DELTA TAU DATA SYSTEMS, INC.

UMAC ACCESSORY 24E3

POWER PMAC AXIS INTERFACE

HARDWARE REFERENCE

PRELIMINARY

DOCUMENT # 3A0-604002-360

November, 2011

Delta Tau Data Systems 21314 Lassen St. Chatsworth, CA 91311

Ph: (818) 998-2095 Fax: (818) 998-7807

e-mail: support@deltatau.com Website: www.deltatau.com

N E W I D E A S I N M O T I O N . . . .

(This page intentionally left blank)

Table of Contents

Introduction

The ACC-24E3 family of products provides a powerful and flexible suite of axis-interface circuitry for UMAC racks controlled by the Power PMAC CPU. The ACC-24E3 provides 2 or 4 channels of axis interface in a 1 or 2-slot (respectively) package.

The ACC-24E3 can be used to process the following types of position feedback:

- Quadrature encoders
- Hall-style commutation sensors
- Serial encoders
- Sinusoidal encoders
- Resolvers

The ACC-24E3 can be used to command the following types of amplifiers and drives:

- Analog velocity-mode
- Analog torque-mode
- Analog sinewave input
- Direct-PWM power block
- Pulse-and-direction input stepper or stepper-replacement servo

The ACC-24E3 is built modularly out of 3 circuit boards for a 2-axis interface, or 6 circuit boards for a 4-axis interface.

A word on the accessory number:

ACC-24 products are Delta Tau’s axis-interface accessories, in many formats. The “E” specifies the “Euro-card” format of the UMAC boards (as opposed to “P” for PC format, “V” for VME format, or “C” for Compact PCI format”. The “3” specifies the use of a PMAC3-style ASIC in the product (as opposed to “2” for a PMAC2-style ASIC, or no final digit for a PMAC1-style ASIC).

The ACC-24E3 is not compatible with the older Turbo PMAC UMAC CPU, or the MACRO 8-axis and MACRO 16-axis CPUs.

Configuration

The ACC-24E3 has a modular design that provides great flexibility in configuring optimal solutions to different styles of axis interfaces. The components that comprise an ACC-24E3 assembly, and the possible assemblies, are described in this section.

Components

An ACC-24E3 is comprised of up to six possible components, assembled from the list described below.

3U-Format Base Board (300-604002-10x)

The 3U-format (100mm x 160mm) base board is the only component that is required in all configurations. It provides the interface to the UBUS32 backplane for communications with the Power PMAC CPU and the "DSPGATE3" ASIC, which provides all of the digital processing logic for two or four channels of axis interface.

It provides a low-profile high-density stack connector near the top edge for a "feedback" mezzanine board for the first two channels, and a low-profile high-density connector near the bottom edge for an "amplifier" mezzanine board for the first two channels. In addition, it provides sockets for a stack connector to an optional 3U-format "piggyback" board for the third and fourth channels. It has front-edge terminal blocks for the "flag" signals for the first two channels.

3U-Format Piggyback Board (301-604002-10x)

The 3U-format (100mm x 160mm) piggyback board can stack on top of the 3U-format base board to provide an interface path for the third and fourth channels. In the UMAC rack, it sits one "4T" slot (200mm or 0.8") to the right of the base board. It has long prong connectors on the back side that fit into matching sockets on the front side of the base board.

It provides a low-profile high-density stack connector near the top edge for a "feedback" mezzanine board for the third and fourth channels, and a low-profile high-density connector near the bottom edge for an "amplifier" mezzanine board for the third and fourth channels. It has front-edge terminal blocks for the "flag" signals for the third and fourth channels.

The 3U-format piggyback board uses the same circuit-board design as the 3U-format base board, but has a different set of components installed.

Digital Feedback Mezzanine Board (300-604003-10x)

The digital feedback mezzanine board is installed along the top edge of either the 3U-format base board or the 3U-format piggyback board. In the first case, it provides the connectors and the buffer circuitry for the first two channels of digital position feedback; in the second case it provides the connectors and the buffer circuitry for the third and fourth channels of digital position feedback circuitry.

The types of feedback sensors that can be connected to each channel on this board include:

- Single-ended or differential quadrature encoders with index
- 3-phase digital hall commutation sensors
- Serial encoders of multiple different protocols

In addition, it is possible to provide pulse-frequency-modulated (PFM) outputs in pulse-and-direction format.

The digital feedback mezzanine board is available with either dual 15-pin D-sub connectors or dual 12-point removable terminal blocks.

Analog Feedback Mezzanine Board (300-604004-10x)

The analog feedback mezzanine board is installed along the top edge of either the 3U-format base board or the 3U-format piggyback board. In the first case, it provides the connectors and the buffer circuitry for the first two channels of analog position feedback; in the second case it provides the connectors and the buffer circuitry for the third and fourth channels of analog position feedback circuitry.

The types of feedback sensors that can be connected to each channel on this board include:

- Single-ended or sinusoidal encoders with index
- Resolvers (with excitation provided from the board)
- Serial (digital) encoders of multiple different protocols

Because the optimal signal levels for sinusoidal encoders and resolvers are different, the board must be purchased configured for one or the other type of sensor. The sinusoidal encoder configuration will have a voltage regulator installed in U19 and resistor packs installed in RP1, RP2, RP3, and RP4. The resolver configuration will have an op-amp installed in U36 for the excitation output.

The analog feedback mezzanine board is available only with dual 15-pin D-sub connectors.

Digital Amplifier Mezzanine Board (300-604005-10x)

The digital amplifier mezzanine board is installed along the bottom edge of either the 3U-format base board or the 3U-format piggyback board. In the first case, it provides the connectors and the buffer circuitry for the first two channels of digital amplifier connection; in the second case it provides the connectors and the buffer circuitry for the third and fourth channels of digital position amplifier connection.

This board is intended primarily for interface to direct-PWM “power-block” style amplifiers, which accept the actual pulse-width-modulated on-off commands to the power-transistors. It has two 36-pin Mini-D connectors in the standard power-block signal configuration.

Analog Amplifier Mezzanine Board (300-604006-10x)

The analog amplifier mezzanine board is installed along the bottom edge of either the 3U-format base board or the 3U-format piggyback board. In the first case, it provides the connectors and the buffer circuitry for the first two channels of analog amplifier connection; in the second case it provides the connectors and the buffer circuitry for the third and fourth channels of analog position amplifier connection.

The simplest configuration for this board comes with a single 16-bit digital-to-analog converter (DAC) for each channel, providing a +/-10V analog output. This makes the board suitable for commanding analog velocity-mode and torque-mode amplifiers. This configuration is providing with no optical isolation of the analog circuitry from the digital circuitry, so both analog and digital signals are relative to the same reference voltage.

The next configuration for this board comes with dual 16-bit digital-to-analog converters for each channel. This makes the board suitable for analog velocity-mode, torque-mode, and "sine-wave" amplifiers. A further configuration provides dual 18-bit DACs for each channel. This configuration is desirable for very high precision applications. Both configurations with dual DACs provide optical isolation between the analog and digital circuitry, although in the factory default configuration, this isolation is defeated by the jumper settings.

It is possible to install a third DAC for each channel, but this is not a standard configuration – contact the factory if such a configuration is desired.

The analog amplifier mezzanine board is available with either dual 15-pin D-sub connectors or dual 12-point removable terminal blocks. However, the 18-bit configuration comes only with the D-sub connectors.

Assemblies

The ACC-24E3 is typically purchased as a complete assembly. The assembly can support either two channels or four channels. (Typically, one hardware channel is required per axis, so these configurations are commonly called "two-axis" and "four-axis".)

A two-channel assembly is comprised of a 3U-format base board with a feedback mezzanine board and an amplifier mezzanine board. It occupies a single 4T slot (20 mm, or 0.8", wide) in the UMAC rack.

A four-channel assembly is comprised of a 3U-format base board with a feedback mezzanine board and an amplifier mezzanine board, plus a 3U-format mezzanine board with its own feedback mezzanine board and amplifier mezzanine board. It occupies two 4T slots in the UMAC rack, for a width of 40mm (1.6").

There is a very high number of assembly combinations that is theoretically possible. However, not all of these combinations are offered for sale by Delta Tau. Contact the factory for availability and pricing of an assembly combination that is not on the price list.

Hardware Setup

The ACC-24E3 requires minimal hardware setup before installation. The factory default configuration provides for the most common type of use for the boards.

3U-Format Base Board (300-604002)

If only a single ACC-24E3 is used in an application, it can typically be used in the default hardware configuration. However, if more than one ACC-24E3 is installed in a single UMAC rack, all but one must have their addressing DIP switch settings changed. If you purchase a pre-assembled UMAC rack, these switch settings will have been made by the assembler.

Addressing DIP Switches SW1

The 3U-format base board of the ACC-24E3 has a bank of DIP switches that determine where the accessory appears in the addressing space of the Power PMAC CPU. Each accessory in a Power PMAC UMAC rack that is based on a “DSPGATE3” machine-interface IC must have a different setting of these switches.

The most important thing for the user to realize is the relationship between the switch settings and the resulting “index number” for the IC’s data structure. For example, with the switches in their default settings (all “OFF”, or “OPEN”), the index number is 0, so registers in the IC are accessed using the **Gate3[0]** (or its alias **Acc24E3[0]**) data structure.

The following table lists the possible switch settings, along with the IC index numbers and base addresses they select. The numerical base addresses are for reference only; in general, a user will not need to know these.

SW1-1 (Address bit 11)	SW1-2 (Address bit 12)	SW1-3 (Address bit 13)	SW1-4 (Address bit 14)	Power PMAC “Gate3” Index #	Power PMAC I/O Base Address Offset
OFF(0)	OFF(0)	OFF(0)	OFF(0)	0	$900000
ON(1)	OFF(0)	OFF(0)	OFF(0)	1	$904000
OFF(0)	ON(1)	OFF(0)	OFF(0)	2	$908000
ON(1)	ON(1)	OFF(0)	OFF(0)	3	$90C000
OFF(0)	OFF(0)	ON(1)	OFF(0)	4	$910000
ON(1)	OFF(0)	ON(1)	OFF(0)	5	$914000
OFF(0)	ON(1)	ON(1)	OFF(0)	6	$918000
ON(1)	ON(1)	ON(1)	OFF(0)	7	$91C000
OFF(0)	OFF(0)	OFF(0)	ON(1)	8	$920000
ON(1)	OFF(0)	OFF(0)	ON(1)	9	$924000
OFF(0)	ON(1)	OFF(0)	ON(1)	10	$928000
ON(1)	ON(1)	OFF(0)	ON(1)	11	$92C000
OFF(0)	OFF(0)	ON(1)	ON(1)	12	$930000
ON(1)	OFF(0)	ON(1)	ON(1)	13	$934000
OFF(0)	ON(1)	ON(1)	ON(1)	14	$938000
ON(1)	ON(1)	ON(1)	ON(1)	15	$93C000

Note that these ON/OFF settings are opposite those of the older ACC-24E2*x* axis-interface boards for the order of the resulting IC index numbers.

Each accessory with a DSPGATE3 IC in a UMAC rack, whether an ACC-24E3 or not (e.g. an ACC-5E3), must have a unique set of switch settings to avoid addressing conflicts. Accessories with other types of ICs – such as the DSPGATE1 ICs used in ACC-24E2x boards that employ the **Gate1[*i*]** data structure, the DSPGATE2 ICs used in the ACC-5E that employ the **Gate2[*i*]** data structure, and the IOGATE ICs used in the digital I/O boards that employ the **GateIo[*i*]** data structure – do not have the possibility of addressing conflict with boards based on the DSPGATE3 IC.

Input Flag Resistor Packs

The input flags brought into the board through the terminal blocks TB1 and TB2 are normally 12V to 24V signals. If it is desired to use 5V signals for a channel, a resistor pack can be inserted into a socket for the channel. The added resistance is in parallel with the existing current-limiting resistors, reducing the overall resistance. The resistor pack must be an 8-pin quad SIP pack of 1-kohm resistors (not provided). For Channel 1, it should be inserted into the socket at RP13; for Channel 2, it should be inserted into the socket at RP17.

Note: With the 1-kohm socketed resistor pack installed for a channel, the flag voltages for that channel should be limited to 5V. Higher voltages can damage the resistor pack due to overheating.

3U-Format Piggyback Board (301-604002)

In almost all applications, the default hardware configuration of the 3U-format piggyback board will be usable without change.

Input Flag Resistor Packs

The input flags brought into the board through the terminal blocks TB1 and TB2 are normally 12V to 24V signals. If it is desired to use 5V signals for a channel, a resistor pack can be inserted into a socket for the channel. The added resistance is in parallel with the existing current-limiting resistors, reducing the overall resistance. The resistor pack must be an 8-pin quad SIP pack of 1-kohm resistors (not provided). For Channel 3, it should be inserted into the socket at RP13; for Channel 4, it should be inserted into the socket at RP17.

Note: With the 1-kohm socketed resistor pack installed for a channel, the flag voltages for that channel should be limited to 5V. Higher voltages can damage the resistor pack due to overheating.

Digital Feedback Mezzanine Board (604003)

In almost all applications, the default hardware configuration of the digital feedback mezzanine board will be usable without change.

Jumpers for Enhanced Stepper Drive Interface

The digital feedback mezzanine board provides four “E-point” jumpers that can be used to provide a full stepper drive interface on the encoder connector. All jumpers are OFF by default, and they are seldom used.

Note: The basic “pulse-and-direction” output can optionally be provided from the encoder connectors on what are usually the T, U, V, and W input flag lines without changing any jumper settings. These outputs are enabled if software data structure element

Gate3[*i*].Chan[*j*].OutFlagD is set to 1. This setting is not saved, so must be made after every power-on/reset, usually in a "power-on PLC" program.

If jumper E1 is ON, the amplifier-enable signal for the first channel on the board (hardware channel 1 or 3) will be output on what are normally the channel's encoder A input lines. If the jumper is in its default OFF state, these lines are used for encoder A input.

If jumper E2 is ON, the amplifier-enable signal for the second channel on the board (hardware channel 2 or 4) will be output on what are normally the channel's encoder A input lines. If the jumper is in its default OFF state, these lines are used for encoder A input.

If jumper E3 is ON, the amplifier-fault signal for the first channel on the board is tied to what is normally the channel's encoder B input phase. If the jumper is in its default OFF state, the amplifier fault state is independent of the state of the encoder B input phase. In this case, the amplifier fault signal will usually come through the amplifier mezzanine board.

If jumper E4 is ON, the amplifier-fault signal for the second channel on the board is tied to what is normally the channel's encoder B input phase. If the jumper is in its default OFF state, the amplifier fault state is independent of the state of the encoder B input phase. In this case, the amplifier fault signal will usually come through the amplifier mezzanine board.

Jumpers for Sharing Incremental and Serial Encoders

If jumper E5 (for revisions -108 and newer, released at the beginning of 2012) is in its default state of connecting pins 1 and 2, when **Gate3[*i*].Chan[*j*].SerialEncEna** for the first channel on the board is set to 1, what are normally the channel's incremental encoder C index channel inputs are used instead for the serial encoder SENCENA+/- outputs. (On older board revisions, this is always the case.) If E5 connects pins 2 and 3, these pins are always used for the incremental encoder C index channel inputs, regardless of the setting of **Gate3[*i*].Chan[*j*].SerialEncEna**. Most serial encoder protocols do not need the SENCENA+/- lines, and this setting permits the use of an incremental encoder with index simultaneously with a serial encoder on the same channel.

If jumper E6 (for revisions -108 and newer, released at the beginning of 2012) is in its default state of connecting pins 1 and 2, when **Gate3[*i*].Chan[*j*].SerialEncEna** for the second channel on the board is set to 1, what are normally the channel's incremental encoder C index channel inputs are used instead for the serial encoder SENCENA+/- outputs. (On older board revisions, this is always the case.) If E6 connects pins 2 and 3, these pins are always used for the incremental encoder C index channel inputs, regardless of the setting of **Gate3[*i*].Chan[*j*].SerialEncEna**. Most serial encoder protocols do not need the SENCENA+/- lines, and this setting permits the use of an incremental encoder with index simultaneously with a serial encoder on the same channel.

Analog Feedback Mezzanine Board (604004)

In almost all applications, the default hardware configuration of the analog feedback mezzanine board will be usable without change. This board comes in two different configurations, one for sinusoidal encoders, and one for resolvers. These two configurations have different defaults.

Termination Resistor Packs

The analog feedback mezzanine board has sockets for termination resistor packs for the analog input pairs. In general, these resistor packs should be installed for sinusoidal encoder feedback, and not installed for resolver feedback. The default is for 220-ohm resistors to be installed in the sinusoidal-encoder configuration of the board, and no resistors to be installed in the resolver configuration of the board.

The resistor pack numbers and the signals they affect are:

- RP1: SIN+/-, COS+/-, INDEX+/- for the first channel
- RP2: SIN+/-, COS+/-, INDEX+/- for the second channel
- RP3: ALTSIN+/-, ALTCOS+/- for the first channel
- RP2: ALTSIN+/-, ALTCOS+/- for the second channel

The 220-ohm resistor value was chosen based on typical encoder cabling. It should be suitable for the large majority of applications. However, different resistor values may be used with benefit in some applications. These are 8-pin, 4-resistor SIP packs.

Jumpers for Enhanced Stepper Drive Interface

The analog feedback mezzanine board provides four “E-point” jumpers that can be used to provide a full stepper drive interface on the encoder connector. All jumpers are OFF by default, and they are seldom used.

Note: The basic “pulse-and-direction” output can optionally be provided from the encoder connectors on what are usually the T, U, V, and W input flag lines without changing any jumper settings. These outputs are enabled if software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** is set to 1. This setting is not saved, so must be made after every power-on/reset, usually in a “power-on PLC” program.

If jumper E1 is ON, the amplifier-enable signal for the first channel on the board (hardware channel 1 or 3) will be output on what are normally the encoder A input lines. If the jumper is in its default OFF state, these lines are used for encoder A input.

If jumper E2 is ON, the amplifier-enable signal for the second channel on the board (hardware channel 2 or 4) will be output on what are normally the encoder A input lines. If the jumper is in its default OFF state, these lines are used for encoder A input.

If jumper E3 is ON, the amplifier-fault signal for the first channel on the board is tied to what is normally the channel’s encoder B input phase. If the jumper is in its default OFF state, the amplifier fault state is independent of the state of the encoder B input phase. In this case, the amplifier fault signal will usually come through the amplifier mezzanine board.

If jumper E4 is ON, the amplifier-fault signal for the second channel on the board is tied to what is normally the channel’s encoder B input phase. If the jumper is in its default OFF state, the amplifier fault state is independent of the state of the encoder B input phase. In this case, the amplifier fault signal will usually come through the amplifier mezzanine board.

Jumpers for Sharing Incremental and Serial Encoders

If jumper E5 (for revisions -107 and newer, released at the beginning of 2012) is in its default state of connecting pins 1 and 2, when **Gate3[*i*].Chan[*j*].SerialEncEna** for the first channel on the board is set to 1, what are normally the channel’s incremental encoder C index channel inputs are used instead for the serial encoder SENCENA+/- outputs. (On older board revisions, this is always the case.) If E5 connects pins 2 and 3, these pins are always used for the incremental encoder C index channel inputs, regardless of the setting of **Gate3[*i*].Chan[*j*].SerialEncEna**. Most serial encoder protocols do not need the SENCENA+/- lines, and this setting permits the use of an incremental encoder with index simultaneously with a serial encoder on the same channel.

If jumper E6 (for revisions -107 and newer, released at the beginning of 2012) is in its default state of connecting pins 1 and 2, when **Gate3[*i*].Chan[*j*].SerialEncEna** for the second channel on the board is set to 1, what are normally the channel's incremental encoder C index channel inputs are used instead for the serial encoder SENCENA+/- outputs. (On older board revisions, this is always the case.) If E6 connects pins 2 and 3, these pins are always used for the incremental encoder C index channel inputs, regardless of the setting of **Gate3[*i*].Chan[*j*].SerialEncEna**. Most serial encoder protocols do not need the SENCENA+/- lines, and this setting permits the use of an incremental encoder with index simultaneously with a serial encoder on the same channel.

Digital Amplifier Mezzanine Board (604005)

In almost all applications, the default hardware configuration of the digital amplifier-interface mezzanine board will be usable without change.

Jumper for Amplifier Fault Use

If jumper E1 is ON, the amplifier fault inputs on the connectors drive the amplifier fault state in the IC that the processor uses. This is the default condition. If the jumper is removed (OFF), the inputs on this board's connectors are not used. This is generally only done when the fault input is brought in from a stepper-style drive through the encoder connector on a feedback mezzanine board.

Analog Amplifier Mezzanine Board (604006)

In most applications, the default hardware configuration of analog amplifier-interface mezzanine board will be usable without change. This configuration supplies power to the analog circuitry from the UMAC's own power supply, defeating isolation. Some users will want this analog circuitry isolated from the core UMAC circuitry; these users will need to change the default configuration and provide external power.

Note that in the simplest configuration of this board, with a single 16-bit DAC per channel, no optical isolation is provided, and the jumpers are hardwired to connect the analog and digital power and ground lines together. There are no user-settable jumpers in this configuration.

Analog Circuit Isolation Jumpers

The analog amplifier mezzanine board has three "E-point" jumpers that determine whether the analog circuitry on the board is isolated from the main UMAC circuits or not. The factory default state is that all three jumpers are ON, so there is no isolation. The analog circuits are powered from the UMAC power supply through the UBUS backplane and the ACC-24E3 3U-format board on which this board is mounted.

If the three jumpers are removed (OFF), the analog circuitry on this board is electrically isolated from the main UMAC circuits. In this case, the analog supply for the board must be supplied through one of the two external connectors on the board (J1 or J2 in the D-sub configuration; TB1 or TB2 in the terminal-block configuration). The supply must be +/-12V to +/-15V with a 0V "AGND" connection.

If jumper E1 is ON (default), the analog common AGND is connected to the digital command GND, defeating isolation. If jumper E1 is OFF, AGND and GND are not connected, making isolation possible.

If jumper E2 is ON (default), the UMAC’s +15V supply from the backplane is used to power the analog circuitry on this board. If the jumper is OFF, the UMAC’s +15V supply is not used, and an external supply must be used.

If jumper E3 is ON (default), the UMAC’s -15V supply from the backplane is used to power the analog circuitry on this board. If the jumper is OFF, the UMAC’s -15V supply is not used, and an external supply must be used.

All three jumpers should be in the same ON/OFF state.

Diode protection on the +/-15V supply lines means that there will be no circuit damage if both internal and external supplies are connected. The higher-voltage supply will be used by the board.

Jumper for Amplifier Fault Use

If jumper E4 is ON, the amplifier fault inputs on the connectors drive the amplifier fault state in the IC that the processor uses. This is the default condition. If the jumper is removed (OFF), the inputs on this board’s connectors are not used. This is generally only done when the fault input is brought in from a stepper-style drive through a feedback mezzanine board.

Connections

The ACC-24E3 has multiple connectors to link it with the UMAC rack and the external devices used in the system. The key issues involved in using each connector are described in this section. Individual pin listings are given the “Connector Pinouts” chapter.

3U-Format Base Board (604002)

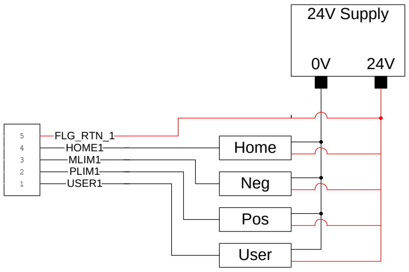
P1 (Rear) UBUS32 Backplane Connector

The P1 connector on the rear of the 3U-format base board installs in a matching connector on the UMAC’s UBUS backplane board. Note that the addressing of the ACC-24E3 by the Power PMAC CPU is not dependent on the physical slot used of the backplane; rather, it is determined by the settings of the DIP switches on SW1 on the board.

TB1, TB2 (Front) Input Flag Connectors

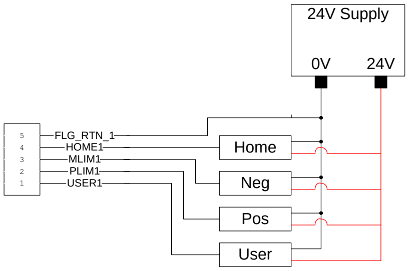
TB1 and TB2 are removable 5-point terminal blocks on the front edge of the board for the connection of the input flags (HOME, PLIM, MLIM, and USER) for Channels 1 and 2, respectively. The flags are 12V to 24V (unless socketed resistor packs are added so they can be 5V signals), sinking or sourcing, optically isolated from the main circuitry on the board.

For sinking drivers, the FL_RT (flag return) line for the channel should be connected to the external +V supply, and each flag connected between the pin on this connector and the 0V supply reference. The following drawing shows typical connections:



Sinking Flags Suggested Wiring

For sourcing drivers, the FL_RT line for the channel should be connected to the external 0V supply reference, and each flag connected between the pin on this connector and the +V supply. The following drawing shows typical connections:



Sourcing Flags Suggested Wiring

TB3 (Front) Position-Compare Output Flag Connector

TB3 is a removable 3-point terminal block on the front edge of the board for the connection of the position-compare outputs for both channels. These are sinking outputs with internal pull-up resistors to 5V. They are not optically isolated from the main circuitry on the board.

3U-Format Piggyback Board (604002)

TB1, TB2 (Front) Input Flag Connectors

TB1 and TB2 are removable 5-point terminal blocks on the front edge of the board for the connection of the input flags (HOME, PLIM, MLIM, and USER) for Channels 3 and 4, respectively. The flags are 12V to 24V (unless socketed resistor packs are added so they can be 5V signals), sinking or sourcing, optically isolated from the main circuitry on the board.

For sinking drivers, the FL_RT (flag return) line for the channel should be connected to the +V supply, and each flag connected between the pin on this connector and the 0V supply reference. For sourcing drivers, the FL_RT line for the channel should be connected to the 0V supply reference, and each flag connected between the pin on this connector and the +V supply. The connections for these flags are the same as for those on the base board, described above.

TB3 (Front) Position-Compare Output Flag Connector

TB3 is a removable 3-point terminable block on the front edge of the board for the connection of the position-compare outputs for both channels. These are sinking outputs with internal pull-up resistors. They are not optically isolated from the main circuitry on the board.

Digital Feedback Mezzanine Board (604003)

J1/TB1, J2/TB2 Encoder Connectors

The encoder connectors for the two channels on the digital feedback mezzanine board can be either 15-pin D-sub connectors or 12-point removable terminal blocks. They support quadrature, Hall-style, and serial-encoder feedback. They also support pulse-and-direction output for stepper and “stepper-replacement” servo drives.

A +5V supply and GND return are available on each connector to power an encoder.

The quadrature and index inputs can be RS-422-style differential signal pairs CHA+/-, CHB+/-, and CHC+/-, or single-ended signals can be wired into the “+” inputs. The “-” inputs are tied to +3V with voltage dividers if not driven externally, permitting the use of single-ended encoders.

The U, V, W, and T flag inputs, usually used for Hall-style commutation sensors and digital-output temperature sensors, are 5V CMOS inputs with internal pull-up resistors. These flags share pins with the serial encoder inputs and the pulse-and-direction outputs; the user must configure in software which use the pins have. If the Hall commutation sensors are used for ongoing phase or servo position (not just power-on phase angle), the U, V, and W signals must be wired into the A+, B+, and C+ inputs on the connector.

The serial encoder lines are RS-422-style differential signal pairs. SENCENA+/- and SENCCLK+/- are always outputs. SENCDAT+/- can be bidirectional or always inputs, depending on the serial encoder protocol. The ACC-24E3 manages the input/output direction control of these lines automatically. These signals share pins with the U, V, W, and T flag inputs, the pulse-and-direction outputs, and the incremental encoder index channels; the user must configure in software which use the pins have.

The PULSE+/- and DIR+/- outputs that can be used for stepper drives are RS-422-style differential signal pairs. With optional jumpers installed on the board, AENA+/- outputs and FAULT+/- inputs are also available as differential signal pairs. These signals share pins with the U, V, W, and T flag inputs and the serial encoder signals; the user must configure in software which use the pins have.

Analog Feedback Mezzanine Board (604004)

J1/TB1, J2/TB2 Encoder/Resolver Connectors

The encoder/resolver connectors for the two channels on the analog feedback mezzanine board are 15-pin D-sub connectors. (Terminal blocks are not available due to the difficulty in obtaining satisfactory shielding.)

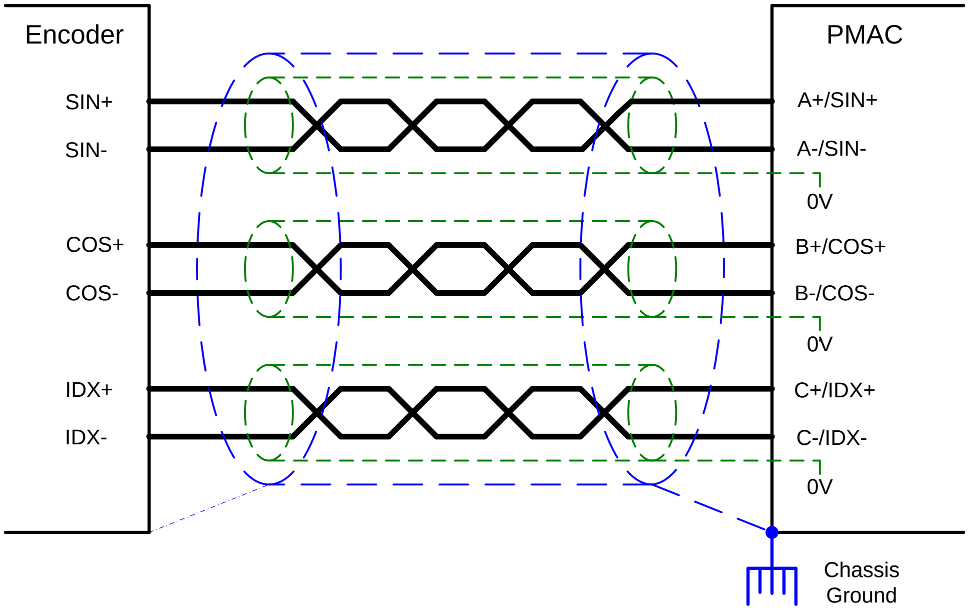
A +5V supply and GND return are available on each connector to power an encoder.

For sinusoidal encoders, the SIN+/-, COS+/-, and INDEX+/- inputs should be 1V peak-to-peak signals in the 0 to 5V range. If differential signal pairs are used, the signals must stay within the 0 to 5V range and both signals of a pair should have the same average and range. If single-ended signals are used, they should be connected to the "+" inputs, and the 2.5V reference voltage available on the BVREF pin should be wired back into the "-" inputs. This reference voltage should also be used at the encoder to "center" the signals on 2.5V.

Proper shielding and grounding of sinusoidal encoder signals is very important to minimize noise on the signal lines. The encoder cable should be kept pyhysically separate from the motor power cable if at all possible. Both motor cable and encoder cable should be shielded. If the motor is driven with PWM signals from the amplifier, chokes should be installed on the cables to "soften" the transitions.

The signal pairs should utilize twisted-pair wiring. Double shielding of the encoder signals is recommended. Optimally, each signal pair has its own inner shield, but it is more common to include all the signals inside the same inner shield. The inner shield(s) should be connected to the signal ground at the controller end only. The outer shield should be connected directly to chassis ground.

The following drawing illustrates the optimal wiring, shielding, and grounding for a sinusoidal encoder.



Suggested Wiring, Shielding, and Grounding for Sinusoidal Encoders

If desired, the ALTSIN+/- and ALTCOS+/- inputs can be connected in the same manner as SIN+/- and COS+/-. Some sinusoidal encoders provide these signals with one cycle per revolution for power-on commutation angle information.

It is possible to connect a digital quadrature encoder into the SIN+/-, COS+/-, and INDEX+/- inputs (for A+/-, B+/-, and C+/- encoder signals). For single-encoded quadrature encoders, it is necessary to tie the SIN-, COS-, and INDEX- inputs to the 2.5V BVREF output on the connector.

For resolvers, the excitation winding should be driven between the RESOUT and GND lines. The SIN+/- and COS+/- inputs come from the sine and cosine windings of the resolvers. These input signals should stay within a +/-5V range. Their magnitude is determined by the magnitude of the excitation output, which is software-controlled on the ACC-24E3, and the winding ratio of the resolver.

If desired, the ALTSIN+/- and ALTCOS+/- inputs can be connected in the same manner as SIN+/- and COS+/-. These inputs can be from a second geared resolver to obtain multi-turn power-on position.

The U, V, W, and T flag inputs, usually used for Hall-style commutation sensors and digital-output temperature sensors, are 5V CMOS inputs. These flags share pins with the serial encoder inputs, the secondary analog inputs, and the pulse-and-direction outputs; the user must configure in software which use the pins have.

The serial encoder lines are RS-422-style differential signal pairs. SENCENA+/- and SENCCLK+/- are always outputs. SENCDAT+/- can be bidirectional or always inputs, depending on the serial encoder protocol. The ACC-24E3 manages the direction control of these lines automatically. These signals share pins with the U, V, W, and T flag inputs, the pulse-and-direction outputs, and the incremental encoder index channels; the user must configure in software which use the pins have.

The PULSE+/- and DIR+/- outputs that can be used for stepper drives are RS-422-style differential signal pairs. With optional jumpers installed on the board, AENA+/- outputs and FAULT+/- inputs are also available as differential signal pairs. These signals share pins with the U, V, W, and T flag inputs, the secondary analog inputs, and the serial encoder signals; the user must configure in software which use the pins have.

Digital Amplifier Mezzanine Board (604005)

J1, J2 Amplifier Connectors

The amplifier connectors on the digital amplifier mezzanine board are 36-pin Mini-D connectors. They are intended for straight-across connection to direct-PWM "power block" amplifiers with standard pinouts using IEEE-1284C cables. All signals are RS-422-style differential signal pairs.

These connectors can also be used for pulse-and-direction input drives (stepper drives or "stepper-replacement" servo drives). In this case, just the Phase D outputs are used for the command values – PULSE+/- comes out on PWMDBOT+/-, and DIR+/- comes out on PWMDTOP+/-. Bit 3 (value 8) of **Gate3[*i*].Chan[*j*].OutputMode** must be set to 1 to enable this signal format.

Analog Amplifier Mezzanine Board (604006)

J1/TB1, J2/TB2 Amplifier Connectors

The amplifier connectors for the two channels on the analog amplifier mezzanine board can be either 15-pin D-sub connectors or 12-point removable terminal blocks. (However, when 18-bit DACs are used, only the D-sub connectors are available. They support +/-10V analog velocity-mode or torque-mode servo drives with a single output per channel. They also support "sine-wave input" servo drives with two +/-10V analog outputs (A and B) per channel.

If the E1, E2, and E3 jumpers on the board are OFF so no analog power comes from the UMAC backplane, analog power must be supplied through one of the connectors on the board, using the A+15V, A-15V, and AGND pins. Either a +/-12V or a +/-15V supply may be used. If these jumpers are ON so the board gets its analog power from the backplane, it is not possible to output these voltages because of protective diodes on the +/-V pins. The AGND line can be used to establish a common reference with the input stage of the amplifier.

The analog outputs can be used either single-ended (DAC+ vs. AGND) or differential (DAC+ vs. DAC-). Remember that the effective voltage when using differential outputs is twice that of using single-ended, and this will affect both the servo-loop gain and the output-limit values.

The amplifer-enable output signals come from a solid-state relay. This provides the maximum flexibility in user wiring. Both normally-open (closed when enabled) and normally-closed (open when enabled) contacts are available, along with a "common" that connects to both. It can be configured for either sinking or sourcing output. Note that there is no software control of the on/off polarity of the amplifier-enable output – this configuration must be done through choice of the output signal used and wiring.

The amplifier-fault input pair FAULT+/- simply connect through an "AC opto" component on the board. When there is at least a 12V difference between the pins in either direction (creating at least +/-5 mA of current), the opto-coupler will turn on sufficiently to create a "0" state in the “fault” status bit of the IC. Note that the fault polarity is determined by the motor setup element **Motor[*x*].AmpFaultLevel**.

Software Setup

Much of the hardware on the ACC-24E3 is software configurable to provide the maximum flexibility. This section explains the software setup that is required to use the ACC-24E3 in different modes of operation.

DSPGATE3 Data Structure Elements

Settings in the "DSPGATE3" machine-interface IC control the software configuration of the ACC-24E3 for a particular application. The Power PMAC has defined a "**Gate3**" data structure to organize the control and status settings of the ASIC. In the Script environment – either buffered program statements or on-line commands – a user can utilize this **Gate3[*i*]** data structure directly, or use its "alias" of the **Acc24E3[*i*]** data structure. In C programs, the user must utilize the **Gate3[*i*]** data structure directly. This manual will use the **Gate3[*i*]** name for maximum generality.

