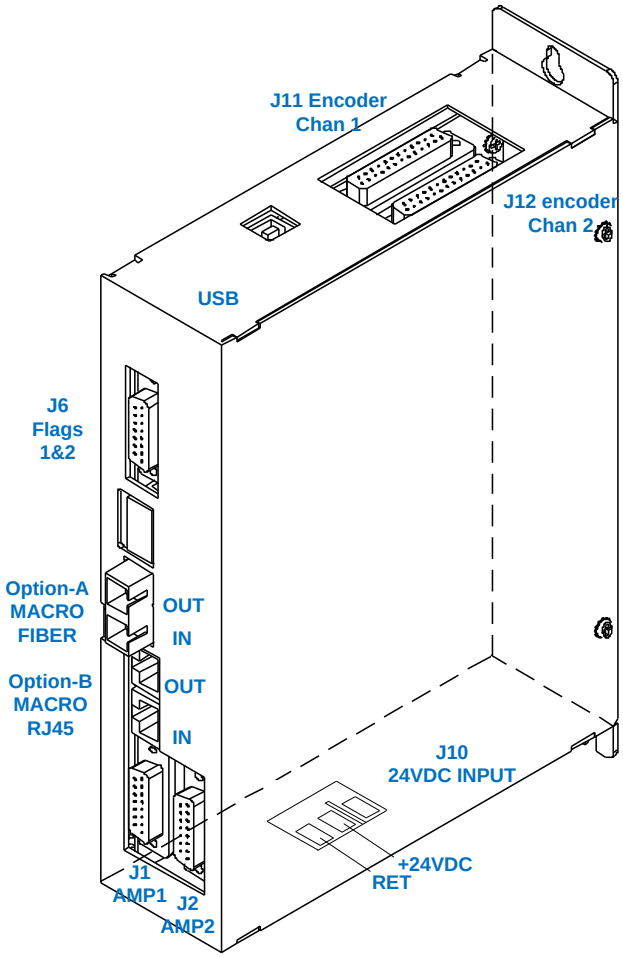
Units must be installed in an enclosure that meets the environmental IP rating of the end product (ventilation or cooling may be necessary to prevent enclosure ambient from exceeding 45° C [113° F]).

WARNING

Installation of electrical control equipment is subject to many regulations including national, state, local, and industry guidelines and rules. General recommendations can be stated but it is important that the installation be carried out in accordance with all regulations pertaining to the installation.

Connector Locations

Below is a drawing of the product with its connectors labeled:



CONNECTOR PINOUTS

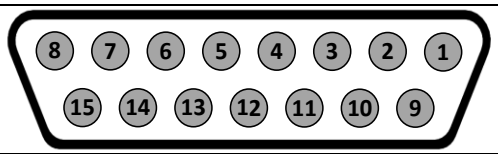
J10: 24 V_{DC} Logic Power Input

An external 24VDC power supply is required to power the logic, flags and DAC output sections of ACC-24M2A through the J10 connector. The polarity of this connection is extremely important. Carefully follow the instructions in the wiring diagram. This connection can be made using 16 AWG wire directly from a protected power supply. In situations where the power supply is shared with other devices, it may be desirable to insert a filter in this connection.

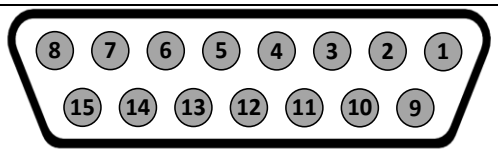
The power supply providing this 24V must be capable of providing an instantaneous current of at least 1.5A to be able to start the DC-to-DC converter in ACC24M2A. In the case where multiple devices are driven from the same 24V supply, it is recommended that each device be wired back to the power supply terminals independently. It is also recommended that the power supply be sized to handle the instantaneous inrush current required for each device.

J10: 3-Pin Edge Connector **Mating: Plated Pins on ACC-24M2A PCB**			 Pins: 3 2 1	
Pin #	**Symbol**	**Function**	**Description**	**Notes**
1	24VDC RET	Common	Logic power return	
2	+24VDC	Input	Logic power input	24V ±10%, 2 A
3	N.C.	N.C.	Not Connected	
Connector is located at the bottom side of the unit. Delta Tau part number: 014-188305-001 Phoenix part number: 1883051				

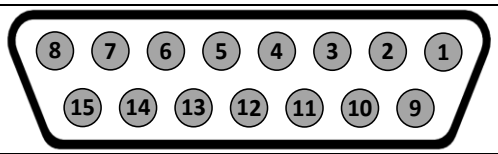
J1: Amplifier Channel 1

J1: DB-15 Female **Mating: DB-15 Male**		
Pin #	**Symbol**	**Description**
1	DAC1_A+	Phase A +analog output
2	DAC1_B+	Phase B +analog output
3	AE_NC_1	Amplifier Enabled Normally Closed
4	AE_NO_1	Amplifier Enabled Normally Open
5	AFAULT_1-	Amplifier Fault input
6	N.C.	Do not connect
7	A+12V	Analog Positive Supply Voltage
8	AGND	Analog Ground
9	DAC1_A-	Phase A +analog output
10	DAC1_B-	Phase B +analog output
11	AE_COM_1	Amplifier Enable Common
12	AFAULT_1+	Amplifier Fault input
13	N.C.	Do not connect
14	AGND	Analog Ground
15	A-12V	Analog Negative Supply Voltage

J2: Amplifier Channel 2

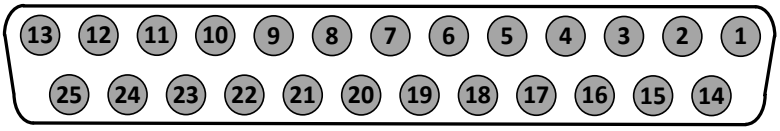
J1: DB-15 Female **Mating: DB-15 Male**		
Pin #	**Symbol**	**Description**
1	DAC2_A+	Phase A +analog output
2	DAC2_B+	Phase B +analog output
3	AE_NC_2	Amplifier Enabled Normally Closed
4	AE_NO_2	Amplifier Enabled Normally Open
5	AFAULT_2-	Amplifier Fault input
6	N.C.	Do not connect
7	A+12V	Analog Positive Supply Voltage
8	AGND	Analog Ground
9	DAC2_A-	Phase A +analog output
10	DAC2_B-	Phase B +analog output
11	AE_COM_2	Amplifier Enable Common
12	AFAULT_2+	Amplifier Fault input
13	N.C.	Do not connect
14	AGND	Analog Ground
15	A-12V	Analog Negative Supply Voltage

J6: Flags and Limits

J6: DB-15 Female **Mating: DB-15 Male**			
Pin #	**Symbol**	**Direction**	**Description**
1	USER1	Input	User Flag for Channel 1
9	PLIM1	Input	Positive Position Limit for Channel 1
2	NLIM1	Input	Negative Position Limit for Channel 1
10	HOME1	Input	Home flag for Channel 1
3	FLG_RTN1	Input	Voltage return for Channel 1's flags
11	EQU1	Output	Position Compare Output for Channel 1
4	USER2	Input	User Flag for Channel 2
12	PLIM2	Input	Positive Position Limit for Channel 2
5	NLIM2	Input	Negative Position Limit for Channel 2
13	HOME2	Input	Home flag for Channel 2
6	FLG_RTN2	Input	Voltage return for Channel 2's flags
14	EQU2	Output	Position Compare Output for Channel 2
7	GND	Input	Digital Ground
15	GND	Input	Digital Ground
8	GND	Input	Digital Ground

J11 & J12: Encoder Feedback, Digital A Quad B

ACC-24M2A accepts inputs from two digital encoders and provides encoder position data to PMAC. J11 is for Encoder 1 and J12 is for Encoder 2. The ACC-24M2A's encoder interface circuitry employs differential line receivers. The differential format provides a means of using twisted pair wiring that allows for better noise immunity when wired into machinery.

J11 & J12: D-sub DB-25F **Mating: D-sub DB-25M**		
Pin #	**Symbol**	**Description**
1	ChA+	Channel A Positive Signal
14	ChA-	Channel A Negative Signal
2	ChB+	Channel B Positive Signal
15	ChB-	Channel A Negative Signal
3	ChC+	Channel C Positive Signal
16	ChC-	Channel C Negative Signal
12 / 24*	ENCPWR/5V	Encoder Power (+5VDC)
13 / 25	GND	Digital Ground

Note:

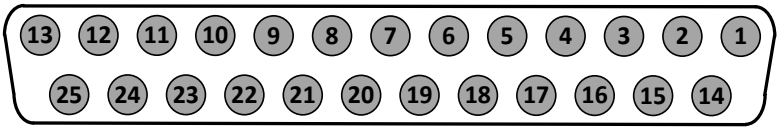
- Do not connect the pins that are not listed.
- *If the encoder being used required +5VDC power, it can be connected to pins 12/24, and grounded on pins 13/25. However, if the encoder has different power requirements, do not connect pins 13/24 and 13/25 to the encoder.
- To twist the ENCPWR/5V and the GND wires together is recommended for better noise immunity.
- Tie together the ACC-24M2A's GND and the encoder's power supply GND if an external power supply is used for the encoder for better noise immunity.

Note

Most applications use pin 12 to supply power to the encoder. However, for encoders that send out initial information at power on, the user should use pin 24 instead of pin 12, and then set MI984=1 on ACC-24M2A in order to manually enable the encoder power after PMAC is powered on.

J11 & J12: Encoder Feedback, SSI

ACC-24M2A accepts inputs from two digital encoders and provides encoder position data to PMAC. J11 is for Encoder 1 and J12 is for Encoder 2. The ACC-24M2A's encoder interface circuitry employs differential line receivers. The differential format provides a means of using twisted pair wiring that allows for better noise immunity when wired into machinery.

J11 & J12: D-sub DB-25F **Mating: D-sub DB-25M**		
Pin #	**Symbol**	**Description**
6	CLK+	Serial Clock Signal Positive
7	DATA+	Serial Data Signal Positive
19	CLK-	Serial Clock Signal Negative
20	DATA-	Serial Data Signal Negative
12/24*	ENCPWR/5V	Encoder Power (+5VDC)
13/25	GND	Digital Ground

Note:

- Do not connect the pins that are not listed.
- *If the encoder being used required +5VDC power, it can be connected to pins 12/24, and grounded on pins 13/25. However, if the encoder has different power requirements, do not connect pins 13/24 and 13/25 to the encoder.
- To twist the ENCPWR/5V and the GND wires together is recommended for better noise immunity.
- Tie together the ACC-24M2A's GND and the encoder's power supply GND if an external power supply is used for the encoder for better noise immunity.

Note

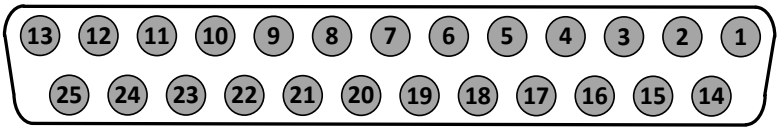
Most applications use pin 12 to supply power to the encoder. However, for encoders that send out initial information at power on, the user should use pin 24 instead of pin 12, and then set MI984=1 on ACC-24M2A in order to manually enable the encoder power after PMAC is powered on.

J11 & J12: Encoder Feedback, Sinusoidal

ACC-24M2A accepts inputs from two digital encoders and provides encoder position data to PMAC. J11 is for Encoder 1 and J12 is for Encoder 2. The ACC-24M2A's encoder interface circuitry employs differential line receivers. The differential format provides a means of using twisted pair wiring that allows for better noise immunity when wired into machinery.

Acc-24M2A with the Sinusoidal Interpolator option accepts inputs from two sinusoidal or quasi-sinusoidal encoders and provides encoder position data to the motion processor. This interpolator creates 4,096 steps per sine-wave cycle. The user must order the appropriate option. ACC-24M2A can be used only with a voltage mode sinusoidal encoder type.

Be sure to use shielded, twisted pair cabling for sinusoidal encoder wiring. Double insulated is the best choice. The sinusoidal signals are very small and must be kept as noise free as possible. Avoid cable routing near noisy motor or driver wiring. Refer to the appendix for tips on encoder wiring.
It is possible to reduce noise in the encoder lines of a motor-based system by the use of inductors that are placed between the motor and the amplifier. Improper grounding techniques may also contribute to noisy encoder signals.

J11 & J12: D-sub DB-25F **Mating: D-sub DB-25M**		
Pin #	**Symbol**	**Description**
1	Sin+	Sinusoidal Signal Positive
14	Sin-	Sinusoidal Signal Negative
2	Cos+	Cosine Signal Positive
15	Cos-	Cosine Signal Negative
3	Index+	Index Pulse Signal Positive
16	Index-	Index Pulse Signal Negative
12/24*	ENCPWR/5V	Encoder Power (+5VDC)
13/25	GND	Ground

Note:

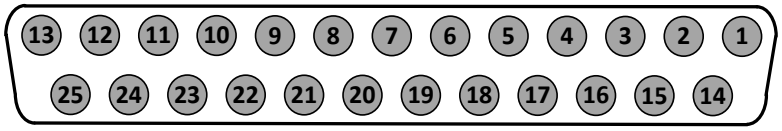
- Do not connect the pins that are not listed.
- *If the encoder being used required +5VDC power, it can be connected to pins 12/24, and grounded on pins 13/25. However, if the encoder has different power requirements, do not connect pins 13/24 and 13/25 to the encoder.
- To twist the ENCPWR/5V and the GND wires together is recommended for better noise immunity.
- Tie together the ACC-24M2A's GND and the encoder's power supply GND if an external power supply is used for the encoder for better noise immunity.

Note

Most applications use pin 12 to supply power to the encoder. However, for encoders that send out initial information at power on, the user should use pin 24 instead of pin 12, and then set MI984=1 on ACC-24M2A in order to manually enable the encoder power after PMAC is powered on.

J11 & J12: Encoder Feedback, EnDat

The Acc-24M2A will read the absolute data from the EnDat (Encoder Data) interface only if the appropriate option is ordered. Its differential format provides a means of using twisted-pair wiring that allows for better noise immunity when wired into machinery.

J11 & J12: D-sub DB-25F **Mating: D-sub DB-25M**		
Pin #	**Symbol**	**Description**
1	Sin+/ ChA+	Sinusoidal Signal Positive/Channel A Positive
2	Cos+/ChB+	Cosine Signal Positive/Channel B Positive
14	Sin-/ChA-	Sinusoidal Signal Negative/Channel A Negative
15	Cos-/ChB-	Cosine Signal Negative/Channel B Negative
6	CLK+	Clock Signal Positive
7	DATA+	Data Signal Positive
19	CLK-	Clock Signal Negative
20	DATA-	Data Signal Negative
12/24	ENCPWR/5V	Encoder Power (+5VDC)
13/25	GND	Ground

Note:

- Do not connect the pins that are not listed.
- *If the encoder being used required +5VDC power, it can be connected to pins 12/24, and grounded on pins 13/25. However, if the encoder has different power requirements, do not connect pins 13/24 and 13/25 to the encoder.
- To twist the ENCPWR/5V and the GND wires together is recommended for better noise immunity.
- Tie together the ACC-24M2A's GND and the encoder's power supply GND if an external power supply is used for the encoder for better noise immunity.

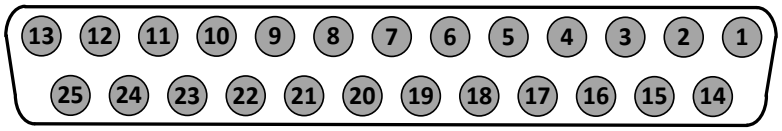
Note

Most applications use pin 12 to supply power to the encoder. However, for encoders that send out initial information at power on, the user should use pin 24 instead of pin 12, and then set MI984=1 on ACC-24M2A in order to manually enable the encoder power after PMAC is powered on.

J11 & J12: Encoder Feedback, HiperFace

ACC-24M2A will read the absolute data from the Hiperface® interface only if the appropriate option is ordered.

J11 & J12: D-sub DB-25F
Mating: D-sub DB-25M



Pin #	Symbol	Description
1	Sin+/ ChA+	Sinusoidal Signal Positive/Channel A Positive
2	Cos+/ChB+	Cosine Signal Positive/Channel B Positive
14	Sin-/ChA-	Sinusoidal Signal Negative/Channel A Negative
15	Cos-/ChB-	Cosine Signal Negative/Channel B Negative
7	DATA+	Clock Signal Positive
20	DATA-	Data Signal Positive
12/24	ENCPWR/5V	Clock Signal Negative
13/25	GND	Data Signal Negative

Note:

- Do not connect the pins that are not listed.
- *If the encoder being used required +5VDC power, it can be connected to pins 12/24, and grounded on pins 13/25. However, if the encoder has different power requirements, do not connect pins 13/24 and 13/25 to the encoder.
- To twist the ENCPWR/5V and the GND wires together is recommended for better noise immunity.
- Tie together the ACC-24M2A's GND and the encoder's power supply GND if an external power supply is used for the encoder for better noise immunity.

Note

Most applications use pin 12 to supply power to the encoder. However, for encoders that send out initial information at power on, the user should use pin 24 instead of pin 12, and then set MI984=1 on ACC-24M2A in order to manually enable the encoder power after PMAC is powered on.

Note

As of the date of the latest revision of this manual, HiperFace is not yet part of the ACC-24M2A firmware.

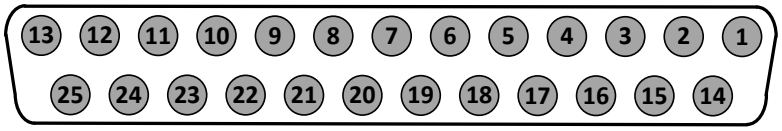
J11 & J12: Encoder Feedback, Resolver

The ACC-24M2A can interface to most industry standard resolvers if the appropriate option is ordered. Typical resolvers requiring 5 to 10 kHz excitation frequencies with voltages ranging from 5 to 10 V peak-to-peak are compatible with this drive.

Fundamentally, the ACC-24M2A connects three differential analog signal pairs to each resolver: a single excitation signal pair, and two analog feedback signal pairs. The wiring diagram below shows an example of how to connect the ACC-24M2A to the Resolver.

The differential format provides a means of using twisted pair wiring that allows for better noise immunity when wired into machinery.

J11 & J12: D-sub DB-25F
Mating: D-sub DB-25M



Pin #	Symbol	Description
4	ResSin+	Resolver Sine Positive
17	ResSin-	Resolver Sine Negative
5	ResCos+	Resolver Cosine Positive
18	ResCos-	Resolver Cosine Negative
11	ResOut	Resolver Output
13/25	GND	GND

Note:

- Do not connect the pins that are not listed.
- *If the encoder being used required +5VDC power, it can be connected to pins 12/24, and grounded on pins 13/25. However, if the encoder has different power requirements, do not connect pins 13/24 and 13/25 to the encoder.
- To twist the ENCPWR/5V and the GND wires together is recommended for better noise immunity.
- Tie together the ACC-24M2A's GND and the encoder's power supply GND if an external power supply is used for the encoder for better noise immunity.

Note

Most applications use pin 12 to supply power to the encoder. However, for encoders that send out initial information at power on, the user should use pin 24 instead of pin 12, and then set MI984=1 on ACC-24M2A in order to manually enable the encoder power after PMAC is powered on.

Universal Serial Bus Port (USB Port)

This connector uses a USB A-B cable to establish communication between the PC and the ACC-24M2A. This type of USB cable could be purchased at any local electronics or computer store. It may be ordered from Delta Tau as well.

Pin #	Symbol	Function
1	VCC	N.C.
2	D-	DATA-
3	D+	DATA+
4	GND	GND
5	SHELL	SHIELD
6	SHELL	SHIELD

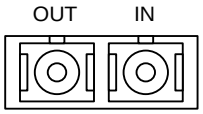
This connector is used only to change the operational firmware, or to perform basic software diagnostic operations. The user can use a serial port terminal window such as Microsoft® HyperTerminal to communicate with the MACRO Device and send ASCII commands to the device. Set the serial port communication settings as follows:

Baud Rate: 38400 if E3 is not jumpered, or 9600 if E3 is jumpered
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: Xon/Xoff

If the PeWin32PRO2 software is installed on the PC, then the USB device should be recognized by the operating system. If the device is not recognized, then contact Technical Support for assistance.

MACRO Fiber Connector

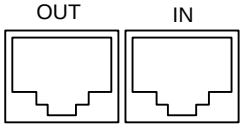
Option A provides the following connector for MACRO communications:

MACRO SC-Style Fiber Connector		 **Front View**
Pin #	**Symbol**	**Function**
1	IN	MACRO Ring Receiver
2	OUT	MACRO Ring Transmitter
Notes: The fiber optic version of MACRO uses 62.5/125 multi-mode glass fiber optic cable terminated in an SC-style connector. The optical wavelength is 1,300 nm.		

The input connector must be inserted into the MACRO output connector of the previous device on the MACRO ring. The output connector must be inserted into the input MACRO connector of the next device on the MACRO ring.

MACRO RJ-45 Copper Connector

Option C Provides the following connector for MACRO communications:

Connector: RJ45 CAT5e **Mating: RJ45 Receptacle**			 **Front View**
Pin #	**Symbol**	**Function**	**Description**
1	DATA+	Data +	Differential MACRO Signal
2	DATA-	Data -	Differential MACRO Signal
3	Unused		Unused terminated pin
4	Unused		Unused terminated pin
5	Unused		Unused terminated pin
6	Unused		Unused terminated pin
7	Unused		Unused terminated pin
8	Unused		Unused terminated pin

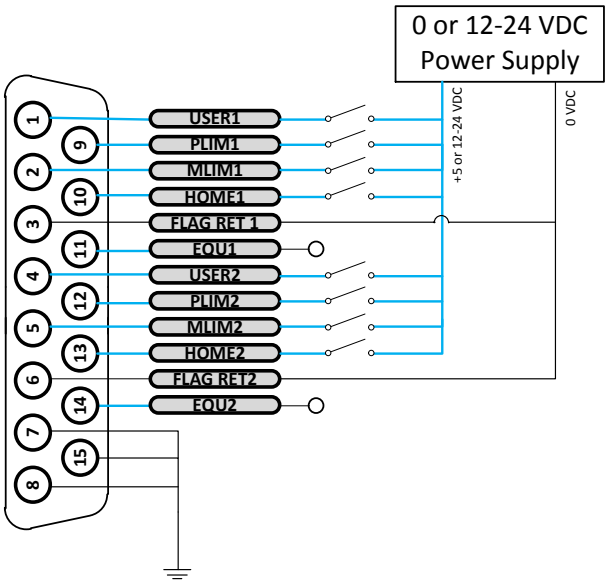
The cable used for MACRO wired connections is CAT5 verified straight-through 8 conductor.

The input connector must be inserted into the MACRO output connector of the previous device on the MACRO ring. The output connector must be inserted into the input MACRO connector of the next device on the MACRO ring.

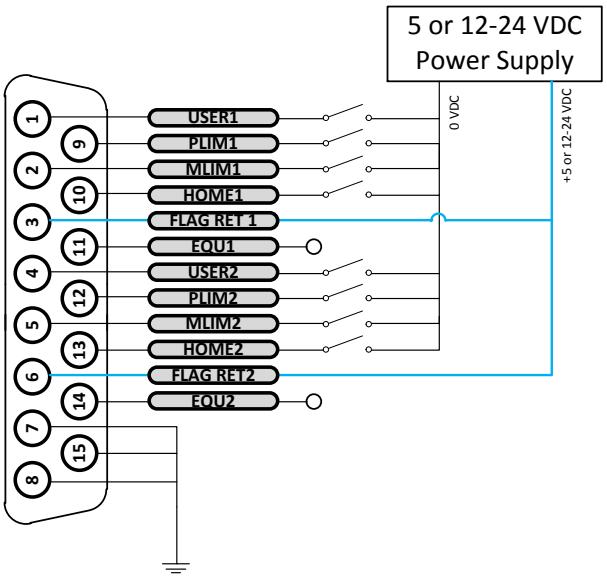
Sample Wiring Diagrams

J6: Flags

Sourcing Flags

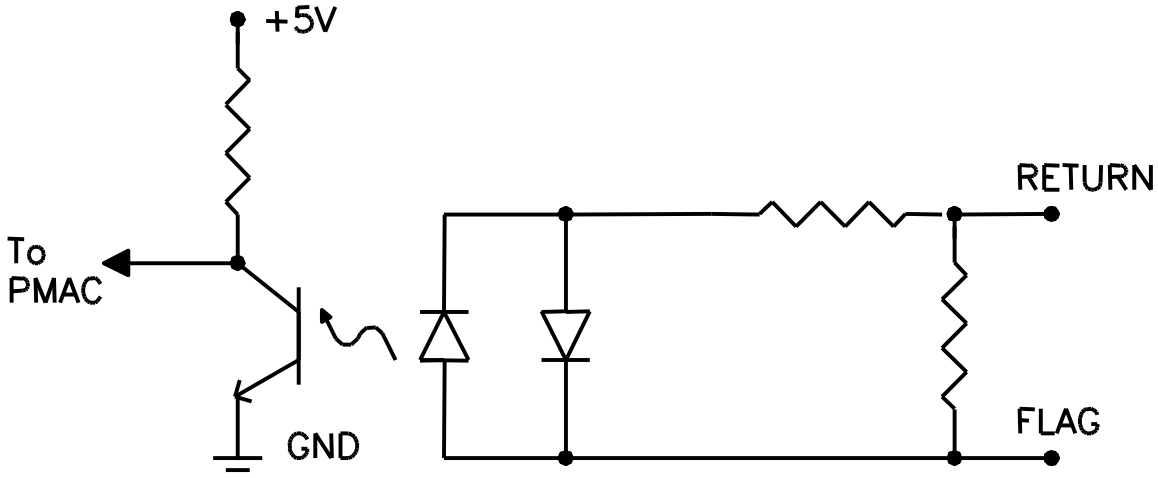


Sinking Flags



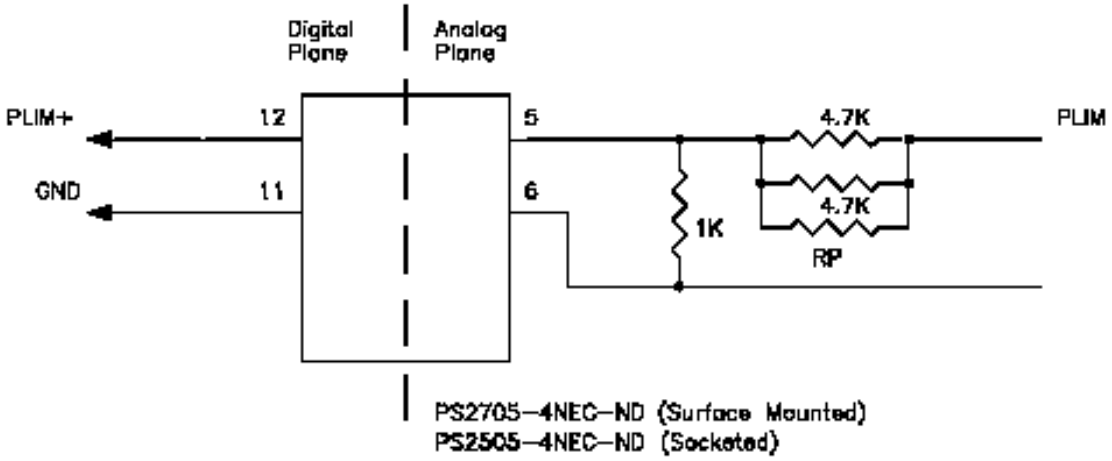
Output IC Diagram

ACC-24M2A allows the use of sinking or sourcing position limits and flags to the controller. The opto-isolator IC used is a PS2705-4NEC-ND quad phototransistor output type (see below).



This IC allows the current to flow from return to flag (sinking) or from flag to return (sourcing).

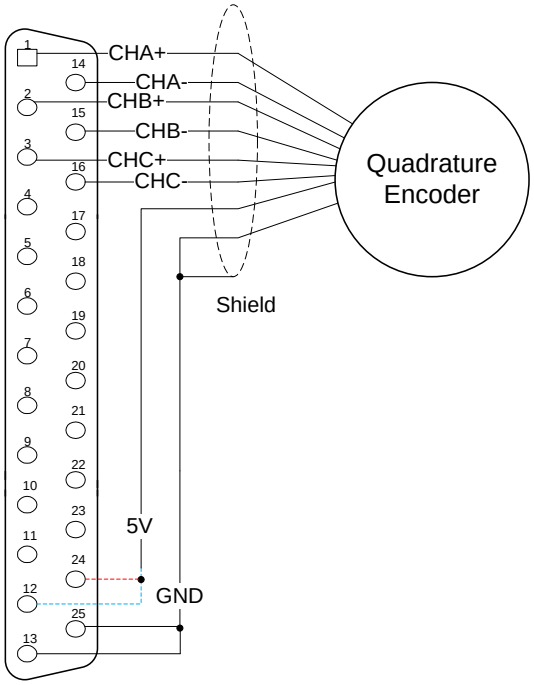
A sample of the internal positive limit circuit for this IC is shown below.