Some of the data structure elements are full 32-bit words, and some are "partial-word" elements of less than 32 bits. Script commands can directly access both types. However, C programs can only access the full-word elements, and to isolate the value of a partial-word element, must explicitly perform the masking and shifting functions in the program.

Elements of the data structure that affect all servo channels are of the form:

Gate3[*i*].*{element name}*

The ASIC index number "***i***" has a range of 0 to 15 and corresponds to the setting of the addressing DIP switches on SW1 of the 3U-format base board.

Elements of the data structure that affect only a single servo channel are of the form:

Gate3[*i*].Chan[*j*].*{element name}*

The channel index number "***j***" has a range of 0 to 3, corresponding to hardware channels 1 to 4, respectively, for the accessory. *The index number for a channel is one less than the corresponding hardware channel number.*

Channel index numbers 0 and 1 correspond to hardware channels 1 and 2, which are on the 3U-format base board and its mezzanine boards, on the left side of a 4-channel assembly. Channel index numbers 2 and 3 correspond to hardware channels 3 and 4, which are on the 3U-format piggyback board and its mezzanine boards, on the right side of a 4-channel assembly.

Write-Protect Key for Setup Elements

In order to prevent potentially dangerous changes to setup elements in the DSPGATE3 IC by unauthorized users, many of the setup elements are "write protected" and cannot be changed unless the "write-protect key" in the IC has been set properly before writing to the protected setup element.

The write-protect key register can be accessed through the element **Gate3[*i*].WpKey**, and the value that must be in this register to enable a write operation to a protected setup element is $AAAAAAAA. The IC automatically clears the key value to 0 after writing to the protected element, so the key must be set once for each write-operation to a protected element.

In a C program, this must be done explicitly. That is, each operation to write a value to a protected setup element must be preceded by an operation to write $AAAAAAAA to **Gate3[*i*].WpKey**.

In the script environment – either buffered program statements or on-line commands – there is a second method that can make the process easier. Before any script operation that writes to a protected setup element, Power PMAC will copy the value in the global memory element **Sys.WpKey** into the actual write-protect key register **Gate3[*i*].WpKey**. This permits the user to write the key value of $AAAAAAAA just once into the global element, and Power PMAC will copy it automatically into the IC register every time it is needed.

Sys.WpKey is *not* a saved value, so it must be set after power-up/reset every time you wish to change protected setup elements in the IC. The special windows in the Integrated Development Environment (IDE) program for setting up the IC automatically manage the value in **Sys.WpKey**.

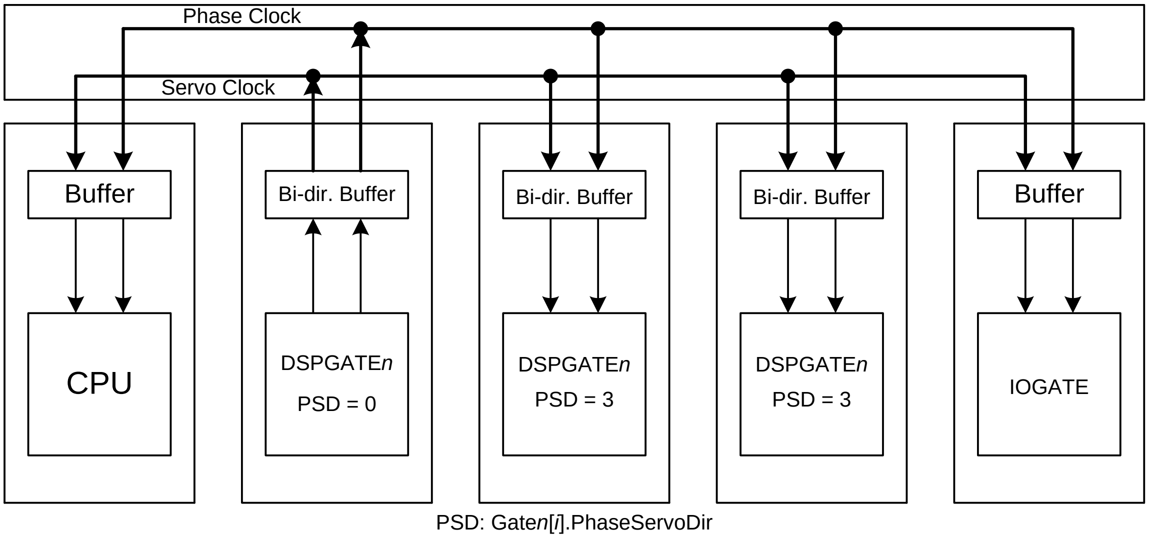
Of course, if you wish to prevent changes to these setup elements in the actual application, **Sys.WpKey** should not be set to the proper key value, and unauthorized users should not be given information about the key value or mechanism.

Software Setup for Clock Signals

The clock frequencies of an ACC-24E3 are set by software data structure elements. Some of these clock frequencies can be used to control the entire Power PMAC system.

Phase and Servo Clock Direction

In a Power PMAC UMAC system, only one machine interface IC is the source of phase and servo clock signals for the entire system. This IC generates its own phase and servo clock signals and outputs them to the rest of the system over the UBUS backplane. The Power PMAC CPU uses these signals as interrupts to drive its phase-commutation and servo-loop algorithms, respectively. Other machine interface ICs use these clock signals to drive their own input and output functions. The following diagram shows how the system clock distribution works:



Power PMAC System Clock Distribution

On the re-initialization of a Power PMAC UMAC system, the CPU will automatically set up the lowest-numbered DSPGATE3 IC found as the source of the system clock signals. It does this by setting **Gate3[*i*].PhaseServoDir** for this IC to 0. It sets **Gate3[*i*].PhaseServoDir** for all of the other ICs it finds to 3, so they will input their phase and servo clock signals.

Phase Clock Frequency

Gate3[*i*].PhaseFreq sets the frequency of the phase clock signal that this IC generates. It is a floating-point number, with units of hertz (Hz). For the IC that is generating the system clock signals, this element determines the phase clock frequency for the entire system. The default value of 9035 specifies a phase clock frequency of 9.035 kHz.

For ICs that are receiving external phase clock signals, it is best to set **Gate3[*i*].PhaseFreq** to the same value as the system clock frequency, or as close as possible. These ICs are locked to the received system clock frequency by a digital phase-locked loop inside the IC, but this circuit works best if the internal frequency is as close as possible to the received frequency.

It is possible to divide down the received phase clock frequency by a factor of 2, 3, or 4 by setting **Gate3[*i*].PhaseClockDiv** to 1, 2, or 3, respectively. This is rarely done, however – the only real reason is that it supports lower PWM frequencies on this IC.

Similarly, for the IC that is outputting its phase clock signal to the system, it is possible to multiply the output phase clock frequency by a factor of 2, 3, or 4 by setting **Gate3[*i*].PhaseClockMult** to 1, 2, or 3, respectively. This also is rarely done, and is done only for reasons of providing more flexibility in the setting of PWM frequencies.

Servo Clock Frequency

Gate3[*i*].ServoClockDiv sets the internally generated servo clock frequency for the IC by specifying how many times the frequency is divided down from the phase clock frequency. It has a range of 0 to 15, specifying a division factor of 1 to 16, respectively. The default value is 3, specifying a 4-times division. With the default phase-clock frequency of 9.035 kHz, this yields a servo-clock frequency of 2.259 kHz.

The equation for **Gate3[*i*].ServoClockDiv** is:

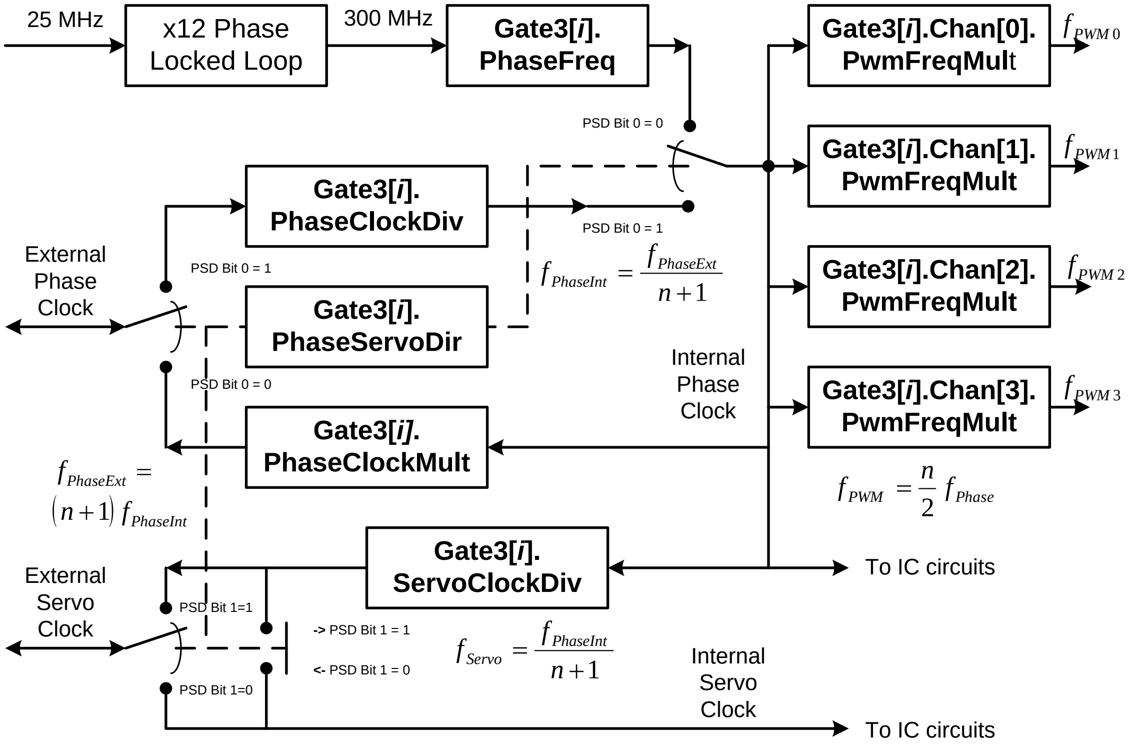
$$Gate3[i].ServoClockDiv = \frac{PhaseFreq(kHz)}{ServoFreq(kHz)} - 1$$

or:

$$ServoFreq(kHz) = \frac{PhaseFreq(kHz)}{(Gate3[i].ServoClockDiv + 1)}$$

For ICs receiving an external servo clock signal, the setting of this element does not really matter, but users are encouraged to set it to the same value as on the IC that is generating the system clock signals.

The following block diagram shows how the clock-direction and clock-frequency controls are used in the ASIC.



DSPGATE3 IC System Clock Generation Controls

The global software setup element **Sys.ServoPeriod** (in milliseconds) must be set to the inverse of the system servo-clock frequency expressed in kilohertz (kHz). This parameter tells the trajectory interpolation algorithms how far to advance the commanded motion equations each servo cycle. (It does not actually set the physical servo clock frequency.) It can be calculated as:

$$Sys.ServoPeriod = 1000 * \frac{(Gate3[i].ServoClockDiv + 1)}{Gate3[i].PhaseFreq}$$

Hardware Clock Frequencies

Each DSPGATE3 has six internal clock signals that control its own hardware features. These clock signals are not shared between ICs, and do not need to be the same on different ICs. Each of these clock signals has a saved setup element that determines its frequency:

- **Gate3[*i*].AdcAmpClockDiv** – Bit clock for amplifier A/D converters
- **Gate3[*i*].AdcEncClockDiv** – Bit clock for encoder A/D converters
- **Gate3[*i*].DacClockDiv** – Bit clock for D/A converters
- **Gate3[*i*].PfmClockDiv** – Clock for PFM pulse-generation circuits
- **Gate3[*i*].EncClockDiv** – Sampling clock for encoder and discrete-input circuits
- **Gate3[*i*].FiltClockDiv** – Secondary filtering clock for discrete inputs

Each of these parameters can take a value "*n*" from 0 to 15, specifying a frequency division factor of 2^n from the source clock frequency. For the first five of the listed clock frequencies, the source clock frequency is 100 MHz, so the possible frequencies are 100 MHz, 50 MHz, 25 MHz, 12.5 MHz, etc., down to about 3 kHz. These parameters each have a default value of 5 for a division factor of 32, yielding a 3.125 MHz frequency for their clock signal. This frequency is suitable for most applications.

For these five clock signals, the following table shows the possible settings of **Gate3[*i*].*xxx*ClockDiv** and the frequencies they produce:

Setting	Divisor	Frequency	Setting	Divisor	Frequency
0	1	100 MHz	8	256	390.6 kHz
1	2	50 MHz	9	512	195.3 kHz
2	4	25 MHz	10	1,024	97.65 kHz
3	8	12.5 MHz	11	2,048	48.82 kHz
4	16	6.25 MHz	12	4,096	24.41 kHz
5	32	3.125 MHz	13	8,192	12.21 kHz
6	64	1.562 MHz	14	16,384	6.104 kHz
7	128	781.2 kHz	15	32,768	3.052 kHz

For the secondary filtering clock, the source frequency is the encoder sampling clock, which has already performed a primary filtering function. This parameter has a default value of 4 for a division factor of 32, yielding a default 195.3 kHz frequency for the secondary digital delay filter.

Software Setup for Position Feedback

The ACC-24E3 provides interfaces for many different types of position feedback sensors, both digital and analog. Each sensor type has some required software configuration in order to use properly.

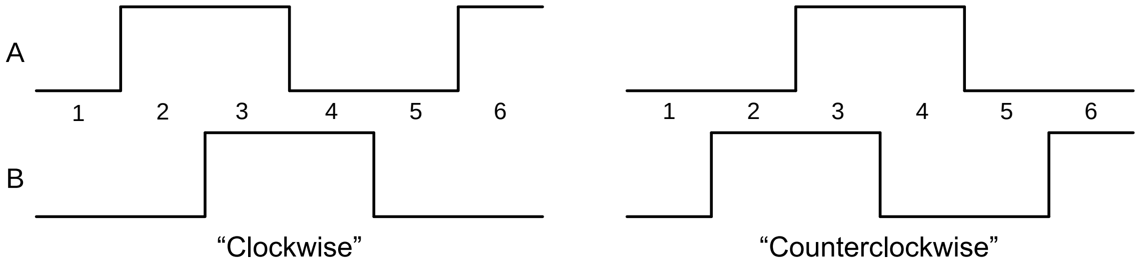
Quadrature Encoders

The most commonly used feedback is digital quadrature encoders. The Power PMAC with ACC-24E3 has many features to provide powerful and flexible, but easy-to-use, functionality with these encoders. The quadrature signals, by convention called “A” and “B” are each nominally of 50% duty cycle and ¼-cycle out of phase with each other, with the direction sense inferred from which signal is leading the other.

IC Hardware Setup

Gate3[*i*].Chan[*j*].EncCtrl determines how the IC will decode the quadrature (or similar) signal connected to the A and B inputs of the digital feedback mezzanine board. Most commonly, a setting of 3 or 7 is used to specify a “times-4” (x4 – four counts per encoder line) decode in either the “clockwise” or “counter-clockwise” direction sense. “Times-2” (x2) and “times-1” (x1) decodes are also possible, but rarely used, as they provide lower count resolution from the same encoder. They are typically only used if the encoder deviates substantially from the 50% duty-cycle and ¼-cycle phase-difference specification.

The following diagram shows the x4 clockwise and counterclockwise decoding for quadrature encoders.



Incremental Quadrature Encoder “Times-4” Decode

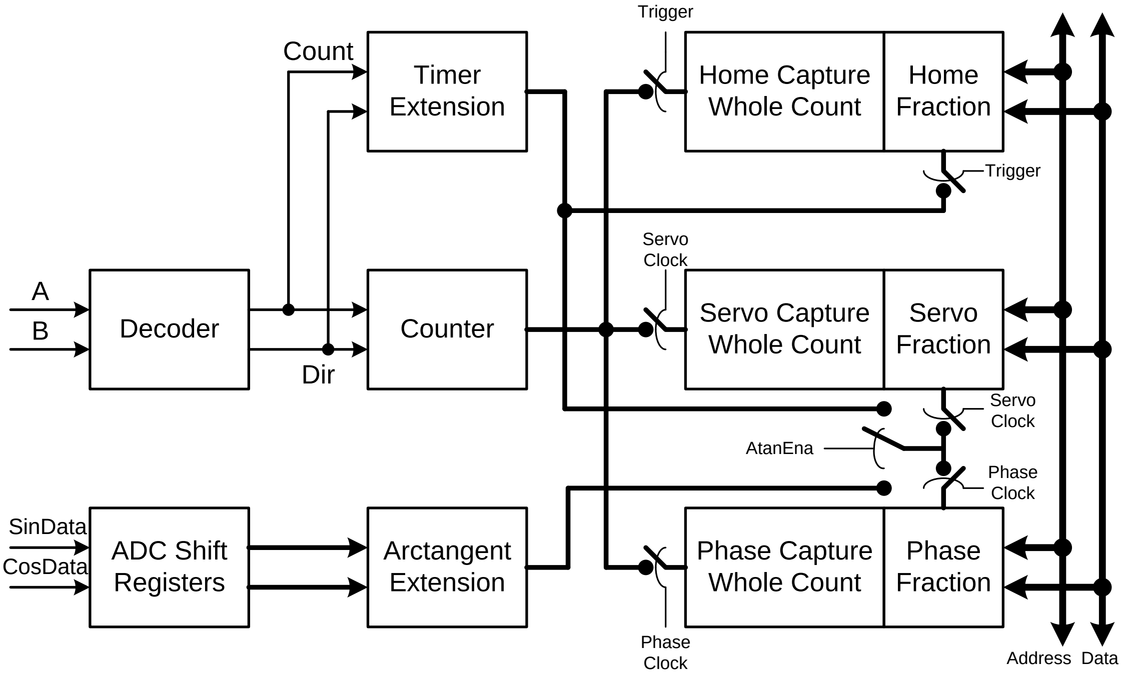
When used for servo-loop feedback, the direction sense of the feedback must match the direction sense of the servo-loop output, or a dangerous runaway condition may result when the loop is closed. That is, a positive servo-loop output must cause the position feedback to increment in the positive direction. *Changing the direction sense of the decode for the feedback encoder of a motor that is operating properly will result in unstable positive feedback and a dangerous runaway condition in the absence of other changes.*

To enable the IC to perform the popular “hardware 1/T extension” to estimate fractional count position based on timer values, the element **Gate3[*i*].Chan[*j*].TimerMode** must be set to its default value of 0. In this mode, the low 8 bits of the 32-bit position value latched into the channel registers **Gate3[*i*].Chan[*j*].ServoCapt** (on the servo-clock interrupt), **Gate3[*i*].Chan[*j*].PhaseCapt** (on the phase-clock interrupt), and **Gate3[*i*].Chan[*j*].HomeCapt** (on the specified trigger condition) will be fractional-count data, yielding a resolution of 1/256 of a count for these registers.

In this mode, the 32-bit register **Gate3[*i*].Chan[*j*].TimerA** contains the position latched on the servo-clock interrupt with 12 bits of fractional-count data, and **Gate3[*i*].Chan[*j*].TimerB** contains the position latched on the specified trigger condition with 12 bits of fractional count data. It is possible to use these registers instead of **Gate3[*i*].Chan[*j*].ServoCapt** and **Gate3[*i*].Chan[*j*].HomeCapt**, respectively, providing 4 additional timer-estimated fractional count data; in most applications, this will not provide a noticeable improvement in performance.

The element **Gate3[*i*].Chan[*j*].AtanEna** should be set to its default value of 0 so that the IC does *not* calculate fractional-count data with arctangent calculations from the encoder ADC registers. (That mode is intended for sinusoidal encoders, as explained below.)

The following diagram shows how the incremental encoder interface circuitry for a channel operates with timer and arctangent extension modes.



DSPGATE3 IC Channel Incremental Encoder Interface Circuitry

Use for Ongoing Commutation Feedback

If quadrature encoder feedback is used for commutation position angle, **Motor[*x*].pPhaseEnc** should be set to **Gate3[*i*].Chan[*j*].PhaseCapt.a** to use the count value (with fractional extension). The user must calculate how many LSBs of this 32-bit register there are per commutation cycle in order to set the commutation scale factor **Motor[*x*].PhasePosSf**, which multiplies the value in the source register to obtain the commutation angle.

With a digital quadrature encoder, there are 8 bits of fractional count data in the **PhaseCapt** register, so there are 256 LSBs per quadrature count. If there were 2000 quadrature counts per commutation cycle, there would be 512,000 LSBs per commutation cycle. In this case, **Motor[*x*].PhasePosSf** would be set to 2048 divided by 512,000. (It is best to enter this value as an expression and let Power PMAC calculate the double-precision floating-point value itself).

As an incremental sensor, a quadrature encoder is not suitable for establishing the absolute phase position angle at power-on. Either a separate sensor that is absolute over a commutation cycle (such as a Hall commutation sensor) is required, or a phasing-search move will be required after power-on.

Use for Ongoing Servo Feedback or Master

To process the resulting position value for servo-loop use (feedback or master) in the servo interrupt, a “Type 1” single-register-read conversion should be specified in the Encoder Conversion Table. (Note that the IC has already performed the 1/T extension in hardware, so the “Type 3” “software 1/T extension” method should not be used.) The IDE’s setup menu for the ECT allows you to specify an entry just by specifying the method, the IC index number, and channel index number. The result will be a value scaled in counts, with 8 bits of fractional-count extension (for a resolution of 1/256 of a count).

To set up the entry manually, make settings of the following type:

EncTable[*n*].pEnc, which specifies the address of this register, should be set to **Gate3[*i*].ServoCapt.a**. (**EncTable[*n*].pEnc1**, which specifies a second source, is not used in this mode; its setting does not matter.)

Conversion parameters **EncTable[*n*].index1**, **.index2**, **.index3**, **.index4**, and **.MaxDelta** are generally all set to 0 to use this register.

To get the output of the conversion table in units of counts (most common), **EncTable[*n*].ScaleFactor** should be set to 1/256.

To use the result of the conversion table entry for outer (position) loop motor feedback, set **Motor[*x*].pEnc** to **EncTable[*n*].a**. To use the result for inner velocity loop motor feedback, set **Motor[*x*].pEnc2** to **EncTable[*n*].a**. To use the result as a master position for the motor’s position-following function, set **Motor[*x*].pMasterEnc** to **EncTable[*n*].a**. To use the result for a time-base master frequency, set **Coord[*x*].pDesTimeBase** to **EncTable[*n*].DeltaPos.a**.

If the servo loop for the motor is closed in the phase interrupt (typically done only for one or two high-bandwidth actuators), a feature enabled by setting **Motor[*x*].PhaseCtrl** bit 3 (value 8) to 1, then the encoder conversion table, **Motor[*x*].pEnc**, and **Motor[*x*].pEnc2** are not used. Instead, Power PMAC directly reads the register whose address is contained in **Motor[*x*].pPhaseEnc** for both inner (velocity) and outer (position) loop servo feedback (regardless of whether Power PMAC is doing phase commutation for the motor or not).

To use the channel’s encoder feedback for this purpose, **Motor[*x*].pPhaseEnc** should be set to **Gate3[*i*].Chan[*j*].PhaseCapt.a**. Whole counts are in the high 24 bits of this 32-bit register, so each count of the encoder is equivalent to 256 (2^8) LSBs of the register. If **Gate3[*i*].Chan[*j*].TimerMode** is set to the default value of 0, the low 8 bits will contain timer-based fractional count estimation. Otherwise, they will be set to 0. Unlike in the conversion table, there is no capability to shift this data around to get the result in units of counts.

As an incremental sensor, a quadrature encoder is not suitable for establishing the absolute servo position at power-on. Either a separate sensor that is absolute over the entire travel of the motor is required, or a homing-search move will be required after power-on.

Use for Trigger Position

To use the IC channel's hardware-captured position value as the "trigger position" for motor triggered moves such as homing-search moves **Motor[*x*].CaptureMode** must be set to 0 to select hardware trigger and hardware capture. Also, **Motor[*x*].pCaptPos** must be set to **Gate3[*i*].Chan[*j*].FlagCapt.a**, the address of the register that latches the present encoder position immediately on the selected input trigger condition. This latches a 32-bit value with whole counts in the high 24 bits. If **Gate3[*i*].Chan[*j*].TimerMode.a** is set to the default value of 0, the low 8 bits will contain a timer-based "1/T" fractional count estimation at the instant of the trigger; otherwise the low 8 bits will contain all zeros.

This data must then be processed to match the scaling and offset of the servo feedback position. With quadrature feedback, most users will want to use only the "whole-count" portion of the captured data, particularly for homing. To do this, **Motor[*x*].CaptPosRightShift** should be set to 8 to shift out the low 8 bits of fractional count from the 32-bit read. **Motor[*x*].CaptPosLeftShift** should be set to 8 to match the 8 bits of fractional-count resolution that the DSPGATE3's 1/T extension uses for servo feedback (yielding 0's in the fractional portion). **Motor[*x*].CaptPosRound** should be set to 1 to enable the half-count offset of the captured position that will match the half-count offset performed on extended servo feedback.

These settings are made automatically when the Power PMAC is re-initialized by the `$$$***` command with an ACC-24E3 and a digital-feedback mezzanine board present. They are also made (with the address based on **Motor[*x*].pEncStatus**) when **Motor[*x*].EncType** is assigned a value of 5 (1/T extension in a PMAC3-style IC) in the Script environment.

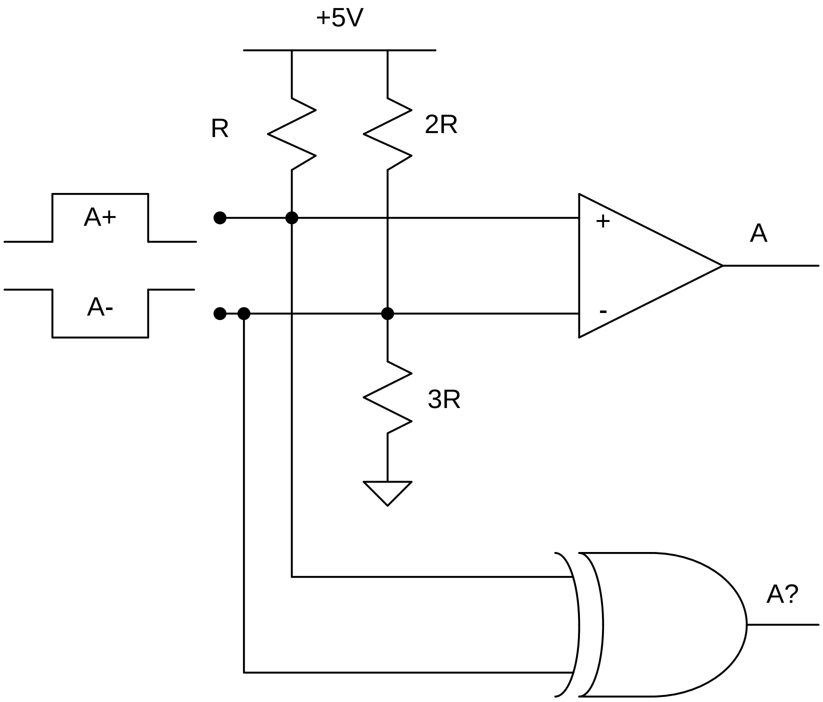
Some users will want to utilize the fractional-count portion of the captured data for the trigger position, particularly for probing and registration functions after the homing-search move. (But note that you cannot reliably control where you stop between whole-count edges, so this may not be of much use in many applications.) To do this, **Gate3[*i*].Chan[*j*].TimerMode**, **Motor[*x*].CaptPosRightShift**, **Motor[*x*].CaptPosLeftShift**, and **Motor[*x*].CaptPosRound** must all be set to 0.

Monitoring for Sensor Loss

The ACC-24E3 has circuitry that can detect loss of signal from differential incremental encoder inputs, and standard Power PMAC algorithms can use this circuitry to automatically disable motors when this loss is detected. The detection circuitry uses the fact that properly working differential input pairs keep the two inputs of the pair in opposite logical states. For the encoder counting circuitry these pairs are input into differential line receivers.

Resistors on the input will pull an undriven input signal to a high enough voltage so that it will be detected as a logical "1". The "+" input of a pair is pulled directly to 5V; the "-" input of a pair is pulled to 3V, which is a high TTL level – this permits the use of single-ended encoder signals connected to the "+" signals to be processed by the differential line receivers.

When the signal pairs are driven properly by differential line drivers in the encoder, one signal will automatically provide a logical high state and the other a logical low state. Feeding these signals into an "exclusive OR" (XOR) gate will produce a "true" output from the gate. However, if the signals no longer drive the circuitry, as would happen if the cable became disconnected, the resistors will put both signals in the logical high state, and the output of the XOR gate will be false. The following diagram shows the principle of the circuitry for one channel of the encoder.



Incremental Encoder Receiver and Loss-Detection Circuitry

If the output of the XOR gate on either the A or B channel of the encoder is false, the encoder is considered “lost”. The status bit **Gate3[*i*].Chan[*j*].LossStatus** in the channel’s status register **Gate3[*i*].Chan[*j*].Status** is set to 1. This is a “transparent” status bit that will return to 0 if the signal is regained. The “latched” status bit **Gate3[*i*].Chan[*j*].LossCapt** in the same register is set to 1 if a loss is detected, and held at 1 until specifically reset (writing a 0 to the bit will reset it).

To employ this circuitry for automatic loss detection, set **Motor[*x*].pEncLoss** to **Gate3[*i*].Chan[*j*].Status.a**. (It can also be set to **Gate3[*i*].Chan[*j*].LossStatus.a**, but will report back as **Gate3[*i*].Chan[*j*].Status.a**). Set **Motor[*x*].EncLossBit** to 28 to specify the **LossStatus** bit number in the register. Set **Motor[*x*].EncLossLevel** to 1 to specify a high-true loss status bit.

Because it is possible for signal transients to cause a momentary setting of this bit (the presence of these transients can be detected by noting that the **LossCapt** status bit is 1 when the **LossStatus** bit is generally 0), it is usually desirable to set **Motor[*x*].EncLossLimit** to a value greater than 0. The accumulated count of loss detection events must exceed this threshold before a fault is declared. Reasonable values for this element will not unduly delay reaction to true faults.

Hall-Style Commutation Sensors

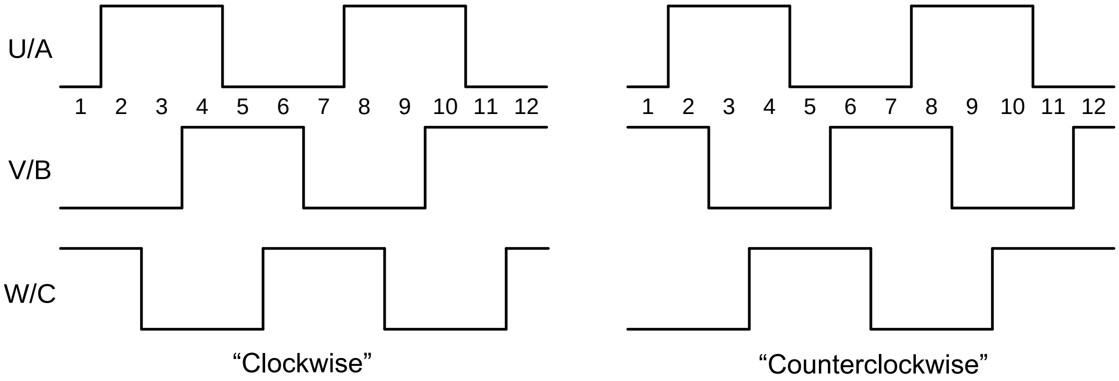
Digital Hall-style commutation sensors are frequently used with ACC-24E3 boards. These signals can be from true magnetic Hall-effect sensors, or from "Hall commutation tracks" on optical incremental encoders that mimic the signal output of the older magnetic sensors. In either case, they should provide three digital signals each of 50% duty cycle, offset either 1/3-cycle from each other ("120° spacing" – most common), or 1/6-cycle from each other ("60° spacing" – less common). The 120°-spacing format is strongly recommended, because the most common failure modes create states – all high or all low – that are not legal states, and so can easily be detected.

Most commonly, they are used to provide approximate power-on phase position when Power PMAC is performing the motor phase commutation, connected to the channel's U, V, and W flag inputs. Occasionally, they are used as the only position feedback, mostly in high-speed, high-power velocity-control applications, connected into the channel's A, B, and C encoder inputs.

IC Hardware Setup

If the Hall sensors are connected into the channel's U, V, and W flag inputs for power-on phase position, it is essential that the pins used on the ACC-24E3 connector not be configured in software for an alternate use. Saved setup element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to its factory default value of 0 so that serial encoder signals are not present on these pins. Non-saved element **Gate3[*i*].Chan[*j*].OutFlagD** must be set to its power-on default value of 0 so that pulse-and-direction signals are not present on these pins.

If the Hall sensors are connected into the channel's A, B, and C encoder inputs for ongoing phase and/or servo position feedback, **Gate3[*i*].Chan[*j*].EncCtrl** must be set to 11 or 15 to specify a "times-6" (x6 – six counts per cycle) decode in either the "clockwise" or "counter-clockwise" direction sense. This permits the Hall signals to be used like a quadrature encoder, except with 3-phase input. The following diagram shows how Hall sensors with 120° spacing are decoded into counts in both direction senses:



Hall Commutation Sensor "Times-6" Decode

Note that this feedback method can provide excellent velocity control at approximately constant speeds, particularly when 1/T extension of the Hall edge count is used, but the low fundamental position resolution means that there cannot be good position control at standstill.

The channel should be set up for timer-based "1/T" fractional count extension with **Gate3[*i*].Chan[*j*].TimerMode** set to the default value of 0. This provides estimation of position to 1/256 of a count. The channel's 32-bit "phase capture" register register

Gate3[*i*].Chan[*j*].PhaseCapt will hold the position latched on the phase interrupt, with the high 24 bits representing “whole counts” (each Hall signal edge is a “count”) and the low 8 bits representing the timer-based fractional-count extension. This permits smooth sinusoidal commutation even though the sensor provides only 6 discrete states per commutation cycle.

The channel’s 32-bit “servo capture” register **Gate3[*i*].Chan[*j*].ServoCapt** will hold the position latched on the servo interrupt, also with the high 24 bits representing “whole counts” and the low 8 bits representing the timer-based fractional-count extension. This extension permits very smooth velocity control, even with the low resolution of most Hall-style signals.

Use for Ongoing Commutation Feedback

In some applications, particularly for velocity control of large motors, often in harsh conditions or at very high speeds, it is not feasible to use an encoder or resolver for ongoing feedback, and Hall sensors are the only sensors available. With the three Hall sensors connected into the A, B, and C encoder inputs, and a “times-6” encoder decode set up, the ACC-24E3 can support this operation.

For ongoing commutation feedback, **Motor[*x*].PhasePos** should be set to **Gate3[*i*].Chan[*j*].PhaseCapt.a**, as for a quadrature encoder, to use the value of the counter latched on the phase interrupt. Since there are 6 counts per cycle and 8 fractional bits per count, **Motor[*x*].PhasePosSf** should be set to 2048/6/256 to convert the data to the 2048-state commutation cycle. The value should be entered as an expression so Power PMAC can compute the exact numerical value to double-precision floating-point resolution.

Use for Power-On Commutation Position

When used for power-on phase commutation position, the three Hall signals can be connected into the U, V, and W flag signals, or the A, B, and C encoder signals on the encoder connector. It does not matter which signal is connected into which input, but any change will affect the offset and potentially the direction sense of the encoded position value.

Motor[*x*].pAbsPhasePos should be set to **Gate3[*i*].Chan[*j*].Status.a** so the absolute phase position read function is enabled and uses the IC channel’s status register where the states of the U, V, and W inputs, or the A, B, and C inputs appear.

When connected into the U, V, and W inputs, **Motor[*x*].AbsPhasePosFormat** should be set to $0400030C to interpret the value in this register correctly. The “04” specifies the 120° Hall spacing format (“05” would specify 60° spacing); the “00” is not used; the “03” specifies using 3 bits of this register; and the “0C” specifies starting at bit 12 (= 0C hex) of the register. This setting tells Power PMAC to read bits 12, 13, and 14 of the specified register, where the W, V, and U inputs, respectively, are found, and to interpret them as 120°-spaced Hall-style sensors.