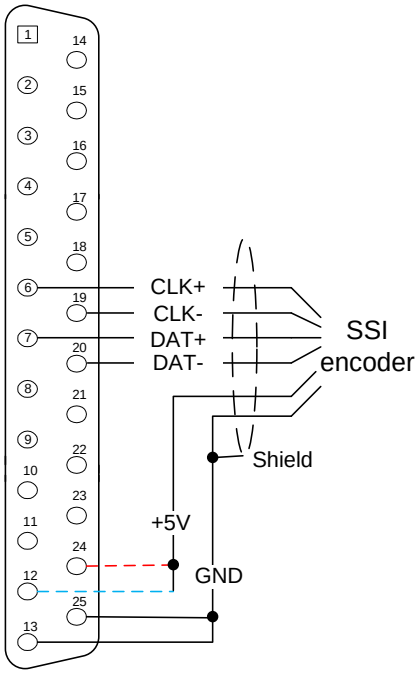
The 4.7K resistor packs used will allow 12–24V flag inputs. If the user wants to use 0–5V flags, then a 1KΩ resistor pack (RP) can be placed in RP7 for Channel 1's flags or in RP8 for Channel 2's flags. If these resistor packs are not added, all flags (± Limits, Home, User, and Amplifier Fault) will be referenced from 12–24V.

J11 & J12: Encoder Feedback, Digital A Quad B

The following wiring diagram shows an example of how to connect a quadrature encoder:



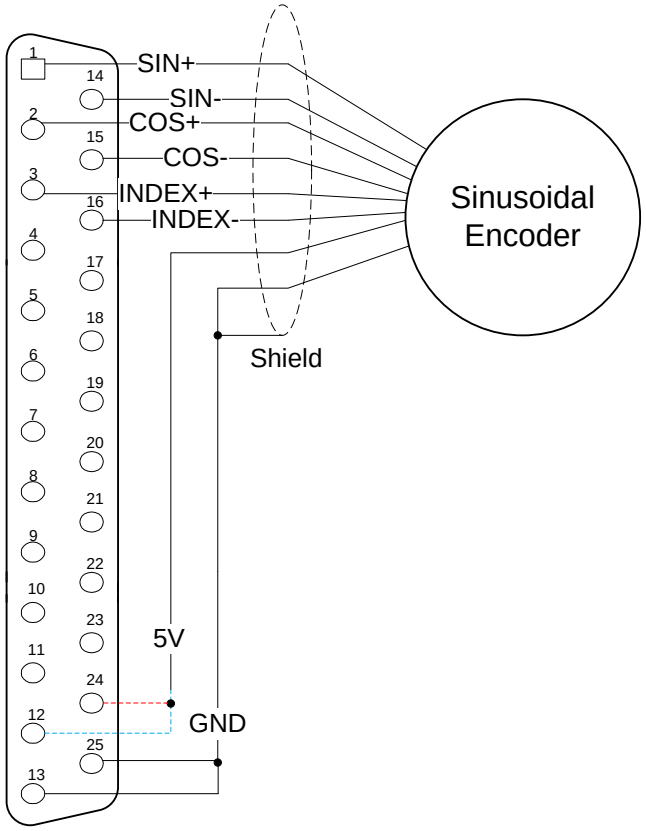
J11 & J12: Encoder Feedback, SSI



J11 & J12: Encoder Feedback, Sinusoidal

Differential Format

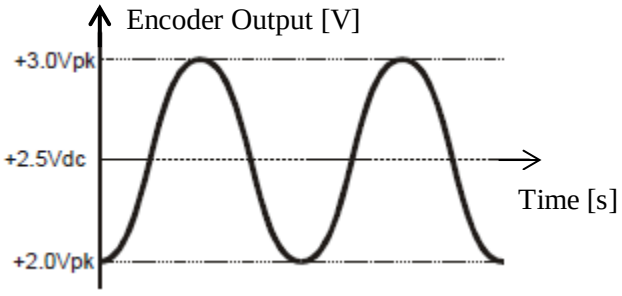
The differential format provides a means of using twisted pair wiring that allows for better noise immunity when wired into machinery.



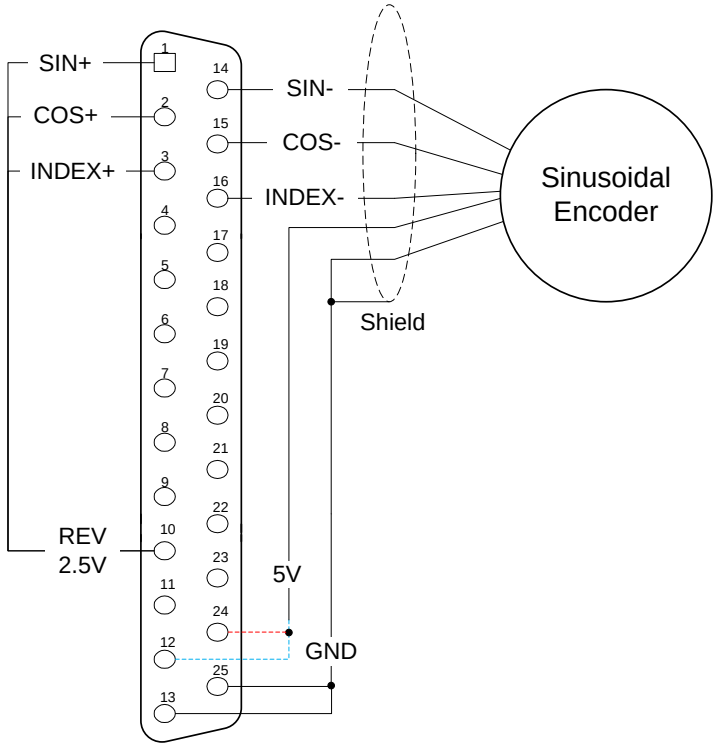
Single Ended Format 1

The single-ended formats provide a simpler means of using a sinusoidal encoder. Typically, fewer wires are needed and the encoders are always of the lower impedance voltage output type.

Note that all the single-ended encoder formats shown here might have velocity-ripple effects at very slow speeds due to the effects of op-amp voltage offsets. These offsets cause the sinusoidal signal to be centered at a value that is slightly different from the reference or servo ground as shown in the signal diagram on the right:



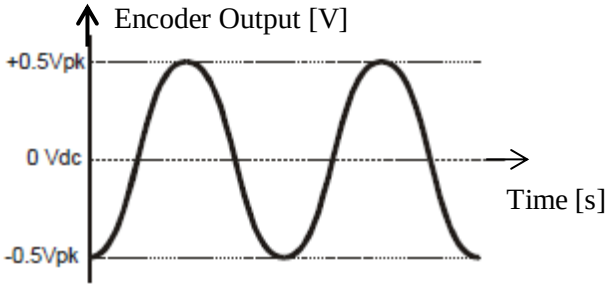
Below is the wiring diagram for Single Ended Format 1:



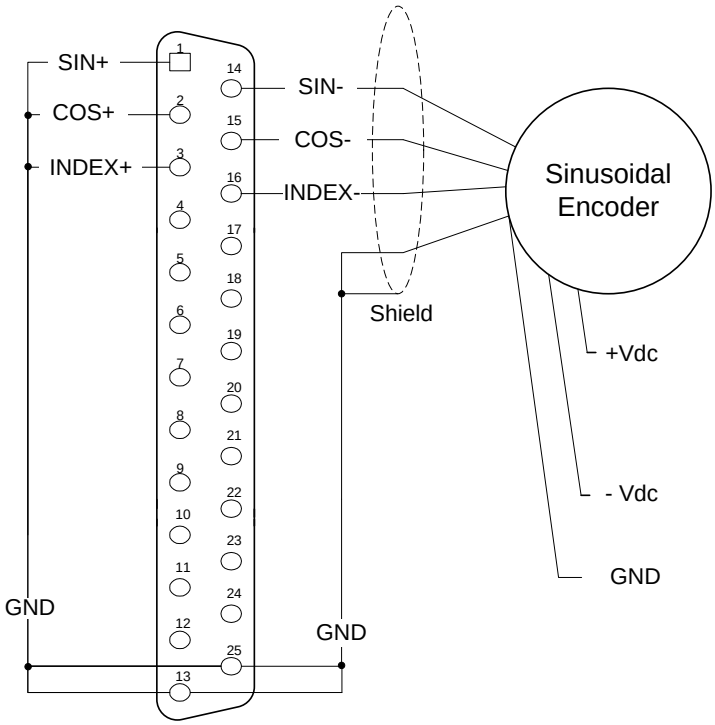
Single Ended Format 2

The diagram shown below is a simple single-ended encoder-wiring interface for encoders with output range at 2-3 Vdc. This encoder has SIN and COS outputs that provide a 1V peak-to-peak output with a voltage offset of 2.5 Vdc. Note that the SIN+, COS+, and INDEX+ lines are tied to the 2.5V internal references on the interpolator card.

The diagram to the right is similar to the signal diagram from the Single Ended Format 1 but with a different voltage offset. This encoder has SIN and COS outputs that provide a 1V peak-to-peak output with a voltage offset of 0.0 Vdc. Note that the SIN-, COS-, and INDEX- lines are tied to the GND on the interpolator card, and the encoder usually requires a bipolar supply.



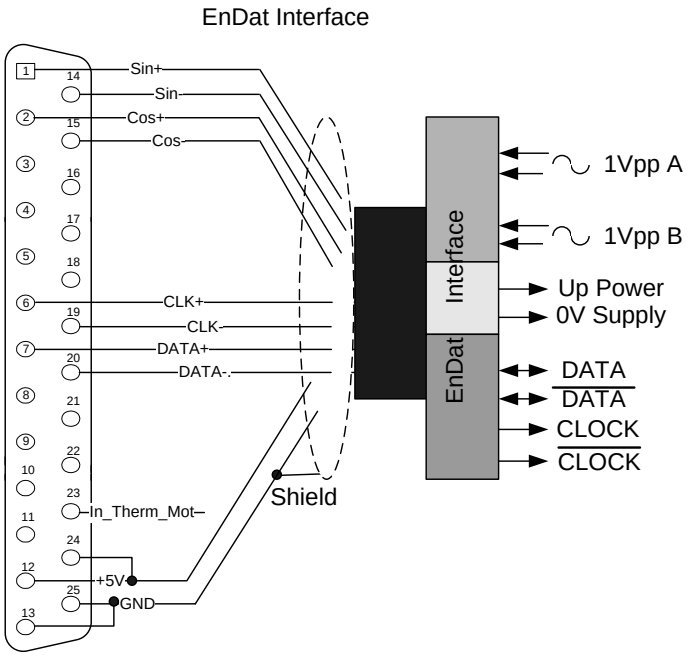
The wiring diagram for Single Ended Format 2 is below:



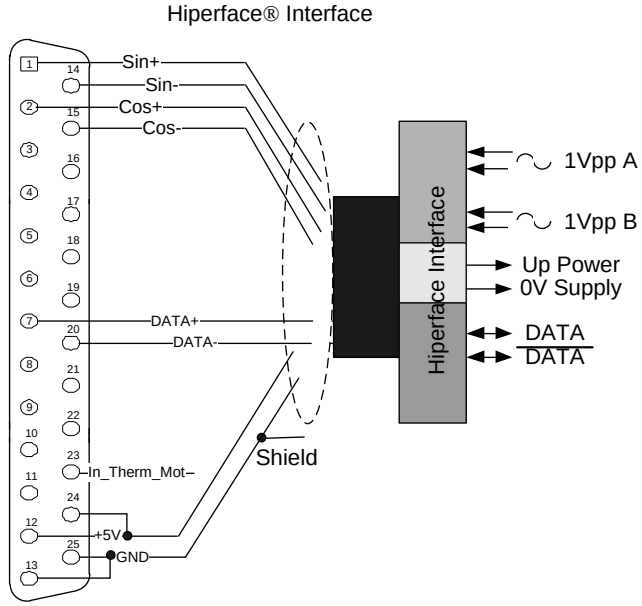
Noise Problems

When problems do occur, the culprit is often electrical noise. When this occurs, attempt to control the high-frequency current paths. If following the grounding instructions does not work, insert chokes in the motor phases. These chokes can be as simple as several wraps of the individual motor leads through a ferrite ring core (such as Micrometals T400-26D). This adds high-frequency impedance to the outgoing motor cable thereby impeding high-frequency noise from leaving the control cabinet. Care should be taken to be certain that the core's temperature is in a reasonable range after installing such devices.

J11 & J12: Encoder Feedback, EnDat



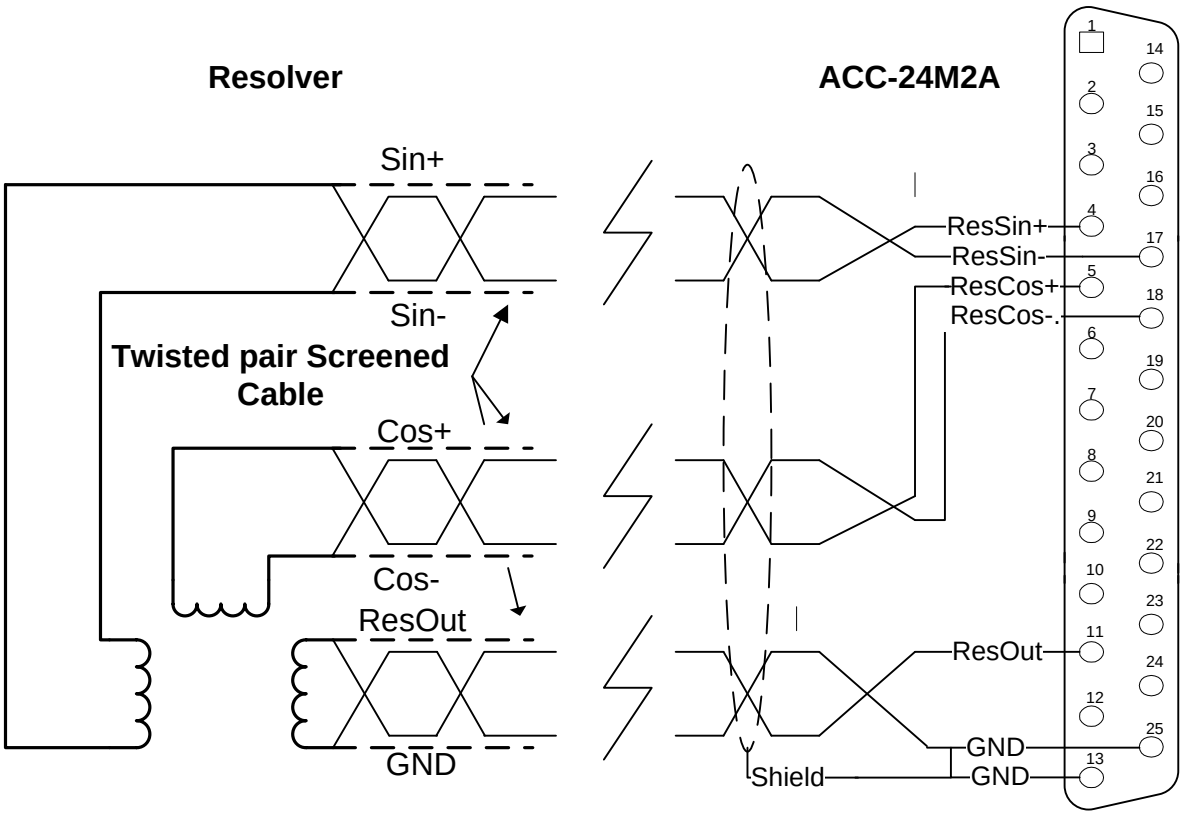
J11 & J12: Encoder Feedback, HiperFace



Note

As of the date of the latest revision of this manual, HiperFace is not yet part of the ACC-24M2A firmware.

J11 & J12: Encoder Feedback, Resolver



Notes:
Terminate shields on pins 13 and 25

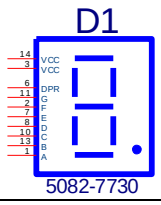
TROUBLESHOOTING

Status LED Indicators

Status Display	Color	Description
7-segment LED	Red	16 numeric codes plus two decimal points
PWR	Green	Lit when logic power is good
WD	Red	Indicates that the watchdog safety circuit has activated, indicating a failure condition.

7-Segment LED Indicator

This indicator reports the status of the unit with respect to the MACRO link, indicating the value of MI974. These are the possible status codes:

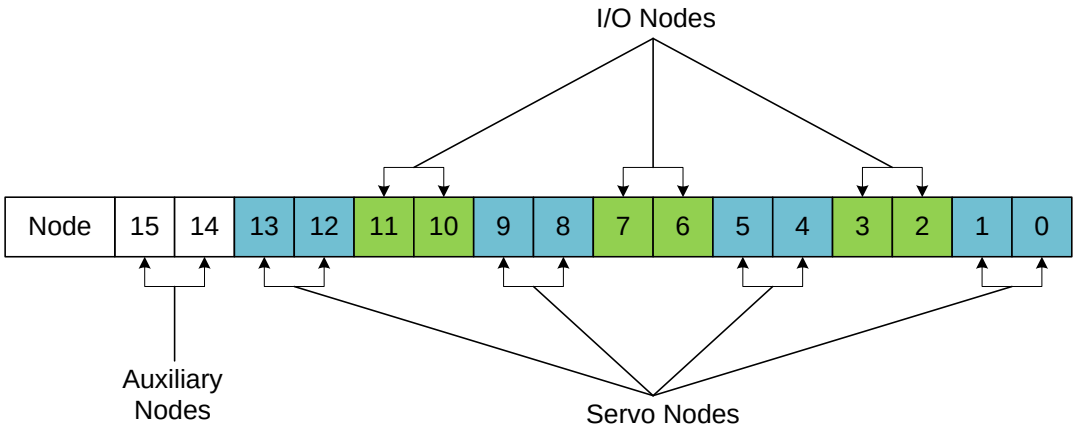
7-Segment LED		
Display	**Description**	**Notes/Cause**
0	Ring Active with no errors	Normal Operation with decimal point blinking
1	One (1) Amp Enable output activated	If an amplifier/motor is connected, it is potentially activated in either open or closed loop form. Exercise caution.
2	Two (1) Amp Enable outputs activated	If an amplifier/motor is connected, it is potentially activated in either open or closed loop form. Exercise caution.
3-9	NA	NA
A	Amplifier Fault	Denotes Amplifier fault condition true. Cleared by enabling amplifier or CLRF.
B	MACRO Ring Break Fault	Break or misconnection in fiber optic or RJ45 ring termination.
C	Configuration change fault	Denotes mismatch between master and slave node configuration. Check MI996 and I6806, etc. for match. Clear with CLRF.
D	MACRO Ring Fault	Ring Data-Error Fault. Too many ring errors or not enough synch packets being received. Node 15 may not be properly enabled.
E	Encoder Fault	Encoder Loss bit condition true (MI927=1). Occurs only when Encoder Loss detection is enabled. Denotes loss of encoder signal. Check encoder wiring and functionality.
F	NA	NA

CONFIGURING WITH TURBO PMAC

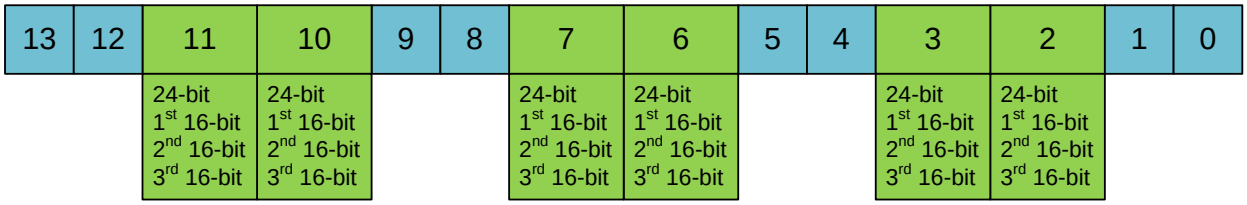
Quick Review: Nodes and Addressing

Each MACRO IC consists of 16 nodes: 2 auxiliary, 8 servo, and 6 I/O nodes.

- Auxiliary nodes are Master/Control registers and internal firmware use.
- Servo nodes carry information such as feedback, commands, and flags for motor control.
- I/O nodes are by default unoccupied and are user configurable for transferring miscellaneous data.



Each I/O node consists of 4 registers; one 24-bit and three 16-bit registers for a total of 72 bits of data.



A given MACRO Station can be populated with either a MACRO8 or MACRO16 CPU:

- MACRO8 supports only 1 MACRO IC (IC#0).
- MACRO16 supports 2 MACRO ICs (IC#0 and IC#1).

The I/O node addresses ($C0XX) for each of the Station MACRO ICs are:

	Station MACRO IC #0 Node Registers					
Node	**2**	**3**	**6**	**7**	**10**	**11**
24-bit	X:$C0A0	X:$C0A4	X:$C0A8	X:$C0AC	X:$C0B0	X:$C0B4
16-bit	X:$C0A1	X:$C0A5	X:$C0A9	X:$C0AD	X:$C0B1	X:$C0B5
16-bit	X:$C0A2	X:$C0A6	X:$C0AA	X:$C0AE	X:$C0B2	X:$C0B6
16-bit	X:$C0A3	X:$C0A7	X:$C0AB	X:$C0AF	X:$C0B3	X:$C0B7

	Station MACRO IC #1 Node Registers					
Node	**2**	**3**	**6**	**7**	**10**	**11**
24-bit	X:$C0E0	X:$C0E4	X:$C0E8	X:$C0EC	X:$C0F0	X:$C0F4
16-bit	X:$C0E1	X:$C0E5	X:$C0E9	X:$C0ED	X:$C0F1	X:$C0F5
16-bit	X:$C0E2	X:$C0E6	X:$C0EA	X:$C0EE	X:$C0F2	X:$C0F6
16-bit	X:$C0E3	X:$C0E7	X:$C0EB	X:$C0EF	X:$C0F3	X:$C0F7

Note

Non-Turbo PMAC2 Ultralite (legacy) I/O node addresses are the same as Station MACRO IC#0 node registers.

A given Turbo PMAC2 Ultralite (or UMAC with ACC-5E) can be populated with up to 4 MACRO ICs (IC#0, IC#1, IC#2, and IC#3) which can be queried with global variable I4902:

If I4902=	**Populated MACRO IC #s**
$0	None
$1	0
$3	0, 1
$7	0, 1, 2
$F	0, 1, 2, 3

And the I/O node addresses ($7XXXX) for each of the Ultralite MACRO ICs are:

	Ring Controller MACRO IC #0 Node Registers					
Station I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
Ultralite I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
24-bit	X:$78420	X:$78424	X:$78428	X:$7842C	X:$78430	X:$78434
16-bit	X:$78421	X:$78425	X:$78429	X:$7842D	X:$78431	X:$78435
16-bit	X:$78422	X:$78426	X:$7842A	X:$7842E	X:$78432	X:$78436
16-bit	X:$78423	X:$78427	X:$7842B	X:$7842F	X:$78433	X:$78437

	Ring Controller MACRO IC #1 Node Registers					
Station I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
Ultralite I/O Node#	**18**	**19**	**22**	**23**	**26**	**27**
24-bit	X:$79420	X:$79424	X:$79428	X:$7942C	X:$79430	X:$79434
16-bit	X:$79421	X:$79425	X:$79429	X:$7942D	X:$79431	X:$79435
16-bit	X:$79422	X:$79426	X:$7942A	X:$7942E	X:$79432	X:$79436
16-bit	X:$79423	X:$79427	X:$7942B	X:$7942F	X:$79433	X:$79437

	Ring Controller MACRO IC #2 Node Registers					
Station I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
Ultralite I/O Node#	**34**	**35**	**38**	**39**	**42**	**43**
24-bit	X:$7A420	X:$7A424	X:$7A428	X:$7A42C	X:$7A430	X:$7A434
16-bit	X:$7A421	X:$7A425	X:$7A429	X:$7A42D	X:$7A431	X:$7A435
16-bit	X:$7A422	X:$7A426	X:$7A42A	X:$7A42E	X:$7A432	X:$7A436
16-bit	X:$7A423	X:$7A427	X:$7A42B	X:$7A42F	X:$7A433	X:$7A437

	Ring Controller MACRO IC #3 Node Registers					
Station I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
Ultralite I/O Node#	**50**	**51**	**54**	**55**	**58**	**59**
24-bit	X:$7B420	X:$7B424	X:$7B428	X:$7B42C	X:$7B430	X:$7B434
16-bit	X:$7B421	X:$7B425	X:$7B429	X:$7B42D	X:$7B431	X:$7B435
16-bit	X:$7B422	X:$7B426	X:$7B42A	X:$7B42E	X:$7B432	X:$7B436
16-bit	X:$7B423	X:$7B427	X:$7B42B	X:$7B42F	X:$7B433	X:$7B437

Setup Overview

This setup assumes that the Ring Master has already been properly configured to run its own local motors.

In order to set up ACC-24M2A with Turbo PMAC, one must:

1. On the Ring Master, enable one (if ACC-24M2A will only use one motor) to two (using two motors) servo nodes (any two unused servo nodes) per ACC-24M2A

   Variables involved:
   I6840/I6890/I6940/I6990 — MACRO IC Ring Configuration/Status
   I6841/I6891/I6941/I6991 — MACRO IC Node Activate Control

   Also, make sure I78 and I80–I82 have been properly configured on the Master.

2. Establish communication between the Master and the ACC-24M2A using MACRO ASCII Mode and enable one or two servo nodes on ACC-24M2A.

   Variables involved:
   MS{anynode},MI11 MACRO Station Station Number
   MS{anynode},MI995 MACRO Ring Configuration/Status
   MS{anynode},MI996 MACRO Node Activate Control

3. Set up Feedback.
4. Set up Flag and Output Command Registers.
5. Configure I^2T Protection.
6. Perform an Open Loop Test.
7. Tune the Servo Loop.

Setup Step 1: MACRO Connectivity

ACC-24M2A requires that the same number of servo nodes be activated through I6841 as there are motors being used on ACC-24M2A; e.g. two servo nodes should be enabled on the Ring Controller if using two motors, one servo node if using only one motor. I80–I82 and I70–I71 must also be configured. There is a specific set of formulas to use for configuring these, as shown in the following example.

Example: Setting up nodes 0 and 1 to control one ACC-24M2A

```
#define RingCheckPeriod       20             ; Suggested Ring Check Period [msec]
#define FatalPackErr          10             ; Suggested Fatal Packet Error Percentage [%]

I80=INT(RingCheckPeriod *8388607/I10+1)      ; Macro Ring Check Period [Servo Cycles]
I81=INT(I80/(I8+1)* FatalPackErr /100)       ; Macro Maximum Ring Error Count
I82=INT(I80/(I8+1)*(100-FatalPackErr)/100)   ; Macro Minimum Sync Packet Count

I6841=$FC003   // Enable nodes 0 and 1, MACRO IC 0 is master
I6840=$4030    // MACRO IC 0 transmits clocks
// for I70 and I71, use the formula I70=MI996 & $3333, I71=MI996 & $3333 //
MSR0,MI996,P33         // Obtain MI996's value and store it in P33
I70=P33&$3333          // Enable flag transfer for nodes 0 and 1
I71= P33&$3333         // Enable flag transfer for nodes 0 and 1
```

Before proceeding, type SAVE, and then $$$.

Setup Step 2: Communicating with ACC-24M2A over MACRO ASCII

ACC-24M2A has no rotary switches to determine its MACRO Station Number. Therefore, ACC-24M2A uses the Ring Order method to obtain its Station Number. Before the ACC-24M2A has been initialized, it will by default be at MACRO Station #255.

If ACC-24M2A is not at factory default, the user can reinitialize it as follows:

- If using MACRO IC #0, to reinitialize ACC-24M2A, type MS$$$***15, then MSSAV15, then MS$$$15.
- If using MACRO IC #1, type MS$$$***31, then MSSAV31, then MS$$$31.
- If using MACRO IC #2, use MS$$$***47, then MSSAV47, then MS$$$47, and so on for other MACRO IC #s.

Then, establishing communication is as follows:

1. Within PeWin32Pro2, in the Terminal Window, type MACSTA255.

2. Type **I11=n** in order to assign this ACC-24M2A to Station **#n.**

Note

ACC-24M2A must be assigned to any unused Station Number (e.g. **I11=1** to assign ACC-24M2A to Station #1).

If a Macro I/O error is received, make sure I6840, I6841 and I79 are set correctly. Also make sure that the unit has not been assigned a Station number already.

If the Station has already been assigned a Station number, there are two options:

A. Find out the station number **n** and enter **MACSTA<n>,** where **n** is the station number, to initiate MACRO ASCII communication with the Station.