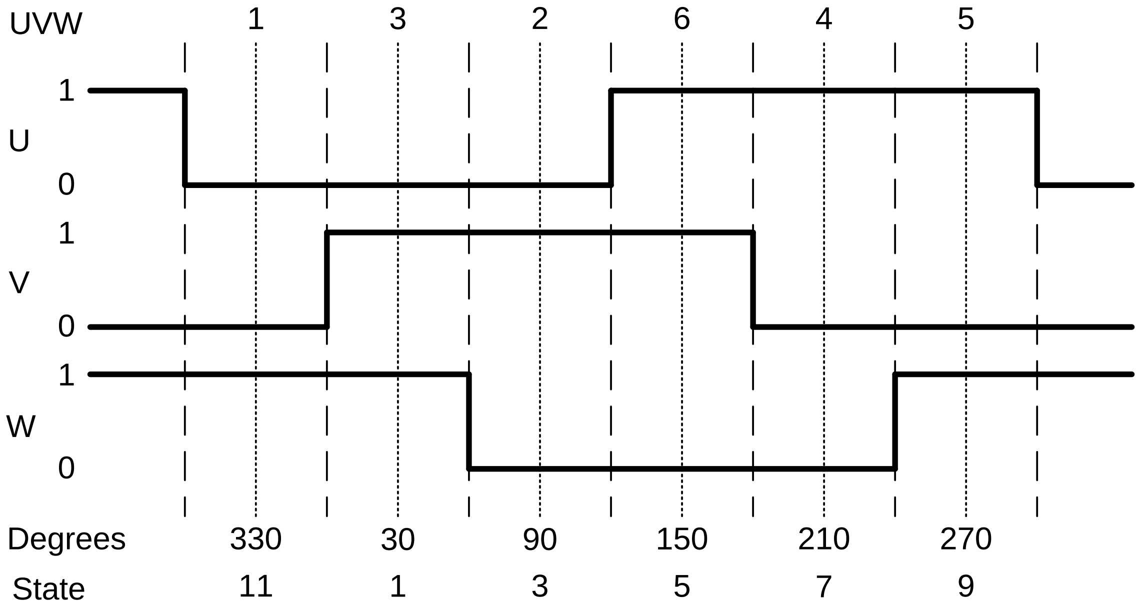
When connected into the A, B, and C inputs, **Motor[*x*].AbsPhasePosFormat** should be set to $04000304 instead, with the final “04” specifying starting at bit 4 of the register, telling Power PMAC to read bits 4, 5, and 6 of the specified register, where the A, B, and C inputs, respectively, are found.

Motor[*x*].AbsPhasePosSf multiplies the sensor value to convert it into commutation cycle units. Power PMAC considers the position value from a 3-phase Hall sensor to have 12 units per cycle – 6 states and 6 edges. (When reading the sensor, it assumes that it will be halfway between the edges.) The commutation cycle has 2048 units per cycle, so the magnitude of **Motor[*x*].AbsPhasePosSf** for Hall sensors should be 2048/12 = 170.667.

The proper sign of **Motor[*x*].AbsPhasePosSf** is determined by the direction sense of the Hall sensors relative to the ongoing commutation position sensor. For Hall sensors with the standard 120^o spacing, if W leads V, and V leads U in the counting-up sense of the ongoing commutation cycle, the direction sense agrees, and **Motor[*x*].AbsPhasePosSf** should be set to (+) 170.667. However, if U leads V, and V leads W in the counting-up sense of the ongoing commutation cycle, the direction sense does not agree, and **Motor[*x*].AbsPhasePosSf** should be set to -170.667.

After scaling the power-on position into commutation units, Power PMAC adds the value of **Motor[*x*].AbsPhasePosOffset** to this to get the power-on commutation angle. This parameter permits the Hall sensor to have a different zero position than the commutation cycle does. Power PMAC considers the zero position in the Hall cycle to be the transition of the V signal when U is low (0) and W is high (1).

The following diagram shows the 120° Hall signal format and the resulting "states". For the 60° spacing, invert the V signal from what is shown.



Hall Commutation Signal Format (120°) and States

The IDE motor setup utility can automatically determine the optimum value of **Motor[*x*].AbsPhasePosOffset**. However, it is not difficult to determine this directly.

The best way "manually" to determine the proper value of **Motor[*x*].AbsPhasePosOffset** is first to drive the (unloaded) motor to the commutation cycle zero point with the "stepper motor" phasing search. (This assumes that basic commutation has already been set up properly.) To do this, set **Motor[*x*].PhaseFindingTime** to a value of 256 or greater (the step time in servo cycles), and **Motor[*x*].PhaseFindingDac** to a large enough value to cause a reliable search (3000 should be enough for an unloaded motor). The phasing search can then be commanded with an on-line **#*x*$** command. (Note that you must address the motor immediately before the command – you cannot simply rely on the currently addressed motor.)

Now put **Motor[*x*].PhasePos** and **Gate3[*i*].Chan[*j*].UVW** (or **ABC** if the encoder inputs are used) in the IDE's Watch window so they can be continually monitored. The first element is the present commutation angle (in the range of 0 to 2048), and the second is the present value of the U, V, and W signal triplet (in the range of 1 to 6, with U having a value of 4, V of 2, and W of 1). Kill the motor with the on-line **#*x*k** command and turn the motor slowly manually until you find the position where the **UVW** value changes between 1 and 3. Note the value of **Motor[*x*].PhasePos** at this point, and put this value in **Motor[*x*].AbsPhasePosOffset**.

This test can also confirm whether the Hall sensor direction sense matches the commutation cycle direction sense or not. If the value of **Motor[*x*].PhasePos** increases as the **UVW** value changes from 1 to 3, the direction senses match, and **Motor[*x*].AbsPhasePosSf** should be set to (+) 170.667. If it decreases, the direction senses do not match, and **Motor[*x*].AbsPhasePosSf** should be set to -170.667.

Because the phase referencing with the Hall sensors is only approximate – it can have a +/-30^o error, a correction will need to be made subsequently. Most commonly, this correction is made at the motor home position, and this home position usually occurs at the index pulse of the encoder (typically the first index pulse inside the home switch). To set up this correction in this case, the value of **Motor[*x*].PhasePos** must be found at the index pulse. It is best to do a homing search on the index pulse with no home offset and read the value of **Motor[*x*].PhasePos** after the move has finished (it can be very difficult to find the index pulse manually). This value can be stored in saved setup element **Motor[*x*].AbsPhasePosForce** for future use. In the actual application, after the homing-search move is completed and the motor settled, the value saved in **Motor[*x*].AbsPhasePosForce** can be copied back into the active element **Motor[*x*].PhasePos** to provide the correction.

Remember to set **Motor[*x*].PhaseFindingTime** back to 0 before saving the other settings to non-volatile memory. Now, an on-line **#*x*$** command (or setting **Motor[*x*].PhaseFindingStep** to 1 in a program) will cause the Power PMAC to read the Hall sensors to establish the phase reference.

Use for Ongoing Servo Feedback

When used for ongoing servo position, the three Hall signals must be connected into the A+, B+, and and C+ signals on the encoder connector, with **Gate3[*i*].Chan[*j*].EncCtrl** set to 11 or 15 to specify a "times-6" (x6 – six counts per cycle) decode in either the "clockwise" or "counter-clockwise" direction sense.

To process the resulting position value for servo-loop use (feedback or master), a "Type 1" single-register-read conversion should be specified in the Encoder Conversion Table. Note that the IC has already performed the 1/T extension in hardware, so the "Type 3" "software 1/T extension" method should not be used.

EncTable[*n*].pEnc, which specifies the address of this register, should be set to **Gate3[*i*].ServoCapt.a**. (**EncTable[*n*].pEnc1**, which specifies a second source, is not used in this mode; its setting does not matter.)

Conversion parameters **EncTable[*n*].index1**, **.index2**, **.index3**, **.index4**, and **.MaxDelta** are generally all set to 0 to use this register.

To get the output of the conversion table in units of counts (most common), **EncTable[*n*].ScaleFactor** should be set to 1/256 to use the low 8 bits of the source register as fraction.

To use the result of the conversion table entry for outer (position) loop motor feedback, set **Motor[*x*].pEnc** to **EncTable[*n*].a**. To use the result for inner velocity loop motor feedback, set **Motor[*x*].pEnc2** to **EncTable[*n*].a**. To use the result as a master position for the motor's position-following function, set **Motor[*x*].pMasterEnc** to **EncTable[*n*].a**. To use the result for a time-base master frequency, set **Coord[*x*].pDesTimeBase** to **EncTable[*n*].DeltaPos.a**.

Use for Trigger Position

When the Hall sensors are used for ongoing servo feedback, it is possible (but rare) to latch the position on a trigger event in hardware into the "home capture" register for the channel. If this is done, the position is treated just as for a standard incremental encoder. Refer to the section on trigger position for incremental encoders, above, for details.

Monitoring for Sensor Loss

When Power PMAC reads Hall sensors for power-on phase position, it knows that only 6 of the 8 possible states are valid. If it detects one of the two invalid states, it considers the read to have failed, and will not set the **Motor[*x*].PhaseFound** status bit to 1. This prevents the motor from being enabled.

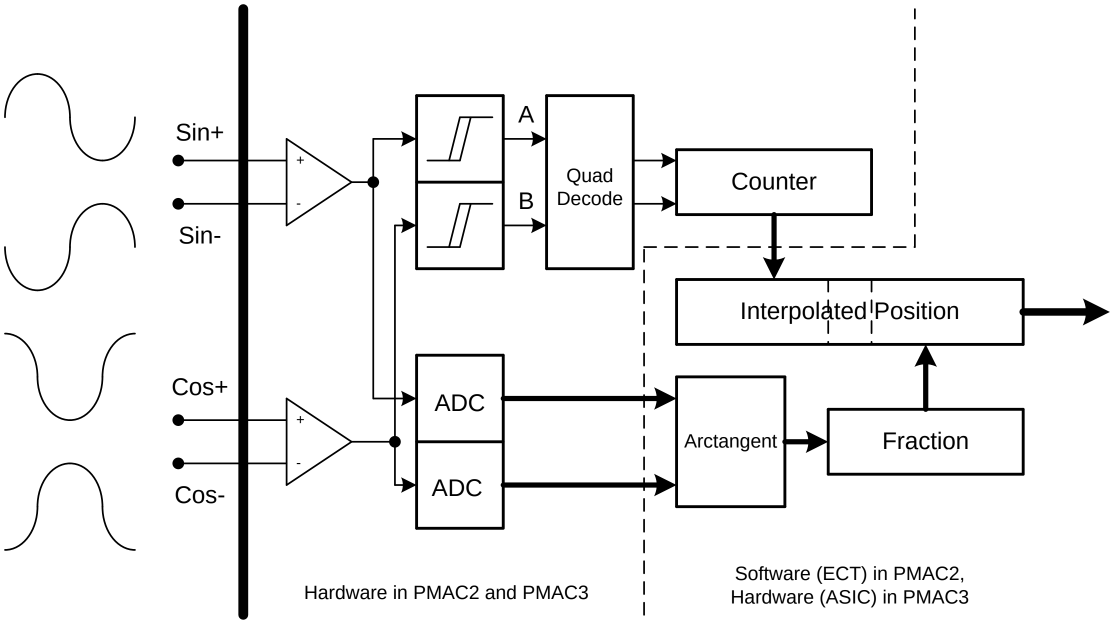
With the 120° spacing, the two invalid states are all signals high (111, or 7) and all signals low (000, or 0). These are the two must likely states for a hardware failure, as a cable disconnection lets the pull-up resistors on the ACC-24E3 bring all signals high, or a loss of power or a short at the sensors brings all signals low. For this reason, the 120° spacing is strongly recommended over the 60° spacing.

Power PMAC does not have automatic routines for checking for loss of 3-phase Hall sensors on an ongoing basis, but it is possible for the user to write such a routine in user application code, such as a background PLC program.

Sinusoidal Encoders

The ACC-24E3 with its analog feedback mezzanine board provides a high-resolution interpolation capability for sinusoidal encoders that does not require intensive calculations by the processor.

The following diagram shows the principle of interpolation that is used in Power PMAC systems. The sine and cosine waveforms are "squared up" into digital quadrature that is sampled and decoded at high frequencies to keep track of the line count. They are also fed into analog-to-digital converters that are sampled every phase and servo cycle. The ASIC automatically computes the arctangent of these values to determine where within a given line the position lies, and combines the whole-line and fractional-line values into a single 32-bit register. By doing these calculations in the ASIC (and simultaneously for all channels), the processor only needs to do a single read of a hardware register, and none of the mathematical manipulation.



Sinusoidal Encoder Interpolation Principle

IC Hardware Setup

Gate3[*i*].Chan[*j*].EncCtrl determines how the IC will decode the digital quadrature signals that are derived from the sinusoidal encoder signals connected to the SIN and COS inputs of the analog feedback mezzanine board. Settings of 3 and 7 specify a "times-4" (x4 – four counts per encoder line) decode in either the "clockwise" or "counter-clockwise" direction sense. Only these two settings may be used for sinusoidal interpolation. These decoded signals are used to determine the "full-count" data from the signal.

When used for servo-loop feedback, the direction sense of the feedback must match the direction sense of the servo-loop output, or a dangerous runaway condition may result when the loop is closed. That is, a positive servo-loop output must cause the position feedback to increment in the positive direction. *Changing the direction sense of the decode for the feedback encoder of a motor that is operating properly will result in unstable positive feedback and a dangerous runaway condition in the absence of other changes.*

Gate3[*i*].Chan[*j*].AtanEna must be set to 1 to enable the IC to use the arctangent calculations performed on the values read by the two "encoder ADCs" on the analog feedback mezzanine board as fractional-count data. In this mode, the low 12 bits of the 32-bit position value latched into the channel registers **Gate3[*i*].Chan[*j*].ServoCapt** (on the servo-clock interrupt) and **Gate3[*i*].Chan[*j*].PhaseCapt** (on the phase-clock interrupt) will be fractional-count data, yielding a resolution of 1/4096 of a quadrature count for these registers. This results in a 16,384-times interpolation per line of the sinusoidal encoder.

Before the IC computes the arctangent of the sine and cosine input values, it adds the value of saved setup element **Gate3[*i*].Chan[*j*].AdcOffset[0]** to the sine value in input register **Gate3[*i*].Chan[*j*].AdcEnc[0]**, and **Gate3[*i*].Chan[*j*].AdcOffset[1]** to the cosine value in input register **Gate3[*i*].Chan[*j*].AdcEnc[1]**. Non-zero values in these offset registers can compensate for biases in the encoder signals or analog receiving circuits.

Timer-based fractional-count estimation can still be used for the capture and compare functions with a sinusoidal encoder by setting **Gate3[*i*].Chan[*j*].TimerMode** to its default value of 0.

Use for Ongoing Commutation Feedback

If sinusoidal encoder feedback is used for commutation position angle, **Motor[*x*].pPhaseEnc** should be set to **Gate3[*i*].Chan[*j*].PhaseCapt.a** to use the count value (with fractional extension). The user must calculate how many LSBs of this 32-bit register there are per commutation cycle in order to set the commutation scale factor **Motor[*x*].PhasePosSf**, which multiplies the value in the source register to obtain the commutation angle.

With a sinusoidal encoder, there are 12 bits of fractional count data, so there are 4096 LSBs per quadrature count, or 16,384 LSBs per line of the encoder. If there were 1024 encoder lines per commutation cycle, there would be 16,777,216 LSBs per commutation cycle. In this case, **Motor[*x*].PhasePosSf** would be set to 2048 divided by 16,777,216. (It is best to enter this value as an expression and let Power PMAC calculate the double-precision floating-point value itself).

As an incremental sensor, a sinusoidal encoder is not suitable for establishing the absolute phase position angle at power-on. Either a separate sensor that is absolute over a commutation cycle (such as a Hall commutation sensor) is required, or a phasing-search move will be required after power-on.

Use for Ongoing Servo Feedback or Master

To process the resulting position value for servo-loop use (feedback or master), a "Type 1" single-register-read conversion should be specified in the encoder conversion table. Note that the IC has already performed the arctangent extension in hardware, so the "Type 5" "software arctangent extension" method should not be used.

EncTable[*n*].pEnc, which specifies the address of this register, should be set to **Gate3[*i*].ServoCapt.a**. (**EncTable[*n*].pEnc1**, which specifies a second source, is not used in this mode; its setting does not matter.)

Some users will set up **EncTable[*n*].index1** and **EncTable[*n*].index2** to create a low-pass "tracking filter" to minimize the effect of high-frequency noise on the analog inputs. Most users will set **EncTable[*n*].MaxDelta** to 0 to disable any derivative limits.

To get the output of the conversion table in units of quadrature counts (most common), **EncTable[*n*].ScaleFactor** should be set to 1/4096.

To use the result of the conversion table entry for outer (position) loop motor feedback, set **Motor[*x*].pEnc** to **EncTable[*n*].a**. To use the result for inner velocity loop motor feedback, set **Motor[*x*].pEnc2** to **EncTable[*n*].a**. To use the result as a master position for the motor's position-following function, set **Motor[*x*].pMasterEnc** to **EncTable[*n*].a**. To use the result for a time-base master frequency, set **Coord[*x*].pDesTimeBase** to **EncTable[*n*].DeltaPos.a**.

If the servo loop for the motor is closed in the phase interrupt (typically used only for one or two high-bandwidth actuators), a feature enabled by setting **Motor[*x*].PhaseCtrl** bit 3 (value 8) to 1, then the encoder conversion table, **Motor[*x*].pEnc**, and **Motor[*x*].pEnc2** are not used for this motor. Instead, Power PMAC directly reads the register whose address is contained in **Motor[*x*].pPhaseEnc** for both inner (velocity) and outer (position) loop servo feedback (regardless of whether Power PMAC is doing phase commutation for the motor or not).

To use the channel's encoder feedback for this purpose, **Motor[*x*].pPhaseEnc** should be set to **Gate3[*i*].Chan[*j*].PhaseCapt.a**. If **Gate3[*i*].Chan[*j*].AtanEna** is set to 1, whole counts are in the high 20 bits of this 32-bit register, so each quadrature count of the encoder is equivalent to 4096 (2^8) LSBs of the register. The low 12 bits will contain arctangent-based fractional count estimation. Therefore, 1 LSB of the register is 1/4096 of a quadrature count, or 1/16,384 of an encoder line. Unlike with the encoder conversion table, there is no capability to shift or scale this value into different units.

As an incremental sensor, a sinusoidal encoder is not suitable for establishing the absolute servo position at power-on. Either a separate sensor that is absolute over the entire travel of the motor is required, or a homing-search move will be required after power-on.

Use for Trigger Position

To use the IC channel's hardware-captured position value as the "trigger position" for motor triggered moves such as homing-search moves **Motor[*x*].CaptureMode** must be set to 0 to select hardware trigger and hardware capture. Also, **Motor[*x*].pCaptPos** must be set to **Gate3[*i*].Chan[*j*].FlagCapt.a**, the address of the register that latches the present encoder position immediately on the selected input trigger condition. This latches a 32-bit value with whole counts in the high 24 bits. If **Gate3[*i*].Chan[*j*].TimerMode.a** is set to the default value of 0, the low 8 bits will contain a timer-based "1/T" fractional count estimation at the instant of the trigger; otherwise the low 8 bits will contain all zeros.

This data must then be processed to match the scaling and offset of the servo feedback position. With sinusoidal feedback, most users will want to both the whole-count and fractional-count portions of the captured data. To do this, **Motor[*x*].CaptPosRightShift** should be set to 0 so none of the fractional-count data is eliminated. **Motor[*x*].CaptPosLeftShift** should be set to 4 to match the 12 bits of fractional-count resolution of the DSPGATE3's arctangent extension uses for servo feedback. **Motor[*x*].CaptPosRound** should be set to 0 to disable the half-count offset of the captured position, because both the arctangent extension used in the servo and the 1/T extension used in the capture have the same half-count offset.

These settings are made automatically when the Power PMAC is re-initialized by the `$$$***` command with an ACC-24E3 and an analog-feedback mezzanine board present. They are also made (with the address based on **Motor[*x*].pEncStatus**) when **Motor[*x*].EncType** is assigned a value of 6 (arctangent extension in a PMAC3-style IC) in the Script environment.

Some users will want to eliminate the fractional-count portion of the captured data for the trigger position, particularly for the homing-search move, so the reference position is at a zero-crossing point of the sine or cosine signal. To do this, **Motor[*x*].CaptPosRightShift** should be set to 8 to shift out the low 8 bits of fractional count from the 32-bit read. **Motor[*x*].CaptPosLeftShift** should be set to 12 to match the 12 bits of fractional-count resolution that the DSPGATE3's arctangent extension used for servo feedback (yielding 0's in the fractional portion). **Motor[*x*].CaptPosRound** should be set to 1 to enable the half-count offset of the captured position that will match the half-count offset performed on extended servo feedback.

Monitoring for Sensor Loss

The ACC-24E3 has circuitry that can detect loss of signal from sinusoidal encoder inputs, and standard Power PMAC algorithms can use this circuitry to automatically disable motors when this loss is detected. The detection circuitry uses the digital "sine" and "cosine" values from the analog-to-digital converters, and computes the sum of squares of these values, which should be roughly constant in an application, and above a minimum threshold for a properly working and connected encoder.

The value of this signal magnitude measurement can be found in the 16-bit element **Gate3[*i*].Chan[*j*].SumOfSquares**. If all 4 of the most significant bits of this element are 0, meaning that the magnitude of the signal is less than 1/16 of the maximum, the status bit **Gate3[*i*].Chan[*j*].SosError**, part of the full-word element **Gate3[*i*].Chan[*j*].Status**, is set to 1.

To employ this circuitry for automatic loss detection, set **Motor[*x*].pEncLoss** to **Gate3[*i*].Chan[*j*].Status.a**. (It can also be set to **Gate3[*i*].Chan[*j*].SosError.a**, but will report back as **Gate3[*i*].Chan[*j*].Status.a**). Set **Motor[*x*].EncLossBit** to 31 to specify the **SosError** bit number in the register. Set **Motor[*x*].EncLossLevel** to 1 to specify a high-true loss status bit.

Because it is possible for signal transients to cause a momentary setting of this, it is usually desirable to set **Motor[*x*].EncLossLimit** to a value greater than 0. The accumulated count of loss detection events must exceed this threshold before a fault is declared. Reasonable values for this element will not unduly delay reaction to true faults.

Serial Encoders

The ACC-24E3 provides direct interface to multiple popular serial encoder protocols through either its digital or analog feedback mezzanine boards. The configuration of the common hardware interface for a particular protocol and encoder is accomplished through software setup.

IC Hardware Setup

The DSPGATE3 IC has one multi-channel saved setup element for serial encoder configuration, and two channel-specific saved setup elements.

Multi-Channel Configuration: Gate3[i].SerialEncCtrl

The serial encoder interface is set up first through the use of multi-channel saved setup element **Gate3[*i*].SerialEncCtrl**, which establishes the protocol, clock frequency, and triggering used for serial encoders on all channels of the IC. This 32-bit element, reported as 8 hexadecimal digits, has multiple components for defining the interface to the encoders. Note that the components cannot be accessed as separate elements.

The full 32-bit element is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SerialClockMDiv								*SerialClockNDiv*				--	--	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

The first two components of the element specify the serial-encoder clock frequency. Depending on the protocol, this frequency can represent the serial bit rate directly, or a higher internal "oversampling" frequency, particularly for those protocols without an explicit clock signal to the encoder.

The high 8 bits, represented by the first two hex digits, form the component ***SerialClockMDiv*** specifying the initial division factor *M* for the serial clock frequency. The internal 100 MHz clock signal is divided by *M*+1 to obtain an intermediate clock frequency for the encoder. This is the first of two division stages to create the resulting serial clock frequency. Because this division factor can go up to 256, frequencies down to about 400 kHz can be supported directly in this first stage without the need for further division.

The next 4 bits, represented by the third hex digit, form the component ***SerialClockNDiv*** specifying the second-stage division factor *N* for the serial clock frequency. The intermediate clock frequency obtained in the first stage is divided by 2^N to obtain the final serial clock frequency. Generally this component is set to 0 (for divide by 1 at this stage) except for protocols with very low frequency such as Hiperface that are generally only used for power-up position.

For example, to create a 5.0 MHz serial encoder clock frequency, a division by 20 from the 100 MHz internal clock is necessary. The initial division factor *M* should be set to be 19 (13 hex) and the second division factor *N* should be set to 0. The first three hex digits of the element should therefore be set to $130.

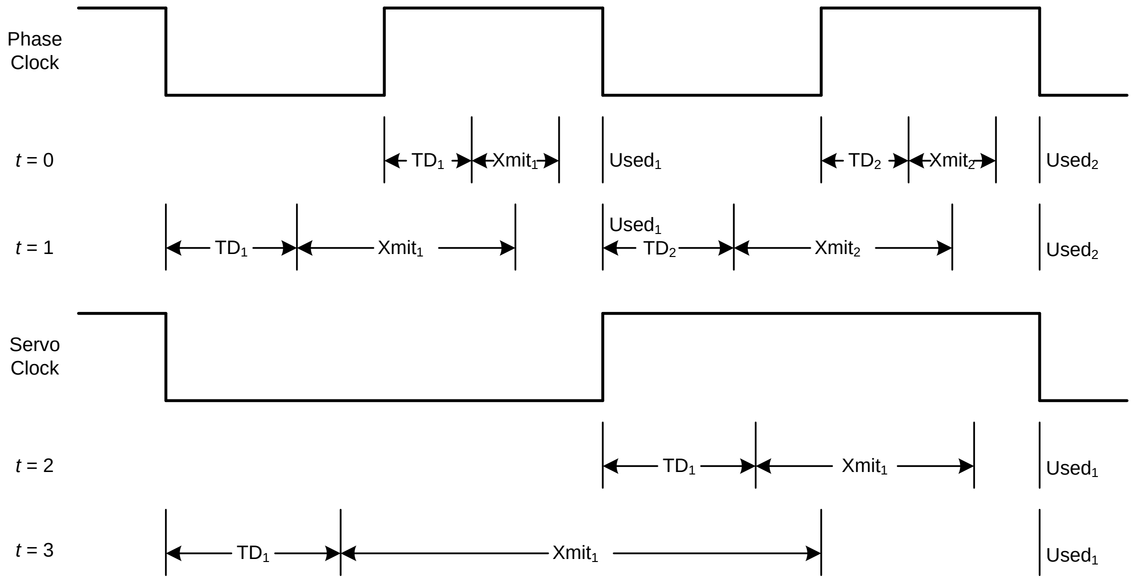
Bits of the fourth hex digit specify which edge (rising or falling) of which clock signal (servo or phase clock) causes the triggering of an encoder read operation. The "2's" bit of this digit is component ***SerialTriggerClockSel*** (***TC***), set to 0 to trigger on the phase clock, and to 1 to trigger on the servo clock. The "1's" bit is component ***SerialTriggerEdgeSel*** (***TE***), set to 0 to trigger on the rising edge of the selected clock, and to 1 to trigger on the falling edge. The “4’s” bit and the “8’s” bit of this digit are not used. This digit can therefore take four possible values:

- 0: Rising edge of phase clock (most common)
- 1: Falling edge of phase clock
- 2: Rising edge of servo clock
- 3: Falling edge of servo clock

If the position data is required for the commutation phase position, the phase clock should always be used for the trigger. If the position data is only required for servo position, the higher-frequency phase clock should still be used if possible to minimize the delay between trigger and use.

Power PMAC needs to be ready to read the resulting data on the falling edge of the phase and/or servo clock signals, which interrupt the processor to perform the appropriate calculations. Generally, the minimum possible delay from triggering to this point is desired to optimize performance. A value of 0 for this digit should be used if the data will be available in time.

The following diagram shows the signal timing for these four options. The “*t*” in this diagram is the fourth hex digit of **Gate3[*i*].SerialEncCtrl**.



t: Trigger select code – clock and edge

TD$_n$: Trigger delay from edge for cycle *n*

Xmit$_n$: Data transmission for cycle *n*

Used$_n$: Data used by software for cycle *n*

Serial Encoder Interface Timing Diagram

The next 8 bits, represented by the fifth and sixth hex digits, form the component ***SerialTrigDelay*** and specify the delay from the selected clock edge to the triggering of the encoder, expressed in serial clock signal cycles (whose frequency is set by the first three digits). This delay is indicated as “TD” in the above timing diagram. Most users will leave this at the default value of 0 (no delay), although some will want to increase this to minimize the delay between triggering and use of the resulting data.

The last 8 bits, represented by the seventh and eighth hex digits, form the component ***SerialProtocol***, and specify which encoder protocol is to be used. It presently can take the following values:

- 0: No serial protocol (quadrature)
- 1: SPI
- 2: SSI
- 3: Heidenhain EnDat2.1/2.2
- 4: Stegmann Hiperface
- 5: Yaskawa Sigma I
- 6: Yaskawa Sigma II/III
- 7: Tamagawa
- 8: Panasonic
- 9: Mitutoyo
- 10: Kawasaki

For example, to specify the SSI protocol with a 4 MHz clock, triggered on the rising edge of the phase clock after a 64 clock-cycle (16µsec) delay, **Gate3[*i*].SerialEncCtrl** should be set to $18004002. This sets the *M* divisor factor to 24 (18 hex) and the *N* divisor factor to 0 to divide the internal 100 MHz clock by 25 to obtain the external 4 MHz clock signal, ***SerialTrigDelay*** to 64 (40 hex) and selects protocol 2 (02 hex).

1				8				0				0				4				0				0				2			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	0	0	0	0	0	-	-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
SerialClockMDiv								*SerialClockNDiv*				*--*	*--*	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Control Word: Gate3[i].Chan[j].SerialEncCmd

Each channel has a "control word" for its own serial encoder interface, implemented in the 32-bit element **Gate3[*i*].Chan[*j*].SerialEncCmd**. This 32-bit element, reported as 8 hexadecimal digits, has multiple components for defining the interface to the encoders. Note that the components cannot be accessed as separate elements.

The full 32-bit element is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SerialEncCmdWord																*Parity*		*TM*	*TE*	*GB*	*Rdy*	*StatusBits*				*NumBits*					

The high 16 bits (bits 31 – 16), represented by the first four hex digits, form the component ***SerialEncCmdWord*** specifying the serial command word that is to be sent out to the encoder in some protocols.

The next 2 bits (bits 15 – 14), represented in the fifth hex digit, form the component ***SerialEncParity***, which specifies the type of parity checking to be performed in those protocols that support parity checking. A value of 0 in these two bits specifies no parity; a value of 1 (adding 4 to the value of the digit) specifies odd parity; a value of 2 (adding 8 to the value of the digit) specifies even parity. (A value of 3 is reserved for future use.)

The next bit (bit 13), also part of the fifth hex digit, is the component ***SerialEncTrigMode***. If this bit is 0, continuous sampling (every phase or servo clock cycle) is specified. If it is 1 (adding 2 to the value of the digit), "one-shot" sampling is specified – this is typically selected for power-on position in the slower protocols.

The next bit (bit 12), also part of the fifth hex digit, is the component ***SerialEncTrigEna***. If this bit is 0, no sampling is performed. If it is 1 (adding 1 to the value of the digit), sampling is enabled. Note that if ***SerialEncTrigMode*** is set to 1 for "one-shot" sampling, the act of sampling automatically clears this bit, and this bit must be "manually" set back to 1 to obtain another sample.

The next bit (bit 11), part of the sixth hex digit, is the component ***SerialEncGtoB***. If this bit is 0, the serial position data received in the SSI protocol is assumed to be in numerical binary format, and no format conversion is performed. If this bit is 1 (adding 8 to the value of the digit), the SSI data received is assumed to be in Gray-code format, and it is automatically converted to numerical binary format for use by the software.

The next bit (bit 10), also part of the sixth hex digit, is the status component ***SerialEncDataReady***. This read-only bit is automatically set to 0 while communication is active with the encoder, and automatically set to 1 (adding 4 to the value of the digit) when the communication is complete and a position value is ready to be read by the processor.

The next 4 bits (bits 09 – 06), parts of the sixth and seventh hex digits, form the component ***SerialEncStatusBits***. This component specifies the number of status bits returned from the encoder in the SPI protocol. The valid range is 0 to 12.

The last 6 bits (bits 05 – 00), parts of the seventh and eighth hex digits, form the component ***SerialEncNumBits***. This component specifies the number of position data bits to be received from the encoder in the SPI, SSI, and EnDat protocols.

Single-Channel Enable Control: Gate3[i].Chan[j].SerialEncEna

The single-bit saved setup element **Gate3[*i*].Chan[*j*].SerialEncEna** (part of the full-word element **Gate3[*i*].Chan[*j*].InCtrl**) determines whether the serial encoder interface buffers for the channel on the ACC-24E3 are enabled or not. If it is set to the default value of 0, the buffers are not enabled, and the shared pins on the connectors can be used for their alternate purposes – the T, U, V, and W input flags, and the differential index channel inputs.

If **Gate3[*i*].Chan[*j*].SerialEncEna** is set to 1, the serial encoder interface buffers are enabled, so these pins can be used to connect to serial encoders. In this case, these pins cannot be used for their alternate purpose.

Note: With the digital feedback mezzanine boards, starting with revision 108 (due for release at the beginning of 2012), and with the analog feedback mezzanine boards, starting with revision 107 (due for release at the beginning of 2012), jumpers have been added to permit the use of the index pulse inputs C+ and C- instead of the SENCENA+ and SENCENA- outputs when **Gate3[*i*].Chan[*j*].SerialEncEna** is set to 1. If jumper E5 (for the lower-numbered channel on the board) or jumper E6 (for the higher-numbered channel) connects pins 2 and 3, the index pulse inputs will be used.

Supplemental Quadrature Encoder Implementation

When the protocol code in **Gate3[*i*].SerialEncCtrl** is set to 0, none of the serial encoder protocols is enabled, and the serial encoder interface circuitry for the channel can be used to connect to a digital quadrature encoder. Note that this interface is independent of the dedicated incremental encoder interface for the channel, permitting the use of two quadrature encoders on the same channel.

Connections

The quadrature interface is a 4-wire interface, using 2 differential signal pairs, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	A+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	B+/-

Note that there is no index channel in this quadrature-encoder interface.

Multi-Channel Setup: Gate3[i].SerialEncCtrl

To select this supplemental quadrature interface on all channels of the ACC-24E3, simply set **Gate3[*i*].SerialEncCtrl** to its default value of $0.

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

For this supplemental quadrature interface, the value of **Gate3[*i*].Chan[*j*].SerialEncCmd** does not matter. It can be left at its default value of $0.

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel to accept the quadrature signals, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

For this supplemental quadrature interface, **Gate3[*i*].Chan[*j*].SerialEncDataA** contains the encoder count value latched on the most recent servo interrupt, with the high 24 bits containing whole-count data referenced to the power-on/reset position, and the low 8 bits containing timer-based fractional-count estimation data. Its format is equivalent to that of **Gate3[*i*].Chan[*j*].ServoCapt** in its default mode. **Gate3[*i*].Chan[*j*].SerialEncDataA** is not used in this mode.

SPI Protocol Implementation

The SPI (Serial Peripheral Interface) is a popular serial data interface for a variety of devices, including DACs and ADCs. There are several serial encoders that employ this protocol.

Connections

The SPI interface is a 6-wire interface, using all 3 differential signal pairs, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	SDO+/-
SENCENA+/-	to:	CSn+/-
SENCCLK+/-	to:	SCK+/-

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for an SPI encoder.

SerialClockMDiv:	= (100 / f_{bit}) – 1	// Serial clock frequency = bit transmission frequency
SerialClockNDiv:	= 0	// No further division unless *f* < 400 kHz
SerialTrigClockSel:	= 0	// Use phase clock if possible
SerialTrigEdgeSel:	= 0	// Use rising clock edge if possible
SerialTrigDelay:	= 0	// Can increase from 0 if possible to reduce latency
SerialProtocol:	= $01	// Selects SPI protocol

For example, for a 4 MHz bit transmission rate, ***SerialClockMDiv*** = (100 / 4) **-** 1 = 24 ($18) and **Gate3[*i*].SerialEncCtrl** is set to $18000001 for triggering on the rising edge of phase clock without delay.

1				8				0				0				0				0				0				1			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SerialClockMDiv								*SerialClockNDiv*				--	--	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for an SPI encoder.

SerialEncCmdWord:	= 0	// No command word supported for SPI protocol
SerialEncParity:	= 0	// No parity check supported for SPI protocol
SerialEncTrigMode:	= 0	// Continuous triggering
SerialEncTrigEna:	= 1	// Enable triggering
SerialEncGtoB:	= 0	// No Gray-to-binary conversion supported
SerialEncDataReady:	= 0	// Read-only status bit
SerialEncStatusBits:	= ??	// Encoder-specific number of status bits returned
SerialEncNumBits:	= ??	// Encoder-specific number of position bits returned

For example, for an SPI encoder with 28 position bits and 4 status bits, **Gate3[*i*].Chan[*j*].SerialEncCmd** would be set to $0000111C. (It may report back as $0000151C if the data-ready status bit is set.)