B. Reset the station number of all the Stations by entering **MACSTA0** and then enter **STN=0**.

3. Hit **CTRL+T** (**^T**) to exit MACRO ASCII Mode.

4. Type **MACSTAn** where **n** is the Station Number assigned in step 2 (e.g. **MACSTA1** to open ASCII communication with Station #1).

5. Assign the node and master number with MI996.
   For example, to assign the Station to Nodes 0 and 1 on Master IC #0 on the ACC-24M2A, type:

```
MI996=$FC003
```

6. Set MI995=$80.
   Example:

```
MI995=$80
```

7. Hit **CTRL+T** (**^T**) to exit MACRO ASCII Mode.

Setup Step 3: Motor Setup

Clocks

For simplicity, set the max phase and clock dividers the same as the ring controller, but note that the servo rate on the Slave Station is independent and can be set to a different frequency.

```
MS{anynode},I992= Value of I7000 (or I6800)     // Max Phase Clock
MS{anynode},I997= Value of I7001 (or I6801)     // Phase Clock Divider
MS{anynode},I998= Value of I7002 (or I6802)     // Servo Clock Divider
```

Note

The Phase clock on the MACRO Station must be the same as the Ring Controller's, but the Servo Clock can be different.

Example: When Nodes 0 and 1 are being used for ACC-24M2A, setting default clocks

```
MS0,MI992=6527
MS0,MI997=0
MS0,MI998=3
```

Then, issue MSSAV15 followed by MS$$$15 to save the changes on the Station.

Activating Motors and Disabling Commutation

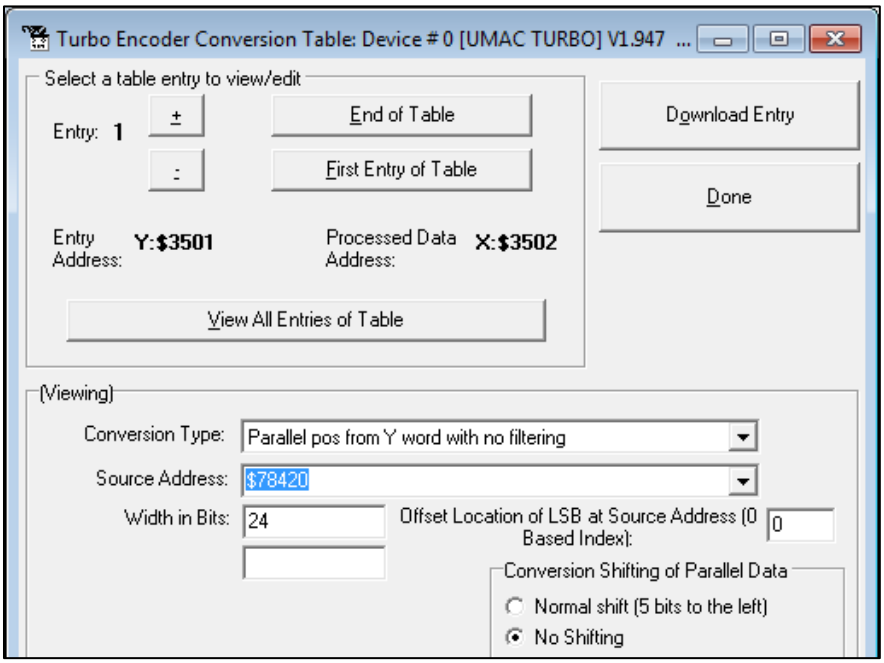
The user must activate the motor he or she wants to use, and then disable commutation for those motors, because the ACC-24M2A runs only non-PMAC-commutated motors. On the Ring Controller, the variable I4900 reports which Servo ICs are present in a Brick, Brick LV, or other Turbo PMAC controller. Knowing that each Servo IC services 4 axes, querying I4900 will reveal how many local channels are occupied and thus the number of the 1st available motor on a Macro Ring. The corresponding Ixx00 (for activating the motor) and Ixx01 (for commutation settings) settings are given in the rightmost columns:

If I4900 Returns	Servo ICs Present	Local Motors	First Motor# On The Ring	Activating 2-Axis Slave	Deactivating Commutation
$0	None	None	1	I100,2,100=1	I101,2,100=0
$1	IC0 only (4-axis)	1 thru 4	5	I500,2,100=1	I501,2,100=0
$3	IC0, and IC1(8-axis)	1 thru 8	9	I900,2,100=1	I901,2,100=0

Motor Feedback

First, the user must make Encoder Conversion Table (ECT) entries on the Ring Controller to read the feedback coming back from the ACC-24M2A on servo nodes. This applies to all feedback types that ACC-24M2A uses.

Use the ECT entry type Parallel Y-Word, No Filtering, 24 bits wide, No Shifting, No Offset. Make sure to select the address based on the correct nodes enabled for this ACC-24M2A. One can set up the ECT entry using PeWin32Pro2 by clicking (from within the software) on Configure→Encoder Conversion Table, showing this window:



The only field the user needs to change on this screen is the Source Address and the Entry Number. Make the Source Address the correct address depending on the node to which this ECT entry corresponds. Make the Entry Number whatever is desired as long as it does not conflict with an ECT entry currently used for another motor.

Example: Motors 1–2 on Nodes 0 and 1, respectively

```
I8000=$2F8420 // Unfiltered parallel pos of location Y:$78420, Node 0
I8001=$18000
I8002=$2F8424 // Unfiltered parallel pos of location Y:$78424, Node 1
I8003=$18000
```

Then, point this motor's Ixx03 and Ixx04 to the numerical hex value Processed Data Address listed the ECT window shown above.

Example: Motors 1–2 Ixx03 and Ixx04 setting:

```
I103=$3502 I104=$3502
I203=$3503 I204=$3503
```

The only exception to this would be if the user wants to use dual feedback on ACC-24M2A and is therefore using both encoder channels for one motor, in which case the user must make one ECT entry for each encoder and point Ixx03 to the position encoder and Ixx04 to the velocity encoder.

Digital A Quad B

The user must configure the Encoder Conversion Table on the ACC-24M2A itself as follows:

Example: ACC-24M2A with two motors, one on Node 0, one on Node 1

```
// ACC-24M2A ECT Setup for Quadrature Encoders
MS0,MI120=$0C090       ; 1/T Extension of Incremental Encoder Ch1
MS0,MI121=$0C098       ; 1/T Extension of Incremental Encoder Ch2

// ACC-24M2A ECT Output Setup
MS0,MI101=$10  ; Output from 1st line of ECT (MI120)
MS0,MI102=$11  ; Output from 2nd line of ECT (MI121)
```

If the user wants to change the direction of the encoder feedback, he or she can either:

- Swap the motor's leads
- Change MS<node>, MI910:
  If MI910=3, set it to 7 (clockwise rotation is positive)
  If MI910=7, set it to 3 (counterclockwise rotation is positive)

Sinusoidal

The user must configure the Encoder Conversion Table on the ACC-24M2A itself as follows:

Example: ACC-24M2A with two motors, one on Node 0, one on Node 1

```
// ACC-24M2A ECT Setup for Sinusoidal Encoders
// Channel 1
MS0,MI120=$F0C090             //  Data Source Address location
MS0,MI121=$FF00               // A/D Converter Address Setup
MS0,MI122=0 // Sine/Cosine Bias

// Channel 2
MS0,MI123=$F0C098             //  Data Source Address location
MS0,MI124=$FF20               // A/D Converter Address Setup

MS0,MI125=0 // Sine/Cosine Bias

// ACC-24M2A ECT Output Setup
MS0,MI101=$12  // Output from 3rd line of ECT (MI122)
MS0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Note that the third line of the entry for each channel (in this example, MI122 for Channel 1 and MI124 for Channel 2) contains the bias in the A/D converter values. This line should contain the value that the A/D converters report when they should ideally report zero. The MACRO Station subtracts this value from both A/D readings before calculating the arctangent. Many users will leave this value at 0, but it is particularly useful to remove the offsets of single-ended analog encoder signals. If it appears that the encoder has an offset, the user can compensate for it in these variables. This line is scaled so that the maximum A/D converter reading provides the full value of the 24-bit register (+/-2^{23}). Generally, it is set by reading the A/D converter values directly as 24-bit values (in this example, from Y:$C090 for Channel 1 and from Y:$C098 for Channel 2), computing the average value over a cycle or cycles, and entering this value here.

For more detail on how the Sinusoidal Interpolation works in PMAC, see Appendix D.

Note

At this point of the setup process, you should be able to move the motor/encoder shaft by hand and see encoder counts in the position window

SSI

ACC-24M2A can be configured to process SSI encoder feedback as a binary parallel word in 12, 16, 20, or 24-bit format. As with all feedback, this data is transferred across the MACRO ring to be used as position and/or velocity feedback. Each SSI device requires three lines of the ECT.

In the second line of each SSI ECT entry, the number of bits to process is specified. So, there are four examples given below.

In the third line, specify the maximum change per servo cycle of the encoder counts that is expected. This is typically equal to 1.25 times the maximum expected velocity of the motor. The units of this entry are whatever the units of the input register are, typically 1/32 of a count. For example, to limit the change in one servo cycle to 64 counts with an input register in units of 1/32 count, this third line would be 64*32 = 2048.

In the examples below, the user must specify the maximum count change per servo cycle on the lines which end with "-User Input" in the comments.

Example: ACC-24M2A with two motors, each with a 12-bit SSI encoder, one on Node 0, one on Node 1

```
#define MaxVelCh1      0 // Maximum count change per servo cycle, Channel 1 -User Input
#define MaxVelCh2      0 // Maximum count change per servo cycle, Channel 2 -User Input

// ACC-24M2A ECT Setup
//Channel 1
MS0,MI120=$30FF54      //  Data Source Address location
MS0,MI121=$000FFF      // 12-bit SSI conversion
MS0,MI122=MaxVelCh1*32

//Channel 2
MS0,MI123=$30FF74      //  Data Source Address location
MS0,MI124=$000FFF      // 12-bit SSI conversion
MS0,MI125=MaxVelCh2*32

// ACC-24M2A ECT output setup
MS0,MI101=$12  // Output from 3rd line of ECT (MI122)
MS0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Example: ACC-24M2A with two motors, each with a 16-bit SSI encoder, one on Node 0, one on Node 1

```
#define MaxVelCh1      0 // Maximum count change per servo cycle, Channel 1 -User Input
#define MaxVelCh2      0 // Maximum count change per servo cycle, Channel 2 -User Input

// ACC-24M2A ECT Setup
//Channel 1
MS0,MI120=$30FF54      //  Data Source Address location
MS0,MI121=$00FFFF      // 16-bit  SSI conversion
MS0,MI122=MaxVelCh1*32

//Channel 2
MS0,MI123=$30FF74      //  Data Source Address location
MS0,MI124=$00FFFF      // 16-bit  SSI conversion
MS0,MI125=MaxVelCh2*32

// ACC-24M2A ECT output setup
MS0,MI101=$12  // Output from 3rd line of ECT (MI122)
MS0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Example: ACC-24M2A with two motors, each with a 20-bit SSI encoder, one on Node 0, one on Node 1

```
#define MaxVelCh1      0 // Maximum count change per servo cycle, Channel 1 -User Input
#define MaxVelCh2      0 // Maximum count change per servo cycle, Channel 2 -User Input

// ACC-24M2A ECT Setup
//Channel 1
MS0,MI120= $30FF54     //  Data Source Address location
MS0,MI121=$0FFFFF      //  20-bit SSI conversion
MS0,MI122=MaxVelCh1*32

//Channel 2
MS0,MI123= $30FF74     //  Data Source Address location
MS0,MI124=$0FFFFF      //  20-bit SSI conversion
MS0,MI125=MaxVelCh2*32

// ACC-24M2A ECT output setup
MS0,MI101=$12  // Output from 3rd line of ECT (MI122)
MS0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Example: ACC-24M2A with two motors, each with a 24-bit SSI encoder, one on Node 0, one on Node 1

```
#define MaxVelCh1      0 // Maximum count change per servo cycle, Channel 1 -User Input
#define MaxVelCh2      0 // Maximum count change per servo cycle, Channel 2 -User Input

// ACC-24M2A ECT Setup
//Channel 1
MS0,MI120= $30FF54     //  Data Source Address location
MS0,MI121=$FFFFFF      //  24-bit SSI conversion
MS0,MI122=MaxVelCh1*32

//Channel 2
MS0,MI123= $30FF74     //  Data Source Address location
MS0,MI124=$FFFFFF      //  24-bit SSI conversion
MS0,MI125=MaxVelCh2*32

// ACC-24M2A ECT output setup
MS0,MI101=$12  // Output from 3rd line of ECT (MI122)
MS0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Note

If the direction decode variable, MS<node>, MI910, is changed the user must save the setting, **MSSAVE{node}** and reset the card **MS$$${node}** before the fractional direction sense matches.

Note

At this point of the setup process, you should be able to move the motor/encoder shaft by hand and see encoder counts in the position window

Resolver

ECT Setup

ACC-24M2A has up to two channels of resolver inputs. The inputs may be used as feedback or master reference signals for the PMAC servo loops. The basic configuration of the drive contains one 10-bit fixed resolution tracking resolver-to-digital (R-to-D) converters, with an optional second resolver when a dual axis driver is ordered. ACC-24M2A creates the AC excitation signal (ResOut) for up to two resolvers, accepts the modulated sine and cosine signals back from these resolvers, demodulates the signals and derives the position of the resolver from the resulting information, in an absolute sense if necessary.

The specifics for this configuration are as follows (Ch1 and Ch2):

Example: ACC-24M2A with two motors, each with a resolver, one on Node 0, one on Node 1, wherein the clockwise direction of the motor's shaft's rotation is positive

```
// ACC-24M2A ECT Setup
// Channel 1
MS0,MI120=$E0FF00             // Data Source Address location ,CW
MS0,MI121=$00FF5C             // A/D Converter Address Setup
MS0,MI122=0 // Sine/Cosine Bias –User Input

// Channel 2 CW
MS0,MI123=$E0FF20             // Data Source Address location, CW
MS0,MI124=$00FF5C             // A/D Converter Address Setup
MS0,MI125=0 // Sine/Cosine Bias –User Input

// ACC-24M2A ECT Output Setup
MS0,MI101=$12  // Output from 3rd line of ECT (MI122)
MS0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Example: ACC-24M2A with two motors, each with a resolver, one on Node 0, one on Node 1, wherein the counterclockwise direction of the motor's shaft's rotation is positive

```
// ACC-24M2A ECT Setup
// Channel 1
MS0,MI120=$E8FF00             // Data Source Address location ,CCW
MS0,MI121=$00FF5C             // A/D Converter Address Setup
MS0,MI122=0 // Sine/Cosine Bias –User Input

// Channel 2 CW
MS0,MI123=$E8FF20             // Data Source Address location, CCW
MS0,MI124=$00FF5C             // A/D Converter Address Setup
MS0,MI125=0 // Sine/Cosine Bias –User Input

// ACC-24M2A ECT Output Setup
MS0,MI101=$12  // Output from 3rd line of ECT (MI122)
MS0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Note that the third line of the entry for each channel (in this example, MI122 for Channel 1 and MI124 for Channel 2) contains the bias in the A/D converter values. This line should contain the value that the A/D converters report when they should ideally report zero. The MACRO Station subtracts this value from both A/D readings before calculating the arctangent. Many users will leave this value at 0, but it is particularly useful to remove the offsets of single-ended analog encoder signals. If it appears that the encoder has an offset, the user can compensate for it in these variables. This line is scaled so that the maximum A/D converter reading provides the full value of the 24-bit register (+/-2^{23}). Generally, it is set by reading the A/D converter values directly as 24-bit values (in this example, from Y:$C090 for Channel 1 and from Y:$C098 for Channel 2), computing the average value over a cycle or cycles, and entering this value here.

Note

If the direction decode variable, MS<node>, MI910, is changed the user must save the setting, **MSSAVE{node}** and reset the card **MS$$${node}** before the fractional direction sense matches.

Configuring Excitation Frequency

After setting up the ECT, the user then must set three MI-Variables for the Resolvers to function correctly.

The ResOut signal (i.e. the Resolver's excitation frequency) emitted from the ACC-24M2A is derived from the Phase Clock frequency of the MACRO set by MI992 and MI997. The user has the ability to select the excitation frequency to be equal with the Phase Clock frequency (default) by setting **MS<node>,MI982** equal to 0. Or, the user can use lower frequencies by increasing the value of MI982.

MI982 affects the excitation frequency as follows:

MI982 Setting	Excitation Frequency
MI982=1	(Phase Clock Frequency)/2
MI982=2	(Phase Clock Frequency)/4
MI982=3	(Phase Clock Frequency)/6

Configuring the Excitation Signal's Gain

Additionally, the user needs to set the Excitation output gain for the system's resolvers by setting **MS<node>,MI981**.

MI981 affects the excitation signal's gain as follows:

MI981 Setting	Excitation Signal Gain
MI981=0	2.5 V_{pp}
MI981=1	5.0 V_{pp}
MI981=2	7.5 V_{pp}
MI981=3	10.0 V_{pp}

Configuring the Excitation Signal's Phase Offset

Finally the resolver excitation phase time offset, **MS<node>, MI980**, needs to be set. The optimum setting of MI980 depends on the L/R time constant of the resolver circuit. Therefore, MI980 should be set interactively to maximize the magnitudes of the feedback ADC values.

For each channel, there are two ADC registers which hold the sin and cosine values. For Channel 1, the base/first ADC register address is Y:$FF00 and the second ADC register address is Y:$FF01; For Channel 2, the base/first ADC register address is Y:$FF20 and the second ADC register address is Y:$FF21. There is no MI-Variable to directly address these registers, so MI198 (*Direct Read/Write Format and Address*) and MI199 (*Direct Read/Write Variable*) will be used here. For each channel, both ADCs should be observed during setup. Notice that MI199 can only be pointed to one register at one time so it must be configured twice throughout the following procedure.

Procedure for Configuring MI980 on Channel 1

The procedure for configuring MI980 for Channel 1 is as follows:

1. In PeWin32Pro2, open a Watch Window (View→Watch Window).
2. Press "Insert" and type **MS<node>,MI199**, where <node> is the node number of this ACC-24M2A's motor (e.g. if this motor is on Node 0, type **MS0,MI199**).
3. In the Terminal Window (View→Terminal), type **MS<node>,MI198=$6DFF00**, where <node> is this motor's node (as in step 2). This points MI199 to the Channel 1's ADC1.
4. Rotate the motor on this channel. Observe **MS<node>,MI199** in the Watch Window. If it saturates to ±32767, the resolver gain (MI981) is too high. Decrease MI981 until the MI199 just barely saturates to ±32767. If it does not saturate, type **MS<node>,MI198=$6DFF01** in the Terminal Window (which sets MI199 to point to Channel 1's ADC2) and then repeat step 4.
5. Set MI199 to point to the ADC which saturated; that is, if ADC1 saturated, type **MS<node>,MI198=$6DFF00** in the Terminal Window, or if ADC2 saturated, type **MS<node>,MI198=$6DFF01** in the Terminal Window.
6. Position the motor's shaft such that the ADC value is close to the maximum value observed throughout one revolution of the motor's shaft. At this point, the other ADC should be close to 0.
7. Increase MI980 by increments of 25. The ADC value should start to increase slowly. If it decreases, instead start with MI980=255 and then decrease MI980 by increments of 25. The ADC value should increase up to a maximum point and then start to decrease again. Set MI980 to the value that produced the largest absolute ADC value achieved throughout the process of adjusting MI980.
8. If the maximum absolute value of this ADC is less than 16,000, increase the gain of the resolver by increasing MI981.

Procedure for Configuring MI980 on Channel 2
The procedure for configuring MI980 for Channel 2 is as follows:

1. In PeWin32Pro2, open a Watch Window (View→Watch Window).
2. Press "Insert" and type **MS<node>,MI199**, where <node> is the node number of this ACC-24M2A's motor (e.g. if this motor is on Node 0, type **MS0,MI199**).
3. In the Terminal Window (View→Terminal), type **MS<node>,MI198=$6DFF20**, where <node> is this motor's node (as in step 2). This points MI199 to the Channel 1's ADC1.
4. Rotate the motor on this channel. Observe **MS<node>,MI199** in the Watch Window. If it saturates to ±32767, the resolver gain (MI981) is too high. Decrease MI981 until the MI199 just barely saturates to ±32767. If it does not saturate, type **MS<node>,MI198=$6DFF21** in the Terminal Window (which sets MI199 to point to Channel 1's ADC2) and then repeat step 4.
5. Set MI199 to point to the ADC which saturated; that is, if ADC1 saturated, type **MS<node>,MI198=$6DFF20** in the Terminal Window, or if ADC2 saturated, type **MS<node>,MI198=$6DFF21** in the Terminal Window.
6. Position the motor's shaft such that the ADC value is close to the maximum value observed throughout one revolution of the motor's shaft. At this point, the other ADC should be close to 0.
7. Increase MI980 by increments of 25. The ADC value should start to increase slowly. If it decreases, instead start with MI980=255 and then decrease MI980 by increments of 25. The ADC value should increase up to a maximum point and then start to decrease again. Set MI980 to the value that produced the largest absolute ADC value achieved throughout the process of adjusting MI980.
8. If the maximum absolute value of this ADC is less than 16,000, increase the gain of the resolver by increasing MI981.

Note

At this point of the setup process, you should be able to move the motor/encoder shaft by hand and see encoder counts in the position window

Flags

On the Ring Controller, the flags (Ixx25) must point to the servo node's flag addresses used for the motors on ACC-24M2A.

Example: Motors 1–2 on Nodes 0 and 1, respectively

```
I125=$3440
I225=$3441
```

Note

These examples configure only motors 1–2. If you are configuring other motors, refer to the Turbo PMAC Software Reference Manual under the entry for Ixx25, under the "Turbo PMAC2 Ultralite" table for a list of the addresses for Ixx25.

Then, on the Ring Controller, the flag control (Ixx24) variable must be set up for each motor on ACC-24M2A as follows:

Ixx24 Setting	Description
$40001	Overtravel limits enabled
$60001	Overtravel limits disabled

Example: Flag Control for Motors 1–2

```
I124,2,100=$40001 // Motor 1-2 have overtravel limits enabled
```

Output Commands

On the Ring Controller, the output command address must be set to the ACC-24M2A's motors' servo node addresses directly.

Example: Motors 1–2 on Nodes 0 and 1, respectively

```
I102=$078420 // Motor 1's output command address is Node 0
I202=$078424 // Motor 2's output command address is Node 1
```

Note

These examples configure only motors 1–2. If you are configuring other motors, refer to the Turbo PMAC Software Reference Manual under the entry for Ixx02, under the under "Turbo PMAC2 Ultralite" table for a list of the addresses for Ixx02.

I2T Settings

The I2T overcurrent protection should be configured for each motor on ACC-24M2A. Below is an example with some formulas for setting up I2T; the user simply needs to fill in the values specified by "-User Input" in the comments on that line:

Example: Configuring I2T Protection for Motors 1–2

```
I15=0                          ; Trigonometric calculation in degrees
#define MaxPhaseFreq   P7000   ; Max Phase Clock [KHz]
#define PWMClk         P7001   ; PWM Clock [KHz]
#define PhaseClk       P7002   ; Phase Clock [KHz]
#define ServoClk       P7003   ; Servo Clock [KHz]
MaxPhaseFreq=117964.8/(2*I6800+3)
PWMClk=117964.8/(4*I6800+6)
PhaseClk=MaxPhaseFreq/(I6801+1)
ServoClk=PhaseClk/(I6802+1)

#define Axis1MinContCurrent   3      ; Continuous Current Limit for Axis 1 [Amps] -User Input
#define Axis1MinPeakCurrent   9      ; Instantaneous Current Limit for Axis 1 [Amps] -User Input
#define Axis1AmpPeakInstCurrent 16.3 ; Peak Instant. Current of Amplifier [Amps] -User Input
#define Axis1I2TOnTime        2      ; Time allowed at peak Current [sec]