0				0				0				0				1				1				1				C			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	-	-	0	1	0	0	0	1	1	1	0	0
SerialEncCmdWord																*Parity*		*TM*	*TE*	*GB*	*Rdy*	*StatusBits*				*NumBits*					

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

The DSPGATE3 IC can support up to 32 bits of position data and 12 bits of status data from an SPI encoder, as specified by the channel command word. There is no general specification as to how many of the position bits are “single-turn” data and how many (if any) are “multi-turn”. There is no general specification as to what particular status bits mean.

For an SPI encoder, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Single/Multi-Turn Position

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F 11	F 10	F 9	F 8	F 7	F 6	F 5	F 4	F 3	F 2	F 1	F 0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Status Field

Bits P*n* represent the bits of single-turn and multi-turn position. Bits F*n* represent bits of the status field.

SSI Protocol Implementation

The SSI (Synchronous Serial Interface) is a popular serial data interface for a variety of devices. Many serial encoders use the SSI interface, with several variations.

Connections

The SSI interface is a 4-wire interface, using 2 differential signal pairs, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	DATA+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	CLOCK+/-

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for an SSI encoder.

SerialClockMDiv:	= (100 / f_{bit}) - 1	// Serial clock frequency = bit transmission frequency
SerialClockNDiv:	= 0	// No further division unless *f* < 400 kHz
SerialTrigClockSel:	= 0	// Use phase clock if possible
SerialTrigEdgeSel:	= 0	// Use rising clock edge if possible
SerialTrigDelay:	= 0	// Can increase from 0 if possible to reduce latency
SerialProtocol:	= $02	// Selects SSI protocol

For example, for a 2.5 MHz bit transmission rate, ***SerialClockMDiv*** = (100 / 2.5) - 1 = 39 ($23) and **Gate3[*i*].SerialEncCtrl** is set to $23000002 for triggering on the rising edge of phase clock without delay.

2				3				0				0				0				0				0				2			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	1	1	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
SerialClockMDiv								*SerialClockNDiv*				--	--	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for an SSI encoder.

SerialEncCmdWord:	= 0	// No command word supported for SSI protocol
SerialEncParity:	= ??	// Encoder-specific parity check
SerialEncTrigMode:	= 0	// Continuous triggering
SerialEncTrigEna:	= 1	// Enable triggering
SerialEncGtoB:	= ??	// Encoder-specific data format
SerialEncDataReady:	= 0	// Read-only status bit
SerialEncStatusBits:	= 0	// No status bits supported for SSI protocol
SerialEncNumBits:	= ??	// Encoder-specific number of position bits returned

For example, for an SSI encoder with 25 position bits in Gray-code format with odd parity, **Gate3[*i*].Chan[*j*].SerialEncCmd** would be set to $00005819. (It may report back as $0005C19 if the data-ready status bit is set.)

0				0				0				0				5				8				1				9			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0	1	1	-	-	-	-	-	0	1	1	0	0	1
SerialEncCmdWord																*Parity*		*TM*	*TE*	*GB*	*Rdy*	*Status*				*NumBits*					

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

The DSPGATE3 IC can support up to 32 bits of position data and a parity error bit from an SSI encoder, as specified by the channel command word. There is no specification as to how many of the position bits are "single-turn" data and how many (if any) are "multi-turn".

For an SSI encoder, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P 31	P 30	P 29	P 28	P 27	P 26	P 25	P 24	P 23	P 22	P 21	P 20	P 19	P 18	P 17	P 16	P 15	P 14	P 13	P 12	P 11	P 10	P 9	P 8	P 7	P 6	P 5	P 4	P 3	P 2	P 1	P 0
Single/Multi-Turn Position																															

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E 0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PE																															

Bits P*n* represent the bits of single-turn and multi-turn position. Bit E0 represents the parity error bit.

EnDat2.1/2.2 Protocol Implementation

EnDat is a serial encoder protocol developed by Heidenhain. Encoders with the EnDat protocol are available from multiple vendors, including Heidenhain. The older EnDat2.1 protocol is typically used only for power-on position. The newer EnDat2.2 protocol is typically used for ongoing position as well.

Connections

The EnDat interface is a 4-wire interface, using 2 differential signal pairs, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	DATA+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	CLOCK+/-

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for an EnDat encoder. The serial clock frequency is set 25 times higher than the external clock frequency, which is the bit transmission frequency, to permit the implementation of delay compensation, which allows high-frequency transmission over long distances.

SerialClockMDiv:	= $(4 / f_{bit}) - 1$	// Serial clock freq. = 25x bit transmission freq.
SerialClockNDiv:	= 0	// No further division
SerialTrigClockSel:	= 0	// Use phase clock if possible
SerialTrigEdgeSel:	= 0	// Use rising clock edge if possible
SerialTrigDelay:	= 0	// Can increase from 0 if possible to reduce latency
SerialProtocol:	= $03	// Selects EnDat protocol

For example, for a 2.0 MHz bit transmission rate, ***SerialClockMDiv*** = (4 / 2) - 1 = 1 ($01) and **Gate3[*i*].SerialEncCtrl** is set to $01000003 for triggering on the rising edge of phase clock without delay.

0				1				0				0				0				0				0				3			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
SerialClockMDiv								*SerialClockNDiv*				--	--	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The DSPGATE3 IC EnDat interface supports two 6-bit command codes to the encoder: 000111 ($07) for reporting position, and 101010 ($2A) for resetting the encoder. These 6 bits fit at the low end of the 16-bit ***SerialEncCmdWord*** command field of **Gate3[*i*].Chan[*j*].SerialEncCmd**.

The most significant bit of ***SerialEncCmdWord***, which is bit 31 of the full-word element, is the *StartDelayComp* control bit. Setting this bit to 1 starts a delay identification and compensation cycle that measures the propagation delay between the encoder and the controller. The delay is measured three times, and the average of these delay values is used in the compensation. When these calculations are done, the *StartDelayComp* bit is automatically cleared. This delay identification operation must be performed after every power-up cycle. Delay compensation permits high bit transmission rates over very long cables.

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for position reporting from an EnDat encoder.

SerialEncCmdWord:	= 7	// Command word for encoder to report position
SerialEncParity:	= 0	// No parity check supported for EnDat protocol
SerialEncTrigMode:	= 0	// Continuous triggering (EnDat2.2)
SerialEncTrigEna:	= 1	// Enable triggering

SerialEncGtoB: = 0 // No Gray code supported for EnDat protocol
SerialEncDataReady: = 0 // Read-only status bit
SerialEncStatusBits: = 0 // No status bits supported for EnDat protocol
SerialEncNumBits: = ?? // Encoder-specific number of position bits returned

For example, for an EnDat2.2 encoder with 37 position bits, **Gate3[*i*].Chan[*j*].SerialEncCmd** would be set to $00071025 for continuous position reporting. (It may report back as $00071425 if the data-ready status bit is set.)

0				0				0				7				1				0				2				5			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	-	-	-	-	-	-	-	-	-	0	0	0	1	1	1	-	-	0	1	-	-	-	-	-	-	1	0	0	1	0	1
DC								*SerialEncCmdWord*								*Parity*		*TM*	*TE*	*GB*	*Rdy*		*Status*					*NumBits*			

To perform the delay identification and compensation cycle on this encoder, set **Gate3[*i*].Chan[*j*].SerialEncCmd** to $80071025, then wait for the MSB to clear.

8				0				0				7				1				0				2				5			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	-	-	-	-	-	-	-	-	-	0	0	0	1	1	1	-	-	0	1	-	-	-	-	-	-	1	0	0	1	0	1
DC								*SerialEncCmdWord*								*Parity*		*TM*	*TE*	*GB*	*Rdy*		*Status*					*NumBits*			

This same encoder can be reset with a command word value of 42 ($2A) sent in one-shot mode with **Gate3[*i*].Chan[*j*].SerialEncCmd** set to $002A3025.

0				0				2				A				3				0				2				5			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	-	-	-	-	-	-	-	-	-	1	0	1	0	1	0	-	-	1	1	-	-	-	-	-	-	1	0	0	1	0	1
								SerialEncCmdWord								*Parity*		*TM*	*TE*	*GB*	*Rdy*		*Status*					*NumBits*			

For an EnDat2.1 encoder with 34 position bits, **Gate3[*i*].Chan[*j*].SerialEncCmd** would be set to $00073018 for one-shot position reporting (at power-up).

0				0				0				7				3				0				1				8			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	-	-	-	-	-	-	-	-	-	0	0	0	1	1	1	-	-	1	1	-	-	-	-	-	-	0	1	1	0	0	0
DC								*SerialEncCmdWord*								*Parity*		*TM*	*TE*	*GB*	*Rdy*		*Status*					*NumBits*			

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

The DSPGATE3 IC can support up to 48 bits of position data from an EnDat encoder, as specified by the channel command word. There is no specification as to how many of the position bits are "single-turn" data and how many (if any) are "multi-turn".

For an EnDat encoder, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															Single/Multi-Turn Position																

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	E	E	-	-	-	-	-	-	-	-	-	-	-	-	-	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
2	1	0														47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
TE	*CE*	*EB*																													

Bits P*n* represent the bits of single-turn and multi-turn position. Bits E*n* represent the error bits (E0 is an error reported by the encoder, E1 is a CRC error detected by the IC, and E2 is a timeout error detected by the IC.

Hiperface Protocol Implementation

Hiperface is a serial encoder protocol from Stegmann that provides digital binary numeric absolute position at power-on over its "parameter" channel. Ongoing position from this encoder style is from analog sine/cosine signals.

Connections

The Hiperface parameter-channel interface is a 1-wire interface, using 1 differential signal pair, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	DATA+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	(no connect)

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for a Hiperface encoder. Because there is no explicit clock signal, the serial clock frequency is set 20 times higher than the bit transmission frequency to "oversample" the input data stream. For the default 9600 baud transmission of the Hiperface encoder, this clock frequency should be 192 kHz. This requires the two-stage division by both ***SerialClockMDiv*** and ***SerialClockNDiv***. It is recommended to divide down as much as possible in the first stage to get as close to the ideal frequency as possible

SerialClockMDiv:	= $82	// Divide by 130 to get ~768 kHz
SerialClockNDiv:	= 2	// Divide by 4 to get ~ 192 kHz
SerialTrigClockSel:	= 1	// Use servo clock
SerialTrigEdgeSel:	= 1	// Use falling clock edge
SerialTrigDelay:	= 0	// No delay
SerialProtocol:	= $04	// Selects Hiperface protocol

For example, for a 9600 bit/sec transmission rate, ***SerialClockMDiv*** = 130 ($82), ***SerialClockNDiv*** = 2 and **Gate3[*i*].SerialEncCtrl** is set to $82230004 for triggering on the falling edge of servo clock without delay. Since this is a "one-shot" read, the selection of the triggering clock edge does not matter much.

8				2				2				3				0				0				0				4			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	0	0	0	1	0	-	-	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
SerialClockMDiv								*SerialClockNDiv*				--	--	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The DSPGATE3 IC Hiperface interface supports three 8-bit command codes to the encoder: $42 for reporting position, $50 for reporting status, and $53 for resetting the encoder. These 8 bits fit at the low end of the 16-bit ***SerialEncCmdWord*** command field of **Gate3[*i*].Chan[*j*].SerialEncCmd**.

The high 8 bits of the ***SerialEncCmdWord*** contain the address of the encoder in the interface. The Hiperface protocol permits up to 8 separate encoders to be "daisy-chained" on a single multi-drop interface. While this can be done using an ACC-24E3, it is expected that each channel of the ACC-24E3 will be connected to a separate individual encoder, simplifying the wiring. In this configuration, this address field can either match the encoder's address value (+ $40), or it can be set to $FF (broadcast mode).

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for position reporting from an Hiperface encoder.

SerialEncCmdWord:	= $4042	// Encoder at address 0 to report position
SerialEncParity:	= ??	// Encoder-specific parity check
SerialEncTrigMode:	= 1	// One-shot triggering
SerialEncTrigEna:	= 1	// Enable triggering (cleared after one-shot)
SerialEncGtoB:	= 0	// No Gray code supported for Hiperface protocol
SerialEncDataReady:	= 0	// Read-only status bit
SerialEncStatusBits:	= 0	// No status bits supported for Hiperface protocol
SerialEncNumBits:	= ??	// Encoder-specific number of position bits returned

For example, for a Hiperface encoder at user address 0 with odd parity, **Gate3[*i*].Chan[*j*].SerialEncCmd** would be set to $40427000 for one-shot position reporting. (It may report back as $40426400 when the trigger-enable bit is cleared and the data-ready status bit is set.)

4				0				4				2				7				0				0				0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-
DC							*SerialEncCmdWord*									*Parity*		*TM*	*TE*	*GB*	*Rdy*		*Status*					*NumBits*			

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

The DSPGATE3 IC supports 32 bits of position data and 12 bits of status data from a Hiperface encoder

For a Hiperface encoder, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															Single/Multi-Turn Position																

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	E	E	E	C	C	C	C	C	C	C	C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	2	1	0	7	6	5	4	3	2	1	0																				
Error Bits				*Error Code*																											

Bits P*n* represent the bits of single-turn and multi-turn position. Bits C*n* represent bits of the encoder's error code (only reported on status request). Bits E*n* represent error bits (E0 is an error reported by the encoder, E1 is a parity error detected by the IC, E2 is a checksum error detected by the IC, and E3 is a timeout error).

Yaskawa Sigma I Protocol Implementation

The Yaskawa Sigma I absolute encoder protocol provides absolute turns count position data in ASCII text format on specific request. This request also causes the encoder to output synthesized quadrature to provide the absolute single-turn position (all without motion). After this, it continues to output synthesized quadrature to reflect the ongoing position from actual motion.

Connections

The Yaskawa Sigma I absolute position interface is a 3-wire interface, using 1 differential signal pair, and one single-ended signal, plus power and ground. It also uses the quadrature interface The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	PA0+/-
SENCENA-	to:	SEN
SENCENA+	to:	(no connect)
SENCCLK+/-	to:	(no connect)
CHA+/-	to:	PA0+/-
CHB+/-	to:	PB0+/-

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for a Yaskawa Sigma I absolute encoder. Because there is no explicit clock signal, the serial clock frequency is set 20 times higher than the bit transmission frequency to "oversample" the input data stream. For the default 9600 baud transmission of the Yaskawa Sigma I encoder, this clock frequency should be 192 kHz. This requires the two-stage division by both ***SerialClockMDiv*** and ***SerialClockNDiv***. It is recommended to divide down as much as possible in the first stage to get as close to the ideal frequency as possible

SerialClockMDiv:	= $82	// Divide by 130 to get ~768 kHz
SerialClockNDiv:	= 2	// Divide by 4 to get ~ 192 kHz
SerialTrigClockSel:	= 1	// Use servo clock
SerialTrigEdgeSel:	= 1	// Use falling clock edge
SerialTrigDelay:	= 0	// No delay
SerialProtocol:	= $05	// Selects Yaskawa Sigma I protocol

For example, for a 9600 bit/sec transmission rate, ***SerialClockMDiv*** = 130 ($82), ***SerialClockNDiv*** = 2 and **Gate3[*i*].SerialEncCtrl** is set to $82230005 for triggering on the falling edge of servo clock without delay. Since this is a "one-shot" read, the selection of the triggering clock edge does not matter much.

8				2				2				3				0				0				0				5			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	0	0	0	1	0	-	-	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
SerialClockMDiv								*SerialClockNDiv*				*--*	*--*	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for a Yaskawa Sigma I encoder.

SerialEncCmdWord:	= 1	// Set bit to strobe encoder
SerialEncParity:	= 0	// Fixed parity check for Yaskawa protocol
SerialEncTrigMode:	= 1	// One-shot triggering
SerialEncTrigEna:	= 1	// Enable triggering (cleared on one-shot completion)
SerialEncGtoB:	= 0	// No Gray code supported for Yaskawa protocol
SerialEncDataReady:	= 0	// Read-only status bit
SerialEncStatusBits:	= 0	// No status bits supported for Yaskawa protocol
SerialEncNumBits:	= 0	// Fixed number of position bits returned

Gate3[*i*].Chan[*j*].SerialEncCmd would be set to $00013000 for continuous position reporting. (It may report back as $00002400 if the trigger-enable bit is cleared at the end of the one-shot and the ready status bit is set.)

0				0				0				1				3				0				0				0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-
SerialEncCmdWord																*Parity*		*TM*	*TE*	*GB*	*Rdy*	*Status*				*NumBits*					

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

For an absolute Sigma I encoder, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	C 7	C 6	C 5	C 4	C 3	C 2	C 1	C 0	B 7	B 6	B 5	B 4	B 3	B 2	B 1	B 0	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0
																Multi-Turn Position															

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F 1	F 0	-	-	-	-	-	-	P 7	P 6	P 5	P 4	P 3	P 2	P 1	P 0	S 7	S 6	S 5	S 4	S 3	S 2	S 1	S 0	E 7	E 6	E 5	E 4	E 3	E 2	E 1	E 0
Error Bits								*"P"*								*Sign (+/-)*								*Multi-Turn Position*							

Bits A*n* represent the bits of the ASCII code for the "one's digit" of the turns count; bits B*n* represent bits of the "ten's digit"; bits C*n* represent bits of the "hundred's digit"; bits D*n* represent bits of the "thousand's digit"; bits E*n* represent bits of the "ten-thousand's digit"; bits S*n* represent bits of the ASCII code for the plus or minus sign; bits P*n* represent bits of the ASCII code for the letter “P”. Bits F*n* represent bits of the error field (F0 is a parity error; F1 is a timeout error).

For each of the five numeric ASCII digits, the numeric value of the digit can be obtained by subtracting 48 ($30) from the value of the ASCII code.

Yaskawa Sigma II/III/V Protocol Implementation

Yaskawa has three generations of serial encoder protocols sending numerical binary data that can be used for power-on and ongoing position data. These are the Sigma II, Sigma III, and Sigma V protocols. The differences between the protocols are minor, and the ACC-24E3 can support all three.

Connections

The Yaskawa II/III/V interface is a 2-wire interface, using 1 bi-directional differential signal pair, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	DATA+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	(no connect)

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for a Yaskawa II/III/V encoder.

SerialClockMDiv:	= 0	// 100 MHz serial clock freq. = 25x bit transmission freq.
SerialClockNDiv:	= 0	// No further division
SerialTrigClockSel:	= 0	// Use phase clock if possible
SerialTrigEdgeSel:	= 0	// Use rising clock edge if possible
SerialTrigDelay:	= 0	// Can increase from 0 if possible to reduce latency
SerialProtocol:	= $06	// Selects Yaskawa II/III/V protocol

For example, for the standard 4.0 MHz bit transmission rate, a 100 MHz serial clock frequency is used, and **Gate3[*i*].SerialEncCtrl** is set to $00000006 for triggering on the rising edge of phase clock without delay.

0				0				0				0				0				0				0				6			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
SerialClockMDiv								*SerialClockNDiv*				--	--	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for a Yaskawa Sigma II/III/V encoder.

SerialEncCmdWord:	= 0	// No command word for position reporting in Yaskawa
SerialEncParity:	= 0	// No parity check supported for Yaskawa protocol
SerialEncTrigMode:	= 0	// Continuous triggering
SerialEncTrigEna:	= 1	// Enable triggering
SerialEncGtoB:	= 0	// No Gray code supported for Yaskawa protocol
SerialEncDataReady:	= 0	// Read-only status bit

SerialEncStatusBits: = 0 // No status bits supported for Yaskawa protocol
SerialEncNumBits: = 0 // Fixed number of position bits returned

Gate3[*i*].Chan[*j*].SerialEncCmd would be set to $00001000 for continuous position reporting. (It may report back as $00001400 if the ready status bit is set.)

0				0				0				0				1				0				0				0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	-	-	-	-	-	-	-	-	-	-	-	-
SerialEncCmdWord																*Parity*		*TM*	*TE*	*GB*	*Rdy*	*Status*				*NumBits*					

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

For an absolute Sigma II encoder with 17 bits per revolution, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M 10	M 9	M 8	M 7	M 6	M 5	M 4	M 3	M 2	M 1	M 0	S 16	S 15	S 14	S 13	S 12	S 11	S 10	S 9	S 8	S 7	S 6	S 5	S 4	S 3	S 2	S 1	S 0	-	-	-	-
Multi-Turn Position											*Single-Turn Position*																				

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E 2	E 1	E 0	-	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0	-	-	-	-	-	-	-	M 15	M 14	M 13	M 12	M 11
Error Bits				*Alarm Code*								*Temperature*															*Multi-Turn Position*				

Bits S*n* represent the bits of single-turn position; bits M*n* represent the bits of multi-turn position ("turns count"). Overall position can be read "seamlessly" as a combination of single-turn and multi-turn position values. Bits T*n* represents the bits of the reported temperature (if provided); bits A*n* represent bits of the alarm code; and bits E*n* represent error bits. (E0 is coding error, E1 is CRC error, and E2 is timeout error.)

For an incremental Sigma II encoder with 17 bits per revolution, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C 13	C 12	C 11	C 10	C 9	C 8	C 7	C 6	-	S 16	S 15	S 14	S 13	S 12	S 11	S 10	S 9	S 8	S 7	S 6	S 5	S 4	S 3	S 2	S 1	S 0	-	-	U	V	W	Z
Compensation Position									*Single-Turn Position*																			*Hall/Index*			

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E 2	E 1	E 0	-	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	T 7	T 6	T 5	T 4	T 3	T 2	T 1	T 0	-	-	-	-	-	-	-	-	-	C 16	C 15	C 14
Error Bits				*Alarm Code*								*Temperature*																	*Compensation Pos*		

Bits S*n* represent the bits of single-turn position; bits M*n* represent the bits of multi-turn position ("turns count"). Overall position can be read "seamlessly" as a combination of single-turn and multi-turn position values. Bits T*n* represents the bits of the reported temperature (if provided); bits A*n* represent bits of the alarm code; and bits E*n* represent error bits. (E0 is coding error, E1 is CRC error, and E2 is timeout error.)

For an absolute Sigma III encoder with 20 bits per revolution, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	M	M	M	M	M	M	M	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	-	-	-	-
7	6	5	4	3	2	1	0	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Multi-Turn Position								*Single-Turn Position*																							

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	E	E	-	A	A	A	A	A	A	A	A	T	T	T	T	T	T	T	T	-	-	-	-	M	M	M	M	M	M	M	M
2	1	0		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0					15	14	13	12	11	10	9	8
Error Bits				*Alarm Code*								*Temperature*												*Multi-Turn Position*							

Bits S*n* represent the bits of single-turn position; bits M*n* represent the bits of multi-turn position ("turns count"). Overall position can be read "seamlessly" as a combination of single-turn and multi-turn position values. Bits T*n* represents the bits of the reported temperature (if provided); bits A*n* represent bits of the alarm code; and bits E*n* represent error bits. (E0 is coding error, E1 is CRC error, and E2 is timeout error.)

Tamagawa FA-Coder Protocol Implementation

Tamagawa produces serial encoders with the "FA-Coder" protocol sending numerical binary data that can be used for power-on and ongoing position data.

Connections

The Tamagawa FA-Coder interface is a 2-wire interface, using 1 bi-directional differential signal pair, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	DATA+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	(no connect)

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for a Tamagawa FA-Coder encoder. Because there is no explicit clock signal, the serial encoder clock in the DSPGATE3 must be 20 times the bit transmission frequency to "oversample" the input data stream.

SerialClockMDiv:	= (5 / f_{bit}) - 1	// Serial clock freq. = 20x bit transmission freq.
SerialClockNDiv:	= 0	// No further division
SerialTrigClockSel:	= 0	// Use phase clock if possible
SerialTrigEdgeSel:	= 0	// Use rising clock edge if possible
SerialTrigDelay:	= 0	// Can increase from 0 if possible to reduce latency
SerialProtocol:	= $07	// Selects Tamagawa FA-Coder protocol

For example, for a 2.5 MHz bit transmission rate, a 50 MHz serial clock frequency is used, and **Gate3[*i*].SerialEncCtrl** is set to $01000007 for triggering on the rising edge of phase clock without delay.

0				1				0				0				0				0				0				7			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
SerialClockMDiv								*SerialClockNDiv*				--	--	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for a Tamagawa FA-Coder encoder.

SerialEncCmdWord:	= $1A	// Command word for position reporting in Tamagawa
SerialEncParity:	= 0	// No parity check supported for Tamagawa protocol
SerialEncTrigMode:	= 0	// Continuous triggering
SerialEncTrigEna:	= 1	// Enable triggering
SerialEncGtoB:	= 0	// No Gray code supported for Tamagawa protocol
SerialEncDataReady:	= 0	// Read-only status bit
SerialEncStatusBits:	= 0	// No status bits supported for Tamagawa protocol
SerialEncNumBits:	= 0	// Fixed number of position bits returned

Gate3[*i*].Chan[*j*].SerialEncCmd would be set to $001A1000 for continuous position reporting. (It may report back as $001A1400 if the ready status bit is set.)

0				0				1				A				1				0				0				0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	-	-	0	1	-	-	-	-	-	-	-	-	-	-	-	-
								SerialEncCmdWord								*Parity*		*TM*	*TE*	*GB*	*Rdy*	*Status*				*NumBits*					

If the ***SerialEncCmdWord*** component is set to $BA, $C2, or $62, the multi-turn position value in the encoder is reset to 0. This should be done in "one-shot" mode, making the element equal to $00BA3000, $00C23000, or $00623000, respectively. When the reset operation is done, the component should report as $00BA0200, $00C20200, or $00620200, respectively.

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

For a Tamagawa FA-Coder encoder with 17 bits per revolution and 16 bits of "turns count", **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	S 16	S 15	S 14	S 13	S 12	S 11	S 10	S 9	S 8	S 7	S 6	S 5	S 4	S 3	S 2	S 1	S 0
																						Single-Turn Position									

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	E	-	-	F	F	F	F	A	A	A	A	A	A	A	A	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
1	0			7	6	5	4	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Err Bits				*Status Field*						*Alarm Code*											*Multi-Turn Position*										

Bits S*n* represent the bits of single-turn position; bits M*n* represent the bits of multi-turn position (“turns count”). Overall position can be read as a combination of single-turn and multi-turn position values. However, note that these values are not directly adjacent to each other, so care must be taken in combining them. Bits A*n* represent bits of the alarm code, bits F*n* represent bits of the status field; and bits E*n* represent error bits. (E0 is CRC error, and E1 is timeout error.)

Note that the single-turn and multi-turn position data sections are not continuous. This means that Power-PMAC’s automatic algorithms for assembling power-on servo position covering the entire range of travel of the encoder will not work for this data format, so this must be done “manually” in the user’s application code. Refer to the section *Use for Absolute Power-On Position*, below.

Panasonic Protocol Implementation

Panasonic provides serial encoders sending numerical binary data that can be used for power-on and ongoing position data.

Connections

The Panasonic serial encoder interface is a 2-wire interface, using 1 bi-directional differential signal pair, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	DATA+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	(no connect)

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for a Panasonic serial encoder. Because there is no explicit clock signal, the serial encoder clock in the DSPGATE3 must be 20 times the bit transmission frequency to “oversample” the input data stream.

SerialClockMDiv:	= (5 / f_{bit}) - 1	// Serial clock freq. = 20x bit transmission freq.
SerialClockNDiv:	= 0	// No further division
SerialTrigClockSel:	= 0	// Use phase clock if possible
SerialTrigEdgeSel:	= 0	// Use rising clock edge if possible
SerialTrigDelay:	= 0	// Can increase from 0 if possible to reduce latency
SerialProtocol:	= $08	// Selects Panasonic protocol

For example, for a 2.5 MHz bit transmission rate, a 50 MHz serial clock frequency is used, and **Gate3[*i*].SerialEncCtrl** is set to $01000008 for triggering on the rising edge of phase clock without delay.

0				1				0				0				0				0				0				8			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	01
	SerialClockMDiv							*SerialClockNDiv*				--	--	*TC*	*TE*			*SerialTrigDelay*								*SerialProtocol*					

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for a Panasonic serial encoder.

SerialEncCmdWord:	= $2A	// Command word for position reporting in Panasonic
SerialEncParity:	= 0	// No parity check supported for Panasonic protocol
SerialEncTrigMode:	= 0	// Continuous triggering
SerialEncTrigEna:	= 1	// Enable triggering
SerialEncGtoB:	= 0	// No Gray code supported for Panasonic protocol
SerialEncDataReady:	= 0	// Read-only status bit
SerialEncStatusBits:	= 0	// No status bits supported for Panasonic protocol
SerialEncNumBits:	= 0	// Fixed number of position bits returned

Gate3[*i*].Chan[*j*].SerialEncCmd would be set to $002A1000 for continuous position reporting. (It may report back as $002A1400 if the data-ready status bit is set.)

0				0				2				A				1				0				0				0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	0	0	1	0	1	0	1	0	-	-	0	1	-	-	-	-	-	-	-	-	-	-	-	-
								SerialEncCmdWord								*Parity*		*TM*	*TE*	*GB*	*Rdy*			*Status*				*NumBits*			

If the ***SerialEncCmdWord*** component is set to $4A, $7A, $DA, or $F2, the multi-turn position value in the encoder is reset to 0. This should be done in "one-shot" mode, making the element equal to $004A3000, $007A3000, $00DA3000, or $00F23000, respectively. When the reset operation is done, the component should report as $004A0200, $007A0200, $00DA0200, or $00F20200, respectively. If the ***SerialEncCmdWord*** component is set to $52, the encoder ID value is reported where multi-turn position is normally reported.

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

For a Panasonic serial encoder with 17 bits per revolution and 16 bits of "turns count", **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M 7	M 6	M 5	M 4	M 3	M 2	M 1	M 0	0	0	0	0	0	0	0	S 16	S 15	S 14	S 13	S 12	S 11	S 10	S 9	S 8	S 7	S 6	S 5	S 4	S 3	S 2	S 1	S 0
Multi-Turn Position																								*Single-Turn Position*							

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E 1	E 0	-	-	F 7	F 6	F 5	F 4	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	M 15	M 14	M 13	M 12	M 11	M 10	M 9	M 8
Err Bits				*Status Field*																				*Multi-Turn Position*							

Bits S*n* represent the bits of single-turn position; bits M*n* represent the bits of multi-turn position ("turns count"). Overall position can be read as a combination of single-turn and multi-turn position values. However, note that these values are not directly adjacent to each other, so care must be taken in combining them. Bits F*n* represent bits of the status field; and bits E*n* represent error bits. (E0 is CRC error, and E1 is timeout error.)

Note that the single-turn and multi-turn position data sections are not continuous. This means that Power-PMAC's automatic algorithms for assembling power-on servo position covering the entire range of travel of the encoder will not work for this data format, so this must be done "manually" in the user's application code. Refer to the section *Use for Absolute Power-On Position*, below.

Mitutoyo Protocol Implementation

Mitutoyo provides a serial data interface on its Absolute Linear Scale line of encoders which sends numerical binary data that can be used for power-on and ongoing position data.

Connections

The Mitutoyo serial encoder interface (half-duplex) is a 2-wire interface, using 1 bi-directional differential signal pair, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	DATA+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	(no connect)

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for a Mitutoyo serial encoder. Because there is no explicit clock signal, the serial encoder clock in the DSPGATE3 must be 20 times the bit transmission frequency to "oversample" the input data stream.

SerialClockMDiv:	= (5 / f_{bit}) - 1	// Serial clock freq. = 20x bit transmission freq.
SerialClockNDiv:	= 0	// No further division
SerialTrigClockSel:	= 0	// Use phase clock if possible
SerialTrigEdgeSel:	= 0	// Use rising clock edge if possible
SerialTrigDelay:	= 0	// Can increase from 0 if possible to reduce latency
SerialProtocol:	= $09	// Selects Mitutoyo protocol

For example, for a 2.5 MHz bit transmission rate, a 50 MHz serial clock frequency is used, and **Gate3[*i*].SerialEncCtrl** is set to $01000009 for triggering on the rising edge of phase clock without delay.

0				1				0				0				0				0				0				9			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
SerialClockMDiv								*SerialClockNDiv*				--	--	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for a Mitutoyo serial encoder.

SerialEncCmdWord:	= $01	// Command word for position reporting in Mitutoyo
SerialEncParity:	= 0	// No parity check supported for Panasonic protocol
SerialEncTrigMode:	= 0	// Continuous triggering

SerialEncTrigEna:	= 1	// Enable triggering
SerialEncGtoB:	= 0	// No Gray code supported for Mitutoyo protocol
SerialEncDataReady:	= 0	// Read-only status bit
SerialEncStatusBits:	= 0	// No status bits supported for Mitutoyo protocol
SerialEncNumBits:	= 0	// Fixed number of position bits returned

Gate3[*i*].Chan[*j*].SerialEncCmd would be set to $00011000 for continuous position reporting. (It may report back as $00011400 if the data-ready status bit is set.)

0				0				0				1				1				0				0				0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	0	0	0	0	1	0	0	1	-	-	0	1	-	-	-	-	-	-	-	-	-	-	-	-
SerialEncCmdWord																*Parity*		*TM*	*TE*	*GB*	*Rdy*	*Status*				*NumBits*					

If the ***SerialEncCmdWord*** component is set to $89, the multi-turn position value in the encoder is reset to 0 (after 8 cycles). If the ***SerialEncCmdWord*** component is set to $9D, the encoder ID value is reported in bits 8 – 15 of **SerialEncDataB**. If the ***SerialEncCmdWord*** component is set to $85, absolute position is reported, just as if the value were $01.

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

For the Mitutoyo encoder, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Absolute Position																															

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	E	-	-	F	F	F	F	A	A	A	A	A	A	A	A	ID	ID	ID	ID	ID	ID	ID	ID	-	-	-	-	-	-	-	-
1	0			3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Err Bits				*Status Field*				*Alarm Code*								*Encoder ID*															

Bits P*n* represent the bits of absolute position. Bits ID*n* represent bits of the encoder ID (only provide on specific request). Bits A*n* represent bits of the alarm code reported from the encoder. Bits F*n* represent bits of the status field reported from the encoder. Bits E*n* represent bits of the error code detected by the IC (E0 is CRC error, E1 is timeout error).

Kawasaki Protocol Implementation

Mitutoyo provides a serial data interface on its MAR-H32 line of 29-bit absolute encoders (13 bits of single-turn information and 16 bits of turns count) which send numerical binary data that can be used for power-on and ongoing position data.

Connections

The Kawasaki serial encoder interface is a 2-wire interface, using 1 bi-directional differential signal pair, plus power and ground. The signal connections from the ACC-24E3 to the encoder will be:

SENCDAT+/-	to:	DATA+/-
SENCENA+/-	to:	(no connect)
SENCCLK+/-	to:	(no connect)

Multi-Channel Setup: Gate3[i].SerialEncCtrl

The following list shows typical settings of **Gate3[*i*].SerialEncCtrl** for a Kawasaki serial encoder. Because there is no explicit clock signal, the serial encoder clock in the DSPGATE3 must be 20 times the bit transmission frequency to “oversample” the input data stream.

SerialClockMDiv:	= (5 / f_{bit}) - 1	// Serial clock freq. = 20x bit transmission freq.
SerialClockNDiv:	= 0	// No further division
SerialTrigClockSel:	= 0	// Use phase clock if possible
SerialTrigEdgeSel:	= 0	// Use rising clock edge if possible
SerialTrigDelay:	= 0	// Can increase from 0 if possible to reduce latency
SerialProtocol:	= $0A	// Selects Kawasaki protocol

For example, for a 2.5 MHz bit transmission rate, a 50 MHz serial clock frequency is used, and **Gate3[*i*].SerialEncCtrl** is set to $0100000A for triggering on the rising edge of phase clock without delay.

0				1				0				0				0				0				0				A			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
SerialClockMDiv								*SerialClockNDiv*				*--*	*--*	*TC*	*TE*	*SerialTrigDelay*								*SerialProtocol*							

Single-Channel Setup: Gate3[i].Chan[j].SerialEncCmd

The following list shows typical settings of **Gate3[*i*].Chan[*j*].SerialEncCmd** for a Kawasaki serial encoder.

SerialEncCmdWord:	= $00	// No command word in Kawasaki protocol
SerialEncParity:	= 0	// No parity check supported for Kawasaki protocol
SerialEncTrigMode:	= 0	// Continuous triggering
SerialEncTrigEna:	= 1	// Enable triggering
SerialEncGtoB:	= 0	// No Gray code supported for Kawasaki protocol
SerialEncDataReady:	= 0	// Read-only status bit
SerialEncStatusBits:	= 0	// No status bits supported for Kawasaki protocol
SerialEncNumBits:	= 0	// Fixed number of position bits returned

Gate3[*i*].Chan[*j*].SerialEncCmd would be set to $00001000 for continuous position reporting.