// Assuming that motor 1 is the first motor on MACRO
I157=INT(32767*(Axis1MinContCurrent/Axis1AmpPeakInstCurrent)
I169=INT(32767*(Axis1MinPeakCurrent/Axis1AmpPeakInstCurrent)
I158=INT((I169*I169-I157*I157)*ServoClk*1000*Axis1I2TOnTime/(32767*32767))
I257=I157 I269=I169 I258=I158 // Assumes motor 2 is the same as motor 1
```

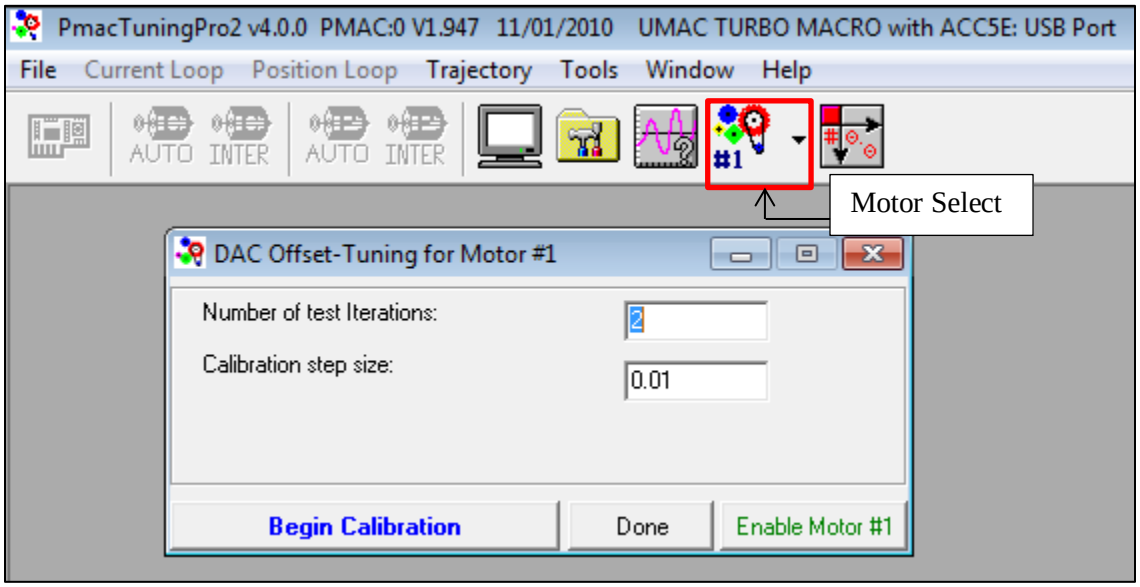
The continuous current limit (Axis1MinContCurrent) and the instantaneous current limit (Axis1MinPeakCurrent) values on the lines with "-User Input" in the comment above should be the smaller of the two limits between your motor and your amplifier's specifications.

DAC Calibration

WARNING

Before performing the DAC Calibration, make sure there is no load attached to the motor, and make sure that the motor can safely and freely move. This step of the setup can generate much motion in the motor.

At this stage in the setup, the user should calibrate the DACs on ACC-24M2A to make sure that when he or she commands 0 volts on the DACs, they actually put out 0 volts. To do this, open PMAC Tuning Pro2 by clicking on Tools→PMAC Tuning Pro2 from within PeWin32Pro2. Then, click Position Loop→DAC Calibration. Make sure there is no load presently connected to the motor. Then, click "Begin Calibration." PMAC will automatically calibrate your DAC. Once it is done, accept the change it makes to Ixx29. You may want to write down this value to add to your setup file. Do this test for both motors (you can select another motor with the Motor Select button):

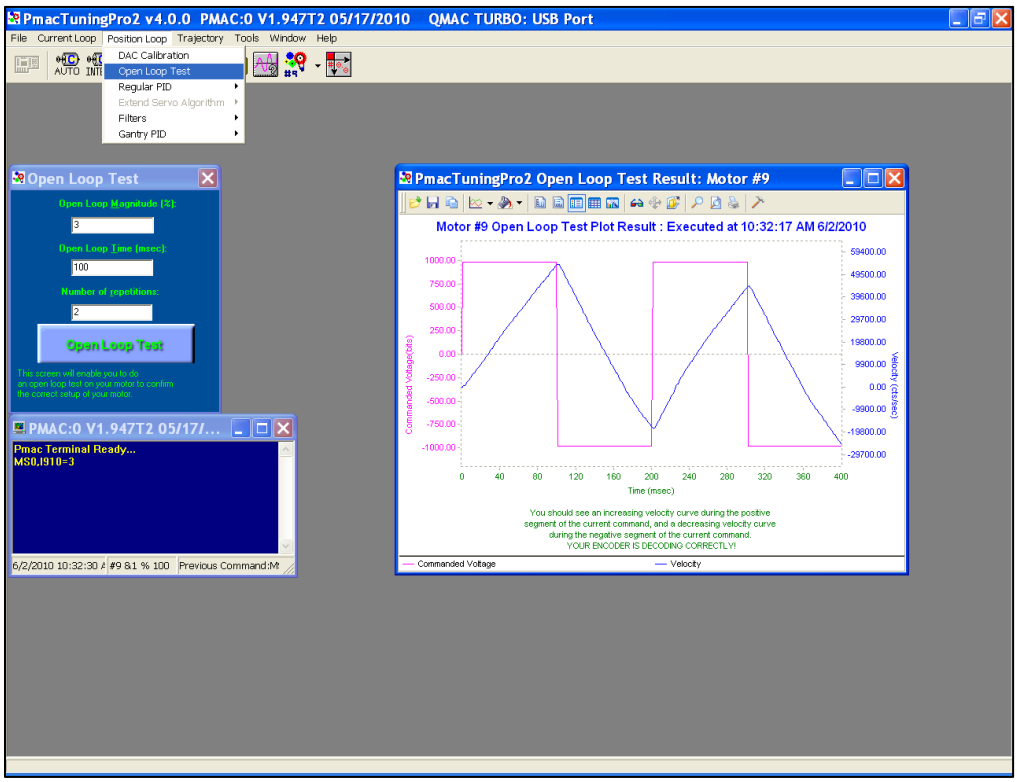


Open Loop Test

WARNING

Before performing the Open Loop Test, make sure there is no load attached to the motor, and make sure that the motor can safely and freely move. This step of the setup can generate much motion in the motor.

The user should now execute an Open-Loop test in order to determine whether the feedback from ACC-24M2A is working properly. To do this, open PMAC Tuning Pro2 from PeWin32Pro2 by clicking on Tools→PMAC Tuning Pro2. Then, click Position Loop→Open Loop Test.



You should see the actual velocity increasing positively while the commanded velocity is positive, the actual velocity decreasing while the commanded velocity is negative.

If you see an erratic response, or an inverted saw tooth, then most likely the encoder decode setting is incorrect. This is on the MACRO side, **MS{node},MI910** has to be changed from 7 to 3, or vice versa.

Servo Loop Tuning

PMAC's Servo Algorithm must be configured to properly control any given system with motors and amplifiers. Configuration is done by adjusting I-Variables (Ixx30 through Ixx35) pertaining to the PID gains. Ixx68 (Friction Feedforward) is also needed. The servo loop gains correspond to I-Variables as follows:

- **Ixx30** Proportional Gain (K_p)
- **Ixx31** Derivative Gain (K_d)
- **Ixx32** Velocity Feedforward (K_{vff})
- **Ixx33** Integral Gain (K_i)
- **Ixx34** Integration Mode
- **Ixx35** Acceleration Feedforward (K_{aff})
- **Ixx68** Friction Feedforward (K_{fff})

Note

The user should connect the load to the motor before tuning the servo loop.

The process of determining proper values of PID gains is called "Tuning." The procedure for tuning is as follows:

1. Set Ixx34 (Motor xx PID Integration Mode) – can be changed on the fly as needed
   =1, position error integration is performed only when Motor xx is not commanding a move
   =0, position error integration is performed always

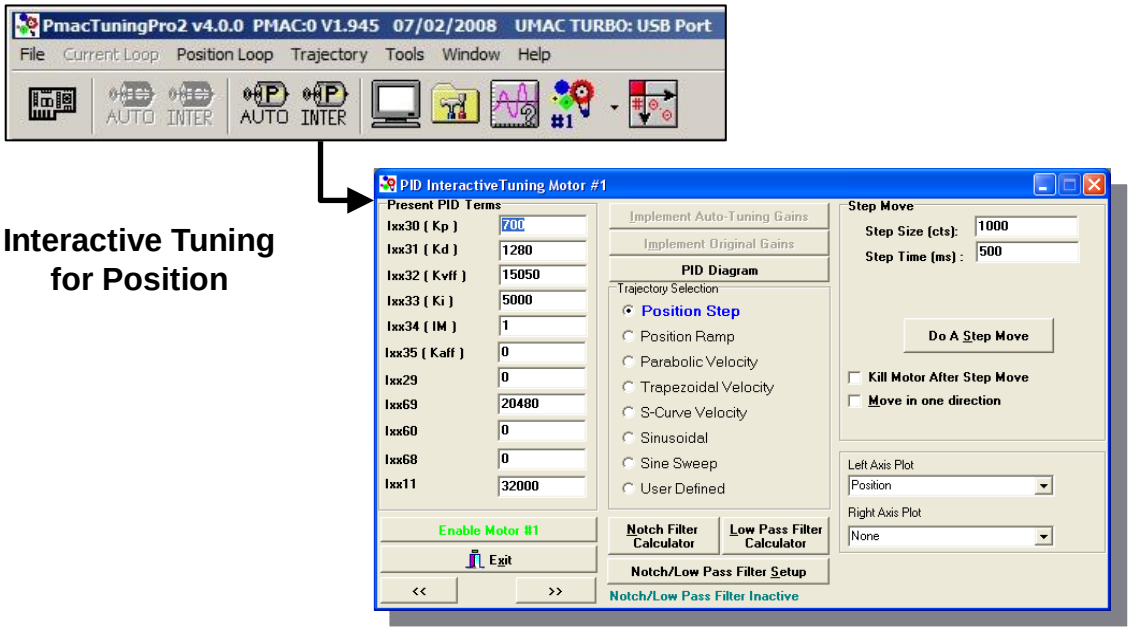
2. Using the Step Response, tune the following parameters in this order:
   Proportional Gain, Kp (Ixx30)
   Derivative Gain, Kd (Ixx31)
   Integral Gain, Ki (Ixx33)

3. Using the Parabolic Move, tune the following parameters in this order:
   Velocity Feedforward, Kvff (Ixx32)
   Acceleration Feedforward, Kaff (Ixx35)
   Friction Feedforward, Kfff (Ixx68)

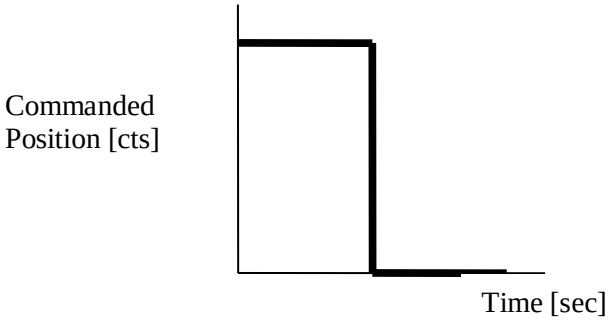
Note

- When tuning the feedforward gains, set Ixx34=1 so that the dynamic behavior of the system may be observed without integrator action. After tuning these, set Ixx34 back to your desired setting.
- Setting Kvff = Kd (Ixx32 = Ixx31) is a good place to start when tuning Kvff.

Steps 2 and 3 should be performed in the Interactive Tuning window in PMAC Tuning Pro2:



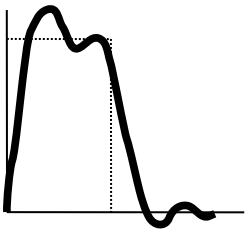
Step 2 (tuning K_p, K_d, and K_i)
Select "Position Step" under "Trajectory Selection." Choose a "Step Size" (under "Step Move") that is within ½ to ¼ of a revolution of the motor if it is a rotary motor, or within ½ to ¼ of one electrical cycle if it is a linear motor. The step move's commanded position profile should look somewhat like this:

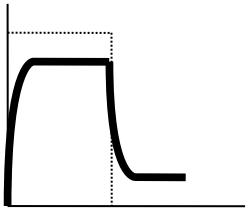


Now, compare your motor's actual position to the commanded position profile. Depending how the actual position looks, adjust the servo loop gains until you achieve the desired response.

Observing the table below, match your actual position response to one of the response shapes below, and then adjust the appropriate gain as listed next to each plot:

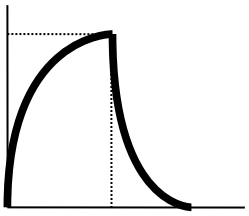
Overshoot and Oscillation
Cause:
Too much Proportional gain or
too little Damping
Fix:
Decrease K_p (Ixx30)
Increase K_d (Ixx31)

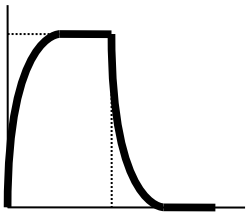




Position Offset
Cause:
Friction or Constant Force
Fix:
Increase K_i (Ixx33)
Increase K_p (Ixx30)

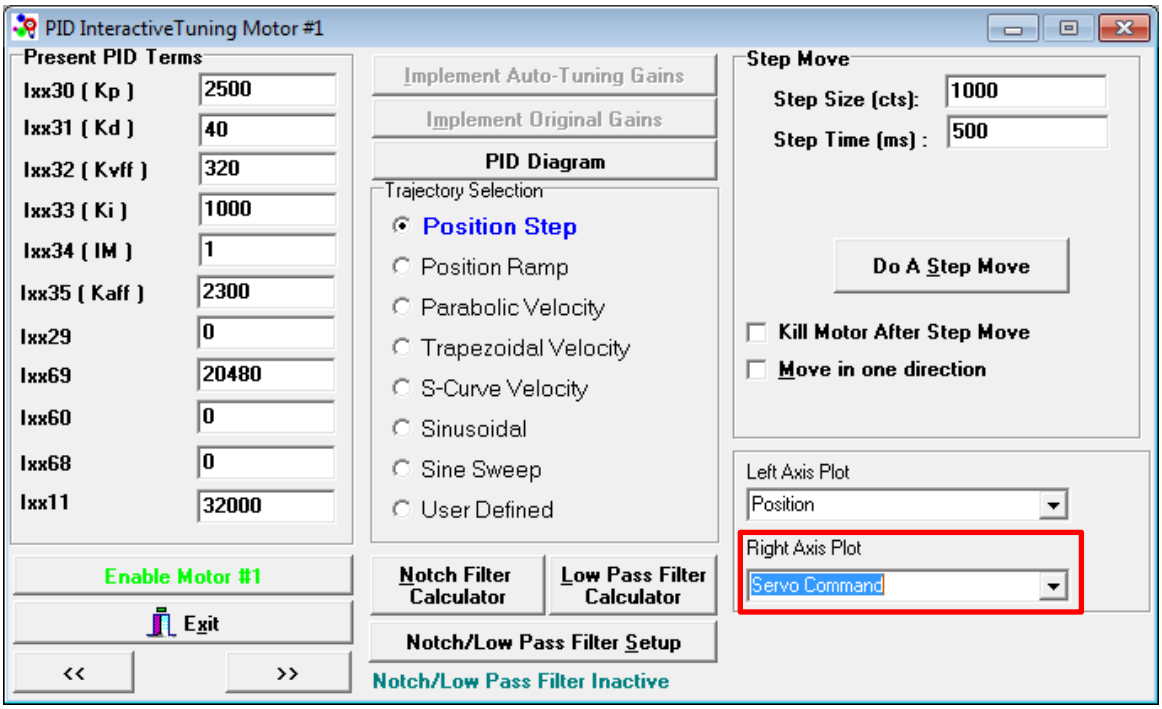
Sluggish Response
Cause:
Too much Damping or
too little Proportional gain
Fix:
Increase K_p (Ixx30) or
Decrease K_d (Ixx31)





Physical System Limitation
Cause:
Limit of the Motor/Amplifier/Load and gain combination
Fix:
Evaluate Performance and
maybe add K_p (Ixx30)

Typically, one should start by increasing K_p until one observes the "Overshoot and Oscillation" condition (upper left corner's plot), and then increase K_d and K_i until the performance goals for the step response are achieved. Be sure when executing the step response that you plot the Servo Command on the Right Axis:

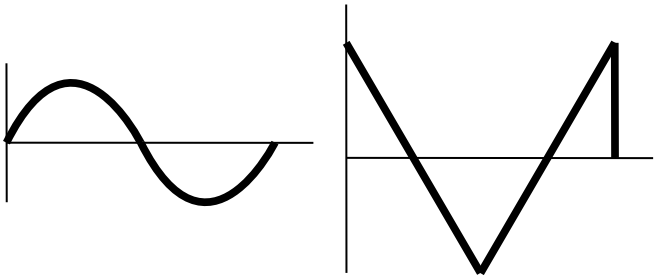


If you see a truncation of the servo command at the beginning of each move, you have reached the maximum output command as determined by Ixx69. In this case, adding more K_p will not improve the Step Response's performance.

Step 3 (Tuning K_{vff}, K_{aff}, and K_{fff})
Select "Parabolic Velocity" under the "Trajectory Selection" in the Interactive Tuning Window. Select a move size and speed that will simulate the fastest, harshest moving conditions you expect your machine to experience. Tune the motor at these settings, and then the motor should be able to handle all easier moves.

After commanding the Parabolic Velocity move, the commanded Velocity Profile and Acceleration Profile should look like this:

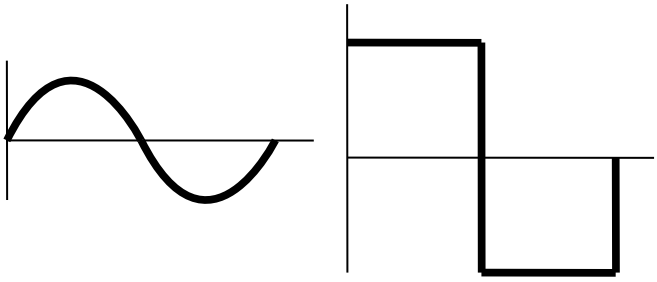
Velocity
Commanded
Profile



Acceleration
Commanded
Profile

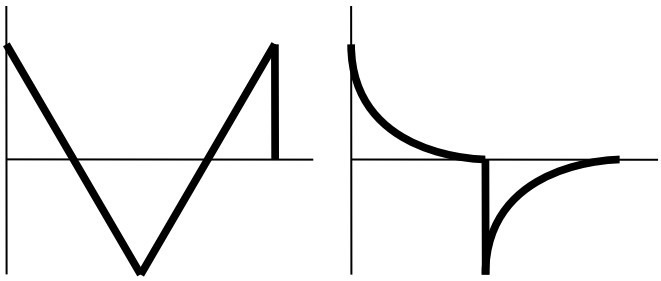
Observing the table below, match your following error response to one of the response shapes below, and then adjust the appropriate gain as listed next to each plot:

High Vel./F.E. Correlation
Cause: Damping
Fix: Increase K_{vff} (Ixx32)



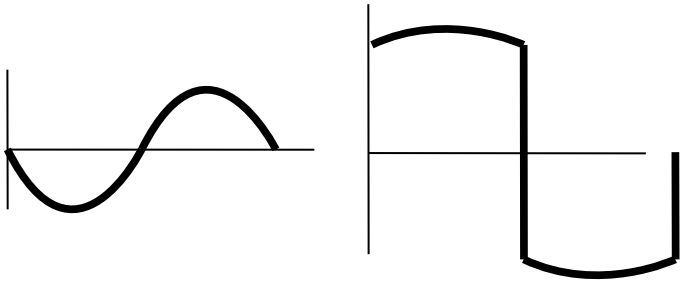
High Vel./F.E. Correlation
Cause: Friction
Fix:
Add Friction Feedforward (Ixx68) and/or turn on Integral Gain (Ixx33, Ixx34)

High Acc./F.E. Correlation
Cause: Inertial Lag
Fix:
Increase K_{aff} (Ixx35)



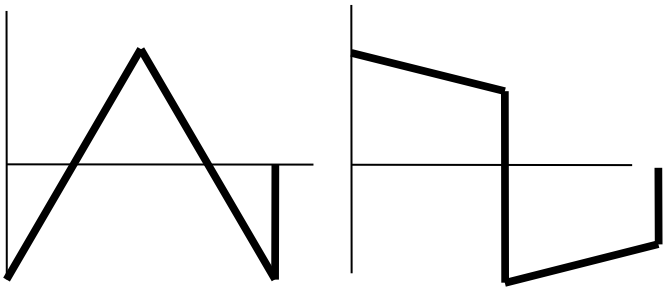
High Acc./F.E. Correlation
Cause:
Physical System Limitation
Fix:
Use softer acceleration or add more Ixx68

Negative Vel./F.E. Correlation
Cause:
Too much Velocity Feedforward
Fix:
Decrease K_{vff} (Ixx32)



High Vel./F.E. Correlation
Cause: Damping & Friction
Fix:
Increase K_{vff} first (Ixx32)
Possibly adjust Ixx68

Negative Acc./F.E. Correlation
Cause:
Too much acceleration Feedforward
Fix:
Decrease K_{aff} (Ixx35)



High Vel./F.E. & Acc./F.E. Correlation
Cause:
Inertial Lag & Friction
Fix:
Increase K_{aff} (Ixx35)
Possibly adjust Ixx68

CONFIGURING WITH POWER PMAC

Quick Review: Nodes and Addressing

Two different types of MACRO interfaces are available for Power PMAC: Gate2-style and Gate3-style.

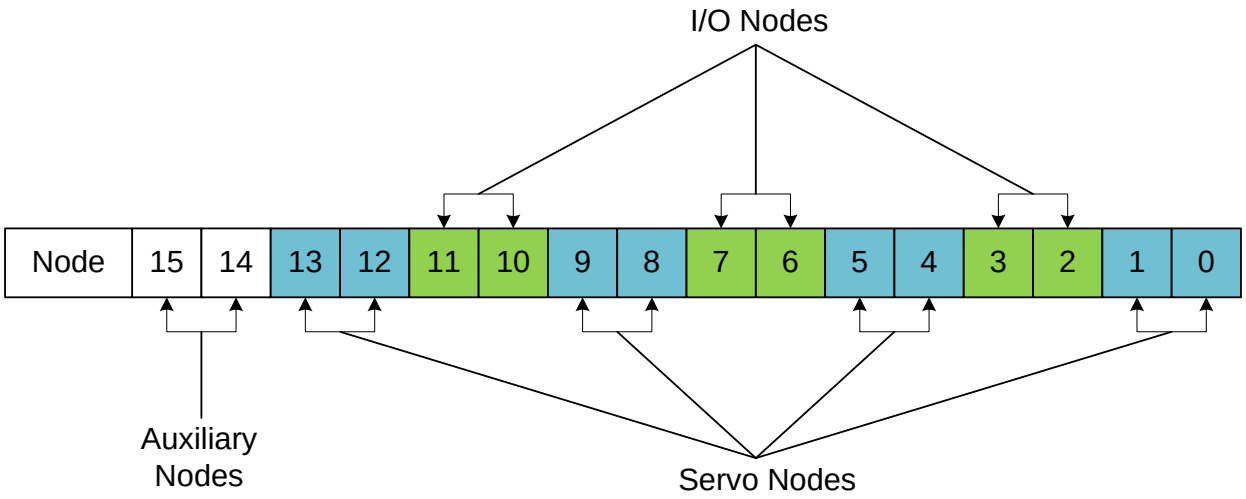
Gate3-style MACRO interfaces have two "banks" of MACRO registers, "Bank A" and "Bank B." As of the date this manual was written, ACC-5E3 is the only Gate3-style MACRO interface for Power PMAC.

Gate2-style MACRO interfaces have up to two ICs, each possessing its own registers for MACRO settings. As of the date this manual was written, ACC-5E is the only Gate2-style MACRO interface for Power PMAC.

Each MACRO IC (for Gate2-style MACRO interfaces) or each MACRO bank (Bank A and Bank B, for Gate3-style MACRO interfaces) consists of 16 nodes: 2 auxiliary, 8 servo, and 6 I/O nodes:

- Auxiliary nodes are Master/Control registers and are for internal firmware use.
- Servo nodes carry information such as feedback, commands, and flags for motor control.
- I/O nodes are by default unoccupied and are user configurable for transferring miscellaneous data.

Each motor that the ring controller controls requires one servo node, and therefore one ACC-5E3 or ACC-5E can control a maximum of 16 motors. The number of I/O nodes used depends on what I/O devices ACC-5E3 or ACC-5E is controlling over the MACRO ring. A visual representation of the nodes' individual functionality is given below:



With Gate3-style MACRO, each node consists of 8 registers: four 32-bit "Input" registers, which can be accessed by the structure **Gate3[*i*].MacroInA[*j*][*k*]** for bank A and **Gate3[*i*].MacroInB[*j*][*k*]** for bank B, and four 32-bit "Output" registers, which can be accessed by the Power PMAC structures **Gate3[*i*].MacroOutA[*j*][*k*]** for bank A and **Gate3[*i*].MacroOutB[*j*][*k*]** for bank B. Gate2-style MACRO interfaces has 4 registers for each node, wherein the input and output data share the registers; they are all grouped into this structure: **Gate2[*i*].Macro[*j*][*k*]**.

Data Organization within Servo Nodes

When controlling non-Gate3 MACRO Stations, each MACRO interface will have its servo node information split up differently within each node ***j*** depending on the commutation method being used. The three modes involved are:

Analog Output Mode
Motor[x].PhaseCtrl = 0
Motor[x].pAdc = 0

UV Commutation Mode (a.k.a. Sinusoidal Commutation Mode)
Motor[x].PhaseCtrl > 0
Motor[x].pAdc = 0

Direct PWM Mode
Motor[x].PhaseCtrl > 0
Motor[x].pAdc > 0 (=**Gate3[*i*].MacroInA[*j*][1]** for MACRO motors on Gate3-style MACRO,
=**Gate2[*i*].Macro[*j*][1]** for Gate2-style)

In Gate3-style MACRO, the contents of each servo node are arranged in each MACRO bank as follows:

Gate3-Style MACRO Bank A Register Structure		
Node Structure	Bit 31	Bit 0
Gate3[i].MacroInA[j][0]	24 bits of feedback information	8 bits of 0
Gate3[i].MacroInA[j][1]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of current sensor ADCA in Direct PWM Mode	16 bits of 0
Gate3[i].MacroInA[j][2]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of current sensor ADCB in Direct PWM Mode	16 bits of 0
Gate3[i].MacroInA[j][3]	16 bits of channel status/flag information	16 bits of 0

Gate3[i].MacroOutA[j][0]	24 bits of servo output command in Analog Output Mode/ 24 bits of DACA output in UV Commutation Mode/ 24 bits of PWMA command in Direct PWM Mode	8 bits of 0
Gate3[i].MacroOutA[j][1]	Not Used in Analog Output Mode/ 16 bits of DACB command in UV Commutation Mode/ 16 bits of PWMB command in Direct PWM Mode	16 bits of 0
Gate3[i].MacroOutA[j][2]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of PWMC command in Direct PWM Mode	16 bits of 0
Gate3[i].MacroOutA[j][3]	16 bits of channel control commands/flag commands	16 bits of 0

Gate3-Style MACRO Bank B Register Structure		
Node Structure	Bit 31	Bit 0
Gate3[i].MacroInB[j][0]	24 bits of feedback information	8 bits of 0
Gate3[i].MacroInB[j][1]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of current sensor ADCA in Direct PWM Mode	16 bits of 0
Gate3[i].MacroInB[j][2]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of current sensor ADCB in Direct PWM Mode	16 bits of 0
Gate3[i].MacroInB[j][3]	16 bits of channel status/flag information	16 bits of 0

Gate3[i].MacroOutB[j][0]	24 bits of servo output command in Analog Output Mode/ 24 bits of DACA output in UV Commutation Mode/ 24 bits of PWMA command in Direct PWM Mode	8 bits of 0
Gate3[i].MacroOutB[j][1]	Not Used in Analog Output Mode/ 16 bits of DACB command in UV Commutation Mode/ 16 bits of PWMB command in Direct PWM Mode	16 bits of 0
Gate3[i].MacroOutB[j][2]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of PWMC command in Direct PWM Mode	16 bits of 0
Gate3[i].MacroOutB[j][3]	16 bits of channel control commands/flag commands	16 bits of 0

In Gate2-style MACRO, the contents of each servo node are arranged in each MACRO IC as follows:

Gate2-Style MACRO Input Register Structure		
Node Structure	Bit 23	Bit 0
Gate2[i].Macro[j][0]	24 bits of feedback information	
Gate2[i].Macro[j][1]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of current sensor ADCA in Direct PWM Mode	8 bits of 0
Gate2[i].Macro[j][2]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of current sensor ADCB in Direct PWM Mode	8 bits of 0
Gate2[i].Macro[j][3]	16 bits of channel status/flag information	8 bits of 0

Gate2-Style MACRO Output Register Structure		
Node Structure	Bit 23	Bit 0
Gate2[i].Macro[j][0]	24 bits of servo output command in Analog Output Mode/ 24 bits of DACA output in UV Commutation Mode/ 24 bits of PWMA command in Direct PWM Mode	
Gate2[i].Macro[j][1]	Not Used in Analog Output Mode/ 16 bits of DACB command in UV Commutation Mode/ 16 bits of PWMB command in Direct PWM Mode	8 bits of 0
Gate2[i].Macro[j][2]	Not Used in Analog Output Mode/ Not Used in UV Commutation Mode/ 16 bits of PWMC command in Direct PWM Mode	8 bits of 0
Gate2[i].Macro[j][3]	16 bits of channel control commands/flag commands	8 bits of 0

Since no Gate3-style MACRO Station products have yet been developed, this is the only node arrangement available until future developments.

Note

I/O Nodes can be arranged in any way desired, and as such, this manual does not have any section describing any specific data arrangement structure within I/O nodes.

Setup Overview

This setup assumes that:

- The Ring Master has already been properly configured to run its own local motors.
- The user is familiar with enabling servo nodes on both the MACRO Ring Controller and Slave Station.

In order to set up ACC-24M2A with Power PMAC, the user must:

1. On the Ring Master, enable one (if ACC-24M2A will only use one motor) to two (using two motors) servo nodes (any two unused servo nodes) per ACC-24M2A.

   Structures involved for Gate3-Style MACRO Interfaces:
   Gate3[i].MacroEnableA — MACRO IC Bank A Node Activate Control
   Gate3[i].MacroEnableB — MACRO IC Bank A Node Activate Control

   Gate3[i].MacroModeA — MACRO IC Bank A Status and Control
   Gate3[i].MacroModeB — MACRO IC Bank B Status and Control

   Structures involved for Gate2-Style MACRO Interfaces:
   Gate2[i].MacroEnable — MACRO IC Node Activate Control
   Gate2[i].MacroMode — MACRO IC Ring Configuration/Status

   Also, make sure **Macro.TestPeriod**, **Macro.TestMaxErrors**, and **Macro.TestReqdSynchs** have been properly configured on the Master.

2. Establish communication between the Master and the ACC-24M2A using MACRO ASCII Mode and enable one or two servo nodes on ACC-24M2A corresponding to the nodes activated on the Master.

   Variables involved:
   MacroSlave{anynode},MI11 MACRO Station Station Number
   MacroSlave{anynode},MI995 MACRO Ring Configuration/Status
   MacroSlave{anynode},MI996 MACRO Node Activate Control

3. Set up Feedback.
4. Set up Flag and Output Command Registers.
5. Configure I2T Protection.
6. Perform an Open Loop Test.
7. Tune the Servo Loop.

Setup Step 1: MACRO Connectivity

ACC-24M2A requires that the same number of servo nodes be activated on the Power PMAC as there are motors being used on ACC-24M2A; e.g. two servo nodes should be enabled on the Ring Controller if using two motors, one servo node if using only one motor. Power PMAC MACRO has three variables for error checking that must also be configured:

Macro.TestPeriod
This is the period in milliseconds at which PMAC checks for errors on the MACRO ring. The recommended value for this variable is 20.

Macro.TestMaxErrors
This is the maximum error count PMAC can receive in one test period (whose duration is specified by **Macro.TestPeriod**) before triggering a fault. The formula for computing this variable is as follows:

Macro.TestMaxErrors=Macro.TestPeriod/10.

Macro.TestReqdSynchs
This is the number of sync packets in one period (whose duration is specified by **Macro.TestPeriod**) that PMAC must receive before triggering an error. The formula for computing this variable is as follows:

Macro.TestReqdSynchs=Macro.TestPeriod-Macro.TestMaxErrors.

Example MACRO Communication Setup

Configuring a Gate3-Style MACRO interface to enable 16 servo nodes and 12 I/O nodes on a Power PMAC, which is acting as a Ring Controller.

```
Sys.WpKey=$AAAAAAAA;

// MACRO Communication Setup
Gate3[0].MacroEnableA=$0FFFFF00;      // Activate 8 Servo Nodes and 6 I/O Nodes of MACRO A
Gate3[0].MacroModeA=$403000;          // Set MACRO A as master
Gate3[0].MacroEnableB=$1FFFFF00;      // Activate 8 Servo Nodes and 6 I/O Nodes of MACRO B
Gate3[0].MacroModeB=$9000;            // Set MACRO B as master to synchronize clock

Macro.TestPeriod=20;                          // MACRO Ring Check Period [msec]
Macro.TestMaxErrors=Macro.TestPeriod/10;      // MACRO Maximum Ring Error Count
Macro.TestReqdSynchs=Macro.TestPeriod - Macro.TestMaxErrors;  // MACRO Minimum Sync Packet Count
```

Setup Step 2: Communicating with ACC-24M2A over MACRO ASCII

ACC-24M2A has no rotary switches to determine its MACRO Station Number. Therefore, ACC-24M2A uses the Ring Order method to obtain its Station Number. Before the ACC-24M2A has been initialized, it will by default be at MACRO Station #255.

If ACC-24M2A is not at factory default, the user can reinitialize it as follows:

- If using MACRO IC #0, to reinitialize ACC-24M2A, type MacroSlave$$$***15, then MacroSlaveSAVE15, then MacroSlave$$$15.
- If using MACRO IC #1, type MacroSlave$$$***31, then MacroSlaveSAVE31, then MacroSlave$$$31.
- If using MACRO IC #2, use MacroSlave$$$***47, then MacroSlaveSAV47, then MacroSlave$$$47, and so on for other MACRO IC #s.

Then, establishing communication is as follows:

1. Within the Power PMAC IDE, in the Terminal Window, type MacroStation255.
2. Type **I11=n** in order to assign this ACC-24M2A to Station **#n.**

Note

ACC-24M2A must be assigned to any unused Station Number (e.g. **I11=1** to assign ACC-24M2A to Station #1).

If a Macro I/O error is received, make sure the Ring Controller's MACRO communication settings are set correctly. Also make sure that the ACC-24M2A has not been assigned a Station number already. If the Station has already been assigned a Station number, there are two options:

A. Find out the station number **n** and enter **MacroStation<n>** (without the angular brackets), where **n** is the station number, to initiate MACRO ASCII communication with the Station.

C. Reset the station number of all the Stations by entering **MacroStation0** and then enter **I11=0**.

3. Type **<MacroStationClose** to exit MACRO ASCII Mode.
4. Type **MacroStation<n>** (without the angular brackets), where **n** is the Station Number assigned in step 2 (e.g. **MacroStation1** to open ASCII communication with Station #1).
5. Assign the node and master number on the ACC-24M2A with MI996.

   For example, to assign the Station to Nodes 0 and 1 on Master IC #0 on the ACC-24M2A, type:

```
MI996=$FC003
```

6. Set MI995=$80.

   Example:

```
MI995=$80
```

7. Type **<MacroStationClose** to exit MACRO ASCII Mode.

Setup Step 3: Motor Setup

Clocks

For simplicity, set the max phase and clock dividers the same as the ring controller, but note that the servo rate on the Geo MACRO Drive is independent and can be set to a different frequency. The variables to use on the MACRO Station for setting clocks are as follows:

MacroSlave{anynode},I992 // Max Phase Clock
MacroSlave{anynode},I997 // Phase Clock Divider
MacroSlave{anynode},I998 // Servo Clock Divider

The formulas for determining what value to which to set these variables are as follows:

$$\text{MI992} = \left(\frac{117964.8}{2 \cdot f_{mp}}\right) - 1,$$

$$\text{MI997} = \left(\frac{f_{mp}}{f_p}\right) - 1,$$

and

$$\text{MI998} = \left(\frac{f_p}{f_s}\right) - 1,$$

where f_{mp} is the desired maximum phase frequency [kHz], f_p is the desired phase clock frequency [kHz], and f_s is the desired servo clock frequency [kHz].

Note

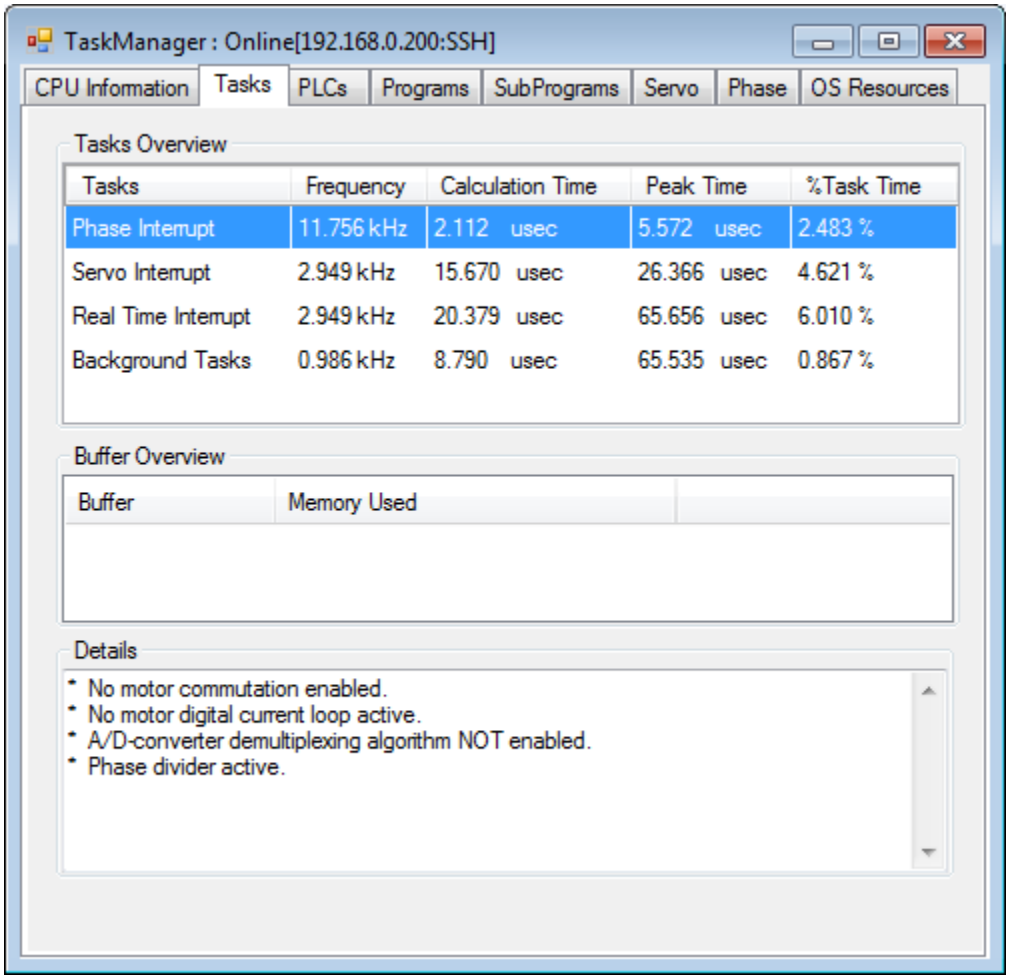
The Phase clock on the MACRO Station must be the same as the Ring Controller's, but the Servo Clock can be different.

Example: When Nodes 0 and 1 are being used for ACC-24M2A, setting default clocks

```
MacroSlave0,MI992=6527
MacroSlave0,MI997=0
MacroSlave0,MI998=3
```

Then, issue MacroSlaveSAVE15 followed by MacroSlave$$$15 to save the changes on the Station.

If you are not sure what your Power PMAC's clocks are, you can check them by (from within the IDE) clicking on Tools→TaskManager, and then clicking on the "Tasks" tab:



The Frequency column displays the clock frequencies in kHz for the different clocks in PMAC.

Phase Interrupt is the Phase Clock
Servo Interrupt is the Servo Clock
Real Time Interrupt is the Real Time Interrupt Clock

After observing these clock frequencies, just use the formulas given on the previous page in order to calculate how to set the clocks on your ACC-24M2A the same as the clocks on your Power PMAC.

Activating Motors and Disabling Commutation

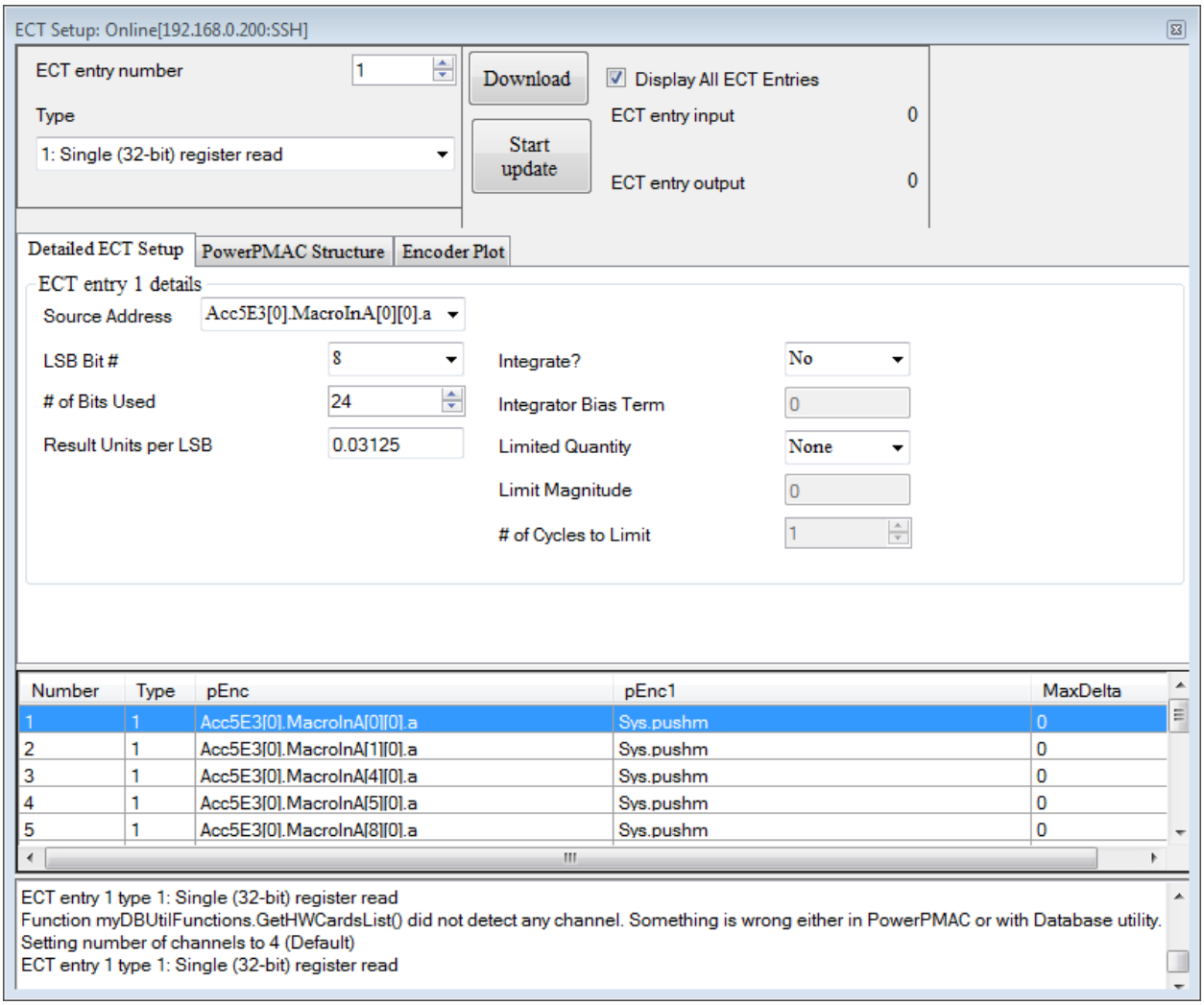
The user must activate the motor he or she wants to use, and then disable commutation for those motors, because the ACC-24M2A runs only non-PMAC-commutated motors. On the Ring Controller, the variable **Sys.Gate1AutoDetect** reports which Gate1-Style Servo ICs are present (such as from ACC-24E2, ACC-24E2A, or ACC-24E2S) in a Power UMAC rack. **Sys.Gate3AutoDetect** detects which Gate3-Style Servo ICs are present in a Power UMAC rack (such as from ACC-24E3). Knowing that each Servo IC services 4 axes, querying **Sys.Gate1AutoDetect** or **Sys.Gate3AutoDetect** will reveal how many local channels are occupied and thus the number of the 1st available motor on a Macro Ring. The corresponding **Motor[x].ServoCtrl** (for activating the motor) and **Motor[x].PhaseCtrl** (for commutation settings) settings are given in the rightmost columns:

AutoDetect Returns	Servo ICs Present	Local Motors	First Motor# On The Ring	Activating 2-Axis Slave	Deactivating Commutation
$0	None	None	1	`Motor[1].ServoCtrl=1` `Motor[2].ServoCtrl=1`	`Motor[1].PhaseCtrl=0` `Motor[2].PhaseCtrl=0`
$1	IC0 only (4 axes)	1–4	5	`Motor[5].ServoCtrl=1` `Motor[6].ServoCtrl=1`	`Motor[5].PhaseCtrl=0` `Motor[6].PhaseCtrl=0`
$3	IC0 and IC1 (8 axes)	1–8	9	`Motor[9].ServoCtrl=1` `Motor[10].ServoCtrl=1`	`Motor[9].PhaseCtrl=0` `Motor[10].PhaseCtrl=0`

Motor Feedback

First, the user must make Encoder Conversion Table (ECT) entries on the Ring Controller to read the feedback coming back from the ACC-24M2A on servo nodes. This applies to all feedback types that ACC-24M2A uses.

Use the ECT entry type "Single (32-bit) register read), LSB Bit #: 8, # of Bits Used: 24. Make sure to select the address based on the correct nodes enabled for this ACC-24M2A. One can set up the ECT entry using the Power PMAC IDE by clicking (from within the software) on Delta Tau→Configure→Encoder Conversion Table, showing this window:



The only field the user needs to change on this screen is the Source Address and the Entry Number. There should be one entry for each motor, each using the Source Address that corresponds to the node used for this motor. That is, make the Source Address match the address of the feedback register of this motor's node. Make the Entry Number whatever is desired as long as it does not conflict with an ECT entry currently used for another motor.

For Gate3-Style MACRO interfaces, use **Gate3[*i*].MacroInA[*j*][0].a** for the encoder feedback Source Address. For Gate2-Style interfaces, use **Gate2[*i*].Macro[*j*][0].a**.

Example: Motors 1–2 on Nodes 0 and 1, respectively, with a Gate3-Style MACRO Interface on card index 0

```
EncTable[1].pEnc=Gate3[0].MacroInA[0][0].a;
EncTable[1].pEnc1=Sys.pushm;
EncTable[1].Index1=8;
EncTable[1].Index2=8;
EncTable[1].Index3=0;
EncTable[1].Index4=0;
EncTable[1].ScaleFactor=1/exp2(EncTable[1].Index1 + 5);

EncTable[2].pEnc=Gate3[0].MacroInA[1][0].a;
EncTable[2].pEnc1=Sys.pushm;
EncTable[2].Index1=8;
EncTable[2].Index2=8;
EncTable[2].Index3=0;
EncTable[2].Index4=0;
EncTable[2].ScaleFactor=1/exp2(EncTable[2].Index1 + 5);
```

Then, point this motor's **Motor[x].pEnc** (position feedback pointer) and **Motor[x].pEnc2** (velocity feedback pointer) to the numerical hex value Processed Data Address listed the ECT window shown above. The user must also set the Encoder Type (**Motor[x].EncType**) to 4.

Example: Motors 1–2 Motor[x].pEnc and Motor[x].pEnc2 setting:

```
Motor[1].pEnc=EncTable[1].a;
Motor[1].pEnc2=EncTable[1].a;
Motor[2].pEnc=EncTable[2].a;
Motor[2].pEnc2=EncTable[2].a;

Motor[1].EncType=4;
Motor[2].EncType=4;
```

Typically, **Motor[x].pEnc** and **Motor[x].pEnc2** point to the same address. However, if the user wants to use dual feedback on ACC-24M2A and is therefore using both encoder channels for one motor, the user must make one ECT entry for each encoder and point **Motor[x].pEnc** to the position encoder's ECT entry's output address and **Motor[x].pEnc2** to the velocity encoder's ECT entry's output address.

Digital A Quad B

The user must configure the Encoder Conversion Table on the ACC-24M2A itself as follows:

Example: ACC-24M2A with two motors, one on Node 0, one on Node 1

```
// ACC-24M2A ECT Setup for Quadrature Encoders
MacroSlave0,MI120=$0C090      // 1/T Extension of Incremental Encoder Ch1
MacroSlave0,MI121=$0C098      // 1/T Extension of Incremental Encoder Ch2

// ACC-24M2A ECT Output Setup
MacroSlave0,MI101=$10  // Output from 1st line of ECT (MI120)
MacroSlave0,MI102=$11  // Output from 2nd line of ECT (MI121)
```

If the user wants to change the direction of the encoder feedback, he or she can either:

- Swap the motor's leads
- Change **MacroSlave<node>, MI910**:
  If MI910=3, set it to 7 (clockwise rotation is positive)
  If MI910=7, set it to 3 (counterclockwise rotation is positive)

Sinusoidal

The user must configure the Encoder Conversion Table on the ACC-24M2A itself as follows:

Example: ACC-24M2A with two motors, one on Node 0, one on Node 1

```
// ACC-24M2A ECT Setup for Sinusoidal Encoders
// Channel 1
MacroSlave0,MI120=$F0C090            //  Data Source Address location
MacroSlave0,MI121=$FF00              // A/D Converter Address Setup
MacroSlave0,MI122=0 // Sine/Cosine Bias –User Input

// Channel 2
MacroSlave0,MI123=$F0C098            //  Data Source Address location
MacroSlave0,MI124=$FF20              // A/D Converter Address Setup

MacroSlave0,MI125=0 // Sine/Cosine Bias –User Input

// ACC-24M2A ECT Output Setup
MacroSlave0,MI101=$12  // Output from 3rd line of ECT (MI122)
MacroSlave0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Note that the third line of the entry for each channel (in this example, MI122 for Channel 1 and MI124 for Channel 2) contains the bias in the A/D converter values. The user should enter into this line (indicated by "–User Input" in the comment of that line) the value that the A/D converters report when they should ideally report zero. The MACRO Station subtracts this value from both A/D readings before calculating the arctangent. Many users will leave this value at 0, but to remove the offsets of single-ended analog encoder signals is particularly useful. If it appears that the encoder has an offset, the user can compensate for it in these variables. This line is scaled so that the maximum A/D converter reading provides the full value of the 24-bit register (+/-2^{23}). Generally, it is set by reading the A/D converter values directly off of the Station as 24-bit values (in this example, from Y:$C090 for Channel 1 and from Y:$C098 for Channel 2), computing the average value over a cycle or cycles, and entering this value here.

For more detail on how Sinusoidal Interpolation works in PMAC, see Appendix D.

Note

At this point of the setup process, you should be able to move the motor/encoder shaft by hand and see encoder counts in the position window

SSI

ACC-24M2A can be configured to process SSI encoder feedback as a binary parallel word in 12, 16, 20, or 24-bit format. As with all feedback, this data is transferred across the MACRO ring to be used as position and/or velocity feedback. Each SSI device requires three lines of the ECT.

In the second line of each SSI ECT entry, the number of bits to process is specified. So, there are four examples given below.

In the third line, specify the maximum change per servo cycle of the encoder counts that is expected. This is typically equal to 1.25 times the maximum expected velocity of the motor. The units of this entry are whatever the units of the input register are, typically 1/32 of a count. For example, to limit the change in one servo cycle to 64 counts with an input register in units of 1/32 count, this third line would be 64*32 = 2048.

In the examples below, the user must specify the maximum count change per servo cycle on the lines which end with "-User Input" in the comments.

Example: ACC-24M2A with two motors, each with a 12-bit SSI encoder, one on Node 0, one on Node 1

```
#define MaxVelCh1      0 // Maximum count change per servo cycle, Channel 1 -User Input
#define MaxVelCh2      0 // Maximum count change per servo cycle, Channel 2 -User Input

// ACC-24M2A ECT Setup
//Channel 1
MacroSlave0,MI120=$30FF54     //  Data Source Address location
MacroSlave0,MI121=$000FFF     // 12-bit SSI conversion
MacroSlave0,MI122=MaxVelCh1*32

//Channel 2
MacroSlave0,MI123=$30FF74     //  Data Source Address location
MacroSlave0,MI124=$000FFF     // 12-bit SSI conversion
MacroSlave0,MI125=MaxVelCh2*32

// ACC-24M2A ECT output setup
MacroSlave0,MI101=$12  // Output from 3rd line of ECT (MI122)
MacroSlave0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Example: ACC-24M2A with two motors, each with a 16-bit SSI encoder, one on Node 0, one on Node 1

```
#define MaxVelCh1      0 // Maximum count change per servo cycle, Channel 1 -User Input
#define MaxVelCh2      0 // Maximum count change per servo cycle, Channel 2 -User Input

// ACC-24M2A ECT Setup
//Channel 1
MacroSlave0,MI120=$30FF54     //  Data Source Address location
MacroSlave0,MI121=$00FFFF     // 16-bit  SSI conversion
MacroSlave0,MI122=MaxVelCh1*32

//Channel 2
MacroSlave0,MI123=$30FF74     //  Data Source Address location
MacroSlave0,MI124=$00FFFF     // 16-bit  SSI conversion
MacroSlave0,MI125=MaxVelCh2*32

// ACC-24M2A ECT output setup
MacroSlave0,MI101=$12  // Output from 3rd line of ECT (MI122)
MacroSlave0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Example: ACC-24M2A with two motors, each with a 20-bit SSI encoder, one on Node 0, one on Node 1

```
#define MaxVelCh1      0 // Maximum count change per servo cycle, Channel 1 -User Input
#define MaxVelCh2      0 // Maximum count change per servo cycle, Channel 2 -User Input

// ACC-24M2A ECT Setup
//Channel 1
MacroSlave0,MI120=$30FF54     //  Data Source Address location
MacroSlave0,MI121=$0FFFFF     //  20-bit SSI conversion
MacroSlave0,MI122=MaxVelCh1*32

//Channel 2
MacroSlave0,MI123=$30FF74     //  Data Source Address location
MacroSlave0,MI124=$0FFFFF     //  20-bit SSI conversion
MacroSlave0,MI125=MaxVelCh2*32

// ACC-24M2A ECT output setup
MacroSlave0,MI101=$12  // Output from 3rd line of ECT (MI122)
MacroSlave0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Example: ACC-24M2A with two motors, each with a 24-bit SSI encoder, one on Node 0, one on Node 1

```
#define MaxVelCh1      0 // Maximum count change per servo cycle, Channel 1 -User Input
#define MaxVelCh2      0 // Maximum count change per servo cycle, Channel 2 -User Input

// ACC-24M2A ECT Setup
//Channel 1
MacroSlave0,MI120=$30FF54     //  Data Source Address location
MacroSlave0,MI121=$FFFFFF     //  24-bit SSI conversion
MacroSlave0,MI122=MaxVelCh1*32

//Channel 2
MacroSlave0,MI123=$30FF74     //  Data Source Address location
MacroSlave0,MI124=$FFFFFF     //  24-bit SSI conversion
MacroSlave0,MI125=MaxVelCh2*32

// ACC-24M2A ECT output setup
MacroSlave0,MI101=$12  // Output from 3rd line of ECT (MI122)
MacroSlave0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Note

If the direction decode variable, **MacroSlave<node>, MI910**, is changed the user must save the setting, **MSSAVE{node}** and reset the card **MS$$${node}** before the fractional direction sense matches.

Note

At this point of the setup process, you should be able to move the motor/encoder shaft by hand and see encoder counts in the position window

Resolver

ECT Setup

ACC-24M2A has up to two channels of resolver inputs. The inputs may be used as feedback or master reference signals for the PMAC servo loops. The basic configuration of the drive contains one 10-bit fixed resolution tracking resolver-to-digital (R-to-D) converters, with an optional second resolver when a dual axis driver is ordered. ACC-24M2A creates the AC excitation signal (ResOut) for up to two resolvers, accepts the modulated sine and cosine signals back from these resolvers, demodulates the signals and derives the position of the resolver from the resulting information, in an absolute sense if necessary.

The specifics for this configuration are as follows (Ch1 and Ch2):

Example: ACC-24M2A with two motors, each with a resolver, one on Node 0, one on Node 1, wherein the clockwise direction of the motor's shaft's rotation is positive

```
// ACC-24M2A ECT Setup
// Channel 1
MacroSlave0,MI120=$E0FF00             //  Data Source Address location ,CW
MacroSlave0,MI121=$00FF5C             // A/D Converter Address Setup
MacroSlave0,MI122=0 // Sine/Cosine Bias -User Input

// Channel 2 CW
MacroSlave0,MI123=$E0FF20             //  Data Source Address location, CW
MacroSlave0,MI124=$00FF5C             // A/D Converter Address Setup
MacroSlave0,MI125=0 // Sine/Cosine Bias -User Input

// ACC-24M2A ECT Output Setup
MacroSlave0,MI101=$12  // Output from 3rd line of ECT (MI122)
MacroSlave0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Example: ACC-24M2A with two motors, each with a resolver, one on Node 0, one on Node 1, wherein the counterclockwise direction of the motor's shaft's rotation is positive

```
// ACC-24M2A ECT Setup
// Channel 1
MacroSlave0,MI120=$E8FF00             //  Data Source Address location ,CCW
MacroSlave0,MI121=$00FF5C             // A/D Converter Address Setup
MacroSlave0,MI122=0 // Sine/Cosine Bias -User Input

// Channel 2 CW
MacroSlave0,MI123=$E8FF20             //  Data Source Address location, CCW
MacroSlave0,MI124=$00FF5C             // A/D Converter Address Setup
MacroSlave0,MI125=0 // Sine/Cosine Bias -User Input