0				0				0				1				1				0				0				0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	-	-	-	-	-	-	-	-	-	-	-	-
SerialEncCmdWord																*Parity*		*TM*	*TE*	*GB*	*Rdy*	*Status*				*NumBits*					

Single-Channel Enable: Gate3[i].Chan[j].SerialEncEna

To enable the serial-encoder interface circuitry on the ACC-24E3 for this channel, set **Gate3[*i*].Chan[*j*].SerialEncEna** to 1.

Data Registers: Gate3[i].Chan[j].SerialEncDataA/B

For the Kawasaki encoder, **Gate3[*i*].Chan[*j*].SerialEncDataA** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	M	M	M	M	M	M	M	M	M	M	M	C	C	S	S	S	S	S	S	S	S	S	S	S	S	S	S	I	I	I	I
11	10	9	8	7	6	5	4	3	2	1	0	1	0	13	12	11	10	9	8	7	6	5	4	3	2	1	0	3	2	1	0
Multi-Turn Position												*Correction*		*Single-Turn Position*														*Interp Pos*			

For this encoder, **Gate3[*i*].Chan[*j*].SerialEncDataB** is configured as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	E	E	-	-	A	A	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	M	M	M	M
2	1	0			2	1	0																					15	14	13	12
Error Bits					*Alarm Code*																							*Multi-Turn*			

Bits I*n* represent bits of interpolated position. Bits S*n* represent the bits of single-turn position. Bits C*n* represent bits of the multi-turn correction data. Bits M*n* represent bits of multi-turn position. Bits A*n* represent bits of the alarm code reported from the encoder (A0 is INPALM, A1 is ABSALM, A2 is the busy flag). Bits E*n* represent bits of the error code detected by the IC (E0 is coding error, E1 is CRC error, E2 is timeout error).

Use for Ongoing Commutation Feedback

If Power PMAC is performing commutation for a brushless motor with a serial encoder, the encoder position will be used for commutation feedback each phase cycle. Note that in this case, **Gate3[*i*].SerialEncCtrl** should be set up to query the encoder every phase cycle, not just every servo cycle.

Ongoing Commutation Feedback Address: Motor[x].pPhaseEnc

To use serial encoder position for the ongoing commutation position angle, **Motor[*x*].pPhaseEnc** should be set to **Gate3[*i*].Chan[*j*].SerialEncDataA.a** to use the data received from the encoder.

In most cases, there will be enough position data in **SerialEncDataA** to cover at least a full commutation cycle (pole pair of the motor). However, this is not strictly necessary. It is necessary to have enough position data in this single register that the change between consecutive phase cycles will always be less than half of the range of the position data used. For example, if the data only covers only 1/4 of a commutation cycle, it must always change at a rate less than 1/8 of a commutation cycle per phase update.

In the Yaskawa Sigma I protocol, the ongoing position feedback comes from simulated quadrature with 8192 counts per motor revolution, so the instructions for using quadrature encoder feedback, above, should be used.

Feedback Shifting: Motor[x].PhaseEncRightShift, PhaseEncLeftShift

If the position data from the encoder data in **SerialEncDataA** can roll over during operation, it is required that the data that ends up in bit 31 of the resulting 32-bit value be real position data, shifted “left” if necessary, so that Power PMAC can handle rollover of this value properly. Many users will want this bit to be the most significant bit of single-turn data for a rotary encoder, and the examples in this section will use that convention. This shifting is required for serial encoder data formats such as Tamagawa and Panasonic, where the bits in **SerialEncDataA** above the single-turn data do not contain position information.

If the LSB of the encoder data is not in bit 0 of **SerialEncDataA**, it can be desirable to "shift out" the bits below this (to the right) so they cannot interfere with the resulting value. (Note that this shift right operation is often not necessary, as Power PMAC usually only uses 11 bits of position data in a commutation cycle, but many users prefer to do this anyway.)

These tasks are accomplished using **Motor[*x*].PhaseEncRightShift** and **Motor[*x*].PhaseEncLeftShift**. Setting **PhaseEncRightShift** to the number of the bit in **SerialEncDataA** where the LSB of encoder data is found (0 for most protocols, 4 or 6 in some Yaskawa protocols) causes the encoder data to be shifted initially so the LSB is in bit 0.

For a rotary encoder, setting **Motor[*x*].PhaseEncLeftShift** to the quantity (32 minus the number of bits per revolution plus **PhaseEncRightShift**) then causes the MSB of single-turn data to end up in bit 31 of the register. For example, if the encoder has 18 bits per revolution (262,144 LSBs per revolution) and the LSB is found in bit 0 of **SerialEncDataA**, **Motor[*x*].PhaseEncRightShift** should be set to 0, and **Motor[*x*].PhaseEncLeftShift** should be set to (32 - 18 + 0) = 14. If the encoder has 17 bits per revolution (131,072 LSBs per revolution) and the LSB is found in bit 4 of **SerialEncDataA**, **Motor[*x*].PhaseEncRightShift** should be set to 4, and **Motor[*x*].PhaseEncLeftShift** should be set to (32 - 17 + 4) = 19. If the total position data in this register will not roll over during operation, this shifting is not required, but it is acceptable to do for rotary motors, and will not add any computational time.

For a linear motor and encoder, where the size of the commutation cycle is generally not a power of 2, there is no "number of bits in a commutation cycle". Even so, it is necessary to have real position data end up in bit 31 of the register if the data in **SerialEncDataA** can roll over during the application, performing any shifting necessary to obtain this result.

Ongoing Commutation Feedback Scaling: Motor[x].PhasePosSf

The user must calculate how many LSBs of this 32-bit register (after any shifting) there are per commutation cycle in order to set the commutation scale factor **Motor[*x*].PhasePosSf**, which multiplies the value in this intermediate register to obtain the commutation angle, in units of 1/2048 of a commutation cycle. If the MSB of single-turn data is in bit 31 of this register so there are 2^{32} (4G) register LSBs per motor revolution, the scale factor can be calculated by the equation:

$$PhasePosSf = \frac{2048}{4{,}294{,}967{,}296} * \frac{PolesPer\,\mathrm{Re}\,v}{2}$$

For the common 4-pole rotary motor (which has 2 commutation cycles per revolution) this reduces to:

$$PhasePosSf = \frac{1}{1{,}048{,}576}$$

For example, an 8-pole brushless rotary motor has a Tamagawa serial encoder with 17 bits per revolution providing true position data in bits 0 – 16 of **SerialEncDataA**, with no position data above that in this register. The data is shifted left 15 bits so the MSB is in bit 31 of the result. **Motor[*x*].PhasePosSf** can be calculated as:

$$PhasePosSf = \frac{2048}{4{,}294{,}967{,}296} * \frac{8}{2} = \frac{1}{524{,}288}$$

The linear motor/encoder calculations for **PhasePosSf** are typically done a little differently because the number of encoder LSBs per commutation cycle is not necessarily a power of 2, and probably will not be. For a linear serial encoder where the LSB from the encoder ends up in bit *M* of the register after any shifting, the scale factor can be calculated from the following equation:

$$PhasePosSf = \frac{2048}{LSBsPerCommCycle} * 2^M$$

For example, if a linear scale with 10-nanometer resolution is used on a linear motor with a pole-pair pitch of 60 millimeters and the encoder LSB is in bit 0 of the source after any shifting, **Motor[*x*].PhasePosSf** should be set to 2048 / (60,000,000/10), or 3.41333… x 10^{-4}. It is best to enter these values as expressions and let Power PMAC calculate the resulting values to full double-precision resolution.

Use for Power-On Commutation Position

If the serial encoder on a synchronous brushless motor provides absolute (as opposed to incremental) position data over at least one commutation cycle of the motor, it can be used for power-on commutation position, eliminating the need for a phasing search move to establish the commutation position reference.

Power-On Commutation Feedback Address: Motor[x].pAbsPhasePos

If the serial-encoder position value is used to determine the absolute power-on phase position angle, **Motor[*x*].pAbsPhasePos** should be set to **Gate3[*i*].Chan[*j*].SerialEncDataA.a** to use the position data received from the encoder.

(In the Yaskawa Sigma I protocol, the absolute single-turn position value is found in **Gate3[*i*].Chan[*j*].PhaseCapt** in units of 1/256 count from synthesized quadrature after the absolute position has been queried, so **Motor[*x*].pAbsPhasePos** should be set to the address of this register.)

Power-On Commutation Feedback Format: Motor[x].AbsPhasePosFormat

Motor[*x*].AbsPhasePosFormat must be set to tell Power PMAC how to read and interpret the data in this input register. This 32-bit value consists of 4 independent bytes that specify the starting bit to use, the number of bits to use, how to use data in subsequent registers (if any), and how to interpret the selected data from the register. It is represented in the hex format $*aabbccdd*, with each pair of hex digits representing one of the bytes. For serial absolute encoders, the first byte (*aa*) should be set to $00 to specify numerical binary format.

The second byte (*bb*) specifies has a second register should be used for this absolute position, if necessary. (If a second register is not used, this byte setting does not matter, but it is usually left at the default of $00.) A second register will be used if the total number of bits specified, given the starting bit number specified, means that the data cannot be gotten entirely from a single register. In this case, the register at the next address will also be used – if the first register is specified as **Gate3[*i*].Chan[*j*].SerialEncDataA**, then the second register will automatically be **Gate3[*i*].Chan[*j*].SerialEncDataB**. Values of $00 to $1F (0 to 31) in *bb* specify the starting bit number to use in this second register.

The third byte (*cc*), which specifies the number of bits to be used (up to 32) should be set to a number of bits at least as big as that necessary to cover one commutation cycle when the size of the commutation cycle can be expressed as a power of 2 LSBs of the encoder, which should be true for all rotary serial encoders. (It is OK to specify more, but not less.)

If the size of the commutation cycle cannot be expressed as a power of 2 LSBs of the encoder, which will generally be the case for linear encoders, then a number of bits at least as big as that necessary to cover the entire travel of the motor must be used. If this requires more than 32 bits starting at the encoder LSB, then enough of the least significant bits of the encoder should not be used (done by increasing the value in byte *dd*) to reduce the total bits used to 32 or less.

The fourth byte (*dd*) should be set to the number of the lowest bit used in **SerialEncDataA** for the absolute position, generally, the LSB of the encoder position data. (Note that there is no pre-shifting of data for absolute phase position.)

For example, if a rotary encoder has 18 ($12) bits of single-turn resolution starting in bit 0 of **SerialEncDataA**, so it is not required to use a second register, **Motor[*x*].AbsPhasePosFormat** would be set to $00001200.

If a rotary encoder has 20 ($14) bits of single-turn resolution starting in bit 4 of **SerialEncDataA**, **Motor[*x*].AbsPhasePosFormat** would be set to $00001404.

If a linear encoder with 10-nanometer resolution is used on a linear brushless motor with a pole-pair pitch of 60 millimeters and a total travel of 1200 millimeters, the commutation cycle cannot be expressed as a power of 2 of the encoder LSBs, so enough bits must be read to cover the entire range of travel of the motor. This range is 120 million LSBs of the encoder position, which is over 2^{26} LSBs, so 27 ($1B) bits must be read. If the encoder LSB is found in bit 0 of **SerialEncDataA**, then **Motor[*x*]AbsPhasePosFormat** should be set to $00001B00.

Power-On Commutation Feedback Scaling: Motor[x].AbsPhasePosSf

Motor[*x*].AbsPhasePosSf must be set to convert the data into units of the commutation table. It multiplies the formatted data value that has been read and the resulting data is assumed to have units of 1/2048 of a commutation cycle. It is computed like **Motor[*x*].PhasePosSf**, except it uses specified formatted data without any shifting operations. For an encoder with *N* bits of single-turn resolution, it can be calculated by the equation:

$$AbsPhasePosSf = \frac{2048}{2^N} * \frac{PolesPer\,\mathrm{Re}\,v}{2}$$

For an encoder with 18 bits of single-turn resolution on a 4-pole motor, it can be calculated as 4,096 / 262,144, or 1 / 64. If you enter the value as an expression, Power PMAC can calculate the resulting value exactly.

Power-On Commutation Feedback Offset: Motor[x].AbsPhasePosOffset

Motor[*x*].AbsPhasePosOffset must be set to specify the difference between the encoder's zero position and Power PMAC's commutation zero angle, in commutation units (1/2048 of a commutation cycle). The proper value is usually found by forcing the motor to the commutation zero angle with a "stepper-motor" phasing search move, reading the value of the encoder in **Gate3[*i*].Chan[*j*].SerialEncDataA** at this position, multiplying this value by **Motor[*x*].AbsPhasePosSf**, and writing the negative of this product into **AbsPhasePosOffset**.

Use for Ongoing Servo Feedback or Master

Serial encoder position data is commonly used for servo-loop position feedback, both for the outer (position) loop and inner (velocity) loop. Sometimes it is used as master position for a motor's position following (electronic gearing) function. In both cases, the data from the IC must be processed with an entry in the encoder conversion table, and the result of the entry used by the motor.

Encoder Conversion Table Entry Method: EncTable[n].Type

To process the resulting position value for servo-loop use (feedback or master), a "Type 1" single-register-read conversion should be specified in the encoder conversion table. **EncTable[*n*].Type** should be set to 1.

Encoder Conversion Table Entry Source Address: EncTable[n].pEnc

EncTable[*n*].pEnc, which specifies the address of this register, should be set to **Gate3[*i*].SerialEncDataA.a**. (**EncTable[*n*].pEnc1**, which specifies a second source, is not used in this mode; its setting does not matter.) Note that if there is more significant position data in **SerialEncDataB**, it is not used on an ongoing basis, although it can be used to establish absolute power-on position. Position data over the full range of travel is not required each servo cycle, as Power PMAC can roll over and extend the position data it does use each cycle as long as it reads enough data so that this data cannot travel half of its range in a single servo cycle.

Encoder Conversion Table Entry Data Shifting: EncTable[n].index1, index2

Conversion parameters **EncTable[*n*].index1** and **EncTable[*n*].index2** are set depending on which bits of the 32-bit register contain real position data. For a serial encoder with *N* bits of data starting with bit *M*, **index2**, which specifies the "shift right" operation, should be set to *M*. The element **index1**, which specifies the subsequent "shift left" operation, should be set to (32 - *N* + *M*) so the MSB from the encoder register ends up in bit 31 of the resulting register, permitting Power PMAC to handle rollover of this data properly.

Encoder Conversion Table Entry Change Limiting: EncTable[n].index3, MaxDelta

Some users will want to protect against possible erroneous readings from the encoder with a "change limiting" filter in the encoder. This filter looks for data changes between consecutive cycles beyond a user-defined threshold, and if it detects this, it presumes the data is erroneous and substitutes an estimate based on recent data.

This filter is enabled by setting **EncTable[*n*].index3** greater than 0. If it is set to 1, the entry compares the magnitude of the first derivative (velocity) of the read data to **EncTable[*n*].MaxDelta**, which is expressed in LSBs of the source data (after the right shift by **index2**) per servo cycle. If the change exceeds this limit, the position read this cycle is not used; instead a position is calculated assuming the previous cycle's velocity was maintained. If the data read in subsequent cycles also exceeds the limit, a position is calculated using the **MaxDelta** velocity toward the read position, which permits a change to a new position source.

If **EncTable[*n*].index3** is set to a value greater than 1, the entry compares the magnitude of the second derivative (acceleration) of the read data to **EncTable[*n*].MaxDelta**, which in this case is expressed in LSBs of the source data (after the right shift by **index2**) per servo cycle per servo cycle. If the change exceeds this limit, the position read this cycle is not used; instead a position is calculated assuming the previous cycle's acceleration was maintained. If the read data in subsequent cycles also exceeds the limit, this acceleration is maintained for a total of **index2** servo cycle. If the limit is violated for more servo cycles than this, a position is calculated using the **MaxDelta** acceleration toward the read position, which permits a change to a new position source.

Encoder Conversion Table Entry Scaling: EncTable[n].ScaleFactor

Floating-point element **EncTable[*n*].ScaleFactor** determines the scaling of the entry final result. Most users want this result scaled in units of LSBs of the encoder. To do this for an *N*-bit serial encoder that has been shifted so its MSB is in bit 31, **ScaleFactor** should be set to (1 / 2^{32-N}). For a 24-bit encoder processed this way, it should be set to (1 / 2^8), or (1 / 256). Usually, this value is entered as an expression, and Power PMAC automatically computes the result.

Motor Position Source Addresses: Motor[x].pEnc, pEnc2, pMasterEnc

As with any type of feedback, the processed result of the conversion table can be used as feedback or master by setting **Motor[*x*].pEnc**, **pEnc2**, or **pMasterEnc** to **EncTable[*n*].a**. The data read from the conversion table entry using these functions is automatically scaled into motor units with a multiplication by **Motor[*x*].PosSf**, **PosSf2**, or **MasterPosSf**, respectively. At the default values of 1.0 for these elements, the motor units are the same as the output units from the conversion table, typically LSBs of the encoder.

Use for Power-On Servo Position

If the serial-encoder position value is absolute over the entire travel of the application, it can be used to determine the absolute power-on motor position. This eliminates the need for a homing search move to establish the position reference for the motor.

Power-On Servo Position Address: Motor[x].pAbsPos

Motor[*x*].pAbsPos should be set to **Gate3[*i*].Chan[*j*].SerialEncDataA.a** to use the position data received from the encoder. In rotary-motor applications, the position typically must be absolute over multiple turns of the motor. Note that this function makes no specific distinction between position within a single turn and "turns count" position – the two values must seamlessly combine into a single position number. (A single-turn rotary serial encoder that is absolute over one motor revolution can be used for absolute power-on phase position, but not for absolute power-on servo position if the motor can turn more than one revolution in the application.)

Power-On Servo Position Format: Motor[x].AbsPosFormat

Motor[*x*].AbsPosFormat must be set to tell Power PMAC how to read and interpret the data in this input register. This 32-bit value consists of 4 independent bytes that specify the starting bit to use, the number of bits to use, how to use data in subsequent registers (if any), and how to interpret the selected data from the register. It is represented in the hex format $*aabbccdd*, with each pair of hex digits representing one of the bytes. For serial encoders, *aa* should be set to $00 to specify unsigned numerical binary format or to $01 to specify signed numerical binary format.

The second byte (*bb*) specifies how a second register should be used for this absolute position, if necessary. (If a second register is not used, this byte setting does not matter.) A second register will be used if the total number of bits specified, given the starting bit number specified, means that the data cannot be gotten entirely from a single register. In this case, the register at the next address will also be used – if the first register is specified as **Gate3[*i*].Chan[*j*].SerialEncDataA**, then the second register will automatically be **Gate3[*i*].Chan[*j*].SerialEncDataB**. Values of $00 to $1F (0 to 31) in *bb* specify the starting bit number to use in this second register.

Note that this function requires that the data used in the first register goes all the way to bit 31 of that register, so the bit in the second register specified here is taken to be the next most significant bit. This is not true in all of the serial encoder protocols – if it is not true, this absolute position will need to be assembled in the user application (see below).

The third byte (*cc*) specifies the total number of bits to be used from the encoder for the absolute position. The fourth byte (*dd*) specifies the starting bit number in the first register to use. Note that there is no pre-shifting of data for absolute phase position. If the number of bits minus the starting bit number is greater than 32, a second register will automatically be used.

For example, if the encoder has 18 bits of single-turn resolution and 12 bits of multi-turn resolution for 30 bits total starting in bit 0, and is to be treated as a signed value, **Motor[*x*].AbsPosFormat** would be set to $01001C00.

Power-On Servo Position Scaling: Motor[x].AbsPosSf

Motor[*x*].AbsPosSf is used to multiply the formatted absolute position value read from the specified register to convert it into motor units. If the motor units are equivalent to LSBs of the encoder, it should be set to the default value of 1.0. When the same sensor is used for power-on and absolute position information, as is usually true with serial encoders, it should be set to the same value as **Motor[*x*].PosSf**.

Power-On Servo Position Offset: Motor[x].AbsPosOffset

Motor[*x*].AbsPosOffset specifies the (signed) difference between the absolute sensor's zero position and the motor's zero position, in motor units. When the absolute power-on position is read from the encoder, after the value is scaled into motor units with **Motor[*x*].AbsPosSf**, the value of **Motor[*x*].AbsPosOffset** is added to obtain the motor position.

"Manual Assembly" of Power-On Servo Position

In several of the serial encoder protocols, the single-turn position and multi-turn position data does not appear in a continuous data set in the IC registers. When this is the case, Power PMAC's automatic algorithms cannot assemble a single position value for power-on absolute position, and it is necessary to perform this function in the user application software, typically in a "power-on" PLC program.

In the Tamagawa FA-Coder protocol, the single-turn position appears in bits 0 – 16 of **Gate[*i*].Chan[*j*].SerialEncDataA**, and the multi-turn position appears in bits 0 – 15 of **Gate[*i*].Chan[*j*].SerialEncDataB**. Sample code to combine them into a single value, and to force the motor position to this value is:

```
MyAbsPos = Gate3[0].Chan[0].SerialEncDataA & $1FFFF;
MyAbsPos += (Gate3[0].Chan[0].SerialEncDataB & $FFFF) << 17;
Motor[1].Pos = MyAbsPos * Motor[1].AbsPosSf + Motor[1].AbsPosOffset;
```

In this example, **MyAbsPos** is a user variable, **Motor[1].AbsPosSf** and **Motor[1].AbsPosOffset** are the saved setup elements used for equivalent functions in this custom algorithm, and

Motor[1].Pos is the (uncorrected) actual position value for the motor, which can be written to when the motor is disabled.

In the Panasonic protocol, the single-turn position appears in bits 0 – 16 of **Gate[*i*].Chan[*j*].SerialEncDataA**, and the multi-turn position appears in bits 24 to 31 of this register and bits 0 – 7 of **Gate[*i*].Chan[*j*].SerialEncDataB**. Sample code to combine them into a single value, and to force the motor position to this value is:

```
MyAbsPos = Gate3[0].Chan[0].SerialEncDataA & $1FFFF;
MyAbsPos += (Gate3[0].Chan[0].SerialEncDataA & $FF000000) >> 7;
MyAbsPos += (Gate3[0].Chan[0].SerialEncDataB & $FF) << 24;
Motor[1].Pos = MyAbsPos * Motor[1].AbsPosSf + Motor[1].AbsPosOffset;
```

In the Yaskawa Sigma I protocol, the single-turn position (from synthesized quadrature) appears in **Gate3[*i*].Chan[*j*].ServoCapt** in units of 1/256 of a count, with 8192 counts per revolution. The turns count information appears as ASCII text in **Gate3[*i*].Chan[*j*].SerialEncDataA** and **SerialEncDataB**. Note that the ASCII code for a numerical digit is 48 greater than the value of that digit. Sample code to assemble the full absolute position (optimized for clarity, not brevity or efficiency) is:

```
TempA = Gate3[0].Chan[0].SerialEncDataA;         // Read into memory
TempB = Gate3[0].Chan[0].SerialEncDataB;         // Read into memory
MyAbsPos = (TempA & $FF) - 48;                   // 1’s digit value
MyAbsPos += ((TempA & $FF00) - 48) * 10;         // 10’s digit value
MyAbsPos += ((TempA & $FF0000) - 48) * 100;      // 100’s digit value
MyAbsPos += ((TempA & $FF000000) - 48) * 1000;   // 1000’s digit value
MyAbsPos += ((TempB & $FF) - 48) * 10000;        // 10000’s digit value
if ((TempB & $FF00) >> 8 == 45) {                // Minus sign?
  MyAbsPos = -MyAbsPos;                          // Negate value
}
MyAbsPos *= 8192;             // Convert from turns to counts
MyAbsPos += Gate3[0].Chan[0].ServoCapt/256;      // Add single turn pos
Motor[1].Pos = MyAbsPos * Motor[1].AbsPosSf + Motor[1].AbsPosOffset;
```

In this example, **TempA**, **TempB**, and **MyAbsPos** are user variables, **Motor[1].AbsPosSf** and **Motor[1].AbsPosOffset** are the saved setup elements used for equivalent functions in this custom algorithm, and **Motor[1].Pos** is the (uncorrected) actual position value for the motor, which can be written to when the motor is disabled.

Use for Trigger Position

Direct immediate hardware capture of the serial encoder position is not possible, because the IC only receives a new position once per phase or servo cycle. For triggered moves, **Motor[*x*].CaptureMode** can be set to 1 to specify “software capture” with an input trigger. In this mode, Power PMAC uses the already-processed motor actual position value from the most recent servo cycle as the trigger position. Note that there is a potential delay of up to one real-time-interrupt period (**Sys.RtIntPeriod** + 1 servo-interrupt periods) from the trigger occurrence to the time of the position used to calculate the post-trigger move. This potential delay time should be multiplied by the speed to obtain the uncertainty in the captured position.

The accuracy of the software capture can be improved significantly with Power PMAC’s “timer-assisted software capture” feature, which uses a timer value latched by the trigger to interpolate between servo-cycle positions. This mode is specified by setting **Motor[*x*].CaptureMode** to 3.

Several setup elements of the IC channel must be set to drive the timer circuitry properly for this mode of operation. Refer to the *Basic Motor Moves* chapter of the User's Manual for details.

Monitoring for Sensor Loss

The ACC-24E3 has circuitry that can detect loss of signal from serial encoder inputs in most protocols, and standard Power PMAC algorithms can use this circuitry to automatically disable motors when this loss is detected. In most protocols, the circuitry can detect the lack of response to a position request of the encoder, and will set a "timeout error" bit because of this.

For those protocols with this feature, this timeout error bit appears as bit 31 in the **Gate3[*i*].Chan[*j*].SerialEncDataB** element. Some of the protocols without an explicit timeout error bit have a parity error or CRC error bit that will be set when there is no feedback.

To employ this circuitry for automatic loss detection, set **Motor[*x*].pEncLoss** to **Gate3[*i*].Chan[*j*].Status.a**. (It can also be set to **Gate3[*i*].Chan[*j*].SosError.a**, but will report back as **Gate3[*i*].Chan[*j*].Status.a**). Set **Motor[*x*].EncLossBit** to 31 to specify the timeout error bit number in the register. Set **Motor[*x*].EncLossLevel** to 1 to specify a high-true loss status bit.

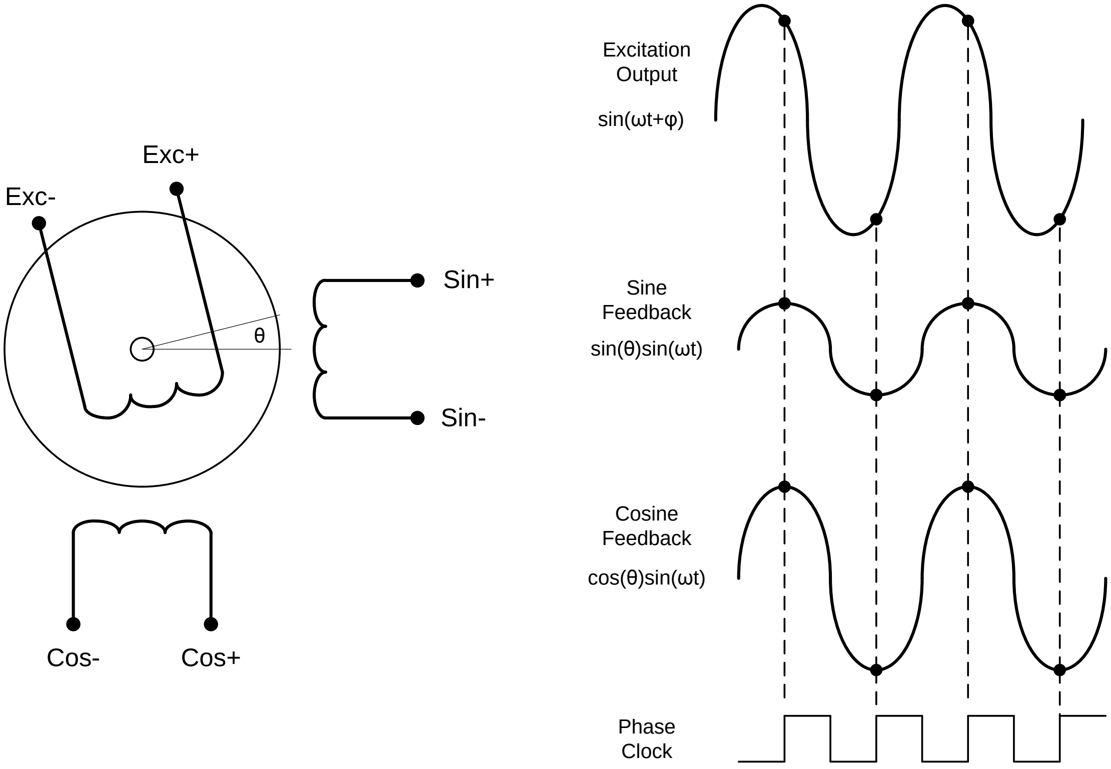
Because it is possible for signal transients to cause a momentary setting of this, it is usually desirable to set **Motor[*x*].EncLossLimit** to a value greater than 0. The accumulated count of loss detection events must exceed this threshold before a fault is declared. Reasonable values for this element will not unduly delay reaction to true faults.

Resolvers

The ACC-24E3 can provide a compact, cost-effective, and high-quality interface for up to four resolvers. The hardware generates an excitation output signal for each resolver, and then processes the returned signals from the "sine" and "cosine" windings.

The accessory performs a "direct" conversion by computing the arctangent of the sine and cosine readings, which have been effectively demodulated by sampling synchronously with the excitation signal, at or near the peak of the carrier frequency component. If noise mitigation is required, a software tracking filter can be implemented in the Encoder Conversion Table that performs the noise-mitigation tasks of a traditional tracking hardware R/D converter.

The following diagram shows the principle of operation for this type of resolver interface. It shows a case where the excitation frequency is one half of the phase clock frequency and the feedback waveforms lag the excitation waveform by about one eighth of a cycle. Note that the feedback signals are sampled twice per cycle, on the rising edge of the phase clock, so that the data is ready for use on the phase or servo interrupt that coincides with the clock's falling edge.



Resolver Interface Principle of Operation

Note that it is possible to connect a second resolver – usually one geared down from the primary resolver to provide multi-turn position information – to the ALTSIN+ and ALTSIN- inputs of the same channel. These inputs are sampled on the falling edge of the phase clock .In this case, the excitation frequency generally should be set equal to the phase clock frequency so the second resolver signals can also be sampled where they have a large magnitude.

IC Hardware Setup

For the resolver interface using the PMAC3 ASIC, there are four setup elements in the IC, one multi-component element for all channels, and three elements for each channel.

Excitation Signal Control: Gate3[i].ResolverCtrl

The key configuration element in the PMAC3 ASIC-based resolver interface is multi-channel element **Gate3[*i*].ResolverCtrl**. This is a 32-bit element, with the only the high 12 bits, represented by the first three hexadecimal digits, presently being used. This element specifies the magnitude, frequency, and phase of the automatically generated sinusoidal excitation signal used for all four channels of the ASIC.

The highest 8 bits of the element, which form component ***ResolverExciteShift***, represented in the first two hex digits of the element, specify the phase shift of the excitation sine wave relative to the phase clock. It can take a value from $00 (0) to $FF (255), in units of 1/512 of an excitation cycle. This component is usually set experimentally, to maximize the magnitude of the feedback signals, which are sampled on the rising edge of the phase clock. Resolvers with different electrical L/R time constants will require different phase shifts to provide maximum readings.

The IDE setup control for resolver feedback does this automatically; it is also possible to do this manually, by monitoring the magnitude of status element **Gate3[*i*].Chan[*j*].SumofSquares** for the ASIC channel. The ASIC automatically computes this value each phase cycle from the readings of input elements **Gate3[*i*].Chan[*j*].AdcEnc[0]** and **AdcEnc[1]**.

The next 2 bits of the element, which form component ***ResolverExciteGain***, represented in the high part of the third hex digit, specify the magnitude of the excitation output. Values of 0, 1, 2, and 3 for this component specify magnitudes of 1/4, 1/2, 3/4, and 1 times, respectively, the maximum magnitude. The highest magnitude that does not cause saturation of the feedback ADCs (which can be detected by values in **SumOfSquares** greater than 32,767) should be used.

The next 2 bits of the element, which form component ***ResolverExciteFreq***, represented in the low part of the third hex digit, specify the frequency of the excitation output. Values of 0, 1, 2, and 3 for this component specify frequencies of 1, 1/2, 1/4, and 1/6 times, respectively, the phase clock frequency. The frequency that comes closest to that recommended by the resolver manufacturer should be used.

For example, to set up an excitation signal with a ¼-cycle (128/512 of a cycle) lag, ½ of the maximum magnitude, and at ¼ of the phase clock frequency, **Gate3[*i*].ResolverCtrl** would be set to $80600000. The $80 (128) in the first two hex digits specifies the lag, the 4's bit in the third digit specifies ½ of the maximum magnitude, and the 2's bit in the third digit specifies ¼ of the phase clock frequency.

Direction Sense Control: Gate3[i].Chan[j].EncCtrl

The direction sense of the resolver conversion for a channel is determined by bit 2 (value 4) of channel saved setup element **Gate3[*i*].Chan[*j*].EncCtrl**. Usually the element value is just changed between its default value of 7 and 3, but for purposes of the resolver conversion, all that matters is the value of bit 2 (value 4).

Note that changing the direction sense on the feedback resolver for a motor with a properly operating servo loop can cause a dangerous runaway condition.

Offset Compensation: Gate3[i].Chan[j].AdcOffset[0], AdcOffset[1]

Before the IC computes the arctangent of the sine and cosine input values, it adds the value of saved setup element **Gate3[*i*].Chan[*j*].AdcOffset[0]** to the sine value in input register **Gate3[*i*].Chan[*j*].AdcEnc[0]**, and **Gate3[*i*].Chan[*j*].AdcOffset[1]** to the cosine value in input register **Gate3[*i*].Chan[*j*].AdcEnc[1]**. Non-zero values in these offset registers can compensate for biases in the resolver signals or analog receiving circuits.

Data Register: Gate3[i].Chan[j].AtanSumOfSqr

The position value automatically computed by the IC hardware from the compensated sine and cosine values is found in the 32-bit status element **Gate3[*i*].Chan[*j*].AtanSumOfSqr**. This full-word element is comprised of two 16-bit elements, each individually accessible in the Script environment. **Gate3[*i*].Chan[*j*].Atan** is found in the high 16 bits, and represents the position value calculated as the arctangent of the compensated sine and cosine readings, with 65,536 states per electrical cycle of the resolver.

Gate3[*i*].Chan[*j*].SumOfSquares is found in the low 16 bits, and is primarily used for diagnostics, both in the initial setup and in operation. In setup, this value can be monitored to optimize the phase shift of the excitation signal with the objective of maximizing its magnitude. During operation, it can be checked to ensure that proper signals are being received. Note that if all 4 of the most significant bits of this element are 0, the channel status bit **Gate3[*i*].Chan[*j*].SosError** is set to 1, indicative of sensor loss.

Use for Ongoing Commutation Feedback

If resolver feedback is used for the ongoing commutation position angle, **Motor[*x*].pPhaseEnc** should be set to **Gate3[*i*].Chan[*j*].AtanSumOfSqr.a** to use the angle value that the IC calculated automatically from the sine and cosine-winding readings. (It can also be set to **Gate3[*i*].Chan[*j*].Atan.a**, because the partial-word element has the same address as the full-word element, but it will report back as the address of the full-word element when queried.)

The actual arctangent value is in the high 16 bits of the 32-bit register, so there are 2^{32} LSBs per cycle of the resolver. The number of LSBs per commutation cycle can be calculated as 4,294,967,296 (2^{32}) divided by the number of commutation cycles per resolver cycle. (For example, if a 2-pole resolver is used on a 4-pole motor, there are two commutation cycles per resolver cycle).

Note that it is possible to eliminate the “sum of squares” data in the low 16 bits of this register by setting **Motor[*x*].PhaseEncRightShift** to 16 to shift these low bits out, and setting **Motor[*x*].PhaseEncLeftShift** to 16 to bring the actual position data back to the top of the resulting value, with 0’s below. (The most significant bit of the position data must end up in bit 31 of the resulting 32-bit register to support rollover properly.) However, this will probably not result in any improvement in performance, as Power PMAC only uses the most significant 11 bits to compute the commutation phase angle in any case.

Motor[*x*].PhasePosSf can be calculated as 2048 divided by the number of LSBs of the source register per commutation cycle. Usually it is best to enter this value as an expression and let Power PMAC calculate the (double-precision floating-point) value, which is typically much smaller than 1.0.

Power PMAC reads the entire 32-bit register each phase cycle and scales the result to the 2048-part commutation cycle. (Note that this means that only the high 11 bits are really needed, as 2^{11}

= 2048.) **Motor[*x*].PhasePosSf** can be calculated as 2048 divided by the number of LSBs of the source register per commutation cycle. One cycle of the resolver position in this register has a range of 4,294,967,298 (2^{32}) LSBs of the register. This covers the travel over one north/south pole pair of the resolver. For the most common 2-pole resolver, this is a full mechanical revolution of the resolver.

For an *n*-pole motor, one mechanical revolution covers *n*/2 commutation cycles of the motor. With the typical 2-pole resolver, one resolver cycle covers these *n*/2 commutation cycles. In the more general case of an n_r-pole resolver on an n_m-pole motor, one resolver cycle covers n_m/n_r resolver cycles.

Therefore, for a 2-pole resolver on a 4-pole motor, the following setting could be used:

Motor[*x*].PhasePosSf = n_m/n_r * 2048/4294967296 = 4/2 * 2048/4294967296 = 1/1048576

Usually it is best to enter this value as an expression and let Power PMAC calculate the (double-precision floating-point) value, which is typically much smaller than 1.0.

Use for Power-On Commutation Position

One of the main advantages of resolvers is that they provide absolute position over a full commutation cycle, so the commutation phase angle can be determined immediately on power-up without the need for a phasing-search move. To utilize this capability in Power PMAC, several saved motor elements must be set up properly.

Motor[*x*].pAbsPhasePos must be set to **Gate3[*i*].Chan[*j*].AtanSumOfSqr.a** to specify the use of the register containing the resolver position. **Motor[*x*].AbsPhasePosFormat** should be set to $00001010 to specify the use of 16 ($10) bits starting at bit 16 ($10) of the register.

Power PMAC reads the selected part of 32-bit register (16 bits in this case) and scales the result to the 2048-part commutation cycle. **Motor[*x*].AbsPhasePosSf** can be calculated as 2048 divided by the number of LSBs of the source register per commutation cycle. One cycle of the resolver position in the selected part of the register has a range of 65,536 (2^{16}) LSBs. This covers the travel over one north/south pole pair of the resolver. For the most common 2-pole resolver, this is a full mechanical revolution of the resolver.

For an *n*-pole motor, one mechanical revolution covers *n*/2 commutation cycles of the motor. With the typical 2-pole resolver, one resolver cycle covers these *n*/2 commutation cycles. In the more general case of an n_r-pole resolver on an n_m-pole motor, one resolver cycle covers n_m/n_r resolver cycles.

Therefore, for a 2-pole resolver on a 4-pole motor, the following setting could be used:

Motor[*x*].AbsPhasePosSf = n_m/n_r * 2048/65536 = 4/2 * 2048/65536 = 1/32

Usually it is best to enter this value as an expression and let Power PMAC calculate the (double-precision floating-point) value, which is typically much smaller than 1.0.

Use for Ongoing Servo Feedback or Master

To process the resulting position value for ongoing servo-loop use (feedback or master), a “Type 1” single-register-read conversion should be specified in the encoder conversion table. **EncTable[*n*].pEnc**, which specifies the address of this register, should be set to

Gate3[*i*].Chan[*j*].AtanSumOfSqr.a. (It could also be set to **Gate3[*i*].Chan[*j*].Atan.a**, using the partial-word element but it will report back as the address of the full-word element.)

EncTable[*n*].pEnc1, which specifies a second source, is not used in this mode; its setting does not matter.

Most users will set up **EncTable[*n*].index1** and **EncTable[*n*].index2** to create a low-pass "tracking filter" to minimize the effect of high-frequency noise on the analog inputs. Refer to the Power PMAC User's Manual chapter on the ECT for details on the setup of this filter. Most users will set **EncTable[*n*].MaxDelta** to 0 to disable any derivative limits.

To get the output of the conversion table in units of an *n*-bit resolver conversion **EncTable[*n*].ScaleFactor** should be set to $½^{32-n}$. For example, to get the units of a 14-bit conversion (most common), with 16,384 states per resolver cycle, **EncTable[*n*].ScaleFactor** should be set to $1/2^{18}$, or 1/262,144. To get the units of a 16-bit conversion (remembering that there is likely to be noise on the lower bits), it should be set to $1/2^{16}$, or 1/65,536. It is best to enter this value as an expression and let Power PMAC calculate the exact numerical value itself.

Use for Power-On Servo Position

A resolver provides absolute position over a commutation cycle or possibly a full motor revolution, but this seldom covers the full travel of a motor in an application, so typically a homing search move is required to establish the position reference.

If the motor's travel is limited to within a single cycle of the resolver, then it can be used for absolute power-on servo position. In this case, **Motor[*x*].pAbsPos**, **Motor[*x*].AbsPosFormat**, **Motor[*x*].AbsPosSf**, and **Motor[*x*].AbsPosOffset** are set just like there power-on commutation position equivalents, explained above (although the resolver position is scaled into motor position units, not commutation cycle units).

If there is a second resolver on the motor mechanically geared down from the primary resolver, this can be used for multi-turn absolute position. This resolver is connected into the ALTSIN+/- and ALTCOS+/- inputs for the channel and the measured values can be read in **Gate3[*i*].Chan[*j*].AdcEnc[2]** and **AdcEnc[3]**. No automatic calculations are performed on these input values, so user application code is necessary to compute the angle of this resolver and to combine its position with that of the primary resolver.

Use for Trigger Position

Direct immediate hardware capture of the resolver position is not possible, because the IC only receives data for a new position once per phase or servo cycle. For triggered moves, **Motor[*x*].CaptureMode** should be set to 1 to specify "software capture" with an input trigger. In this mode, Power PMAC uses the already-processed motor actual position value from the most recent servo cycle as the trigger position. Note that there is a potential delay of up to one real-time-interrupt period (**Sys.RtIntPeriod** + 1 servo-interrupt periods) from the trigger occurrence to the time of the position used to calculate the post-trigger move. This potential delay time should be multiplied by the speed to obtain the uncertainty in the captured position.

The accuracy of the software capture can be improved significantly with Power PMAC's "timer-assisted software capture" feature, which uses a timer value latched by the trigger to interpolate between servo-cycle positions. This mode is specified by setting **Motor[*x*].CaptureMode** to 3.

Several setup elements of the IC channel must be set to drive the timer circuitry properly for this mode of operation. Refer to the *Basic Motor Moves* chapter of the User's Manual for details.

Monitoring for Sensor Loss

The ACC-24E3 has circuitry that can detect loss of signal from sinusoidal inputs such as those from a resolver, and standard Power PMAC algorithms can use this circuitry to automatically disable motors when this loss is detected. The detection circuitry uses the digital "sine" and "cosine" values from the analog-to-digital converters, and computes the sum of squares of these values, which should be roughly constant in an application, and above a minimum threshold for a properly working and connected resolver.

The value of this signal magnitude measurement can be found in the 16-bit element **Gate3[*i*].Chan[*j*].SumOfSquares**. If all 4 of the most significant bits of this element are 0, meaning that the magnitude of the signal is less than 1/16 of the maximum, the status bit **Gate3[*i*].Chan[*j*].SosError**, part of the full-word element **Gate3[*i*].Chan[*j*].Status**, is set to 1.

To employ this circuitry for automatic loss detection, set **Motor[*x*].pEncLoss** to **Gate3[*i*].Chan[*j*].Status.a**. (It can also be set to **Gate3[*i*].Chan[*j*].SosError.a**, but will report back as **Gate3[*i*].Chan[*j*].Status.a**). Set **Motor[*x*].EncLossBit** to 31 to specify the **SosError** bit number in the register. Set **Motor[*x*].EncLossLevel** to 1 to specify a high-true loss status bit.

Because it is possible for signal transients to cause a momentary setting of this, it is usually desirable to set **Motor[*x*].EncLossLimit** to a value greater than 0. The accumulated count of loss detection events must exceed this threshold before a fault is declared. Reasonable values for this element will not unduly delay reaction to true faults.

MLDT Sensors

The ACC-24E3 can be set up for direct interface with "externally excited" Magnetostrictive Linear Displacement Transducers (MLDTs), such as MTS's Temposonics® brand. MLDTs can provide absolute position information in rugged environments; they are particularly well suited to hydraulic applications. In this interface, the ACC-24E3 provides a periodic electronic excitation pulse output to the MLDT, receives the echo pulse that returns at the speed of sound in the transducer, and very accurately measures the time between these pulses, which is directly proportional to the distance of the moving member from the stationary base of the transducer. The timer therefore contains a position measurement. The timer operates at a 600 MHz frequency (1.667 nanosecond period), providing a position resolution of just under 5 micrometers in most of these sensors.

IC Hardware Setup

The excitation pulse is generated using the IC's pulse-frequency modulation (PFM) output on Phase D. The differential pulse is output on what is otherwise would be the T and W flag inputs on the encoder connector. **Gate3[*i*].Chan[*j*].OutFlagD** must be set to 1 to enable the PFM outputs on these pins. This is not a saved value, so this assignment must be made after every power-on/reset.

Bit 3 of **Gate1[*i*].Chan[*j*].OutputMode** must be set to 1 (for a value of 8 to 15 for the element) to put the channel's Phase D in PFM (not PWM) mode. (Other bits control the output format for Phases A, B, and C.)

Gate1[*i*].Chan[*j*].PfmWidth sets the output pulse width (time) for the channel's PFM output in units of the PFM clock period. The PFM clock period is set by **Gate3[*i*].PfmClockDiv**; at the default clock frequency of 3.125 MHz, the clock period is 320 nanoseconds, and with setting for **Gate1[*i*].Chan[*j*].PfmWidth** of 5, the pulse width is about 1.5 microseconds, which is suitable for almost all MLDT interfaces.

To put the channel's PFM and timer circuits in the mode to interface with MLDTs, **Gate3[*i*].Chan[*j*].TimerMode** must be set to 1. In this mode, a PFM pulse is output every servo interrupt (regardless of the value in the PFM command register).

The frequency of the pulse output should produce a period just slightly longer than the longest expected response time for the echo pulse. For MLDTs, the response time is approximately 0.35 µsec/mm (9 µsec/inch). On an MLDT 1500 mm (~60 in) long, the longest response time is approximately 540 µsec. The servo update period must be longer than this (i.e. the frequency must be lower). A recommended period between pulse outputs for this device is 600 µsec, for a frequency of 1.667 kHz.

The "echo" return pulse from the sensor must be connected to the CHA+ and CHA- inputs on the encoder connector. With **Gate3[*i*].Chan[*j*].TimerMode** set to 1 for MLDT mode, the timer value representing the sensor position can be found in the high 24 bits of the 32-bit read-only element **Gate1[*i*].Chan[*j*].TimerA**. The timer is cleared to zero at the falling edge of the output pulse. It then counts up at 600 MHz until a rising edge on the return pulse is received, at which point the timer's value is latched into **Gate1[*i*].Chan[*j*].TimerA** where the processor can read it.

Each increment of the time represents a physical length along the sensor that is determined by the timer frequency and the speed of the return pulse in the MLDT, which is the speed of sound in the metal. This speed varies from device to device, but is always approximately the inverse of 0.35 µsec/mm, or 9.0 µsec/inch. With the timer frequency of 600 MHz, the resolution can be calculated as:

$$\mathrm{Re}\,solution\left(\frac{length}{increment}\right)=\frac{1}{TimerFreq}\left(\frac{\mu\sec}{increment}\right)*\mathrm{Re}\,turnSpeed\left(\frac{length}{\mu\sec}\right)$$

$$\cong\frac{1}{600}\frac{\mu\sec}{increment}*\frac{1}{0.35}\frac{mm}{\mu\sec}\cong 0.00476\frac{mm}{increment}\left(210\frac{increments}{mm}\right)$$

$$\cong\frac{1}{600}\frac{\mu\sec}{increment}*\frac{1}{9.0}\frac{inches}{\mu\sec}\cong 0.000185\frac{inches}{increment}\left(5400\frac{increments}{inch}\right)$$

Use for Ongoing Servo Feedback or Master

To process the timer value latched by the IC on the receipt of the return pulse, a "Type 1" single-register read conversion should be specified in the Encoder Conversion Table. **EncTable[*n*].pEnc**, which specifies the address of this register, should be set to **Gate1[*i*].Chan[*j*].TimerA.a**.

EncTable[*n*].pEnc1, which specifies a second source, is not used in this mode; its setting does not matter.

The position data is found in the high 24 bits of the source register. **EncTable[*n*].index2** should be set to 8 to shift this data right 8 bits, putting the timer LSB in bit 0 of the result. Most users will then shift the data back left, setting **EncTable[*n*].index1** to 8. **EncTable[*n*].ScaleFactor** should be set to 1 / 256 so the output units are in timer increments.

It is strongly recommended that the entry implement a velocity or acceleration limit as a protection against missed or spurious echo pulses. To implement a velocity limit, set **EncTable[*n*].index3** to 0 to specify a velocity (not acceleration) limit, and set **EncTable[*n*].MaxDelta** to a value slightly greater than the greatest true velocity the system will see, expressed in timer units per servo cycle.

To implement an acceleration limit, set **EncTable[*n*].index3** to a value greater than 0 to specify acceleration limit, and set **EncTable[*n*].MaxDelta** to a value slightly greater than the greatest true acceleration the system will see, expressed in timer units per servo cycle per servo cycle.

Set **EncTable[*n*].index4** to 0 so the result is not integrated, and the maximum change filtering can be used..

To use the result of the conversion table entry for outer (position) loop motor feedback, set **Motor[*x*].pEnc** to **EncTable[*n*].a**. To use the result for inner velocity loop motor feedback, set **Motor[*x*].pEnc2** to **EncTable[*n*].a**. To use the result as a master position for the motor's position-following function, set **Motor[*x*].pMasterEnc** to **EncTable[*n*].a**.

Use for Absolute Power-On Position

MLDTs provide absolute position information, but Power PMAC must be specifically set up to use it for this purpose. (Otherwise, it considers the power-on/reset to position to be zero, no matter what the sensor value is.)

Motor[*x*].pAbsPos should be set to **Gate1[*i*].Chan[*j*].TimerA.a** to read the timer value register for absolute power-on position. **Motor[*x*].AbsPosFormat** should be set to $00001808 to read 24 ($18) bits starting at bit 8 ($08) and interpret them as unsigned binary ($00). The scale factor element **Motor[*x*].AbsPosSf** should be set to the same value as the ongoing scale factor element **Motor[*x*].PosSf** (which is often 1.0 so the motor units are timer increments).

Use for Trigger Position

Direct immediate hardware capture of the MLDT position is not possible, because the IC only receives a new position once per phase or servo cycle. For triggered moves, **Motor[*x*].CaptureMode** should be set to 1 to specify "software capture" with an input trigger. In this mode, Power PMAC uses the already-processed motor actual position value from the most recent servo cycle as the trigger position. Note that there is a potential delay of up to one real-time-interrupt period (**Sys.RtIntPeriod** + 1 servo-interrupt periods) from the trigger occurrence to the time of the position used to calculate the post-trigger move. This potential delay time should be multiplied by the speed to obtain the uncertainty in the captured position.

Note that the timer-assisted software capture feature is not readily available to improve the accuracy of the trigger position capture with MLDTs, because the timer circuits are set up in MLDT mode. A separate ASIC channel would need to be used for the timer-assistance feature.

Software Setup for Flags

Each channel on the ACC-24E3 supports multiple input and output "flag" signals. There are basically four sets of flags on each channel, providing different types of functionality:

- Amplifier flags: amplifier-enable output and amplifier-fault input
- Overtravel-limit flags: positive-end limit and negative-end limit
- Position-capture flags: home, limits, and "user" flags; encoder index
- Position-compare output flags: "EQU" outputs

Motor Flag Addresses

Each motor has four "flag address" setup elements that specify which register the motor software will access for particular flag functions. To use flags from a channel of the ACC-24E3, these elements should be set as follows:

Motor[*x*].pAmpEnable = **Gate3[*i*].Chan[*j*].OutCtrl.a**	// Amplifier enable output
Motor[*x*].pAmpFault = **Gate3[*i*].Chan[*j*].Status.a**	// Amplifier fault input
Motor[*x*].pLimits = **Gate3[*i*].Chan[*j*].Status.a**	// Overtravel limit inputs
Motor[*x*].pCaptFlag = **Gate1[*i*].Chan[*j*].Status.a**	// Trigger flag inputs
Motor[*x*].pEncStatus = **Gate3[*i*].Chan[*j*].Status.a**	// Encoder status info

Motor[*x*].pCaptFlag is automatically set to the same value as **Motor[*x*].pEncStatus** on re-initialization, and when **Motor[*x*].EncType** is assigned a value of 5 or 6 in the Script environment to specify use of a PMAC3-style interface channel.

Except for **Motor[*x*].pCaptFlag** and **Motor[*x*].pEncStatus**, setting the address value to 0 disables that flag function. The position compare function is tied directly to the encoder; it does not utilize any automatic motor functions.

It is also necessary to specify which bit number(s) in the specified registers are used for the respective functions. The following settings specify the standard values to use the named flags on an ACC-24E3 for their named functions. These are the default setting made when the Power PMAC is re-initialized using the **`$$$***`** command with an ACC-24E3 present. They are also set automatically when **Motor[*x*].EncType** is assigned a value of 5 or 6 in the Script environment to specify use of a PMAC3-style interface channel.

Motor[*x*].AmpEnableBit = 8	// Amp enable bit in output flag register
Motor[*x*].AmpFaultBit = 7	// Amp fault bit in input flag register
Motor[*x*].CaptFlagBit = 20	// Trigger bit in input flag register
Motor[*x*].LimitBits = 9	// First limit flag bit in input flag register

Flag Polarities

The Power PMAC always writes a value of 0 to the selected amplifier-enable output bit to disable the amplifier and a value of 1 to enable the amplifier. Hardware resets or failures such as watchdog timer trips automatically force values of 0 into the standard amplifier-enable output bits to disable the attached amplifiers. For this reason, the software polarity of the amplifier-enable output is not user-programmable.

The Power PMAC always considers values of 0 in the selected limit input bits to mean that the motor is not in the limit, and values of 1 to mean that the motor has hit the limit. With the opto-isolation circuitry on the ACC-24E3, current must be flowing through the isolators (in either direction) to produce a value of 0 in the register, meaning that normally-closed limit switches (opening on the limit) must be used. Since the most likely circuit failures produce an open circuit, this is the more fail-safe configuration. For this reason, the software polarity of the overtravel-limit inputs is not user-programmable.

Since there is no standardization of the fault outputs of servo amplifiers, the software polarity of the amplifier-fault input to the Power PMAC is user programmable. Power PMAC considers the value in the selected amplifier-fault bit that is equal to **Motor[*x*].AmpFaultLevel** to be the fault state, and the opposite logical level to mean there is no fault.

Error Flag Filtering

The input flags are automatically passed through a digital delay filter that samples them at the encoder sampling clock frequency set by **Gate3[*i*].EncClockDiv** (3.125 MHz default) and performs a best-two-of-three voting each clock cycle to eliminate single-cycle spikes. If the saved setup element **Gate3[*i*].Chan[*j*].FlagFilt2Ena** is set to 1, these flag inputs are also passed through a second digital delay filter operating at the (usually) lower secondary filtering clock frequency set by **Gate3[*i*].FiltClockDiv** before appearing in the memory-mapped register. This secondary filter can be useful in high-noise environments to reject spurious faults on the overtravel-limit and amplifier-fault flags.

Capture Trigger Control

Each channel on the DSPGATE3 IC has a hardware "position-capture" circuit that latches the encoder counter position immediately upon receiving a pre-selected input trigger condition. Three setup elements for the IC channel determine what the position-capture trigger condition will be for the channel. **Gate3[*i*].Chan[*j*].CaptCtrl** specifies whether the channel's encoder index is used to create the trigger or not, and which edge triggers if it is; also whether an input flag is used create the trigger or not, and which edge triggers if it is.

If **CaptCtrl** specifies the use of an input flag, **Gate3[*i*].Chan[*j*].FlagSel** specifies which one of the four types of input flags is used to create the trigger – HOME (= 0), PLIM (= 1), MLIM (= 2), or USER (= 3) – and **Gate3[*i*].Chan[*j*].CaptFlagChan** specifies the index of the channel on the IC the specified flag type comes from (usually the same channel as the encoder; this is the default). However, the ability to specify a different channel's flag to capture this channel's encoder position permits one flag signal to capture the position of multiple encoders, as in a contact probe.

If **Gate3[*i*].Chan[*j*].TimerMode** is set to the default value of 0, the encoder position captured by the trigger will include the fractional count value estimated by the built-in "hardware 1/T" extension. This is true even for sinusoidal encoders, where the fractional count value for servo and phase tasks is calculated by the arctangent function. The captured position value can be found in **Gate3[*i*].Chan[*j*].HomeCapt** with 24 bits of integer count and 8 bits of fraction, and (if **TimerMode** = 0) in **Gate3[*i*].Chan[*j*].TimerB** with 20 bits of integer count and 12 bits of fraction.

This hardware position capture feature can be used automatically in motor "triggered" moves: homing-search moves, jog-until-trigger moves, and program move-until-trigger. These moves end at a preset distance from the position captured by the trigger.

Position-Compare Outputs

Each channel on the DSPGATE3 IC has a "position-compare" circuit for its incremental encoder interface, in which the present encoder count value is compared every encoder sample clock cycle (3.125 MHz default, up to 100 MHz, as set by **Gate3[*i*].EncClockDiv**) to user-set values in the **Gate3[*i*].Chan[*j*].CompA** and **Gate3[*i*].Chan[*j*].CompB** elements. If the count value has matched or passed one of these values in the most recent encoder sample clock cycle, the internal compare state for the channel is toggled. The value of the channel's internal compare state can be monitored in the 1-bit status element **Gate3[*i*].Chan[*j*].Equ**. This compare function can be used to trigger external events very quickly and precisely when a position is reached.

These 32-bit "compare" registers have 12 bits of fractional count resolution, and so have units of 1/4096 of a quadrature count. If **Gate3[*i*].Chan[*j*].TimerMode** for the encoder channel used (see below) is set to the default value of 0, the IC will be calculating a timer-based estimate of the fractional-count position every encoder sample clock cycle, and using this estimate in the comparison to the **CompA** and **CompB** values. This is true even for sinusoidal encoders, where the fractional count value for servo and phase tasks is calculated by the arctangent function. This capability allows the compare outputs to be triggered at resolutions much finer than a quadrature count.

The present value of the counter can be found in **Gate3[*i*].Chan[*j*].ServoCapt**. For a digital encoder, this value has 8 bits of fractional count (so appears a factor of 16 smaller than **CompA** and **CompB**). For an analog encoder (with **Gate3[*i*].Chan[*j*].AtanEna** = 1), this value as 12 bits of fractional count, so is scaled the same as **CompA** and **CompB**. In both cases, if **Gate3[*i*].Chan[*j*].TimerMode** is 0, the element **Gate3[*i*].Chan[*j*].TimerA** contains the most recent position with 12 bits of timer-estimated fractional count, so scaled the same as **CompA** and **CompB**.

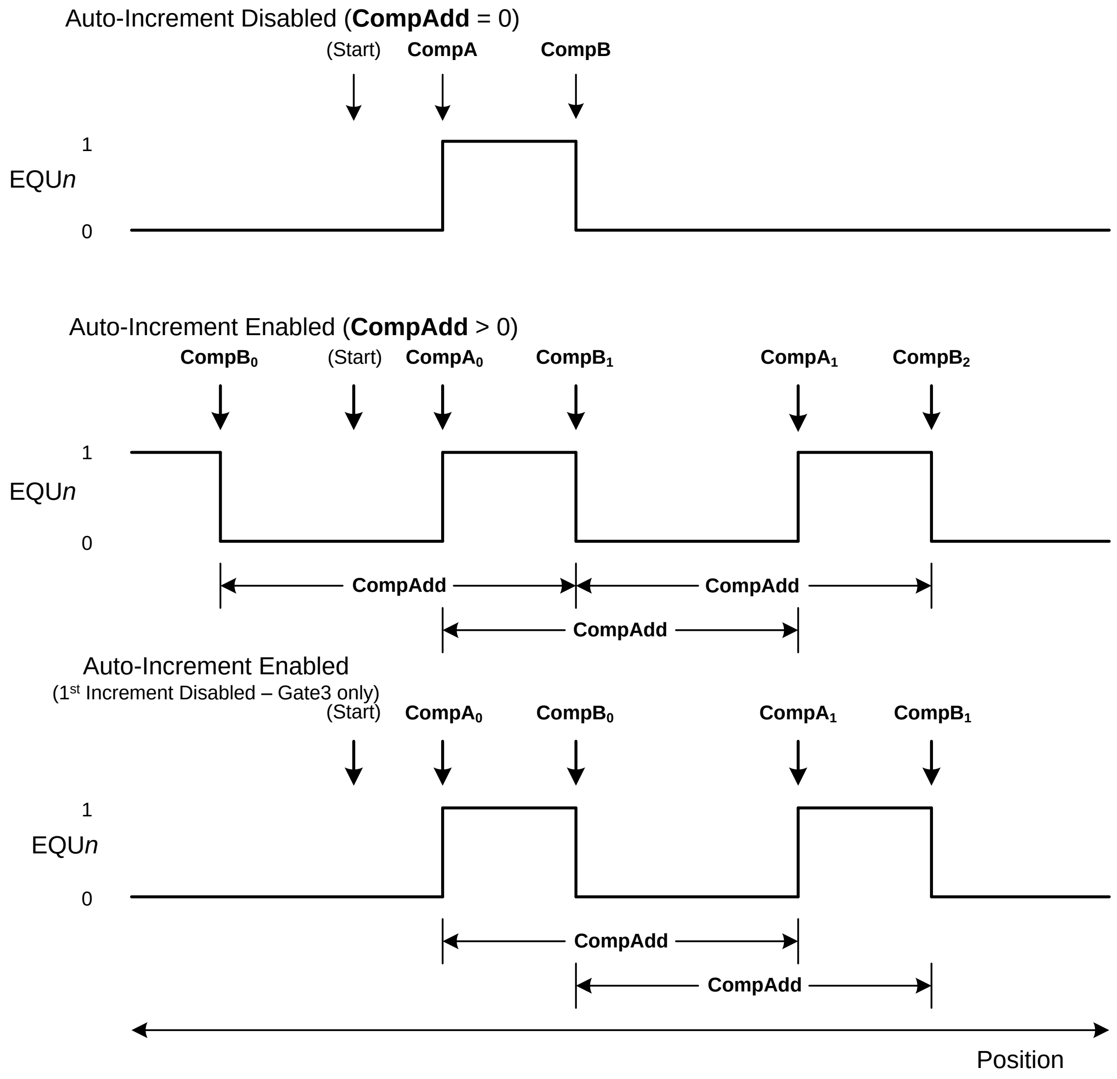
If **Gate3[*i*].Chan[*j*].Equ1Ena** is set to the default value of 0, the channel's position-compare circuit uses the same channel's encoder count value. However, if it is set to 1, it uses the encoder count value of the first encoder channel on the IC (**Chan[0]**). This permits sophisticated functions such as the automatic "windowing" of compare outputs.

The internal compare states for the channels on the IC can be logically processed before being output on the EQU*n* lines. Each channel's compare output signal can use 1 to 4 of the IC's internal compare states, as determined by the 4-bit mask in **Gate3[*i*].Chan[*j*].EquOutMask** as inputs to a logical AND function (the default is to use only the same channel's internal compare state). This logical combination can then be inverted or not as determined by the 1-bit element **Gate3[*i*].Chan[*j*].EquOutPol**. The value of the channel's compare signal output can be monitored in the 1-bit status element **Gate3[*i*].Chan[*j*].EquOut**.

The position values in the **CompA** and **CompB** registers can be "auto-incremented" if the value of **Gate3[*i*].Chan[*j*].CompAdd** is set to a non-zero value. The 32-bit **CompAdd** value also has 12 bits of fractional count data, for units of 1/4096 of a quadrature count. As the position in the **CompA** register is passed and the compare state toggled, the (signed) value of **CompAdd** is automatically added to **CompB**, and as the position in the **CompB** register is passed, the value of **CompAdd** is automatically added to **CompA**.

The present value of the internal compare state for a channel can be set by writing to the 2-bit element **Gate3[*i*].Chan[*j*].EquWrite**. Bit 1 (value 2) of this element is the value to be forced into the internal compare state for the channel. Bit 0 (value 1) is the forcing flag; after the value has been forced, this bit is automatically reset to 0 for confirmation. So, to force a value of 1 into the internal compare state, a value of 3 should be written into **EquWrite** (which will end up as a value of 1). To force a value of 0 into the internal compare state, a value of 1 should be written into **EquWrite** (which will end up as a value of 0).

The following diagram shows the position-compare waveforms that can be generated with and without auto-increment, and with and without inhibiting the first auto-increment.



Position Compare State Waveforms

Forcing a value into the internal compare state through **EquWrite** inhibits the auto-increment operation on the first compare event that follows. This means that the initial **CompA** and **CompB** values set to start a sequence of compare pulses do not have to “bracket” the present position, making it much easier to specify a sequence while the encoder is counting. Writing to either **CompA** or **CompB** re-enables the first auto-increment operation, requiring that the initial values “bracket” the present position, but permitting operation in either direction from the starting point.

Software Setup for Amplifiers

Each channel on an ACC-24E3 can support multiple different types of amplifier interfaces. Some will require the analog amplifier-interface mezzanine board, and some will require the digital amplifier-interface mezzanine board. All require that some saved setup data structure elements be configured properly in software.

Analog Velocity and Torque-Mode Amplifiers

The analog amplifier-interface mezzanine board can be used to create a +/-10V analog servo output commands for velocity-mode and torque-mode amplifiers. This can be done whether the board is configured with one or two DACs per axis (although if the second DAC is present, it will not be used for this purpose). This board has high-precision resistor-bridge DACs with serial data inputs. To interface properly with these circuits, several settings must be made properly.

DAC Signal Setup

To obtain the required output signals from the IC channel for analog outputs on Phase A or B, both Phase A and B outputs must be put in DAC mode. This means that both bits 0 and 1 of **Gate3[*i*].Chan[*j*].OutputMode** must be 1, making the resulting value 3, 7, 11, or 15, depending on the desired output signals for Phase C or D (if any). The ACC-24E3 DAC circuitry requires "non-inverted" signals from Phase A and B, so bit 0 of **Gate3[*i*].Chan[*j*].OutputPol** must be set to 0, making the resulting value 0 or 2, depending on the inversion for Phase D.

When the standard 16-bit DACs are used, **Gate3[*i*].DacStrobe** must be set to $7FFF0000. The motor element **Motor[*x*].DacShift** must be set to the default value of 0. The value of **Gate3[*i*].Chan[*j*].PackOutData** does not matter in this case, as the command value in the high 16 bits of **Gate3[*i*].Chan[*j*].Dac[0]** (for Phase A) is used in the same way in packed and unpacked mode.

When the optional 18-bit DACs are used, **Gate3[*i*].DacStrobe** must be set to $7FFFFF00, as these DACs use a 24-bit field. The motor element **Motor[*x*].DacShift** must be set to 6 to shift the 18-bit data properly within the 24-bit field. **Gate3[*i*].Chan[*j*].PackOutData** should be set to 0 to keep 2 16-bit values from being "packed" into a single 32-bit register, overwriting the low bits of the 18-bit DAC.

Motor Addressing and Mode Setup

The motor element **Motor[*x*].pDac** should be set to **Gate3[*i*].Chan[*j*].Dac[0].a** to specify the use of the channel's Phase A analog output (usual), or to **Gate3[*i*].Chan[*j*].Dac[1].a** to specify the use of the channel's Phase B analog output for the velocity or torque command. Note that when the value of **Motor[*x*].pDac** is queried, it will be reported as **Gate3[*i*].Chan[*j*].Pwm[*k*].a** because the address of the PWM register is the same as that of the DAC register.

Motor[*x*].PhaseCtrl should be set to 0 to disable all phase-interrupt tasks, as the motor commutation is done in the amplifier or the motor in these modes, or to 8 if the servo loop is to be closed in the high-frequency phase interrupt, as may be useful for some certain special high-bandwidth actuators, such as galvanometers and fast-tool servos.

Analog Sine-Wave Amplifiers

The analog amplifier-interface mezzanine board can be used to create dual +/-10V analog servo output commands for “sine-wave input” amplifiers if it is configured with two DACs per axis. This type of amplifier is used when Power PMAC is performing the commutation for the motor, but not the current-loop closure.

DAC Signal Setup

The analog amplifier mezzanine board has high-precision resistor-bridge DACs with serial data inputs. To interface properly with these circuits, several settings must be made correctly.

To obtain the required output signals from the IC channel for analog outputs on Phase A and B, both Phase A and B outputs must be put in DAC mode. This means that both bits 0 and 1 of **Gate3[*i*].Chan[*j*].OutputMode** must be 1, making the resulting value 3, 7, 11, or 15, depending on the desired output signals for Phase C or D (if any). The ACC-24E3 DAC circuitry requires “non-inverted” signals from Phase A and B, so bit 0 of **Gate3[*i*].Chan[*j*].OutputPol** must be set to 0, making the resulting value 0 or 2, depending on the inversion for Phase D.

When the standard 16-bit DACs are used, **Gate3[*i*].DacStrobe** must be set to $7FFF0000. The motor element **Motor[*x*].DacShift** must be set to the default value of 0. The value of **Gate3[*i*].Chan[*j*].PackOutData** does not matter in this case, as the command values in the high 16 bits of **Gate3[*i*].Chan[*j*].Dac[0]** and **Dac[1]** (for Phases A and B) are used in the same way in packed and unpacked mode.

When the optional 18-bit DACs are used, **Gate3[*i*].DacStrobe** must be set to $7FFFFF00, as these DACs use a 24-bit data field. The motor element **Motor[*x*].DacShift** must be set to 6 to shift the 18-bit usable data properly within the 24-bit field. **Gate3[*i*].Chan[*j*].PackOutData** must be set to 0 to keep 2 16-bit values from being “packed” into a single 32-bit register, overwriting the low bits of the 18-bit DAC.

Motor Addressing and Mode Setup

The motor element **Motor[*x*].pDac** should be set to **Gate3[*i*].Chan[*j*].Dac[0].a** to specify the use of the channel’s Phase A analog output for the first phase and the channel’s Phase B analog output for the second phase. Note that when the value of **Motor[*x*].pDac** is queried, it will be reported as **Gate3[*i*].Chan[*j*].Pwm[0].a** because the address of the PWM register is the same as that of the DAC register.

Motor[*x*].PhaseCtrl should be set to 1 for Power PMAC to perform the commutation tasks with the command output values combined into a single 32-bit register, as the IC expects when **Gate3[*i*].Chan[*j*].PackOutData** is set to 1 to enable this packing.

Motor[*x*].PhaseCtrl should be set to 4 for Power PMAC to perform the commutation tasks with the command output values written to separate registers, as the IC expects when **Gate3[*i*].Chan[*j*].PackOutData** is set to 0 to disable packing two command values into a single register.

The motor element **Motor[*x*].pAdc** must be set to 0 in this mode to disable operation of the motor current loop in Power PMAC.

Direct-PWM Amplifiers

The digital amplifier mezzanine board can be used to create the multi-phase PWM outputs for direct-PWM "power block" amplifiers. This type of amplifier is used when Power PMAC is performing both the phase commutation for the motor and the current-loop closure. The PWM output for each motor phase represents the voltage command for that phase that is required to get the desired current level.

(Note that for direct-PWM control of brush DC motors, the Power PMAC's commutation algorithm is activated, but with settings that defeat the AC nature of the commutation, as the true commutation is done in the motor. Enabling the Power PMAC commutation algorithm permits the current-loop algorithms to be performed for the motor.)

PWM Signal Setup

To obtain the required output signals from the IC channel for PWM outputs on Phases A, B, and C, these phase outputs must be put in PWM mode. This means that bits 0, 1, and 2 of **Gate3[*i*].Chan[*j*].OutputMode** must be 0, making the resulting value 0 or 8, depending on the desired output signals for Phase D (if any). Most direct-PWM amplifiers require "non-inverted" signals from Phases A, B, and C, so bit 0 of **Gate3[*i*].Chan[*j*].OutputPol** should be set to 0, making the resulting value 0 or 2, depending on the inversion for Phase D.

The PWM frequency for the channel is specified by **Gate3[*i*].Chan[*j*].PwmFreqMult**, which can take a value *n* from 0 to 7, setting the channel's PWM frequency to (*n* + 1) / 2 times the internal phase-clock frequency for the IC.

The PWM "dead time" for the channel (the time between the start of turn-off of one of the phase transistors and the start of turn-on of the other) is specified by **Gate3[*i*].Chan[*j*].PwmDeadTime**, in units of 53.33 nanoseconds. The optimal setting is dependent on the amplifier requirements.