// ACC-24M2A ECT Output Setup
MacroSlave0,MI101=$12  // Output from 3rd line of ECT (MI122)
MacroSlave0,MI102=$15  // Output from 6th line of ECT (MI125)
```

Note that the third line of the entry for each channel (in this example, MI122 for Channel 1 and MI124 for Channel 2) contains the bias in the A/D converter values. This line (indicated by "-User Input" in the comments on that line) should contain the value that the A/D converters report when they should ideally report zero. The MACRO Station subtracts this value from both A/D readings before calculating the arctangent. Many users will leave this value at 0, but it is particularly useful to remove the offsets of single-ended analog encoder signals. If it appears that the encoder has an offset, the user can compensate for it in these variables. This line is scaled so that the maximum A/D converter reading provides the full value of the 24-bit register (+/-2^{23}). Generally, it is set by reading the A/D converter values directly as 24-bit values (in this example, from Y:$C090 for Channel 1 and from Y:$C098 for Channel 2), computing the average value over a cycle or cycles, and entering this value here.

Note

If the direction decode variable, **MacroSlave<node>, MI910**, is changed the user must save the setting, **MSSAVE{node}** and reset the card **MS$$${node}** before the fractional direction sense matches.

Configuring Excitation Frequency

After setting up the ECT, the user then must set three MI-Variables for the Resolvers to function correctly.

The ResOut signal (i.e. the Resolver's excitation frequency) emitted from the ACC-24M2A is derived from the Phase Clock frequency of the MACRO set by MI992 and MI997. The user has the ability to select the excitation frequency to be equal with the Phase Clock frequency (default) by setting **MacroSlave<node>,MI982** equal to 0. Or, the user can use lower frequencies by increasing the value of MI982.

MI982 affects the excitation frequency as follows:

MI982 Setting	Excitation Frequency
MI982=1	(Phase Clock Frequency)/2
MI982=2	(Phase Clock Frequency)/4
MI982=3	(Phase Clock Frequency)/6

Configuring the Excitation Signal's Gain

Additionally, the user needs to set the Excitation output gain for the system's resolvers by setting **MacroSlave<node>,MI981**.

MI981 affects the excitation signal's gain as follows:

MI981 Setting	Excitation Signal Gain
MI981=0	2.5 V_{pp}
MI981=1	5.0 V_{pp}
MI981=2	7.5 V_{pp}
MI981=3	10.0 V_{pp}

Configuring the Excitation Signal's Phase Offset

Finally the resolver excitation phase time offset, **MacroSlave<node>, MI980**, needs to be set. The optimum setting of MI980 depends on the L/R time constant of the resolver circuit. Therefore, MI980 should be set interactively to maximize the magnitudes of the feedback ADC values.

For each channel, there are two ADC registers which hold the sin and cosine values. For Channel 1, the base/first ADC register address is Y:$FF00 and the second ADC register address is Y:$FF01; For Channel 2, the base/first ADC register address is Y:$FF20 and the second ADC register address is Y:$FF21. There is no MI-Variable to directly address these registers, so MI198 (*Direct Read/Write Format and Address*) and MI199 (*Direct Read/Write Variable*) will be used here. For each channel, both ADCs should be observed during setup. Notice that MI199 can only be pointed to one register at one time so it must be configured twice throughout the following procedure.

Procedure for Configuring MI980 on Channel 1

The procedure for configuring MI980 for Channel 1 is as follows:

9. In the Power PMAC IDE, open a Watch Window (click Delta Tau→View→Watch).
10. Into a field in the Watch Window, type **MacroSlave<node>,MI199**, where **<node>** is the node number of this ACC-24M2A's motor (e.g. if this motor is on Node 0, type **MacroSlave0,MI199**).
11. In the Terminal Window (from within the IDE, click Delta Tau→View→Terminal), type **MacroSlave<node>,MI198=$6DFF00**, where **<node>** is this motor's node (as in step 2). This points MI199 to the Channel 1's ADC1.
12. Rotate the motor on this channel. Observe **MacroSlave<node>,MI199** in the Watch Window. If it saturates to ±32767, the resolver gain (MI981) is too high. Decrease MI981 until the MI199 just barely saturates to ±32767. If it does not saturate, type **MacroSlave<node>,MI198=$6DFF01** in the Terminal Window (which sets MI199 to point to Channel 1's ADC2) and then repeat step 4.
13. Set MI199 to point to the ADC which saturated; that is, if ADC1 saturated, type **MacroSlave<node>,MI198=$6DFF00** in the Terminal Window, or if ADC2 saturated, type **MacroSlave<node>,MI198=$6DFF01** in the Terminal Window.
14. Position the motor's shaft such that the ADC value is close to the maximum value observed throughout one revolution of the motor's shaft. At this point, the other ADC should be close to 0.
15. Increase MI980 by increments of 25. The ADC value should start to increase slowly. If it decreases, instead start with MI980=255 and then decrease MI980 by increments of 25. The ADC value should increase up to a maximum point and then start to decrease again. Set MI980 to the value that produced the largest absolute ADC value achieved throughout the process of adjusting MI980.
16. If the maximum absolute value of this ADC is less than 16,000, increase the gain of the resolver by increasing MI981.

Procedure for Configuring MI980 on Channel 2
The procedure for configuring MI980 for Channel 2 is as follows:

9. In the Power PMAC IDE, open a Watch Window (click Delta Tau→View→Watch Window).
10. Into a field in the Watch Window, type **MacroSlave<node>,MI199**, where **<node>** is the node number of this ACC-24M2A's motor (e.g. if this motor is on Node 0, type **MacroSlave0,MI199**).
11. In the Terminal Window (from within the IDE, click Delta Tau→View→Terminal), type **MacroSlave<node>,MI198=$6DFF20**, where **<node>** is this motor's node (as in step 2). This points MI199 to the Channel 1's ADC1.
12. Rotate the motor on this channel. Observe **MacroSlave<node>,MI199** in the Watch Window. If it saturates to ±32767, the resolver gain (MI981) is too high. Decrease MI981 until the MI199 just barely saturates to ±32767. If it does not saturate, type **MacroSlave<node>,MI198=$6DFF21** in the Terminal Window (which sets MI199 to point to Channel 1's ADC2) and then repeat step 4.
13. Set MI199 to point to the ADC which saturated; that is, if ADC1 saturated, type **MacroSlave<node>,MI198=$6DFF20** in the Terminal Window, or if ADC2 saturated, type **MacroSlave<node>,MI198=$6DFF21** in the Terminal Window.
14. Position the motor's shaft such that the ADC value is close to the maximum value observed throughout one revolution of the motor's shaft. At this point, the other ADC should be close to 0.
15. Increase MI980 by increments of 25. The ADC value should start to increase slowly. If it decreases, instead start with MI980=255 and then decrease MI980 by increments of 25. The ADC value should increase up to a maximum point and then start to decrease again. Set MI980 to the value that produced the largest absolute ADC value achieved throughout the process of adjusting MI980.
16. If the maximum absolute value of this ADC is less than 16,000, increase the gain of the resolver by increasing MI981.

Note

At this point of the setup process, you should be able to move the motor/encoder shaft by hand and see encoder counts in the position window

Flags

On the Ring Controller, the motors' flag pointers must point to the servo node's flag addresses used for the motors on ACC-24M2A.

Example: Motors 1–2 on Nodes 0 and 1, respectively, using ACC-5E3 in a Power UMAC

```
Motor[1].pEncCtrl=Acc5E3[0].MacroOutA[0][3].a
Motor[1].pEncStatus=Acc5E3[0].MacroInA[0][3].a
Motor[1].pAmpEnable=Acc5E3[0].MacroOutA[0][3].a
Motor[1].pAmpFault=Acc5E3[0].MacroInA[0][3].a
Motor[1].pCaptFlag=Acc5E3[0].MacroInA[0][3].a
Motor[1].pPhaseEnc=Acc5E3[0].MacroInA[0][0].a
Motor[1].pAdc=Acc5E3[0].MacroInA[0][1].a

Motor[2].pEncCtrl=Acc5E3[0].MacroOutA[1][3].a
Motor[2].pEncStatus=Acc5E3[0].MacroInA[1][3].a
Motor[2].pAmpEnable=Acc5E3[0].MacroOutA[1][3].a
Motor[2].pAmpFault=Acc5E3[0].MacroInA[1][3].a

Motor[2].pCaptFlag=Acc5E3[0].MacroInA[1][3].a
Motor[2].pPhaseEnc=Acc5E3[0].MacroInA[1][0].a
Motor[2].pAdc=Acc5E3[0].MacroInA[1][1].a
```

Then, on the Ring Controller, the flag control variables must be set up for each motor on ACC-24M2A.

Example: Motors 1–2 on Nodes 0 and 1, respectively, with overtravel limits enabled, using ACC-5E3 in a Power UMAC

```
Motor[1].pLimits=Acc5E3[0].MacroInA[0][3].a
Motor[1].LimitBits=25
Motor[1].CaptPosRound=1
Motor[1].CaptPosRightShift=0
Motor[1].CaptPosLeftShift=13
Motor[1].CaptFlagBit=19
Motor[1].AmpFaultBit=23
Motor[1].AmpEnableBit=22
Motor[1].AmpFaultLevel=0

Motor[2].pLimits=Acc5E3[0].MacroInA[1][3].a
Motor[2].LimitBits=25
Motor[2].CaptPosRound=1
Motor[2].CaptPosRightShift=0
Motor[2].CaptPosLeftShift=13
Motor[2].CaptFlagBit=19
Motor[2].AmpFaultBit=23
Motor[2].AmpEnableBit=22
Motor[2].AmpFaultLevel=0
```

Example: Motors 1–2 on Nodes 0 and 1, respectively, with overtravel limits disabled

```
Motor[1].pLimits=0;
Motor[1].LimitBits=25
Motor[1].CaptPosRound=1
Motor[1].CaptPosRightShift=0
Motor[1].CaptPosLeftShift=13
Motor[1].CaptFlagBit=19
Motor[1].AmpFaultBit=23
Motor[1].AmpEnableBit=22
Motor[1].AmpFaultLevel=0

Motor[2].pLimits=0;
Motor[2].LimitBits=25
Motor[2].CaptPosRound=1
Motor[2].CaptPosRightShift=0
Motor[2].CaptPosLeftShift=13
Motor[2].CaptFlagBit=19
Motor[2].AmpFaultBit=23
Motor[2].AmpEnableBit=22
Motor[2].AmpFaultLevel=0
```

Output Commands

On the Ring Controller, the output command address must be set to the ACC-24M2A's motors' servo node addresses directly.

Example: Motors 1–2 on Nodes 0 and 1, respectively, using ACC-5E3 in a Power UMAC

```
Motor[1].pDac=Acc5E3[0].MacroOutA[0][0].a;
Motor[2].pDac=Acc5E3[0].MacroOutA[1][0].a;
```

Note

These examples configure only motors 1–2. If you are configuring other motors, refer to the Power PMAC Software Reference Manual for the different settings the aforementioned structures can take.

I²T Settings

The I²T overcurrent protection should be configured for each motor on ACC-24M2A. Below is an example with some formulas for setting up I²T; the user simply needs to fill in the values specified by "-User Input" in the comments on that line:

Example: Configuring I²T Protection for Motors 1–2

```
#define Axis1MinContCurrent    3       ; Continuous Current Limit for Axis 1 [Amps] -User Input
#define Axis1MinPeakCurrent    9       ; Instantaneous Current Limit for Axis 1 [Amps] -User Input
#define Axis1AmpPeakInstCurrent 16.3   ; Peak Instant. Current of Amplifier [Amps] -User Input
#define Axis1I2TOnTime         2       ; Time allowed at peak current [sec]

// Assuming that motor 1 is the first motor on MACRO
Motor[1].I2TSet=32767*Axis1MinContCurrent/Axis1AmpPeakInstCurrent
Motor[1].MaxDac=32767*Axis1MinPeakCurrent/Axis1AmpPeakInstCurrent
Motor[1].I2TTrip=(Motor[1].MaxDac*Motor[1].MaxDac - Motor[1].I2TSet*Motor[1].I2TSet +
                    Motor[1].IdCmd*Motor[1].IdCmd)*Axis1I2TOnTime
Motor[2].I2TSet=Motor[1].I2TSet // Assumes motor 2 is the same as motor 1
Motor[2].MaxDac=Motor[1].MaxDac // Assumes motor 2 is the same as motor 1
Motor[2].I2TTrip=Motor[1].I2TTrip // Assumes motor 2 is the same as motor 1
```

The continuous current limit (Axis1MinContCurrent) and the instantaneous current limit (Axis1MinPeakCurrent) values on the lines with "-User Input" in the comment above should be the smaller of the two limits between your motor and your amplifier's specifications.

DAC Calibration

WARNING

Before performing the DAC Calibration, make sure there is no load attached to the motor, and make sure that the motor can safely and freely move. This step of the setup can generate much motion in the motor.

At this stage in the setup, the user should calibrate the DACs on ACC-24M2A to make sure that when he or she commands 0 volts on the DACs, they actually put out 0 volts. You can do this by means of the automatic DAC calibration program from the Power PMAC System Setup program, easily accessible by means of a TelNet Terminal connection to your PMAC. To open a TelNet connection, from the Windows Start Menu click Start→Run (or just type into the search field if it is Windows Vista/7) and then type:

```
telnet 192.168.0.200
```

Put your Power PMAC's IP Address in the place of 192.168.0.200 (the default IP address) shown above. TelNet will prompt you for a "powerpmac login"; here type:

```
root
```

The password is:

```
deltatau
```

Then, type:

```
cd setup
dir
```

calcdacbias should appear as a program in the list. This is a program that receives two arguments as follows:

calcdacbias <MotorNumber> [<Iterations>]

<MotorNumber> is the number of the motor whose DAC bias you want to calculate. [<Iterations>] is the number of times you want the program to iterate; more iterations generally yields a more accurate result.

Example
To calculate the DAC bias of motor 1 with 10 iterations, type:

```
calcdacbias 1 10
```

Once the program completes, it will issue a command to PMAC to change the DAC bias for this motor. For example, after entering the above command, the final iteration of the program will print something to the effect of:

Command: Motor[1].DacBias = -139.200000

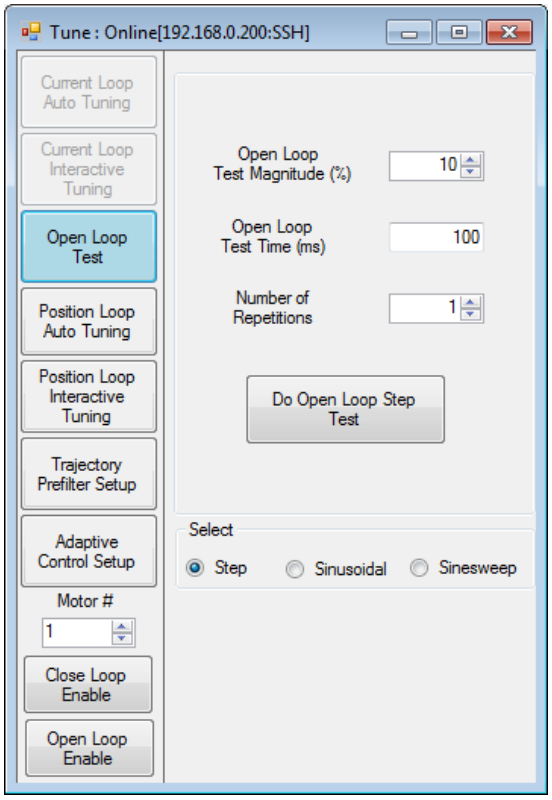
You may want to write down this number and put it into your motor setup file in your project within the Power PMAC IDE for future reference.

Open Loop Test

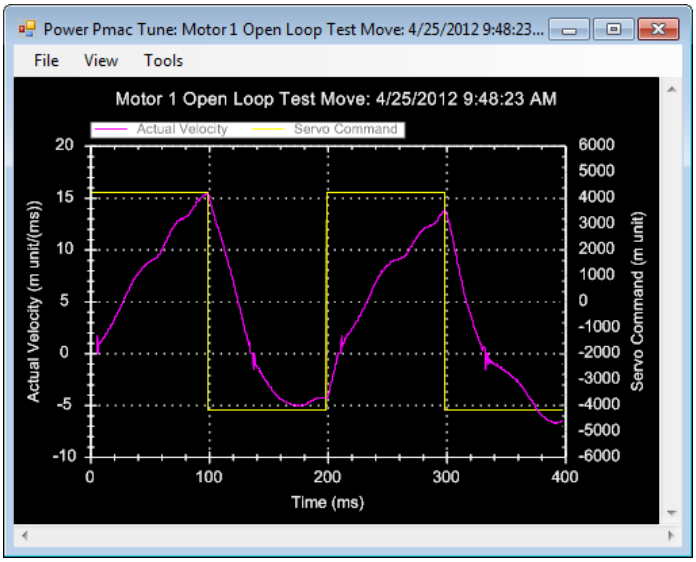
WARNING

Before performing the Open Loop Test, make sure there is no load attached to the motor, and make sure that the motor can safely and freely move. This step of the setup can generate much motion in the motor.

The user should now execute an Open-Loop test in order to determine whether the feedback from ACC-24M2A is working properly. To do this, open Tuning from the Power PMAC IDE by clicking on Tools→Tune. Then, click the Open Loop Test button on the left:



You should see the actual velocity increasing positively while the commanded velocity is positive, the actual velocity decreasing while the commanded velocity is negative, looking something like this:



If you see an erratic response, or an inverted saw tooth, then most likely the encoder decode setting is incorrect. To change this, change **MS{node},MI910** for this motor from 7 to 3, or vice versa.

Servo Loop Tuning

PMAC's Servo Algorithm must be configured to properly control any given system with motors and amplifiers. Configuration is done by adjusting setup structures pertaining to the PID gains. Friction Feedforward is also needed. The servo loop gains correspond to structures as follows:

- **Motor[x].Servo.Kp** Proportional Gain (K_p)
- **Motor[x].Servo.Kvfb** Derivative Gain (K_d)
- **Motor[x].Servo.Kvff** Velocity Feedforward (K_{vff})
- **Motor[x[.Servo.Ki** Integral Gain (K_i)
- **Motor[x].Servo.SwZvInt** Integration Mode
- **Motor[x].Kaff** Acceleration Feedforward (K_{aff})
- **Motor[x].Kfff** Friction Feedforward (K_{fff})

Note

The user should connect the load to the motor before tuning the servo loop.

The process of determining proper values of PID gains is called "Tuning." The procedure for tuning is as follows:

4. Set **Motor[x].Servo.SwZvInt** (Motor xx PID Integration Mode) – can be changed on the fly as needed
   =1, position error integration is performed only when Motor xx is not commanding a move
   =0, position error integration is performed always

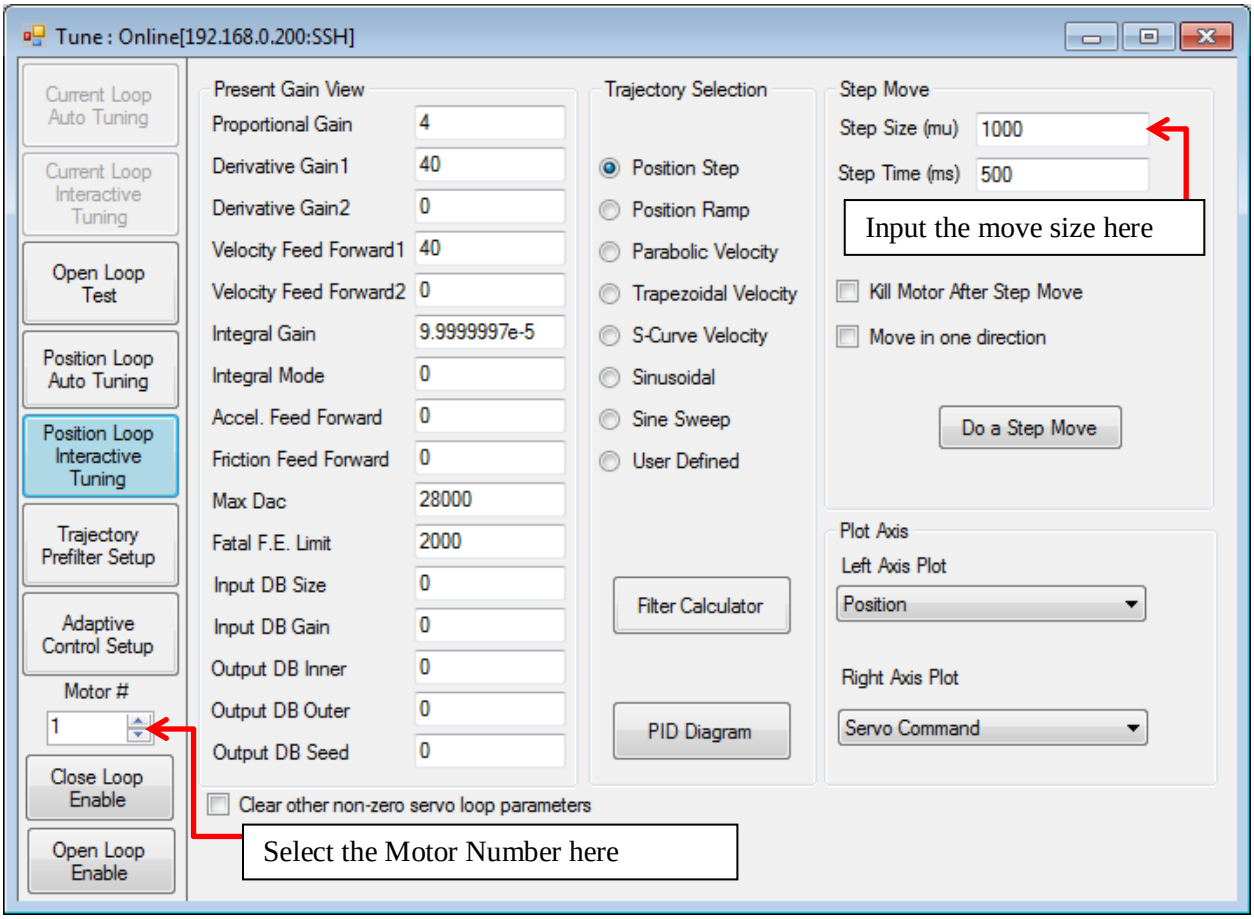
5. Using the Step Response, tune the following parameters in this order:
   Proportional Gain, Kp (**Motor[x].Servo.Kp**)
   Derivative Gain, Kd (**Motor[x].Servo.Kvfb**)
   Integral Gain, Ki (**Motor[x[.Servo.Ki**)

6. Using the Parabolic Move, tune the following parameters in this order:
   Velocity Feedforward, Kvff (**Motor[x].Servo.Kvff**)
   Acceleration Feedforward, Kaff (**Motor[x].Kaff**)
   Friction Feedforward, Kfff (**Motor[x].Kfff**)

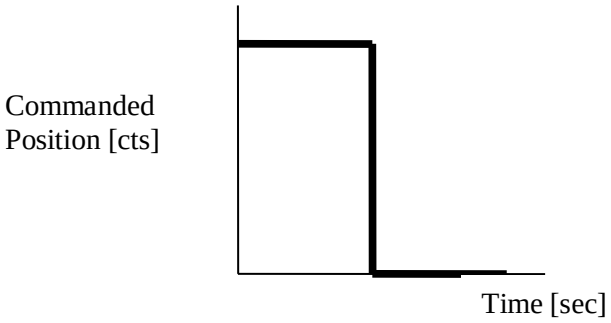
Note

- When tuning the feedforward gains, set **Motor[x].Servo.SwZvInt** =1 so that the dynamic behavior of the system may be observed without integrator action. After tuning these, set **Motor[x].Servo.SwZvInt** back to your desired setting.
- Setting Kvff = Kd (**Motor[x].Servo.Kvff** = **Motor[x].Servo.Kvfb**) is a good place to start when tuning Kvff.

Steps 2 and 3 should be performed in the Interactive Tuning window in Tuning:



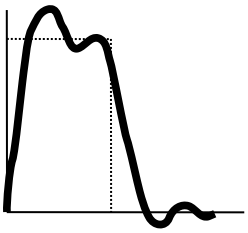
Step 2 (tuning K_p, K_d, and K_i)
Select "Position Step" under "Trajectory Selection." Choose a "Step Size" (under "Step Move") that is within ½ to ¼ of a revolution of the motor if it is a rotary motor, or within ½ to ¼ of one electrical cycle if it is a linear motor. The step move's commanded position profile should look somewhat like this:

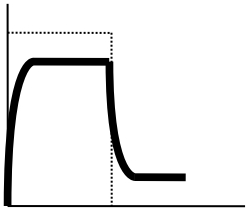


Now, compare your motor's actual position to the commanded position profile. Depending how the actual position looks, adjust the servo loop gains until you achieve the desired response.

Observing the table below, match your actual position response to one of the response shapes below, and then adjust the appropriate gain as listed next to each plot:

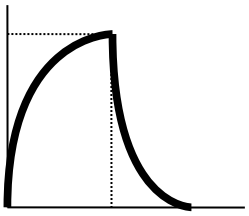
Overshoot and Oscillation
Cause:
Too much Proportional gain or
too little Damping
Fix:
Decrease K_p
Increase K_d

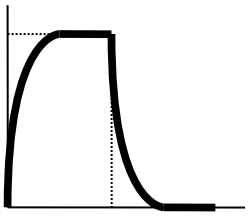




Position Offset
Cause:
Friction or Constant Force
Fix:
Increase K_i
Increase K_p

Sluggish Response
Cause:
Too much Damping or too little Proportional gain
Fix:
Increase K_p or
Decrease K_d

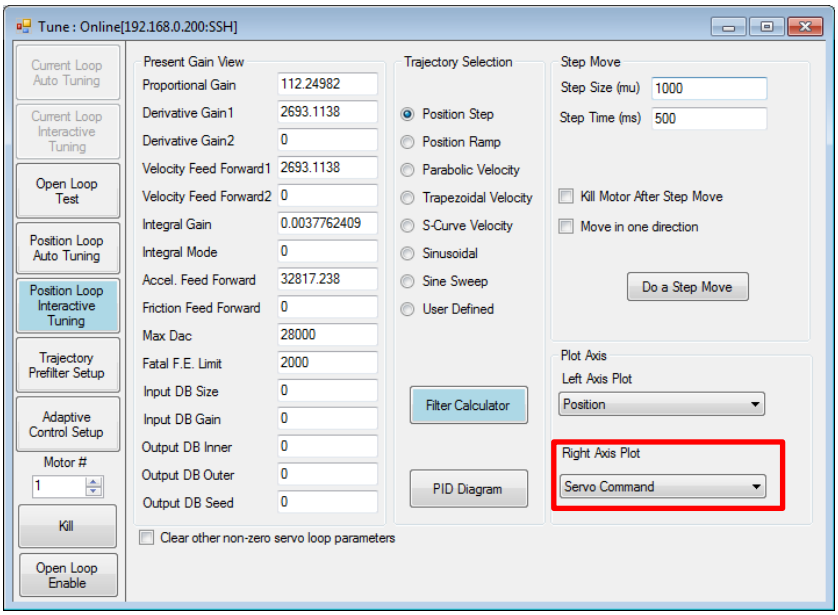




Physical System Limitation
Cause:
Limit of the Motor/Amplifier/Load and gain combination
Fix:
Evaluate Performance and
maybe add K_p

Typically, one should start by increasing K_p until one observes the "Overshoot and Oscillation" condition (upper left corner's plot), and then increase K_d and K_i until the performance goals for the step response are achieved. Be sure when executing the step response that you plot the Servo Command on the Right Axis (see image on right below).

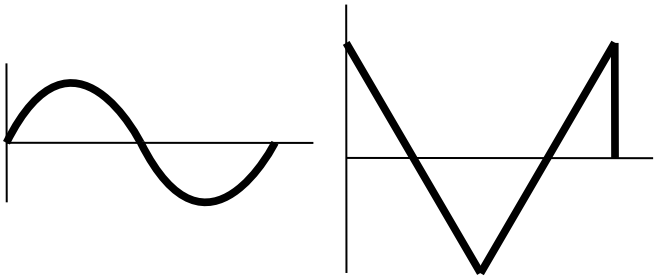
If you see a truncation of the servo command at the beginning of each move, you have reached the maximum output command as determined by **Motor[x].MaxDac**. In this case, adding more K_p will not improve the Step Response's performance.



Step 3 (Tuning K_{vff}, K_{aff}, and K_{fff})
Select "Parabolic Velocity" under the "Trajectory Selection" in the Interactive Tuning Window. Select a move size and speed that will simulate the fastest, harshest moving conditions you expect your machine to experience. Tune the motor at these settings, and then the motor should be able to handle all easier moves.

After commanding the Parabolic Velocity move, the commanded Velocity Profile and Acceleration Profile should look like this:

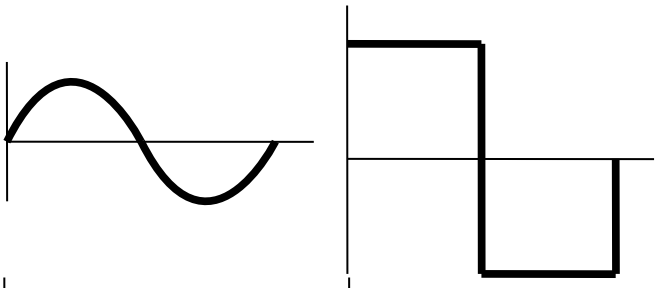
Velocity
Commanded
Profile



Acceleration
Commanded
Profile

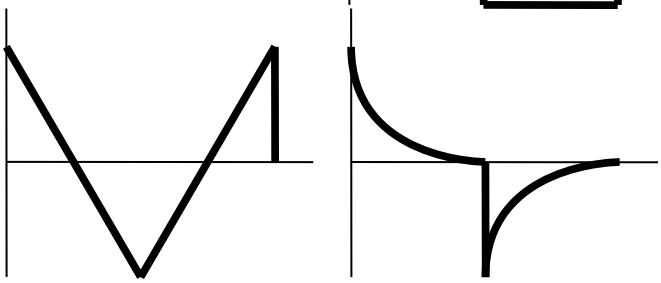
Observing the table below, match your actual position response to one of the response shapes below, and then adjust the appropriate gain as listed next to each plot:

High Vel./F.E. Correlation
Cause: Damping
Fix: Increase K_{vff}



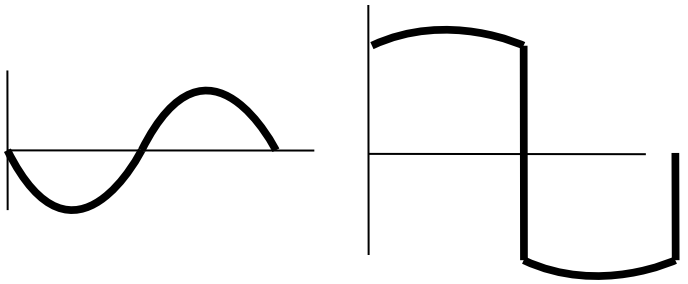
High Vel./F.E. Correlation
Cause: Friction
Fix:
Add K_{fff}
and/or turn on Integral Gain (K_i)

High Acc./F.E. Correlation
Cause: Inertial Lag
Fix:
Increase K_{aff}



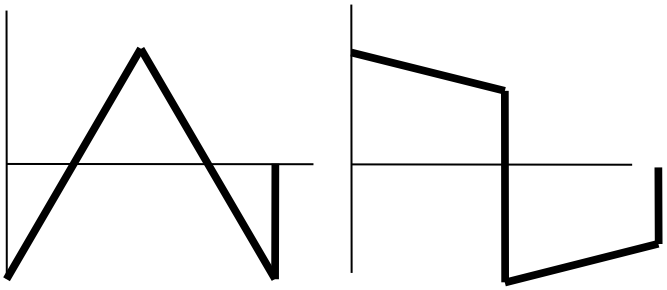
High Acc./F.E. Correlation
Cause:
Physical System Limitation
Fix:
Use softer acceleration or add more K_{fff}

Negative Vel./F.E. Correlation
Cause:
Too much Velocity Feedforward
Fix:
Decrease K_{vff}



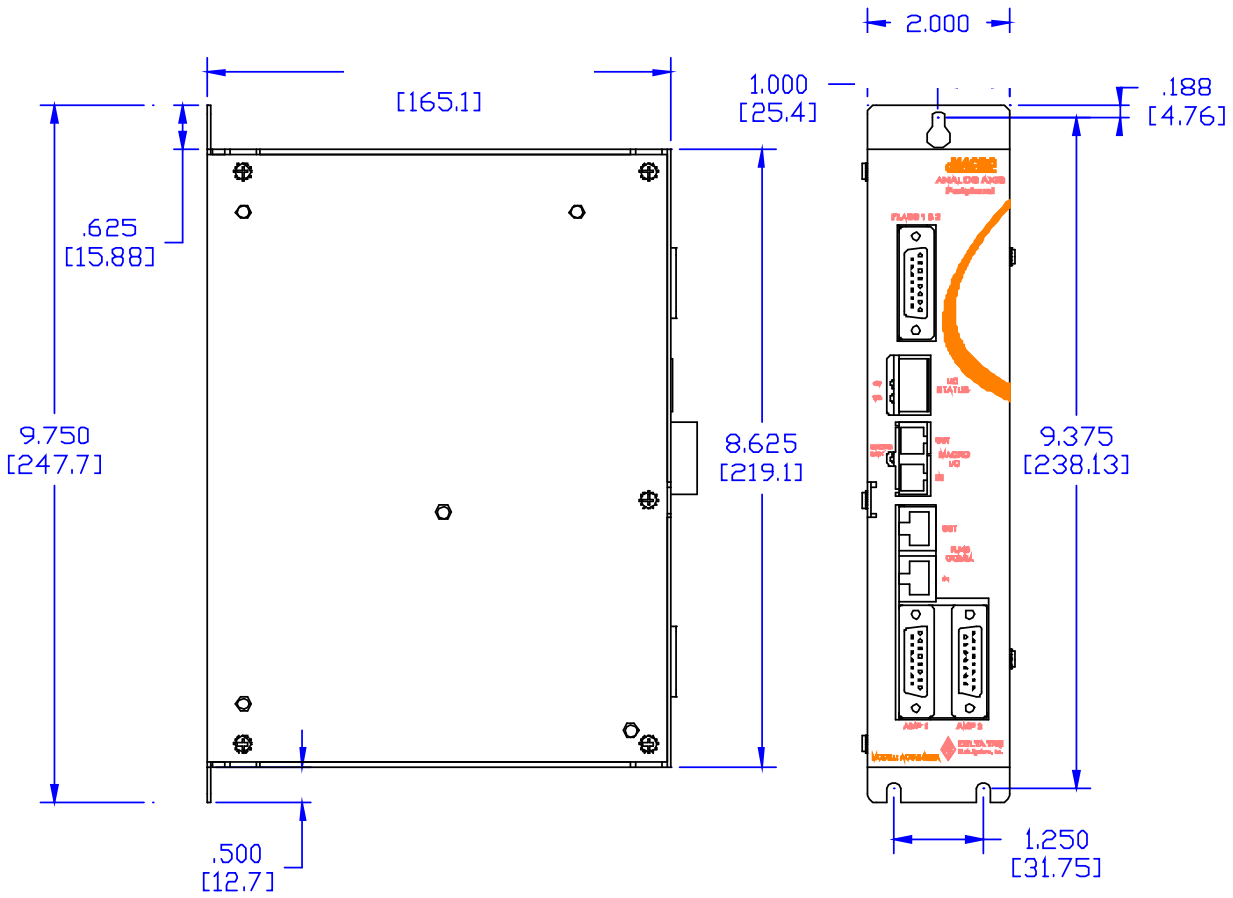
High Vel./F.E. Correlation
Cause: Damping & Friction
Fix:
Increase K_{vff} first
Possibly adjust K_{fff}

Negative Acc./F.E. Correlation
Cause:
Too much Acceleration Feedforward
Fix:
Decrease K_{aff}



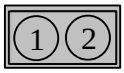
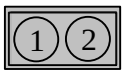
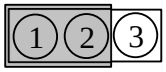
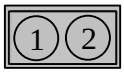
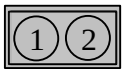
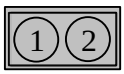
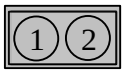
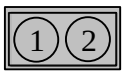
High Vel./F.E. & Acc./F.E. Correlation
Cause:
Inertial Lag & Friction
Fix:
Increase K_{aff}
Possibly adjust K_{fff}

LAYOUT



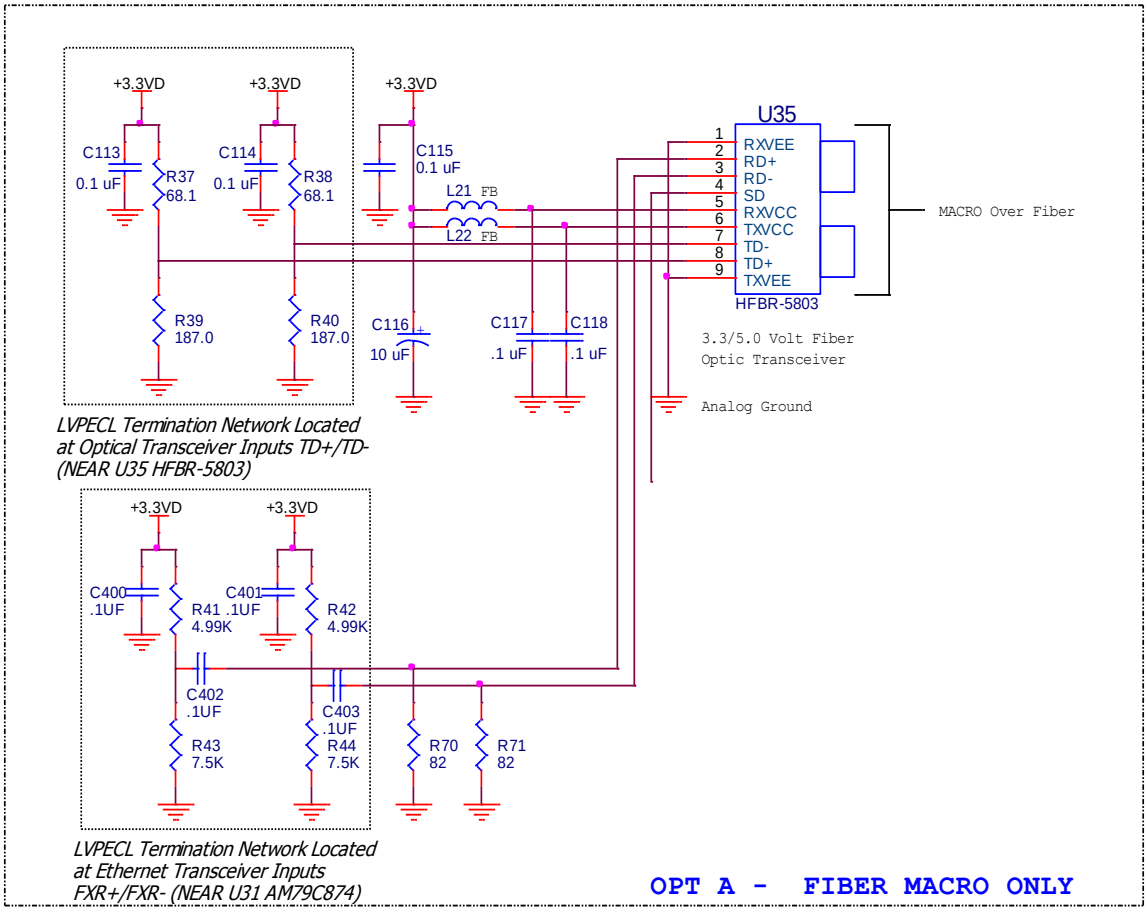
All main dimensions are in units of inches (millimeters are in square brackets).

APPENDIX A: JUMPERS

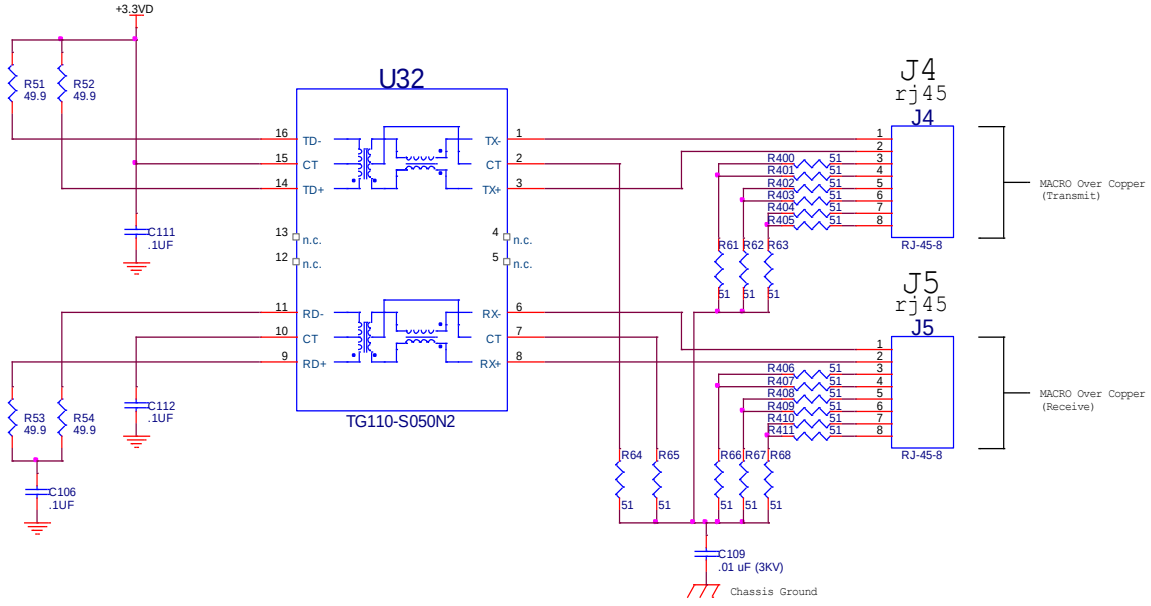
Jumper	Name	Description	Default
E0 	Lattice Download	Remove jumper to disable ability to perform Lattice Download. Jump pins 1 and 2 to enable ability to download.	Not jumpered
E1 	Watchdog Timer Disable	Remove jumper to enable Watchdog Timer. Jump pins 1 and 2 to disable Watchdog Timer (for test purposes only)	Not jumpered
E2 	CPU Mode Operation/Bootstrap	Jump pins 1 and 2 for firmware download through USB port. Jump pins 2 and 3 for normal operation.	Pin 2-3