It is strongly recommended that **Gate3[*i*].Chan[*j*].PackOutData** be set to 1 so that Power PMAC can write the 16-bit Phase A and C PWM commands into a single 32-bit register (and the Phase B and D PWM commands into a single register as well), reducing the number of slow I/O accesses every phase cycle.

ADC Signal Setup

The bit clock frequency for the ADCs is set by **Gate3[*i*].AdcAmpClockDiv**. Usually a value of 4 (for 6.25 MHz) or 5 (for 3.125 MHz) is used. The user should select the highest frequency that the amplifier's ADCs can accept. This frequency is common for all four channels on the IC.

The element **Gate3[*i*].AdcAmpStrobe** specifies the "strobe word" for the amplifier ADCs on all channels of the IC. Generally a value of $FFFFFE is used, but this may vary depending on the amplifier.

The element **Gate3[*i*].AdcAmpHeaderBits** tells the IC how many "header" bits precede the true ADC numerical data in the returned serial data stream. Most current-feedback ADCs in power-block amplifiers have 1 header bit.

The element **Gate3[*i*].AdcAmpUtoS** should be set to the default value of 0 for all presently known power-block amplifiers, because they all use ADCs that provide the data in the needed signed numerical format. A value of 1 would convert data from an unsigned ADC into signed format.

The element **Gate3[*i*].AdcAmpDelay** is almost always set to the default value of 0. Non-zero values permit a slight delay in the strobing of the ADCs to compensate for delays in the PWM outputs.

It is strongly recommended that **Gate3[*i*].Chan[*j*].PackInData** be set to 1 so that Power PMAC can read the 16-bit Phase A and B current-feedback values from a single 32-bit register, reducing the number of slow I/O accesses every phase cycle. This must be set to the same value as **Gate3[*i*].Chan[*j*].PackOutData**.

Motor Addressing and Mode Setup

The motor element **Motor[*x*].pDac** should be set to **Gate3[*i*].Chan[*j*].Pwm[0].a** to specify the use of the channel's Phase A PWM output for the first phase, the channel's Phase B PWM output for the second phase, the channel's Phase C PWM output for the third phase, and the channel's Phase D PWM output for the fourth phase (if any).

The motor element **Motor[*x*].pAdc** should be set to **Gate3[*i*].Chan[*j*].AdcAmp[0].a** to specify the use of the channel's two primary amplifier ADC registers.

Motor[*x*].PhaseCtrl should be set to 1 for Power PMAC to perform the commutation tasks with pairs of command output values and current-feedback values combined into single 32-bit registers, as the IC expects when **Gate3[*i*].Chan[*j*].PackOutData** and **Gate3[*i*].Chan[*j*].PackInData** are set to 1 to enable this packing.

If **Gate3[*i*].Chan[*j*].PackOutData** and **Gate3[*i*].Chan[*j*].PackInData** are set to 0 to disable this packing, **Motor[*x*].PhaseCtrl** should be set to 4 so Power PMAC performs the commutation tasks using separate registers for each phase's output and feedback.

Pulse-and-Direction Amplifiers

Pulse-and-direction outputs can be obtained either on the digital amplifier-interface mezzanine board using the Phase D outputs or on the digital feedback mezzanine board using the auxiliary flag (T, U, V, and W) lines. To enable output of these signals on the digital feedback mezzanine board, the element **Gate3[*i*].Chan[*j*].OutFlagD** must be set to 1. This is not a saved element, so it must be set to 1 after every power-on or reset – this is usually done in a "power-on PLC program".

Pulse-and-direction outputs are typically used for traditional stepper motor drives where the motors operate open loop. These outputs can also be used for "stepper-replacement" servo drives, where the position loop is closed in the drive. In both cases, the Power PMAC motor commanding the pulse output will require a simulated feedback to close a loop and ensure that the correct number of pulses have been generated.

IC Hardware Setup

To obtain signals from the pulse-frequency modulation (PFM) circuitry on the Phase D outputs of the IC, bit 3 (value 8) of **Gate3[*i*].Chan[*j*].OutputMode** must be set to 1, making the value of the 4-bit element 8 or higher (depending on the desired mode – if any – for Phases A, B, and C).

Gate3[*i*].Chan[*j*].PackOutData must be set to 0 (not the default) so that the Phase D output comes from the Phase D register **Gate3[*i*].Chan[*j*].Pfm**, and not the low 16 bits of the Phase B register. This permits the full use of the 24 active bits of the **Pfm** register (which appear in the high 24 bits of the 32-bit bus).

Gate3[*i*].Chan[*j*].PfmFormat must be set to the default value of 0 so that the PFM signals are output in pulse-and-direction (and not quadrature) format.

Generally, bit 1 of **Gate3[*i*].Chan[*j*].OutputPol** is set to the default value of 0 for high-true pulses on the PULSE+ line. The polarity can be inverted either by using the PULSE- line or by setting this bit to 1.

The single-bit element **Gate3[*i*].Chan[*j*].PfmDirPol** can be used to invert the sense of the direction output in software.

There are several possibilities for providing the feedback value for Power PMAC to close a position loop and ensure that the correct number of pulses have been output. In the preferred method, **Gate3[*i*].Chan[*j*].TimerMode** should be set to 3 to use the channel's timer as a pulse counter for the generated pulse train. In this mode, the accumulated pulse count can be read in **Gate3[*i*].Chan[*j*].TimerA**.

Note that this setup leaves the channel's encoder circuitry available to process a real encoder, which could be used for confirmation of position. It is not recommended to use real encoder feedback to close the loop in this mode of operation.

In an alternate method, **Gate3[*i*].Chan[*j*].EncCtrl** is set to 8 and the pulse train is fed into the channel's encoder counter. In this mode, **Gate3[*i*].Chan[*j*].ServoCapt** contains the accumulated pulse count in units of 1/256 count (8 bits of fractional count information). If **Gate3[*i*].Chan[*j*].TimerMode** is set to its default value of 0, the low 8 bits will contain a timer-based fractional count estimation, which can provide slightly smoother trajectories, but may lead to dithering at rest. If **TimerMode** is set to 2 or 3, there is no fractional count estimation (the fractional component is fixed at ½ count). Note that in this method, the encoder decode and count circuitry for the channel is not available for use with an external encoder.

Encoder Conversion Table Setup

To process the resulting position value in the preferred method for servo-loop feedback, a "Type 1" single-register-read conversion should be specified in the encoder conversion table. **EncTable[*n*].pEnc**, which specifies the address of this register, should be set to **Gate3[*i*].Chan[*j*].TimerA.a**. (**EncTable[*n*].pEnc1**, which specifies a second source, is not used in this mode; its setting does not matter.)

In the alternate method, **EncTable[*n*].pEnc** should be set to **Gate3[*i*].Chan[*j*].ServoCapt.a** to use the latched encoder counter value

Conversion parameters **EncTable[*n*].index1**, **.index2**, **.index3**, **.index4**, and **.MaxDelta** are generally all set to 0 to use this register.

To get the output of the conversion table in units of pulses (most common) in the preferred method, **EncTable[*n*].ScaleFactor** should be set to 1.0. In the alternate method using the **ServoCapt** register, it should be set to 1/256 to get the output in units of counts.

Motor Addressing and Mode Setup

The motor element **Motor[*x*].pDac** should be set to **Gate3[*i*].Chan[*j*].Pfm.a** to specify the use of the channel's Phase D PFM output.

Motor[*x*].pEnc and **Motor[*x*].pEnc2** should be set to **EncTable[*n*].a**, where "*n*" is the index of the table entry that processed the **TimerA** register used as a pulse counter.

It is recommended that the following servo gain values be used to close a simulated loop:

- **Motor[*x*].Servo.Kp** = 40
- **Motor[*x*].Servo.Kvfb** = 0
- **Motor[*x*].Servo.Kvff** = 40
- **Motor[*x*].Servo.Ki** = 0.001

Many users will want to create a half count of deadband in the servo loop to prevent dithering between adjacent steps at rest if the command value stopped in between them. To do this, set **Motor[*x*].BreakPosErr** to 0.5 (if the motor units are equal to pulses), and **Motor[*x*].Kbreak** to 0.0.

Connector Pinouts

3U-Format Base Board (604002)

P1 UBUS32 Backplane Connector Board

This connector provides the interface between the ACC-24E3 and the UBUS32 backplane that connects the board to the CPU. It is automatically connected when the ACC-24E3 is installed properly in the UMAC rack. In general, the user will not need to know what signals are present on this connector; this listing is provided primarily for reference.

Pin #	Row A	Row B	Row C
1	+5V	+5V	+5V
2	GND	GND	GND
3	BD09	BD00	BD08
4	BD11	BD01	BD10
5	BD13	BD02	BD12
6	BD15	BD03	BD14
7	BD17	BD04	BD16
8	BD19	BD05	BD18
9	BD21	BD06	BD20
10	BD23	BD07	BD22
11	BD25	(*reserved*)	BD24
12	BD27	(*reserved*)	BD26
13	BD29	(*reserved*)	BD28
14	BD31	BCSDIR	BD30
15	(*reserved*)	BCS0-	(*reserved*)
16	BA01	BCS1-	BA00
17	BA04	BCS5-	BA02
18	BA03	BA15	BA05
19	BCS3-	BA07	BCS2-
20	BA06	BA08	BCS4-
21	BCS12-	BA09	BCS10-
22	BCS16-	BA10	BCS14-
23	BA14	BA11	BA13
24	BRD-	BA12	BWR-
25	(*reserved*)	DPRCS1-	(*reserved*)
26	WAIT-	VMECS1-	BRESET
27	PHASE+	UMAC_INT-	SERVO+
28	PHASE-	INT1-	SERVO-
29	AGND	INT2-	AGND
30	A-15V	PWM_ENA	A+15V
31	GND	GND	GND
32	+5V	+5V	+5V

Notes:

1. These signals are provided primarily for reference, as this is typically an “internal” connector inside the system without direct user access.
2. “B” as the first letter means “buffered”

TB1 (Front) First Channel Input Flags

This connector is found in the middle of the front edge of the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot. Pin 1 is at the bottom of the connector.

Pin #	Symbol	Function	Description	Notes
1	USER1	Input	Gen. purpose capture flag	Sourcing or sinking, isolated
2	PLIM1	Input	Positive limit flag	Sourcing or sinking, isolated
3	MLIM1	Input	Negative limit flag	Sourcing or sinking, isolated
4	HOME1	Input	Home capture flag	Sourcing or sinking, isolated
5	FL_RT1	Return	Return for all flags	+V (12-24V) or 0V

TB2 (Front) Second Channel Input Flags

This connector is found at the bottom of the front edge of the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot. Pin 1 is at the bottom of the connector.

Pin #	Symbol	Function	Description	Notes
1	USER2	Input	Gen. purpose capture flag	Sourcing or sinking, isolated
2	PLIM2	Input	Positive limit flag	Sourcing or sinking, isolated
3	MLIM2	Input	Negative limit flag	Sourcing or sinking, isolated
4	HOME2	Input	Home capture flag	Sourcing or sinking, isolated
5	FL_RT2	Return	Return for all flags	+V (12-24V) or 0V

TB3 (Front) Position-Compare Output Flags

This connector is found at the top of the front edge of the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot. Pin 1 is at the bottom of the connector.

Pin #	Symbol	Function	Description	Notes
1	GND	Common	Digital reference voltage	
2	EQU1	Output	Position compare flag	Sinking, internal 330-ohm pullup to 5V
3	EQU2	Output	Position compare flag	Sinking, internal 330-ohm pullup to 5V

3U-Format Piggyback Board (604002)

TB1 (Front) Third Channel Input Flags

This connector is found in the middle of the front edge of the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot. Pin 1 is at the bottom of the connector.

Pin #	Symbol	Function	Description	Notes
1	USER3	Input	Gen. purpose capture flag	Sourcing or sinking, isolated
2	PLIM3	Input	Positive limit flag	Sourcing or sinking, isolated
3	MLIM3	Input	Negative limit flag	Sourcing or sinking, isolated
4	HOME3	Input	Home capture flag	Sourcing or sinking, isolated
5	FL_RT3	Return	Return for all flags	+V (12-24V) or 0V

TB2 (Front) Fourth Channel Input Flags

This connector is found at the bottom of the front edge of the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot. Pin 1 is at the bottom of the connector.

Pin #	Symbol	Function	Description	Notes
1	USER4	Input	Gen. purpose capture flag	Sourcing or sinking, isolated
2	PLIM4	Input	Positive limit flag	Sourcing or sinking, isolated
3	MLIM4	Input	Negative limit flag	Sourcing or sinking, isolated
4	HOME4	Input	Home capture flag	Sourcing or sinking, isolated
5	FL_RT4	Return	Return for all flags	+V (12-24V) or 0V

TB3 (Front) Position-Compare Output Flags

This connector is found at the top of the front edge of the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot. Pin 1 is at the bottom of the connector.

Pin #	Symbol	Function	Description	Notes
1	GND	Common	Digital reference voltage	
2	EQU3	Output	Position compare flag	Sinking, internal 330-ohm pull-up to 5V
3	EQU4	Output	Position compare flag	Sinking, internal 330-ohm pull-up to 5V

First Digital Feedback Mezzanine Board (604003) Terminal Block Version

TB1 (Top) First Channel Encoder Input: 12-Point Terminal Block

This connector is found at the front of the top edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot. Pin 1 of the terminal block is closest to the back of the rack.

This connector is compatible with encoder connectors on ACC-24E2x boards. A cable properly wired for one of those boards can be plugged into this connector without modification.

Pin #	Symbol	Function	Description	Notes
1	CHA1+ AENA1+	Input Output	A positive quadrature Amplifier enable positive	Also PULSE+ input For stepper drives
2	CHA1- AENA1-	Input Output	A negative quadrature Amplifier enable negative	Also PULSE- input For stepper drives
3	CHB1+ FAULT1+	Input Input	B positive quadrature Amplifier fault positive	Also DIR+ input From stepper drives
4	CHB1- FAULT1-	Input Input	B negative quadrature Amplifier fault negative	Also DIR- input From stepper drives
5	CHC1+ SENCENA1+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
6	CHC1- SENCENA1-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
7	+5V	Output	Digital supply voltage	Power for encoder
8	GND	Common	Digital reference voltage	Return from encoder
9	CHU1 DIR1+ SENCCLK1+	Input Output Output	U Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
10	CHV1 DIR1- SENCCLK1-	Input Output Output	V Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
11	CHW1 PULSE1+ SENCDAT1+	Input Output Bidirect.	W Hall sensor signal PFM output pulse Serial encoder data	For commutation angle For stepper control For multiple protocols
12	CHT1 PULSE1- SENCDAT1-	Input Output Bidirect.	GP input flag PFM output pulse Serial encoder data	Can be for overtemp For stepper control For multiple protocols

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for CHA*n*+ and CHA*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the CHB*n*+ and CHB*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -108 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

5. To use the CHA*n*+/- and CHB*n*+/- quadrature inputs as PULSE*n*+/- and DIR*n*+/- inputs, the saved software data structure element **Gate3[*i*].Chan[*j*].EncCtrl** that specifies how these signals are decoded must be set to 0 or 4. For the standard quadrature decode, this element must be set to one of the quadrature-decode values – usually 3 or 7. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

TB2 (Top) Second Channel Encoder Input: 12-Point Terminal Block

This connector is found at the back of the top edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot. Pin 1 of the terminal block is closest to the back of the rack.

This connector is compatible with encoder connectors on ACC-24E2x boards. A cable properly wired for one of those boards can be plugged into this connector without modification.

Pin #	Symbol	Function	Description	Notes
1	CHA2+ AENA2+	Input Output	A positive quadrature Amplifier enable positive	Also PULSE+ input For stepper drives
2	CHA2- AENA1-	Input Output	A negative quadrature Amplifier enable negative	Also PULSE- input For stepper drives
3	CHB2+ FAULT2+	Input Input	B positive quadrature Amplifier fault positive	Also DIR+ input From stepper drives
4	CHB1- FAULT1-	Input Input	B negative quadrature Amplifier fault negative	Also DIR- input From stepper drives
5	CHC2+ SENCENA2+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
6	CHC2- SENCENA2-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
7	+5V	Output	Digital supply voltage	Power for encoder
8	GND	Common	Digital reference voltage	Return from encoder
9	CHU2 DIR2+ SENCCLK2+	Input Output Output	U Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
10	CHV2 DIR2- SENCCLK2-	Input Output Output	V Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
11	CHW2 PULSE2+ SENCDAT2+	Input Output Bidirect.	W Hall sensor signal PFM output pulse Serial encoder data	For commutation angle For stepper control For multiple protocols
12	CHT2 PULSE2- SENCDAT2-	Input Output Bidirect.	GP input flag PFM output pulse Serial encoder data	Can be for overtemp For stepper control For multiple protocols

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for CHA*n*+ and CHA*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the CHB*n*+ and CHB*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0.

Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".Starting with revision -108 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

5. To use the CHA*n*+/- and CHB*n*+/- quadrature inputs as PULSE*n*+/- and DIR*n*+/- inputs, the saved software data structure element **Gate3[*i*].Chan[*j*].EncCtrl** that specifies how these signals are decoded must be set to 0 or 4. For the standard quadrature decode, this element must be set to one of the quadrature-decode values – usually 3 or 7. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

First Digital Feedback Mezzanine Board (604003) D-Sub Version

J1 (Top) First Channel Encoder Input: 15-Pin D-Sub Connector

This connector is found at the front of the top edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin #	Symbol	Function	Description	Notes
1	CHT1 PULSE1- SENCDAT1-	Input Output Bidirect.	GP input flag PFM output pulse Serial encoder data	Can be for overtemp For stepper control For multiple protocols
2	CHV1 DIR1- SENCCLK1-	Input Output Output	V Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
3	GND	Common	Digital reference voltage	Return from encoder
4	CHC1- SENCENA1-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
5	CHB1- FAULT1-	Input Input	B negative quadrature Amplifier fault negative	Also DIR- input From stepper drives
6	CHA1- AENA1-	Input Output	A negative quadrature Amplifier enable negative	Also PULSE- input For stepper drives
7	N.C.		No Connect	Reserved for future use
8	N.C.		No Connect	Reserved for future use
9	CHW1 PULSE1+ SENCDAT1+	Input Output Bidirect.	W Hall sensor signal PFM output pulse Serial encoder data	For commutation angle For stepper control For multiple protocols
10	CHU1 DIR1+ SENCCLK1+	Input Output Output	U Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
11	+5V	Output	Digital supply voltage	Power for encoder
12	CHC1+ SENCENA1+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
13	CHB1+ FAULT1+	Input Input	B positive quadrature Amplifier fault positive	Also DIR+ input From stepper drives
14	CHA1+ AENA1+	Input Output	A positive quadrature Amplifier enable positive	Also PULSE+ input For stepper drives
15	N.C.		No Connect	Reserved for future use

```
  _________________
8 \ ● ● ● ● ● ● ● ● ● /  1
15 \● ● ● ● ● ● ● ● /   9
```

Controller/Encoder Connectors
15-pin D-Sub Receptacle

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for CHA*n*+ and CHA*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the CHB*n*+ and CHB*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -108 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

5. To use the CHA*n*+/- and CHB*n*+/- quadrature inputs as PULSE*n*+/- and DIR*n*+/- inputs, the saved software data structure element **Gate3[*i*].Chan[*j*].EncCtrl** that specifies how these signals are decoded must be set to 0 or 4. For the standard quadrature decode, this element must be set to one of the quadrature-decode values – usually 3 or 7. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

J2 (Top) Second Channel Encoder Input: 15-Pin D-Sub Connector

This connector is found at the back of the top edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin #	Symbol	Function	Description	Notes
1	CHT2 PULSE2- SENCDAT2-	Input Output Bidirect.	GP input flag PFM output pulse Serial encoder data	Can be for overtemp For stepper control For multiple protocols
2	CHV2 DIR2- SENCCLK2-	Input Output Output	V Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
3	GND	Common	Digital reference voltage	Return from encoder
4	CHC2- SENCENA2-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
5	CHB2- FAULT2-	Input Input	B negative quadrature Amplifier fault negative	Also DIR- input From stepper drives
6	CHA2- AENA2-	Input Output	A negative quadrature Amplifier enable negative	Also PULSE- input For stepper drives
7	N.C.		No Connect	Reserved for future use
8	N.C.		No Connect	Reserved for future use
9	CHW2 PULSE2+ SENCDAT2+	Input Output Bidirect.	W Hall sensor signal PFM output pulse Serial encoder data	For commutation angle For stepper control For multiple protocols
10	CHU2 DIR2+ SENCCLK2+	Input Output Output	U Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
11	+5V	Output	Digital supply voltage	Power for encoder
12	CHC2+ SENCENA2+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
13	CHB2+ FAULT2+	Input Input	B positive quadrature Amplifier fault positive	Also DIR+ input From stepper drives
14	CHA2+ AENA2+	Input Output	A positive quadrature Amplifier enable positive	Also PULSE+ input For stepper drives
15	N.C.		No Connect	Reserved for future use

```
   _________________
8 \ ● ● ● ● ● ● ● ● ● /  1
15 \● ● ● ● ● ● ● ● /   9
```

Controller/Encoder Connectors
15-pin D-Sub Receptacle

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for CHA*n*+ and CHA*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the CHB*n*+ and CHB*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must

be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value “*j*” is one less than the corresponding hardware signal channel number “*n*”.

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value “*j*” is one less than the corresponding hardware signal channel number “*n*”. Starting with revision -108 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

5. To use the CHA*n*+/- and CHB*n*+/- quadrature inputs as PULSE*n*+/- and DIR*n*+/- inputs, the saved software data structure element **Gate3[*i*].Chan[*j*].EncCtrl** that specifies how these signals are decoded must be set to 0 or 4. For the standard quadrature decode, this element must be set to one of the quadrature-decode values – usually 3 or 7. Remember that the channel index value “*j*” is one less than the corresponding hardware signal channel number “*n*”.

Second Digital Feedback Mezzanine Board (604003) Terminal Block Version

TB1 (Top) Third Channel Encoder Input: 12-Point Terminal Block

This connector is found at the front of the top edge of the mezzanine board mounted on the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot. Pin 1 of the terminal block is closest to the back of the rack.

This connector is compatible with encoder connectors on ACC-24E2x boards. A cable properly wired for one of those boards can be plugged into this connector without modification.

Pin #	Symbol	Function	Description	Notes
1	CHA3+ AENA3+	Input Output	A positive quadrature Amplifier enable positive	Also PULSE+ input For stepper drives
2	CHA3- AENA3-	Input Output	A negative quadrature Amplifier enable negative	Also PULSE- input For stepper drives
3	CHB3+ FAULT3+	Input Input	B positive quadrature Amplifier fault positive	Also DIR+ input From stepper drives
4	CHB3- FAULT3-	Input Input	B negative quadrature Amplifier fault negative	Also DIR- input From stepper drives
5	CHC3+ SENCENA3+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
6	CHC3- SENCENA3-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
7	+5V	Output	Digital supply voltage	Power for encoder
8	GND	Common	Digital reference voltage	Return from encoder
9	CHU3 DIR3+ SENCCLK3+	Input Output Output	U Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
10	CHV3 DIR3- SENCCLK3-	Input Output Output	V Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
11	CHW3 PULSE3+ SENCDAT3+	Input Output Bidirect.	W Hall sensor signal PFM output pulse Serial encoder data	For commutation angle For stepper control For multiple protocols
12	CHT3 PULSE3- SENCDAT3-	Input Output Bidirect.	GP input flag PFM output pulse Serial encoder data	Can be for overtemp For stepper control For multiple protocols

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for CHA*n*+ and CHA*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the CHB*n*+ and CHB*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -108 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

5. To use the CHA*n*+/- and CHB*n*+/- quadrature inputs as PULSE*n*+/- and DIR*n*+/- inputs, the saved software data structure element **Gate3[*i*].Chan[*j*].EncCtrl** that specifies how these signals are decoded must be set to 0 or 4. For the standard quadrature decode, this element must be set to one of the quadrature-decode values – usually 3 or 7. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

TB2 (Top) Fourth Channel Encoder Input: 12-Point Terminal Block

This connector is found at the back of the top edge of the mezzanine board mounted on the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot. Pin 1 of the terminal block is closest to the back of the rack.

This connector is compatible with encoder connectors on ACC-24E2x boards. A cable properly wired for one of those boards can be plugged into this connector without modification.

Pin #	Symbol	Function	Description	Notes
1	CHA4+ AENA4+	Input Output	A positive quadrature Amplifier enable positive	Also PULSE+ input For stepper drives
2	CHA4- AENA4-	Input Output	A negative quadrature Amplifier enable negative	Also PULSE- input For stepper drives
3	CHB4+ FAULT4+	Input Input	B positive quadrature Amplifier fault positive	Also DIR+ input From stepper drives
4	CHB4- FAULT4-	Input Input	B negative quadrature Amplifier fault negative	Also DIR- input From stepper drives
5	CHC4+ SENCENA4+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
6	CHC4- SENCENA4-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
7	+5V	Output	Digital supply voltage	Power for encoder
8	GND	Common	Digital reference voltage	Return from encoder
9	CHU4 DIR4+ SENCCLK4+	Input Output Output	U Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
10	CHV4 DIR4- SENCCLK4-	Input Output Output	V Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
11	CHW4 PULSE4+ SENCDAT4+	Input Output Bidirect.	W Hall sensor signal PFM output pulse Serial encoder data	For commutation angle For stepper control For multiple protocols
12	CHT4 PULSE4- SENCDAT4-	Input Output Bidirect.	GP input flag PFM output pulse Serial encoder data	Can be for overtemp For stepper control For multiple protocols

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for CHA*n*+ and CHA*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the CHB*n*+ and CHB*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0.

Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -108 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

5. To use the CHA*n*+/- and CHB*n*+/- quadrature inputs as PULSE*n*+/- and DIR*n*+/- inputs, the saved software data structure element **Gate3[*i*].Chan[*j*].EncCtrl** that specifies how these signals are decoded must be set to 0 or 4. For the standard quadrature decode, this element must be set to one of the quadrature-decode values – usually 3 or 7. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

Second Digital Feedback Mezzanine Board (604003) D-Sub Version

J1 (Top) Third Channel Encoder Input: 15-Pin D-Sub Connector

This connector is found at the front of the top edge of the mezzanine board mounted on the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot.

Pin #	Symbol	Function	Description	Notes
1	CHT3 PULSE3- SENCDAT3-	Input Output Bidirect.	GP input flag PFM output pulse Serial encoder data	Can be for overtemp For stepper control For multiple protocols
2	CHV3 DIR3- SENCCLK3-	Input Output Output	V Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
3	GND	Common	Digital reference voltage	Return from encoder
4	CHC3- SENCENA3-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
5	CHB3- FAULT3-	Input Input	B negative quadrature Amplifier fault negative	Also DIR- input From stepper drives
6	CHA3- AENA3-	Input Output	A negative quadrature Amplifier enable negative	Also PULSE- input For stepper drives
7	N.C.		No Connect	Reserved for future use
8	N.C.		No Connect	Reserved for future use
9	CHW3 PULSE3+ SENCDAT3+	Input Output Bidirect.	W Hall sensor signal PFM output pulse Serial encoder data	For commutation angle For stepper control For multiple protocols
10	CHU3 DIR3+ SENCCLK3+	Input Output Output	U Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
11	+5V	Output	Digital supply voltage	Power for encoder
12	CHC3+ SENCENA3+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
13	CHB3+ FAULT3+	Input Input	B positive quadrature Amplifier fault positive	Also DIR+ input From stepper drives
14	CHA3+ AENA3+	Input Output	A positive quadrature Amplifier enable positive	Also PULSE+ input For stepper drives
15	N.C.		No Connect	Reserved for future use

```
   _______________
8 \ ● ● ● ● ● ● ● ● ●  /  1
15 \● ● ● ● ● ● ● ● /   9
```

Controller/Encoder Connectors
15-pin D-Sub Receptacle

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for CHA*n*+ and CHA*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the CHB*n*+ and CHB*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its

default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -108 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

5. To use the CHA*n*+/- and CHB*n*+/- quadrature inputs as PULSE*n*+/- and DIR*n*+/- inputs, the saved software data structure element **Gate3[*i*].Chan[*j*].EncCtrl** that specifies how these signals are decoded must be set to 0 or 4. For the standard quadrature decode, this element must be set to one of the quadrature-decode values – usually 3 or 7. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

J2 (Top) Fourth Channel Encoder Input: 15-Pin D-Sub Connector

This connector is found at the back of the top edge of the mezzanine board mounted on the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot.

Pin #	Symbol	Function	Description	Notes
1	CHT4 PULSE4- SENCDAT4-	Input Output Bidirect.	GP input flag PFM output pulse Serial encoder data	Can be for overtemp For stepper control For multiple protocols
2	CHV4 DIR4- SENCCLK4-	Input Output Output	V Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
3	GND	Common	Digital reference voltage	Return from encoder
4	CHC4- SENCENA4-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
5	CHB4- FAULT4-	Input Input	B negative quadrature Amplifier fault negative	Also DIR- input From stepper drives
6	CHA4- AENA4-	Input Output	A negative quadrature Amplifier enable negative	Also PULSE- input For stepper drives
7	N.C.		No Connect	Reserved for future use
8	N.C.		No Connect	Reserved for future use
9	CHW4 PULSE4+ SENCDAT4+	Input Output Bidirect.	W Hall sensor signal PFM output pulse Serial encoder data	For commutation angle For stepper control For multiple protocols
10	CHU4 DIR4+ SENCCLK4+	Input Output Output	U Hall sensor signal PFM output direction Serial encoder clock	For commutation angle For stepper control For multiple protocols
11	+5V	Output	Digital supply voltage	Power for encoder
12	CHC4+ SENCENA4+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
13	CHB4+ FAULT4+	Input Input	B positive quadrature Amplifier fault positive	Also DIR+ input From stepper drives
14	CHA4+ AENA4+	Input Output	A positive quadrature Amplifier enable positive	Also PULSE+ input For stepper drives
15	N.C.		No Connect	Reserved for future use

```
   _________________
8 \ ● ● ● ● ● ● ● ● ● /  1
15 \● ● ● ● ● ● ● ● /   9
```

Controller/Encoder Connectors
15-pin D-Sub Receptacle

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for CHA*n*+ and CHA*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the CHB*n*+ and CHB*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must

be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -108 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

5. To use the CHA*n*+/- and CHB*n*+/- quadrature inputs as PULSE*n*+/- and DIR*n*+/- inputs, the saved software data structure element **Gate3[*i*].Chan[*j*].EncCtrl** that specifies how these signals are decoded must be set to 0 or 4. For the standard quadrature decode, this element must be set to one of the quadrature-decode values – usually 3 or 7. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

First Analog Feedback Mezzanine Board (604004)

J1 (Top) First Channel Encoder Input: 15-Pin D-Sub Connector

This connector is found at the front of the top edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin #	Symbol	Function	Description	Notes
1	ALTCOS1- CHT1 PULSE1- SENCDAT1-	Input Input Output Bidirect.	Aux cosine negative GP input flag PFM output pulse Serial encoder data	For coarse position Can be for overtemp For stepper control For multiple protocols
2	ALTSIN1- CHV1 DIR1- SENCCLK1-	Input Input Output Output	Aux sine negative V Hall sensor signal PFM output direction Serial encoder clock	For coarse position For commutation angle For stepper control For multiple protocols
3	GND	Common	Digital reference voltage	Return from encoder
4	INDEX1- SENCENA1-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
5	COS1- FAULT1-	Input Input	Cosine negative Amplifier fault negative	For encoder or resolver From stepper drives
6	SIN1- AENA1-	Input Output	Sine negative Amplifier enable negative	For encoder or resolver For stepper drives
7	BVREF1	Output	2.5V reference	Center for sine encoder signals
8	N.C.		No Connect	Reserved for future use
9	ALTCOS1+ CHW1 PULSE1+ SENCDAT1+	Input Input Output Bidirect.	Aux cosine positive W Hall sensor signal PFM output pulse Serial encoder data	For coarse position For commutation angle For stepper control For multiple protocols
10	ALTSIN1+ CHU1 DIR1+ SENCCLK1+	Input Input Output Output	Aux sine positive U Hall sensor signal PFM output direction Serial encoder clock	For coarse position For commutation angle For stepper control For multiple protocols
11	+5V	Output	Digital supply voltage	Power for encoder
12	INDEX1+ SENCENA1+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
13	COS1+ FAULT1+	Input Input	Cosine positive Amplifier fault positive	For encoder or resolver From stepper drives
14	SIN1+ AENA1+	Input Output	Sine positive Amplifier enable positive	For encoder or resolver For stepper drives
15	RESOUT1	Output	Resolver excitation	Programmable frequency, magnitude, and phase

```
   ________________
8 \ ● ● ● ● ● ● ● ● /  1
15 \_● ● ● ● ● ● ● _/   9
```

Controller/Encoder Connectors
15-pin D-Sub Receptacle

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for SIN*n*+ and SIN*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the COS*n*+ and COS*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -107 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

J2 (Top) Second Channel Encoder Input: 15-Pin D-Sub Connector

This connector is found at the back of the top edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin #	Symbol	Function	Description	Notes
1	ALTCOS2- CHT2 PULSE2- SENCDAT2-	Input Input Output Bidirect.	Aux cosine negative GP input flag PFM output pulse Serial encoder data	For coarse position Can be for overtemp For stepper control For multiple protocols
2	ALTSIN2- CHV2 DIR2- SENCCLK2-	Input Input Output Output	Aux sine negative V Hall sensor signal PFM output direction Serial encoder clock	For coarse position For commutation angle For stepper control For multiple protocols
3	GND	Common	Digital reference voltage	Return from encoder
4	INDEX2- SENCENA2-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
5	COS2- FAULT2-	Input Input	Cosine negative Amplifier fault negative	For encoder or resolver From stepper drives
6	SIN2- AENA2-	Input Output	Sine negative Amplifier enable negative	For encoder or resolver For stepper drives
7	BVREF2	Output	2.5V reference	Center for sine encoder signals
8	N.C.		No Connect	Reserved for future use
9	ALTCOS2+ CHW2 PULSE2+ SENCDAT2+	Input Input Output Bidirect.	Aux cosine positive W Hall sensor signal PFM output pulse Serial encoder data	For coarse position For commutation angle For stepper control For multiple protocols
10	ALTSIN2+ CHU2 DIR2+ SENCCLK2+	Input Input Output Output	Aux sine positive U Hall sensor signal PFM output direction Serial encoder clock	For coarse position For commutation angle For stepper control For multiple protocols
11	+5V	Output	Digital supply voltage	Power for encoder
12	INDEX2+ SENCENA2+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
13	COS2+ FAULT2+	Input Input	Cosine positive Amplifier fault positive	For encoder or resolver From stepper drives
14	SIN2+ AENA2+	Input Output	Sine positive Amplifier enable positive	For encoder or resolver For stepper drives
15	RESOUT2	Output	Resolver excitation	Programmable frequency, magnitude, and phase

```
   ________________
8 \ ● ● ● ● ● ● ● ● ●  /  1
15 \● ● ● ● ● ● ● ● /   9
```

Controller/Encoder Connectors
15-pin D-Sub Receptacle

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for SIN*n*+ and SIN*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the COS*n*+ and COS*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its

default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -107 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

Second Analog Feedback Mezzanine Board (604004)

J1 (Top) Third Channel Encoder Input: 15-Pin D-Sub Connector

This connector is found at the front of the top edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the right slot.