E3 	Buffer Request Select Polarity	Remove jumper to allow BRSEL- to 5Vdc Jump pins 1 and 2 to pull BRSEL- to 0Vdc	Not jumpered
E5 	Encoder / Pulse and Direction Ch1	Remove jumper to enable +/-10V analog output on Ch1 Jump pins 1 and 2 to enable Stepper mode output for Ch1	Not jumpered
E6 	Encoder / Pulse and Direction Ch2	Remove jumper to enable +/-10V analog output on Ch1 Jump pins 1 and 2 to enable Stepper mode output on Ch2	Not jumpered
E9 	Stepper Drive Amplifier Enable Ch1	Remove jumper to receive encoder C-Channel input on Ch1. Jump pins 1 and 2 to provide Amp Enable line for Stepper Motor-style amplifier Ch1.	Not jumpered
E10 	Stepper Drive Amplifier Enable Ch2	Remove jumper to receive encoder C-Channel input on Ch2. Jump pins 1 and 2 to provide Amp Enable line for Stepper Motor-style amplifier Ch2.	Not jumpered

APPENDIX B: SCHEMATICS

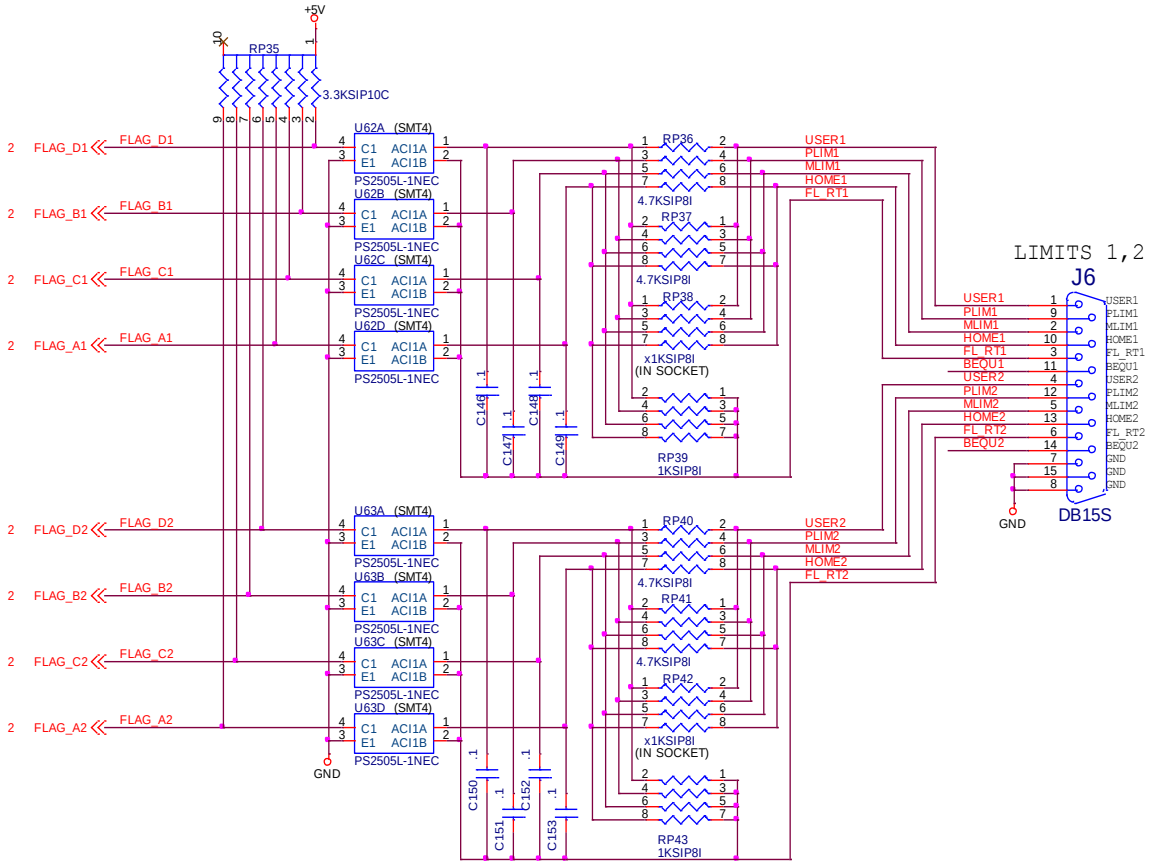
MACRO Fiber Connection



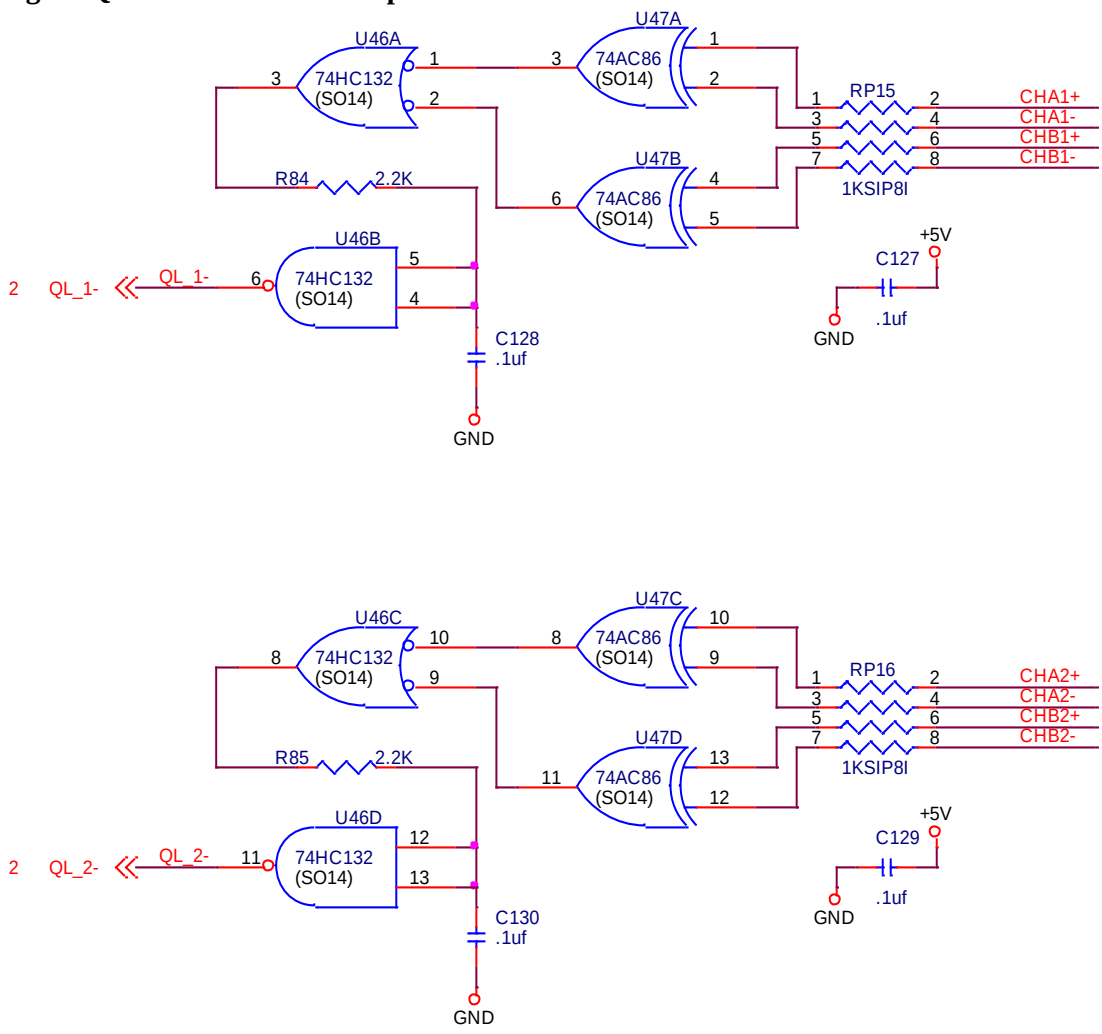
MACRO RJ45 Connection



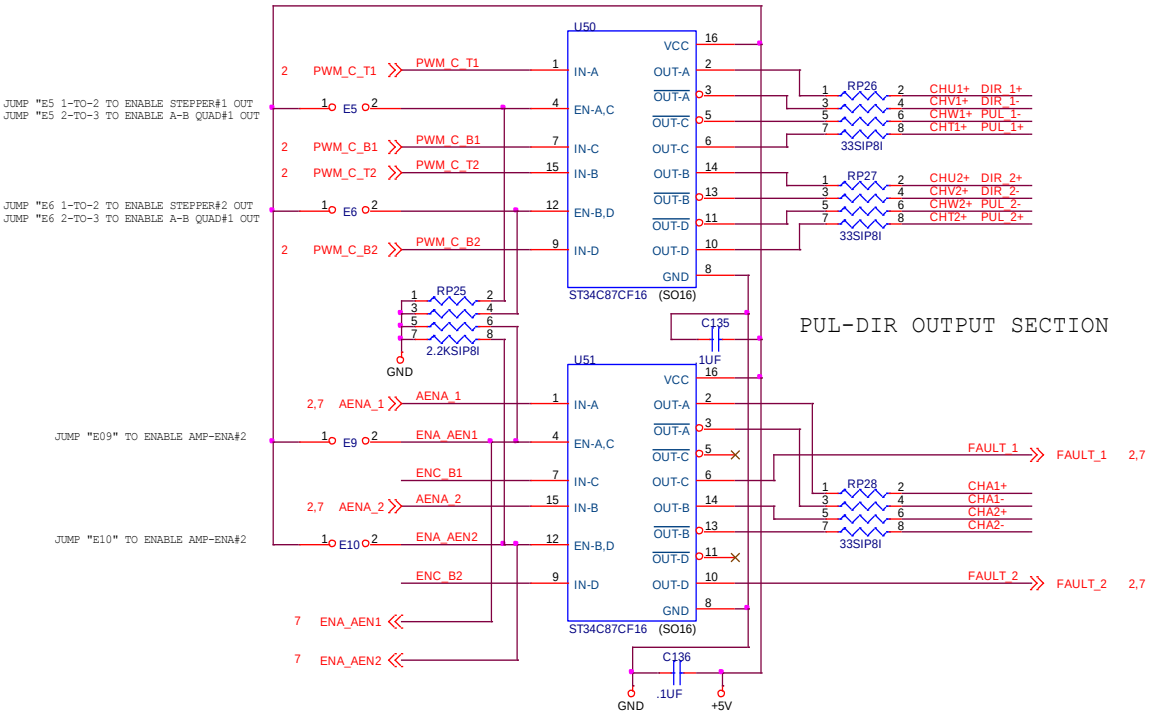
Limit Inputs



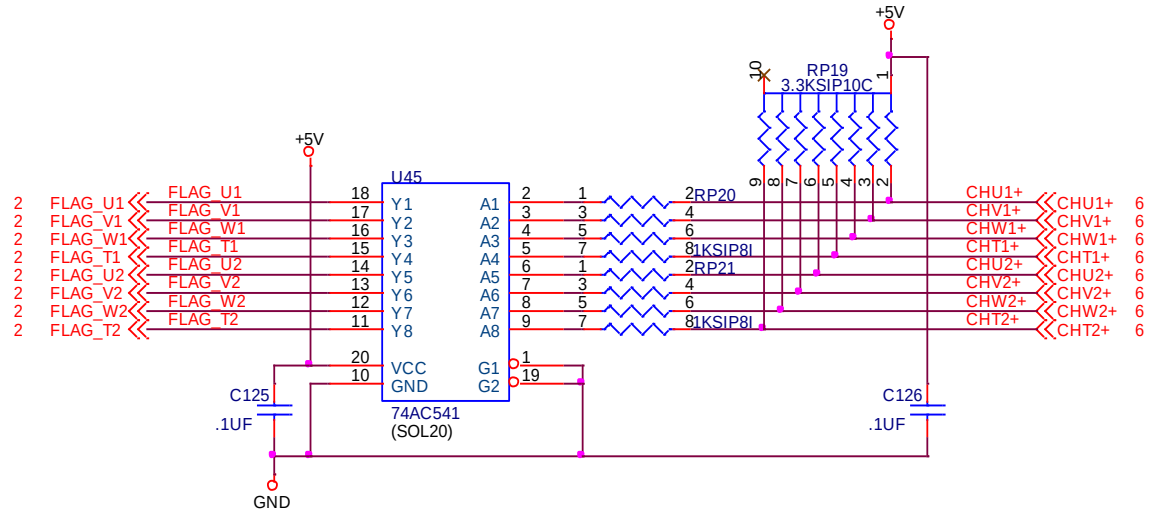
Digital Quadrature Encoder Inputs



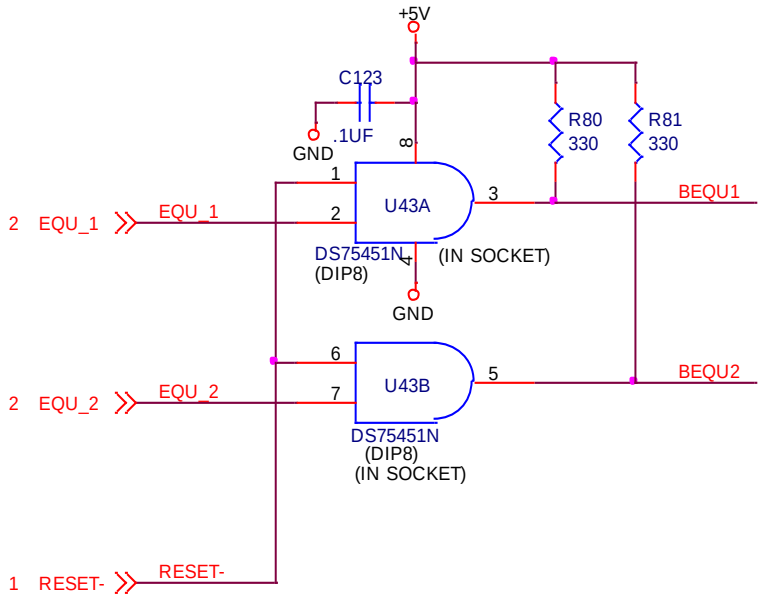
Pulse and Direction Outputs



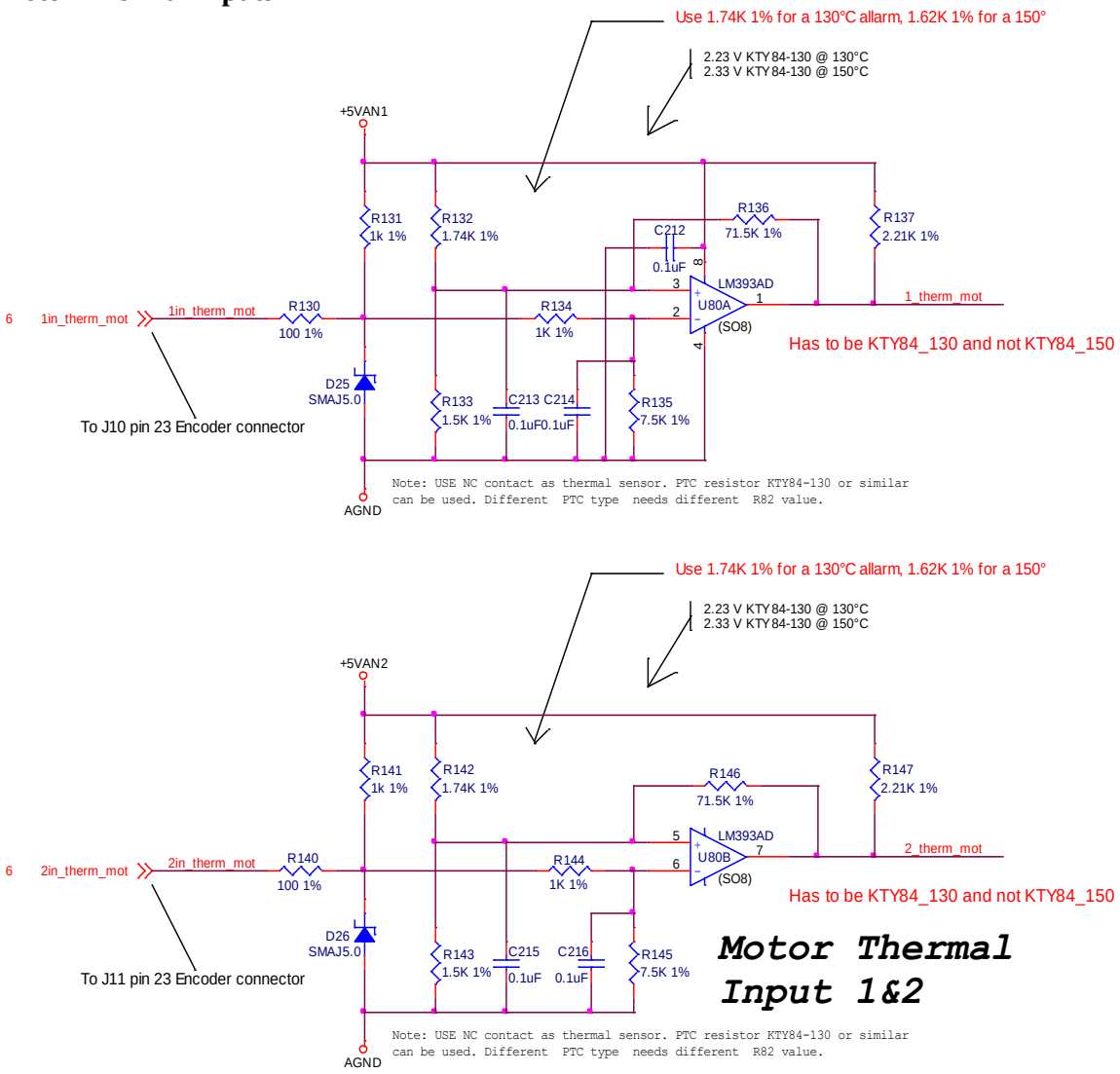
Hall Sensor Inputs



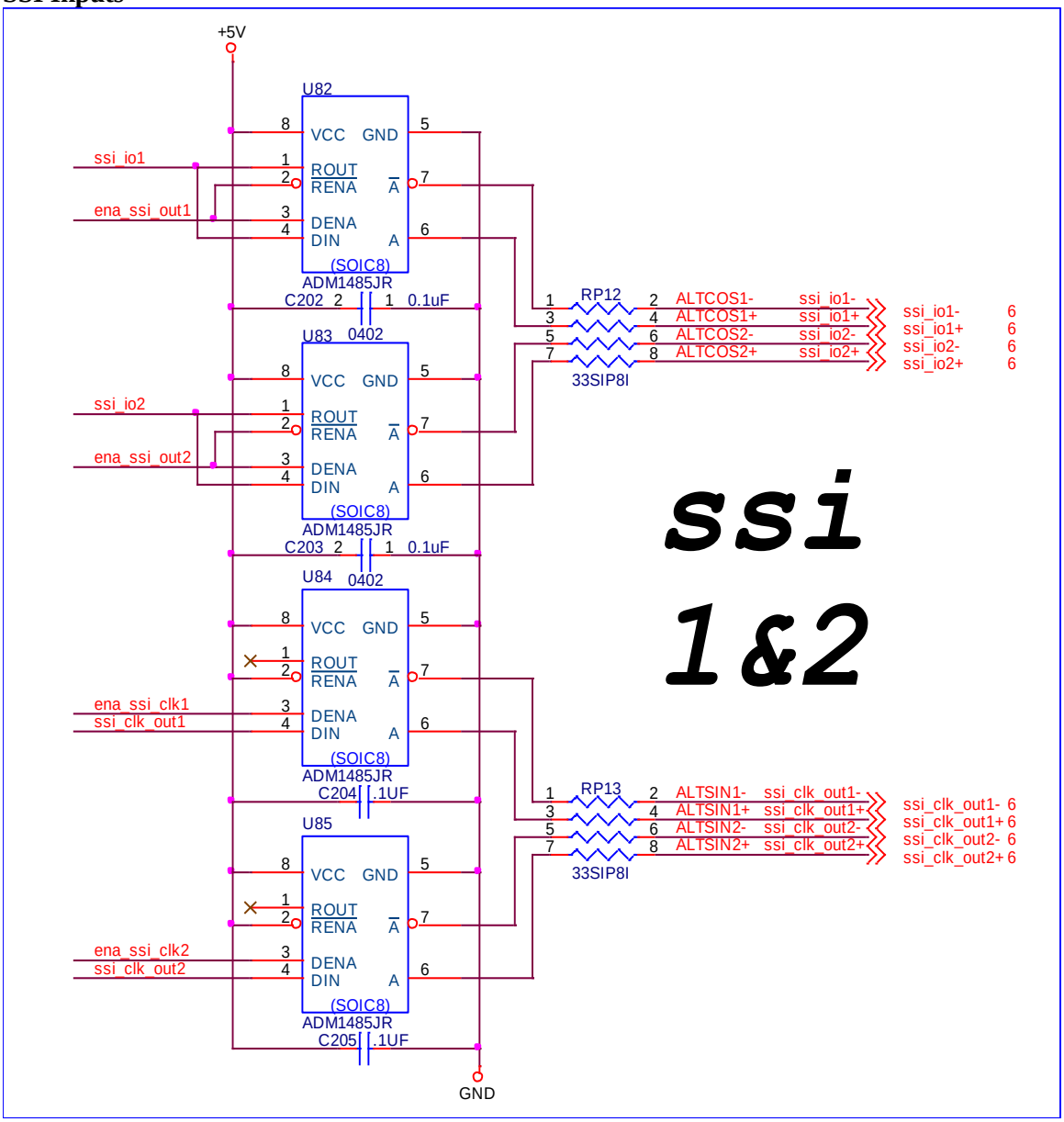
Position Compare Outputs



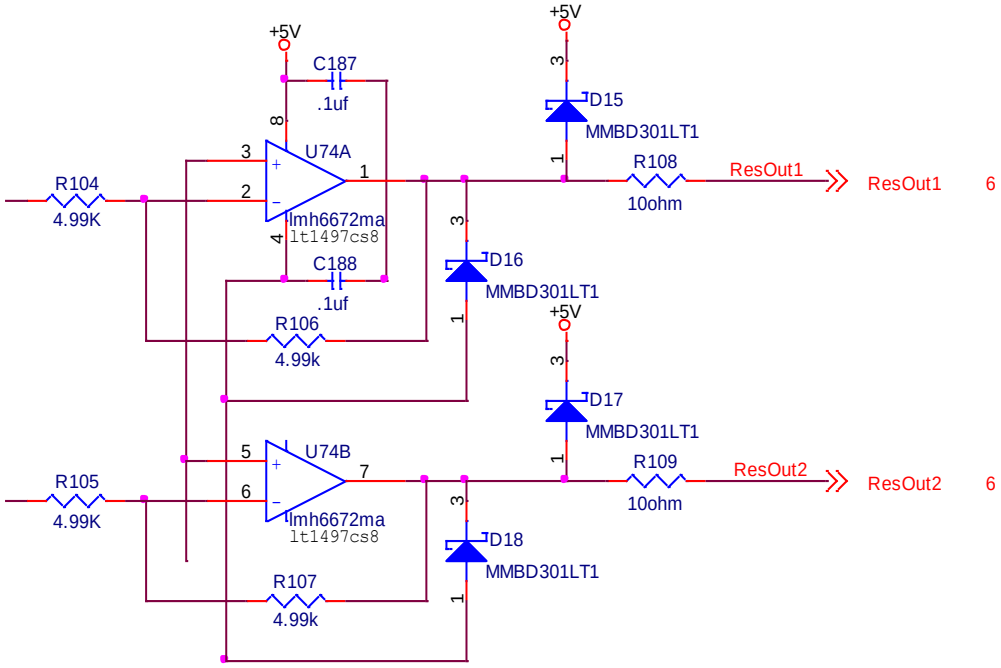
Motor Thermal Inputs



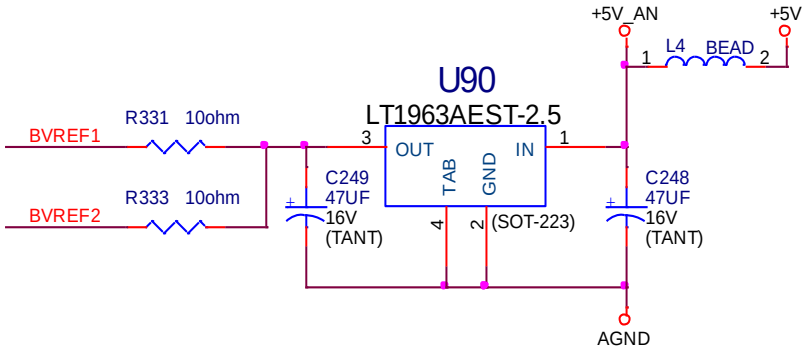
SSI Inputs


+5V
U82
VCC GND
ROUT
RENA
DENA
DIN
A
(SOIC8)
ADM1485JR
C202 0.1uF
0402
U83
C203 0.1uF
U84
C204 .1UF
U85
C205 .1UF
ssi_io1
ena_ssi_out1
ssi_io2
ena_ssi_out2
ena_ssi_clk1
ssi_clk_out1
ena_ssi_clk2
ssi_clk_out2
RP12
RP13
33SIP8I
ALTCOS1-
ALTCOS1+
ALTCOS2-
ALTCOS2+
ALTSIN1-
ALTSIN1+
ALTSIN2-
ALTSIN2+
ssi_io1-
ssi_io1+
ssi_io2-
ssi_io2+
ssi_clk_out1-
ssi_clk_out1+
ssi_clk_out2-
ssi_clk_out2+
ssi 1&2
GND

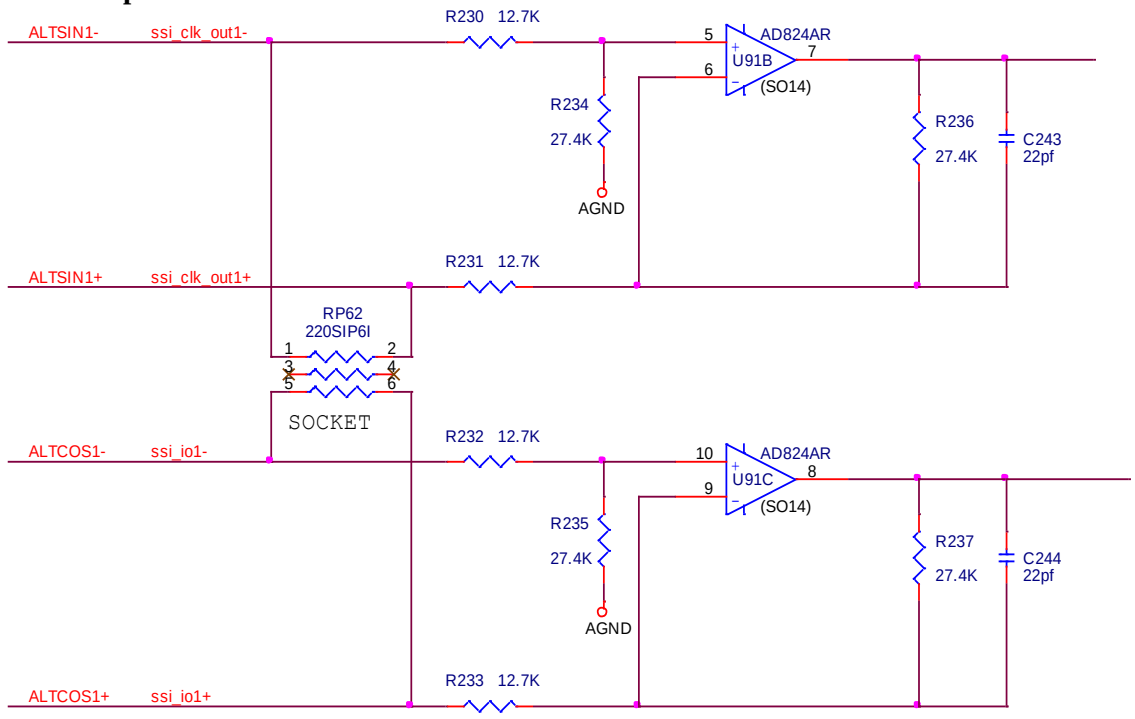
Resolver Outputs



Analog Feedback Voltage Reference Circuit

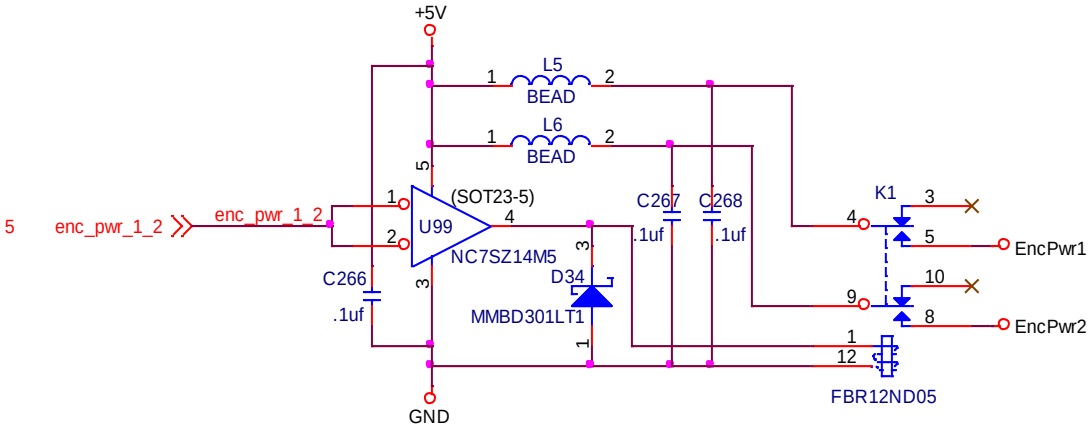


Sin/Cos Inputs

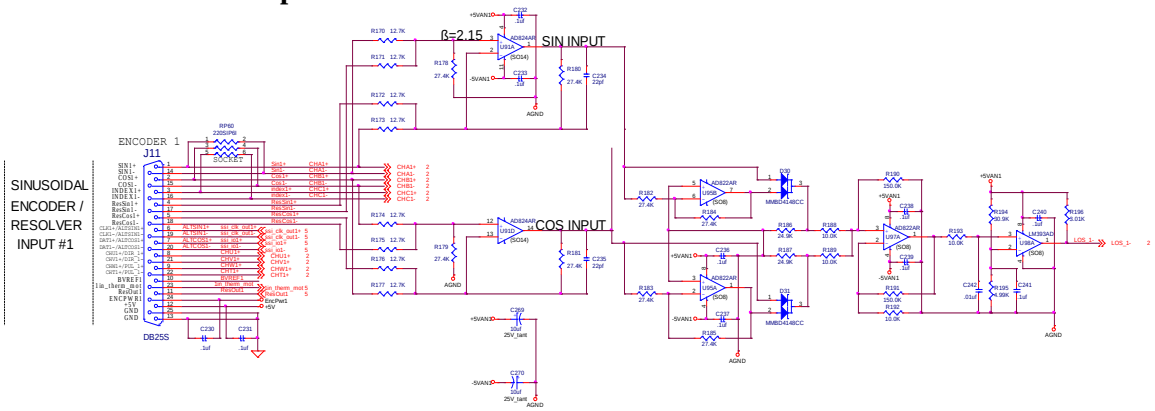


Note: Sin/Cos inputs for Channel #2 are identical to Channel #1.

Encoder Power

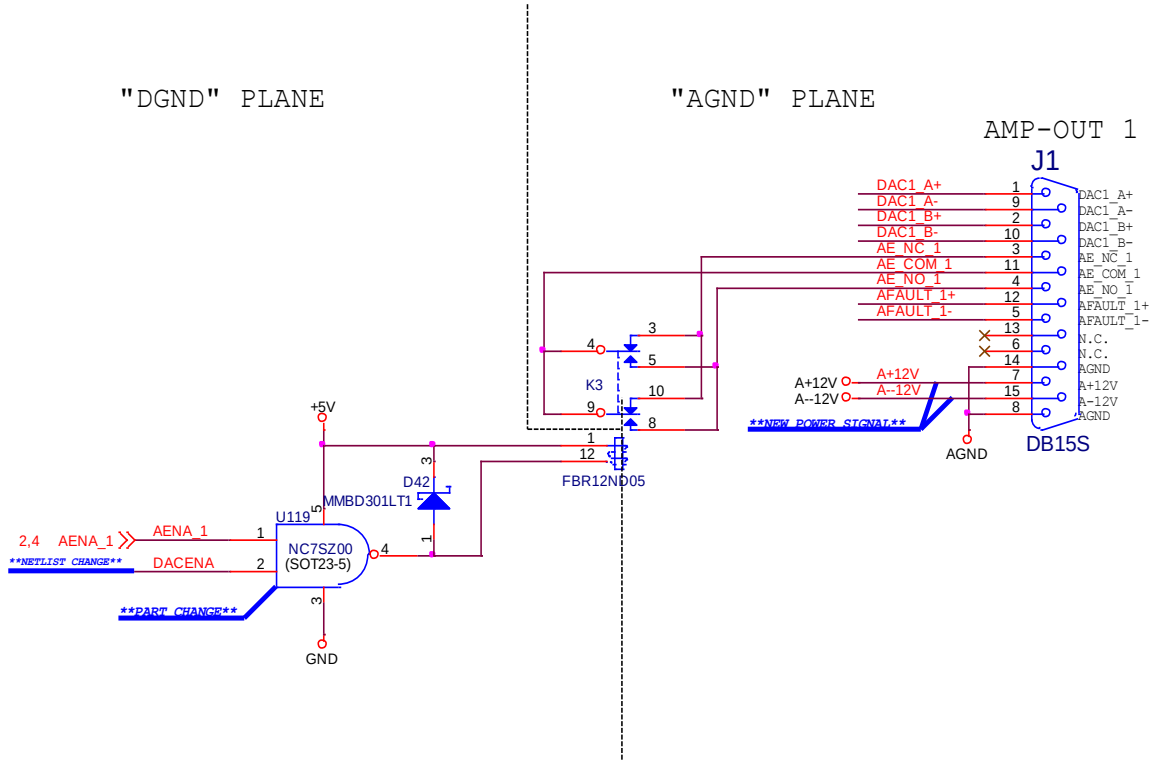


Sinusoidal Encoder Input



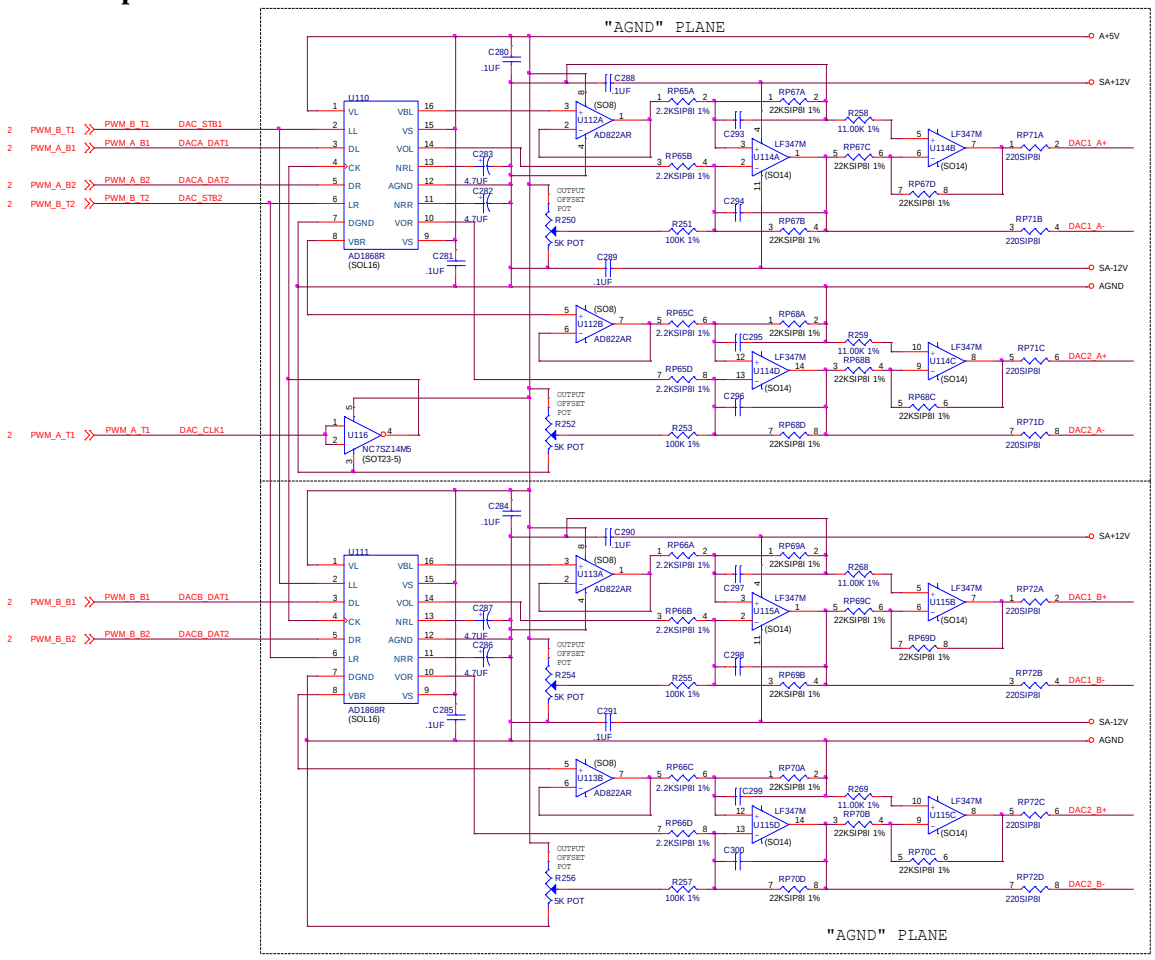
Note: Encoder #2 is identical to Encoder #1.

Amplifier Output

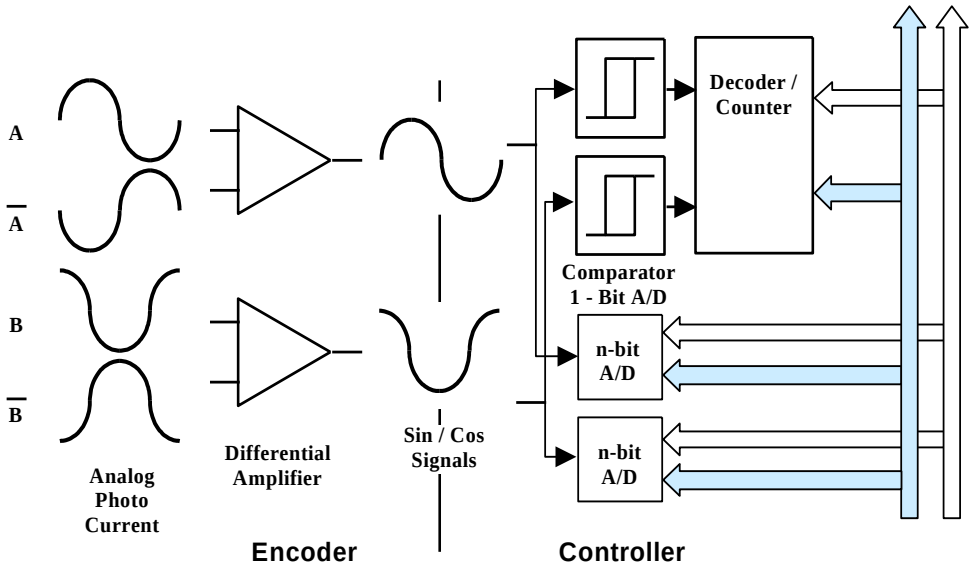


Note: Amplifier Output #2 is identical to Amplifier Output #1.

DAC Outputs


"AGND" PLANE
A+5V
SA+12V
SA-12V
AGND
PWM_B_T1
PWM_A_B1
PWM_A_B2
PWM_B_T2
DAC_STB1
DACA_DAT1
DACA_DAT2
DAC_STB2
PWM_A_T1
DAC_CLK1
PWM_B_B1
PWM_B_B2
DACB_DAT1
DACB_DAT2
U110
U111
AD1868R
(SOL16)
U116
NC7SZ14M5
(SOT23-5)
U112A
U112B
U113A
U113B
AD822AR
(SO8)
U114A
U114B
U114C
U114D
U115A
U115B
U115C
U115D
LF347M
(SO14)
OUTPUT OFFSET POT
R250
R251
R252
R253
R254
R255
R256
R257
5K POT
100K 1%
C280
C281
C282
C283
C284
C285
C286
C287
4.7UF
.1UF
C288
C289
C290
C291
RP65A
RP65B
RP65C
RP65D
RP66A
RP66B
RP66C
RP66D
RP67A
RP67B
RP67C
RP67D
RP68A
RP68B
RP68C
RP68D
RP69A
RP69B
RP69C
RP69D
RP70A
RP70B
RP70C
RP70D
2.2KSIP8I 1%
22KSIP8I 1%
R258
R259
R268
R269
11.00K 1%
C293
C294
C295
C296
C297
C298
C299
C300
RP71A
RP71B
RP71C
RP71D
RP72A
RP72B
RP72C
RP72D
220SIP8I
DAC1_A+
DAC1_A-
DAC2_A+
DAC2_A-
DAC1_B+
DAC1_B-
DAC2_B+
DAC2_B-
"AGND" PLANE

APPENDIX C: SINUSOIDAL INTERPOLATION



The sine and cosine signals from the encoder are processed in two ways in this product (see above diagram). First, they are sent through comparators that square up the signals into digital quadrature and are then sent into the quadrature decoding and counting circuit of the Servo IC on the ACC-24M2A. The units of the hardware counter, which are called hardware counts, are thus ¼ of a line. For most users, this fact is an intermediate value, an internal detail that does not concern them. However, this is important in two cases. First, if the sinusoidal encoder is used for PMAC-based brushless-motor commutation, the hardware counter (not the fully interpolated position value) will be used for the commutation position feedback. Therefore, the units of Ixx71 will be hardware counts. Second, if the hardware position-compare circuits in the Servo IC are used, the units of the compare register are hardware counts. The same is true of the hardware position-capture circuits, but often these scaling issues are handled automatically through the move-until-trigger constructs.

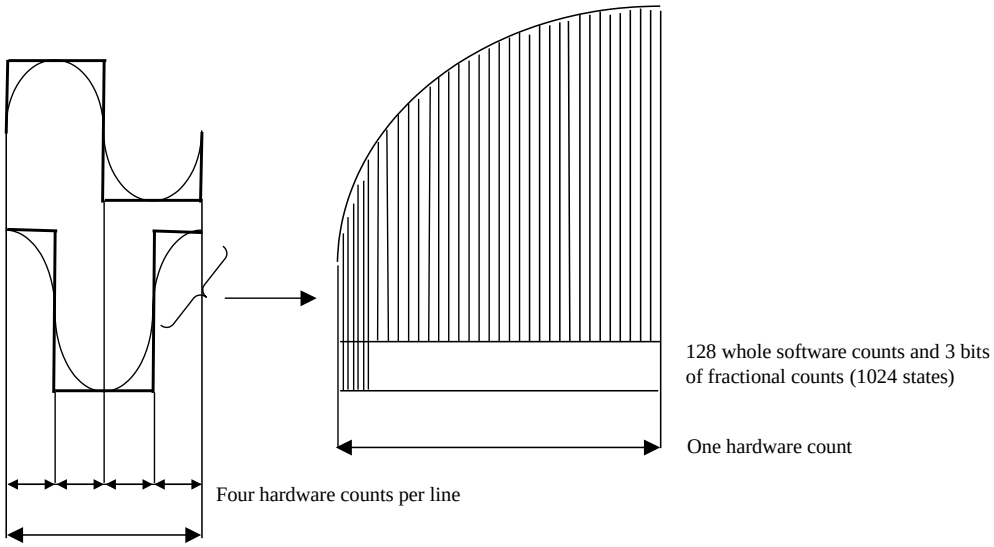
The second, parallel processing of the sine and cosine signals is through analog-to-digital converters, which produce numbers proportional to the input voltages. These numbers are used to calculate mathematically an arctangent value that represents the location within a single line. This is calculated to $1/4096^{th}$ of a line, so there are 4096 unique states per line, or 1024 states per hardware count.
For historical reasons, PMAC expects the position it reads for its servo feedback software to have units of $1/32^{th}$ of a count. That is, PMAC considers the least significant bit (LSB) of whatever it reads for position feedback to have a magnitude of $1/32^{th}$ of a count for the purposes of its software scaling calculations. We call the resulting software units software counts and any software parameter that uses counts from the servo feedback (e.g. jog speed in counts/msec, axis scale factor in counts/engineering-unit) is using these software counts. In most cases, such as digital quadrature feedback, these software counts are equivalent to hardware counts.

However, with the added resolution produced by the ACC-24M2A interpolator option, software counts and hardware counts are no longer the same. The LSB produced by the interpolator (through the encoder conversion table processing) is $1/1024^{th}$ of a hardware count, but PMAC software considers it $1/32^{th}$ of a software count. Therefore, with the ACC-24M2A, a software count is $1/32^{th}$ the size of a hardware count.

The following equations express the relationships between the different units:

1 line = 4 hardware counts = 128 software counts = 4096 states (LSBs)
¼-line = 1 hardware count = 32 software counts = 1024 states (LSBs)
1/128-line = 1/32-hardware count = 1 software count = 32 states (LSBs)
1/4096-line = 1/1024-hardware count = 1/32-software count = 1 state (LSB)

Note that these are all just naming conventions. Even the position data that is fractional in terms of software counts is real. The servo loop can see it and react to it, and the trajectory generator can command to it.



The Interpolator can accept a voltage-source (1 V_{pp}) signal from the encoder. The maximum sine-cycle frequency input is approximately 8 MHz (1,400,000 SIN cycles/sec), which gives a maximum speed of about 5.734 billion steps per second.

When used with a 1000 line sinusoidal rotary encoder, there will be 4,096,000 discrete states per revolution (128,000 software counts). The maximum calculated electrical speed of this encoder would be 1,400 RPS or 84,000 RPM, which exceeds the maximum physical speed of most encoders.

Example 1:
A 4-pole rotary brushless motor has a sinusoidal encoder with 2000 lines. It directly drives a screw with a 5-mm pitch.

For servo control, the interpolated results of the conversion table are used. There are 128 software counts per line, or 256,000 software counts per revolution. With each revolution corresponding to 5 mm on the screw, there are 51,200 software counts per millimeter. The measurement resolution, at 4096 states per line, is 1/8,192,000 of a revolution, or 1/1,638,400 of a millimeter (~0.6 nanometers/state).

Example 2:
A linear brushless motor has a commutation cycle of 60.96 mm (2.4 inches). It has a linear scale with a 20-micron line pitch.

The servo uses the interpolated results of the conversion table. With 128 software counts per line, and 50 lines per millimeter, there are 6400 software counts per millimeter (or 162,560 software counts per inch). The measurement resolution, at 4096 states per line, is 204,800 states per mm (~5 nanometers/state).