Pin #	Symbol	Function	Description	Notes
1	ALTCOS3- CHT3 PULSE3- SENCDAT3-	Input Input Output Bidirect.	Aux cosine negative GP input flag PFM output pulse Serial encoder data	For coarse position Can be for overtemp For stepper control For multiple protocols
2	ALTSIN3- CHV3 DIR3- SENCCLK3-	Input Input Output Output	Aux sine negative V Hall sensor signal PFM output direction Serial encoder clock	For coarse position For commutation angle For stepper control For multiple protocols
3	GND	Common	Digital reference voltage	Return from encoder
4	INDEX3- SENCENA3-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
5	COS3- FAULT3-	Input Input	Cosine negative Amplifier fault negative	For encoder or resolver From stepper drives
6	SIN3- AENA3-	Input Output	Sine negative Amplifier enable negative	For encoder or resolver For stepper drives
7	BVREF3	Output	2.5V reference	Center for sine encoder signals
8	N.C.		No Connect	Reserved for future use
9	ALTCOS3+ CHW3 PULSE3+ SENCDAT3+	Input Input Output Bidirect.	Aux cosine positive W Hall sensor signal PFM output pulse Serial encoder data	For coarse position For commutation angle For stepper control For multiple protocols
10	ALTSIN3+ CHU3 DIR3+ SENCCLK3+	Input Input Output Output	Aux sine positive U Hall sensor signal PFM output direction Serial encoder clock	For coarse position For commutation angle For stepper control For multiple protocols
11	+5V	Output	Digital supply voltage	Power for encoder
12	INDEX3+ SENCENA3+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
13	COS3+ FAULT3+	Input Input	Cosine positive Amplifier fault positive	For encoder or resolver From stepper drives
14	SIN3+ AENA3+	Input Output	Sine positive Amplifier enable positive	For encoder or resolver For stepper drives
15	RESOUT3	Output	Resolver excitation	Programmable frequency, magnitude, and phase

```
   ___________________        Controller/Encoder Connectors
8 \ ● ● ● ● ● ● ● ● ●  /  1   15-pin D-Sub Receptacle
15 \_● ● ● ● ● ● ● ●_/   9
```

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for SIN*n*+ and SIN*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the COS*n*+ and COS*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -107 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

J2 (Top) Fourth Channel Encoder Input: 15-Pin D-Sub Connector

This connector is found at the back of the top edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin #	Symbol	Function	Description	Notes
1	ALTCOS4- CHT4 PULSE4- SENCDAT4-	Input Input Output Bidirect.	Aux cosine negative GP input flag PFM output pulse Serial encoder data	For coarse position Can be for overtemp For stepper control For multiple protocols
2	ALTSIN4- CHV4 DIR4- SENCCLK4-	Input Input Output Output	Aux sine negative V Hall sensor signal PFM output direction Serial encoder clock	For coarse position For commutation angle For stepper control For multiple protocols
3	GND	Common	Digital reference voltage	Return from encoder
4	INDEX4- SENCENA4-	Input Output	Negative index Serial encoder strobe	Acts as capture flag For multiple protocols
5	COS4- FAULT4-	Input Input	Cosine negative Amplifier fault negative	For encoder or resolver From stepper drives
6	SIN4- AENA4-	Input Output	Sine negative Amplifier enable negative	For encoder or resolver For stepper drives
7	BVREF4	Output	2.5V reference	Center for sine encoder signals
8	N.C.		No Connect	Reserved for future use
9	ALTCOS4+ CHW4 PULSE4+ SENCDAT4+	Input Input Output Bidirect.	Aux cosine positive W Hall sensor signal PFM output pulse Serial encoder data	For coarse position For commutation angle For stepper control For multiple protocols
10	ALTSIN4+ CHU4 DIR4+ SENCCLK4+	Input Input Output Output	Aux sine positive U Hall sensor signal PFM output direction Serial encoder clock	For coarse position For commutation angle For stepper control For multiple protocols
11	+5V	Output	Digital supply voltage	Power for encoder
12	INDEX4+ SENCENA4+	Input Output	Positive index Serial encoder strobe	Acts as capture flag For multiple protocols
13	COS4+ FAULT4+	Input Input	Cosine positive Amplifier fault positive	For encoder or resolver From stepper drives
14	SIN4+ AENA4+	Input Output	Sine positive Amplifier enable positive	For encoder or resolver For stepper drives
15	RESOUT4	Output	Resolver excitation	Programmable frequency, magnitude, and phase

```
   __________________         Controller/Encoder Connectors
8 \ ● ● ● ● ● ● ● ● ● /  1    15-pin D-Sub Receptacle
15 \_●_●_●_●_●_●_●_●_/   9
```

Notes:

1. To obtain amplifier-enable outputs AENA*n*+ and AENA*n*- on the encoder input pins for SIN*n*+ and SIN*n*-, jumper E1 (for the first channel) or E2 (for the second channel) for the board must be ON. This jumper must be in its default OFF state to use these pins for encoder input.

2. To use the COS*n*+ and COS*n*- input pins for amplifier-fault inputs, jumper E3 (for the first channel) or E4 (for the second channel) must be ON. This jumper must be in its

default OFF state when bringing the amplifier fault signal in through the amplifier-interface mezzanine board to prevent signal contention.

3. To bring out PULSE*n*+/- and DIR*n*+/- outputs on the CHT/U/V/W*n* input lines, the software data structure element **Gate3[*i*].Chan[*j*].OutFlagD** for the IC and channel must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for serial encoder interface, this element must be set to its power-on default state of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*".

4. To enable serial encoder signals SENCDAT*n*+/-, SENCCLK*n*+/-, and SENCENA*n*+/- on the specified pins, the saved software data structure element **Gate3[*i*].Chan[*j*].SerialEncEna** must be set to 1. When using these pins for flag inputs such as Hall commutation sensors or for pulse-and-direction outputs, this element must be set to its factory default value of 0. Remember that the channel index value "*j*" is one less than the corresponding hardware signal channel number "*n*". Starting with revision -107 of the board (released at the start of 2012), if jumper E5 (for the first channel) or jumper E6 (for the second channel) is set to connect pins 2 and 3, the SENCENA*n*+/- pins will not be used for the serial encoder interface even if **SerialEncEna** is set to 1, permitting their use for the incremental encoder index pulse.

First Digital Amplifier Mezzanine Board (604005)

J1 (Bottom) First Channel Amplifier Output: 36-Pin Mini-D Connector

This connector is found at the front of the bottom edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin#	Symbol	Function	Description	Notes
1	PHACLK1+	Output	Phase clock	For ADC bank select
2	N.C.		No connect	
3	ADCCLK1+	Output	A/D converter clock	
4	ADCSTB1+	Output	A/D converter strobe	
5	ADCDAT1A+	Input	Phase A current data	Serial digital, differential
6	ADCDAT1B+	Input	Phase B current data	Serial digital, differential
7	AENA1+	Output	Amplifier enable	High is enable
8	FAULT1+	Input	Amplifier fault	High is fault
9	PWMATOP1+	Output	Phase A top cmd.	High is ON command
10	PWMABOT1+	Output	Phase A bottom cmd.	High is ON command
11	PWMBTOP1+	Output	Phase B top cmd.	High is ON command
12	PWMBBOT1+	Output	Phase B bottom cmd.	High is ON command
13	PWMCTOP1+	Output	Phase C top cmd.	High is ON command
14	PWMCBOT1+	Output	Phase C bottom cmd.	High is ON command
15	GND	Common	Reference voltage	
16	+5V	Output	Digital power	
17	PWMDTOP1+	Output	Phase D top cmd.	High is ON command
18	PWMDBOT1+	Output	Phase D bottom cmd.	High is ON command
19	PHACLK1-	Output	Phase clock	For ADC bank select
20	N.C.		No connect	
21	ADCCLK1-	Output	A/D converter clock	
22	ADCSTB1-	Output	A/D converter strobe	
23	ADCDAT1A-	Input	Phase A current data	Serial digital, differential
24	ADCDAT1B-	Input	Phase B current data	Serial digital, differential
25	AENA1-	Output	Amplifier enable	Low is enable
26	FAULT1-	Input	Amplifier fault	Low is fault
27	PWMATOP1-	Output	Phase A top cmd.	Low is ON command
28	PWMABOT1-	Output	Phase A bottom cmd.	Low is ON command
29	PWMBTOP1-	Output	Phase B top cmd.	Low is ON command
30	PWMBBOT1-	Output	Phase B bottom cmd.	Low is ON command
31	PWMCTOP1-	Output	Phase C top cmd.	Low is ON command
32	PWMCBOT1-	Output	Phase C bottom cmd.	Low is ON command
33	GND	Common	Reference voltage	
34	+5V	Output	Digital power	
35	PWMDTOP1-	Output	Phase D top cmd.	Low is ON command
36	PWMDBOT1-+	Output	Phase D bottom cmd.	Low is ON command

```
   ________________________________
18 \ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● /  1
36  \●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●/ 19
```

Controller/Drive Connectors
36-pin Mini-D Receptacle

J2 (Bottom) Second Channel Amplifier Output: 36-Pin Mini-D Connector

This connector is found at the back of the bottom edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin#	Symbol	Function	Description	Notes
1	PHACLK2+	Output	Phase clock	For ADC bank select
2	N.C.		No connect	
3	ADCCLK2+	Output	A/D converter clock	
4	ADCSTB2+	Output	A/D converter strobe	
5	ADCDAT2A+	Input	Phase A current data	Serial digital, differential
6	ADCDAT2B+	Input	Phase B current data	Serial digital, differential
7	AENA2+	Output	Amplifier enable	High is enable
8	FAULT2+	Input	Amplifier fault	High is fault
9	PWMATOP2+	Output	Phase A top cmd.	High is ON command
10	PWMABOT2+	Output	Phase A bottom cmd.	High is ON command
11	PWMBTOP2+	Output	Phase B top cmd.	High is ON command
12	PWMBBOT2+	Output	Phase B bottom cmd.	High is ON command
13	PWMCTOP2+	Output	Phase C top cmd.	High is ON command
14	PWMCBOT2+	Output	Phase C bottom cmd.	High is ON command
15	GND	Common	Reference voltage	
16	+5V	Output	Digital power	
17	PWMDTOP2+	Output	Phase D top cmd.	High is ON command
18	PWMDBOT2+	Output	Phase D bottom cmd.	High is ON command
19	PHACLK2-	Output	Phase clock	For ADC bank select
20	N.C.		No connect	
21	ADCCLK2-	Output	A/D converter clock	
22	ADCSTB2-	Output	A/D converter strobe	
23	ADCDAT2A-	Input	Phase A current data	Serial digital, differential
24	ADCDAT2B-	Input	Phase B current data	Serial digital, differential
25	AENA2-	Output	Amplifier enable	Low is enable
26	FAULT2-	Input	Amplifier fault	Low is fault
27	PWMATOP2-	Output	Phase A top cmd.	Low is ON command
28	PWMABOT2-	Output	Phase A bottom cmd.	Low is ON command
29	PWMBTOP2-	Output	Phase B top cmd.	Low is ON command
30	PWMBBOT2-	Output	Phase B bottom cmd.	Low is ON command
31	PWMCTOP2-	Output	Phase C top cmd.	Low is ON command
32	PWMCBOT2-	Output	Phase C bottom cmd.	Low is ON command
33	GND	Common	Reference voltage	
34	+5V	Output	Digital power	
35	PWMDTOP2-	Output	Phase D top cmd.	Low is ON command
36	PWMDBOT2-+	Output	Phase D bottom cmd.	Low is ON command

```
    ______________________________________
18 \ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●  /  1
36 \●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●/ 19
```

Controller/Drive Connectors
36-pin Mini-D Receptacle

Second Digital Amplifier Mezzanine Board (604005)

J1 (Bottom) Third Channel Amplifier Output: 36-Pin Mini-D Connector

This connector is found at the front of the bottom edge of the mezzanine board mounted on the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot.

Pin#	Symbol	Function	Description	Notes
1	PHACLK3+	Output	Phase clock	For ADC bank select
2	N.C.		No connect	
3	ADCCLK3+	Output	A/D converter clock	
4	ADCSTB3+	Output	A/D converter strobe	
5	ADCDAT3A+	Input	Phase A current data	Serial digital, differential
6	ADCDAT3B+	Input	Phase B current data	Serial digital, differential
7	AENA3+	Output	Amplifier enable	High is enable
8	FAULT3+	Input	Amplifier fault	High is fault
9	PWMATOP3+	Output	Phase A top cmd.	High is ON command
10	PWMABOT3+	Output	Phase A bottom cmd.	High is ON command
11	PWMBTOP3+	Output	Phase B top cmd.	High is ON command
12	PWMBBOT3+	Output	Phase B bottom cmd.	High is ON command
13	PWMCTOP3+	Output	Phase C top cmd.	High is ON command
14	PWMCBOT3+	Output	Phase C bottom cmd.	High is ON command
15	GND	Common	Reference voltage	
16	+5V	Output	Digital power	
17	PWMDTOP3+	Output	Phase D top cmd.	High is ON command
18	PWMDBOT3+	Output	Phase D bottom cmd.	High is ON command
19	PHACLK3-	Output	Phase clock	For ADC bank select
20	N.C.		No connect	
21	ADCCLK3-	Output	A/D converter clock	
22	ADCSTB3-	Output	A/D converter strobe	
23	ADCDAT3A-	Input	Phase A current data	Serial digital, differential
24	ADCDAT3B-	Input	Phase B current data	Serial digital, differential
25	AENA3-	Output	Amplifier enable	Low is enable
26	FAULT3-	Input	Amplifier fault	Low is fault
27	PWMATOP3-	Output	Phase A top cmd.	Low is ON command
28	PWMABOT3-	Output	Phase A bottom cmd.	Low is ON command
29	PWMBTOP3-	Output	Phase B top cmd.	Low is ON command
30	PWMBBOT3-	Output	Phase B bottom cmd.	Low is ON command
31	PWMCTOP3-	Output	Phase C top cmd.	Low is ON command
32	PWMCBOT3-	Output	Phase C bottom cmd.	Low is ON command
33	GND	Common	Reference voltage	
34	+5V	Output	Digital power	
35	PWMDTOP3-	Output	Phase D top cmd.	Low is ON command
36	PWMDBOT3-+	Output	Phase D bottom cmd.	Low is ON command

```
     ____________________________________
18 \ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● /  1
36  \●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●/  19
```

Controller/Drive Connectors
36-pin Mini-D Receptacle

J2 (Bottom) Fourth Channel Amplifier Output: 36-Pin Mini-D Connector

This connector is found at the back of the bottom edge of the mezzanine board mounted on the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot.

Pin#	Symbol	Function	Description	Notes
1	PHACLK4+	Output	Phase clock	For ADC bank select
2	N.C.		No connect	
3	ADCCLK4+	Output	A/D converter clock	
4	ADCSTB4+	Output	A/D converter strobe	
5	ADCDAT4A+	Input	Phase A current data	Serial digital, differential
6	ADCDAT4B+	Input	Phase B current data	Serial digital, differential
7	AENA4+	Output	Amplifier enable	High is enable
8	FAULT4+	Input	Amplifier fault	High is fault
9	PWMATOP4+	Output	Phase A top cmd.	High is ON command
10	PWMABOT4+	Output	Phase A bottom cmd.	High is ON command
11	PWMBTOP4+	Output	Phase B top cmd.	High is ON command
12	PWMBBOT4+	Output	Phase B bottom cmd.	High is ON command
13	PWMCTOP4+	Output	Phase C top cmd.	High is ON command
14	PWMCBOT4+	Output	Phase C bottom cmd.	High is ON command
15	GND	Common	Reference voltage	
16	+5V	Output	Digital power	
17	PWMDTOP4+	Output	Phase D top cmd.	High is ON command
18	PWMDBOT4+	Output	Phase D bottom cmd.	High is ON command
19	PHACLK4-	Output	Phase clock	For ADC bank select
20	N.C.		No connect	
21	ADCCLK4-	Output	A/D converter clock	
22	ADCSTB4-	Output	A/D converter strobe	
23	ADCDAT4A-	Input	Phase A current data	Serial digital, differential
24	ADCDAT4B-	Input	Phase B current data	Serial digital, differential
25	AENA4-	Output	Amplifier enable	Low is enable
26	FAULT4-	Input	Amplifier fault	Low is fault
27	PWMATOP4-	Output	Phase A top cmd.	Low is ON command
28	PWMABOT4-	Output	Phase A bottom cmd.	Low is ON command
29	PWMBTOP4-	Output	Phase B top cmd.	Low is ON command
30	PWMBBOT4-	Output	Phase B bottom cmd.	Low is ON command
31	PWMCTOP4-	Output	Phase C top cmd.	Low is ON command
32	PWMCBOT4-	Output	Phase C bottom cmd.	Low is ON command
33	GND	Common	Reference voltage	
34	+5V	Output	Digital power	
35	PWMDTOP4-	Output	Phase D top cmd.	Low is ON command
36	PWMDBOT4-+	Output	Phase D bottom cmd.	Low is ON command

```
    ______________________________
18 \ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● /  1
36  \●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●_●/  19
```

Controller/Drive Connectors
36-pin Mini-D Receptacle

First Analog Amplifier Mezzanine Board (604006), Terminal Block Version

TB1 (Bottom) First Channel Amplifier Output: 12-point Terminal Block

This connector is found at the front of the bottom edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot. Pin 1 of the terminal block is closest to the back of the rack.

Pin #	Symbol	Function	Description	Notes
1	DAC1A+	Output	Phase A analog out pos	+/-10V, ref to AGND
2	DAC1A-	Output	Phase A analog out neg	-/+10V, ref to AGND
3	DAC1B+	Output	Phase B analog out pos	+/-10V, ref to AGND
4	DAC1B-	Output	Phase B analog out neg	-/+10V, ref to AGND
5	AE_NC_1	Output	Amp enable normally closed	Open when enabled
6	AE_COM_1	Return	Amp enable common	For either NO or NC contact
7	AE_NO_1	Output	Amp enable normally open	Closed when enabled
8	FAULT1+	Input	Amp fault input pos	Software polarity control
9	FAULT1-	Input	Amp fault input neg	Software polarity control
10	A+15V	Input	Analog plus supply voltage	Used when E2 OFF
11	A-15V	Input	Analog minus supply voltage	Used when E3 OFF
12	AGND	Common	Analog reference voltage	Isolated when E1 OFF

TB2 (Bottom) Second Channel Amplifier Output: 12-point Terminal Block

This connector is found at the back of the bottom edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot. Pin 1 of the terminal block is closest to the back of the rack.

Pin #	Symbol	Function	Description	Notes
1	DAC2A+	Output	Phase A analog out pos	+/-10V, ref to AGND
2	DAC2A-	Output	Phase A analog out neg	-/+10V, ref to AGND
3	DAC2B+	Output	Phase B analog out pos	+/-10V, ref to AGND
4	DAC2B-	Output	Phase B analog out neg	-/+10V, ref to AGND
5	AE_NC_2	Output	Amp enable normally closed	Open when enabled
6	AE_COM_2	Return	Amp enable common	For either NO or NC contact
7	AE_NO_2	Output	Amp enable normally open	Closed when enabled
8	FAULT2+	Input	Amp fault input pos	Software polarity control
9	FAULT2-	Input	Amp fault input neg	Software polarity control
10	A+15V	Input	Analog plus supply voltage	Used when E2 OFF
11	A-15V	Input	Analog minus supply voltage	Used when E3 OFF
12	AGND	Common	Analog reference voltage	Isolated when E1 OFF

First Analog Amplifier Mezzanine Board (604006), D-Sub Version

J1 (Bottom) First Channel Amplifier Output: 15-pin D-sub Connector

This connector is found at the front of the bottom edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin #	Symbol	Function	Description	Notes
1	DAC1A+	Output	Phase A analog out pos	+/-10V, ref to AGND
2	DAC1B+	Output	Phase B analog out pos	+/-10V, ref to AGND
3	AE_NC_1	Output	Amp enable normally closed	Open when enabled
4	AE_NO_1	Output	Amp enable normally open	Closed when enabled
5	FAULT1-	Input	Amp fault input neg	Software polarity control
6	DAC1C-	Output	Phase C analog out neg	-/+10V, ref to AGND, optional
7	A+15V	Input	Analog plus supply voltage	Used when E2 OFF
8	AGND	Common	Analog reference voltage	Isolated when E1 OFF
9	DAC1A-	Output	Phase A analog out neg	-/+10V, ref to AGND
10	DAC1B-	Output	Phase B analog out neg	-/+10V, ref to AGND
11	AE_COM_1	Return	Amp enable common	For either NO or NC contact
12	FAULT1+	Input	Amp fault input pos	Software polarity control
13	DAC1C+	Output	Phase C analog out pos	+/-10V, ref to AGND, optional
14	AGND	Common	Analog reference voltage	Isolated when E1 OFF
15	A-15V	Input	Analog minus supply voltage	Used when E3 OFF

```
    _______________
8 \  ● ● ● ● ● ● ● ● ●  /  1
15 \_●_●_●_●_●_●_●_●_/   9
```

Controller/Amplifier Connectors
15-pin D-Sub Receptacle

J2 (Bottom) Second Channel Amplifier Output: 15-pin D-sub Connector

This connector is found at the back of the bottom edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the left slot.

Pin #	Symbol	Function	Description	Notes
1	DAC2A+	Output	Phase A analog out pos	+/-10V, ref to AGND
2	DAC2B+	Output	Phase B analog out pos	+/-10V, ref to AGND
3	AE_NC_2	Output	Amp enable normally closed	Open when enabled
4	AE_NO_2	Output	Amp enable normally open	Closed when enabled
5	FAULT2-	Input	Amp fault input neg	Software polarity control
6	DAC2C-	Output	Phase C analog out neg	-/+10V, ref to AGND, optional
7	A+15V	Input	Analog plus supply voltage	Used when E2 OFF
8	AGND	Common	Analog reference voltage	Isolated when E1 OFF
9	DAC2A-	Output	Phase A analog out neg	-/+10V, ref to AGND
10	DAC2B-	Output	Phase B analog out neg	-/+10V, ref to AGND
11	AE_COM_2	Return	Amp enable common	For either NO or NC contact
12	FAULT2+	Input	Amp fault input pos	Software polarity control
13	DAC2C+	Output	Phase C analog out pos	+/-10V, ref to AGND, optional
14	AGND	Common	Analog reference voltage	Isolated when E1 OFF
15	A-15V	Input	Analog minus supply voltage	Used when E3 OFF

```
   _________________           Controller/Amplifier Connectors
8 \ ● ● ● ● ● ● ● ● ●  /  1     15-pin D-Sub Receptacle
15 \_●_●_●_●_●_●_●_●_/   9
```

Second Analog Amplifier Mezzanine Board (604006), Terminal Block Version

TB1 (Bottom) Third Channel Amplifier Output: 12-point Terminal Block

This connector is found at the front of the bottom edge of the mezzanine board mounted on the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot. Pin 1 of the terminal block is closest to the back of the rack.

Pin #	Symbol	Function	Description	Notes
1	DAC3A+	Output	Phase A analog out pos	+/-10V, ref to AGND
2	DAC3A-	Output	Phase A analog out neg	-/+10V, ref to AGND
3	DAC3B+	Output	Phase B analog out pos	+/-10V, ref to AGND
4	DAC3B-	Output	Phase B analog out neg	-/+10V, ref to AGND
5	AE_NC_3	Output	Amp enable normally closed	Open when enabled
6	AE_COM_3	Return	Amp enable common	For either NO or NC contact
7	AE_NO_3	Output	Amp enable normally open	Closed when enabled
8	FAULT3+	Input	Amp fault input pos	Software polarity control
9	FAULT3-	Input	Amp fault input neg	Software polarity control
10	A+15V	Input	Analog plus supply voltage	Used when E2 OFF
11	A-15V	Input	Analog minus supply voltage	Used when E3 OFF
12	AGND	Common	Analog reference voltage	Isolated when E1 OFF

TB2 (Bottom) Fourth Channel Amplifier Output: 12-point Terminal Block

This connector is found at the back of the bottom edge of the mezzanine board mounted on the 3U-format piggyback board. In a 4-channel 2-slot assembly, it will be in the right slot. Pin 1 of the terminal block is closest to the back of the rack.

Pin #	Symbol	Function	Description	Notes
1	DAC4A+	Output	Phase A analog out pos	+/-10V, ref to AGND
2	DAC4A-	Output	Phase A analog out neg	-/+10V, ref to AGND
3	DAC4B+	Output	Phase B analog out pos	+/-10V, ref to AGND
4	DAC4B-	Output	Phase B analog out neg	-/+10V, ref to AGND
5	AE_NC_4	Output	Amp enable normally closed	Open when enabled
6	AE_COM_4	Return	Amp enable common	For either NO or NC contact
7	AE_NO_4	Output	Amp enable normally open	Closed when enabled
8	FAULT4+	Input	Amp fault input pos	Software polarity control
9	FAULT4-	Input	Amp fault input neg	Software polarity control
10	A+15V	Input	Analog plus supply voltage	Used when E2 OFF
11	A-15V	Input	Analog minus supply voltage	Used when E3 OFF
12	AGND	Common	Analog reference voltage	Isolated when E1 OFF

Second Analog Amplifier Mezzanine Board (604006), D-Sub Version

J1 (Bottom) Third Channel Amplifier Output: 15-pin D-sub Connector

This connector is found at the front of the bottom edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the right slot.

Pin #	Symbol	Function	Description	Notes
1	DAC3A+	Output	Phase A analog out pos	+/-10V, ref to AGND
2	DAC3B+	Output	Phase B analog out pos	+/-10V, ref to AGND
3	AE_NC_3	Output	Amp enable normally closed	Open when enabled
4	AE_NO_3	Output	Amp enable normally open	Closed when enabled
5	FAULT3-	Input	Amp fault input neg	Software polarity control
6	DAC3C-	Output	Phase C analog out neg	-/+10V, ref to AGND, optional
7	A+15V	Input	Analog plus supply voltage	Used when E2 OFF
8	AGND	Common	Analog reference voltage	Isolated when E1 OFF
9	DAC3A-	Output	Phase A analog out neg	-/+10V, ref to AGND
10	DAC3B-	Output	Phase B analog out neg	-/+10V, ref to AGND
11	AE_COM_3	Return	Amp enable common	For either NO or NC contact
12	FAULT3+	Input	Amp fault input pos	Software polarity control
13	DAC3C+	Output	Phase C analog out pos	+/-10V, ref to AGND, optional
14	AGND	Common	Analog reference voltage	Isolated when E1 OFF
15	A-15V	Input	Analog minus supply voltage	Used when E3 OFF

```
   _____________________
8 \ ● ● ● ● ● ● ● ● ● / 1
15 \● ● ● ● ● ● ● ● /  9
    ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾
```

Controller/Amplifier Connectors
15-pin D-Sub Receptacle

J2 (Bottom) Fourth Channel Amplifier Output: 15-pin D-sub Connector

This connector is found at the back of the bottom edge of the mezzanine board mounted on the 3U-format base board. In a 4-channel 2-slot assembly, it will be in the right slot.

Pin #	Symbol	Function	Description	Notes
1	DAC4A+	Output	Phase A analog out pos	+/-10V, ref to AGND
2	DAC4B+	Output	Phase B analog out pos	+/-10V, ref to AGND
3	AE_NC_4	Output	Amp enable normally closed	Open when enabled
4	AE_NO_4	Output	Amp enable normally open	Closed when enabled
5	FAULT4-	Input	Amp fault input neg	Software polarity control
6	DAC4C-	Output	Phase C analog out neg	-/+10V, ref to AGND, optional
7	A+15V	Input	Analog plus supply voltage	Used when E2 OFF
8	AGND	Common	Analog reference voltage	Isolated when E1 OFF
9	DAC4A-	Output	Phase A analog out neg	-/+10V, ref to AGND
10	DAC4B-	Output	Phase B analog out neg	-/+10V, ref to AGND
11	AE_COM_4	Return	Amp enable common	For either NO or NC contact
12	FAULT4+	Input	Amp fault input pos	Software polarity control
13	DAC4C+	Output	Phase C analog out pos	+/-10V, ref to AGND, optional
14	AGND	Common	Analog reference voltage	Isolated when E1 OFF
15	A-15V	Input	Analog minus supply voltage	Used when E3 OFF

```
   _________________
8 \ ● ● ● ● ● ● ● ● ● /  1
15 \_●_●_●_●_●_●_●_●_/   9
```

Controller/Encoder Connectors
15-pin D-Sub Receptacle

DSPGATE3 (PMAC3-Style ASIC) Register Elements

This section documents the data structure elements of the “DSPGATE3” ASIC that forms the core of the ACC-24E3. These elements include saved setup elements (shown in bold) that are documented in detail in the Power PMAC Software Reference Manual, non-saved setup/control elements, and read-only status elements.

It also documents the address offset of each register. This is primarily for reference, as most uses of the elements and registers do not require the user to know the numerical address of the register.

To calculate the address offset of a register element from the I/O base address:

1. Add the offset of the base address of the IC in question from the I/O base address, based on the IC type and index number. This offset value is found in the boxed table at the top of the section for each IC type.
2. If a channel-specific register, add the offset of the channel’s base address from the IC’s base address.
3. Add the offset of the register in question from the IC’s base address, or for a channel-specific register, from the channel’s base address.

This value can be used to declare “io” format M-variables.

To find the absolute numerical address of the register element in the Power PMAC memory map:

1. Find the starting address for I/O by querying the value of **Sys.piom**.
2. Add this to the address offset calculated above

This value will match the numerical value reported when the address (**.a**) of the element is queried.

Notes:

1. Bit numbers are given in “Intel” (little-endian) style, where the value of Bit *n* in the word is 2^n.

2. Names given in italics are not actual elements that can be used, but express components within full-word elements with distinct functions.

In the Power PMAC script-language environment, both with buffered program statements and on-line commands, elements that are less than 32 bits are accessed using the actual bit width of the element. For example **Gate1[*i*].Chan[*j*].Status** occupies bits 08 – 31 of a 32-bit word. In the script environment, it is treated as a 24-bit value. **Gate1[*i*].Chan[*j*].HomeFlag** occupies bit 24 of this same word. It can be accessed directly as a single-bit value in the script environment

In the C-language, IC registers can be accessed only as full 32-bit words, even if the registers do not occupy all 32-bit words. So **Gate1[*i*].Chan[*j*].Status** is treated as a 32-bit value, with the low 8 bits being meaningless. The single-bit value **Gate1[*i*].Chan[*j*].HomeFlag** cannot be accessed directly from a C program; it would have to be derived with an expression such as

(Gate1[*i*].Chan[*j*].Status & $01000000) >> 24

Note: Saved setup element names are shown in bold font. Partial-word components that cannot be accessed as separate elements, even in the Script environment, are shown in italic font.

Gate3[#]	0	1	2	3	4	5	6	7
IC Base Offset	$900000	$904000	$908000	$90C000	$910000	$914000	$918000	$91C000
Gate3[#]	8	9	10	11	12	13	14	15
IC Base Offset	$920000	$924000	$928000	$92C000	$930000	$934000	$938000	$93C000

Data Structure	Full-Word Element	Partial-Word Element/*Component*	Offset from ASIC Base	Bits
Gate3[*i*].	Chan[0]		$000	
	Chan[1]		$080	
	Chan[2]		$100	
	Chan[3]		$180	
	PhaseServoClockCtrl		$200	00-31
		EncLatchDelay		00-07
		PhaseServoDir		10-11
		PhaseClockDiv		12-13
		PhaseClockMult		14-15
		PhaseFreq		16-31
	HardwareClockCtrl		$204	00-31
		ServoClockDiv		00-03
		ClockPol		04-07
		FiltClockDiv		08-11
		EncClockDiv		12-15
		PfmClockDiv		16-19
		DacClockDiv		20-23
		AdcEncClockDiv		24-27
		AdcAmpClockDiv		28-31
	DacStrobe		$208	00-31
	AdcAmpCtrl		$20C	00-31
		AdcAmpHeaderBits		00-02
		AdcAmpUtoSConv		03
		AdcStrobeAmpDelay		04-07
		AdcStrobeAmpWord		08-31
	AdcEncCtrl		$210	00-31
		AdcEncHeaderBits		00-02
		AdcEncUtoSConv		03
		AdcStrobeEncDelay		04-07
		AdcStrobeEncWord		08-31
	ResolverCtrl		$214	00-31
		{*reserved*}		00-19
		ResolverExciteFreq		20-21
		ResolverExciteGain		22-23
		ResolverExciteShift		24-31
	SerialEncCtrl		$218	00-31
		SerialProtocol		00-03
		{*reserved*}		04-07
		SerialTrigDelay		08-15
		SerialTrigEdgeSel		16
		SerialTrigClockSel		17

	{reserved}		18-19
	SerialClockNDiv		20-23
	SerialClockMDiv		24-31
WpKey		$21C	00-31
ChipID		$220	00-31
IntCtrl		$224	00-31
	InterruptStatus		00-07
	InterruptSource		08-15
	InterruptEna		16-23
	{reserved}		24-31
MacroError		$228	00-31
EEpromCtrl		$22C	00-31
	EEpromWriteEna		00
	EEpromReadEna		01
	(reserved}		02-31
EEpromData[0]		$230	00-31
EEpromData[1]		$234	00-31
EEpromData[2]		$238	00-31
EEpromData[3]		$23C	00-31
GpioCtrl / **GpioMode[0]**		$240	00-31
GpioMode[1]		$244	00-31
GpioMode[2]		$248	00-31
GpioMode[3]		$24C	00-31
GpioData[0]		$250	00-31
GpioData[1]		$254	00-31
GpioData[2]		$258	00-31
GpioData[3]		$25C	00-31
GpioDir[0]		$260	00-31
GpioDir[1]		$264	00-31
GpioDir[2]		$268	00-31
GpioDir[3]		$26C	00-31
GpioPol[0]		$270	00-31
GpioPol[1]		$274	00-31
GpioPol[2]		$278	00-31
GpioPol[3]		$27C	00-31
ReadAlias[*n*] (*n* = 0 to 14)		n*4+$280	00-31
WriteAlias[*n*] (*n* = 0 to 14)		n*4+$2C0	00-31
MemAlias[*n*] (*n* = 0 to 59)		n*4+$300	00-31
MacroEnableA		$3F0	00-23
MacroModeA		$3F4	00-15
MacroEnableB		$3F8	00-23
MacroModeB		$3FC	00-15
MacroInA[*m*][*n*] (*m* = 0 to 15, *n* = 0 to 3)		m*16+n*4+$400	00-31
MacroOutA[*m*][*n*] (*m* = 0 to 15, *n* = 0 to 3)		m*16+n*4+$500	00-31
MacroInB[*m*][*n*] (*m* = 0 to 15, *n* = 0 to 3)		m*16+n*4+$600	00-31
MacroOutB[*m*][*n*] (*m* = 0 to 15, *n* = 0 to 3)		m*16+n*4+$700	00-31

Chan[#]	0	1	2	3
Offset from IC Base	$000	$080	$100	$180

Data Structure	Full-Word Element	Partial-Word Element/*Component*	Offset from Channel Base	Bits
Gate3[*i*].Chan[*j*].	Status		$000	00-31
		ABPins		00-03
		ABC		04-06
		Fault		07
		HomeFlag		08
		PlusLimit		09
		MinusLimit		10
		UserFlag		11
		UVW		12-14
		T		15
		HallState		16-18
		DemuxInvalid		19
		PosCapt		20-21
		TrigState		22
		Equ		24
		EquOut		25
		LossStatus		28
		LossCapt		29
		CountError		30
		SosError		31
	PhaseCapt		$004	00-31
	ServoCapt		$008	00-31
	AtanSumOfSqr		$00C	00-31
		SumOfSquares		00-15
		Atan		16-31
	TimerA		$010	00-31
	TimerB		$014	00-31
	SerialEncDataA		$018	00-31
	SerialEncDataB		$01C	00-31
	AdcAmp[0]		$020	00-31
	AdcAmp[1]		$024	00-31
	AdcAmp[2]		$028	00-31
	AdcAmp[3]		$02C	00-31
	AdcEnc[0]		$030	00-31
	AdcEnc[1]		$034	00-31
	AdcEnc[2]		$038	00-31
	AdcEnc[3]		$03C	00-31
	Pwm[0] / Dac[0]		$040	00-31
	Pwm[1] / Dac[1]		$044	00-31
	Pwm[2] / Dac[2]		$048	00-31
	Pwm[3] / Pfm		$04C	00-31
	CompA		$050	00-31
	CompB		$054	00-31
	CompAdd		$058	00-31
	SerialEncCmd		$05C	00-31
		SerialPosNumBits		00-05
		SerialStatusNumBits		06-09
		SerialDataReady		10

	SerialGrayBinConv		11
	SerialTrigEna		12
	SerialTrigMode		13
	SerialParityType		14-15
	SerialCmdWord		16-31
AdcOffset[0]		$060	00-31
AdcOffset[1]		$064	00-31
InCtrl		$068	00-31
	EncCtrl		00-03
	TimerMode		04-05
	CaptCtrl		06-09
	CaptFlagSel		10-11
	CaptFlagChan		12-13
	GatedIndexSel		14
	IndexGateState		15
	IndexDemuxEna		16
	FlagFilt2Ena		17
	AtanEna		18
	CountReset		19
	SerialEncEna		20
	PackInData		21-22
	{*reserved*}		23-31
OutCtrl		$06C	00-31
	EquOutMask		00-03
	EquOutPol		04
	Equ1Ena		05
	EquWrite		06-07
	AmpEna		08
	OutFlagB		09
	OutFlagC		10
	OutFlagD		11
	OutputMode		12-15
	OutputPol		16-17
	PfmDirPol		18
	PfmFormat		19
	PwmFreqMult		20-22
	PackOutData		23
	PwmDeadTime		24-31
PfmWidth		$070	00-11
HomeCapt		$074	00-31