Designing a Direct PWM Interface

Introduction

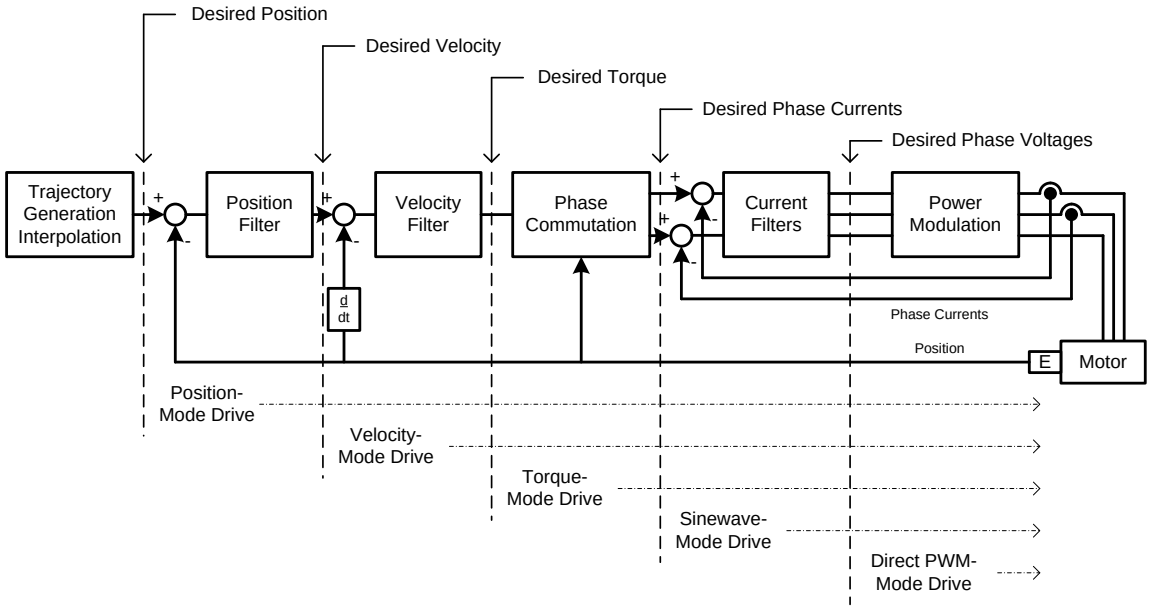
Delta Tau Data Systems' PMAC controller family pioneered the "direct PWM" controller-to-amplifier interface, in which the controller outputs the actual pulse-width-modulated (PWM) on/off signals for the power transistors in the amplifier and receives current feedback information in the form of serial data words from the amplifier

The interface is an open standard, and multiple companies have designed amplifiers compliant with the standard, both for internal use, and for the market. This document explains how to design an interface to be compliant with the standard.

In this mode of operation, the drive does not perform any control functions, except perhaps shunt regulation; it simply provides the power stage, interface, isolation, level shifting, self-protection, basic diagnostic capability, and sensing of the phase currents with analog-to-digital conversion. This style of drive is often called a "power block". The controller performs all of the control functions: position loop, velocity loop, commutation, and current loop.

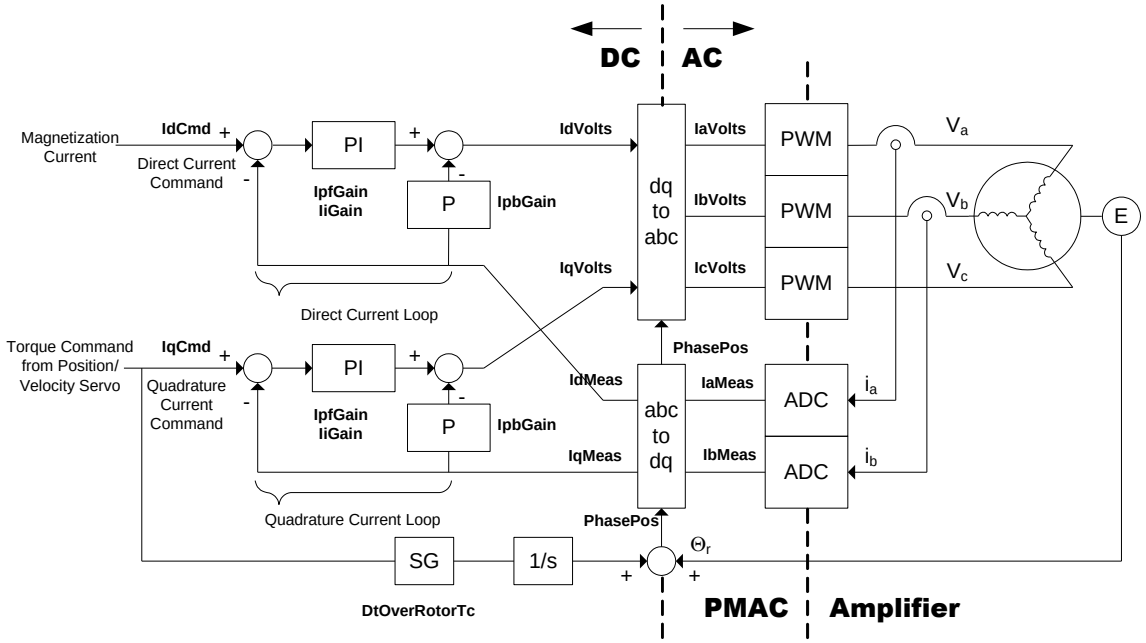
Note that no position feedback device or "flags" (e.g. home, limits, registration) are connected to this style of drive, as the controller performs all of the functionality using that I/O.

The following diagram shows the sequence of operations required to control an electronically commutated motor, and the possible locations in the sequence where tasks can be split between controller and drive. With the direct PWM interface, the split is at the rightmost of the possible locations; everything to the left is done by the controller.



Motor Control Task Sequencing and Possible Divisions

The Delta Tau implementations of the direct PWM interface utilize a specially designed ASIC to generate the PWM output signals and accept the digital current feedback signals. The control tasks – position/velocity-loop closure, commutation, and current-loop closure – are performed in software by the processor. The following diagram shows this operation.



Direct PWM Commutation with Digital Current Loop

The details of the calculations in the PMAC controllers are not directly related to the interface, but are useful both to understand the overall concept, and to test the interface in actual operation.

Connection

The direct PWM interface can be implemented through a cable connection between "box-level" controllers and amplifiers. It can also be implemented through a "stack" connector between "board-level" controllers and amplifiers. Both cases are covered in their own sections below.

Cable Connection

Cable connections will use 36-pin cables conforming to the IEEE-1284C standard for each axis. These have Mini-D 36-pin connectors at both ends. Shielded, twisted-pair cables meeting this standard are commercially available in multiple lengths.

The most commonly used Delta Tau controller interfaces that use these connectors are the UMAC ACC-24E2 and ACC-24E3 PWM axis interface boards.

All of the signals sent in both directions over this cable are transmitted as RS422-style differential signal pairs operating at 5V levels. They are intended to be transmitted with MC3487 or equivalent differential line drivers; they are intended to be received with MC3486 or equivalent differential line receivers.

The pin-out of the 36-pin connector is given at the back of this document.

Inter-Board Connection

Inter-board connections will use 50-pin (2 x 25) stack connectors with 0.1-inch spacing between pins for a set of 4 axes. Mating connectors are available from numerous sources.

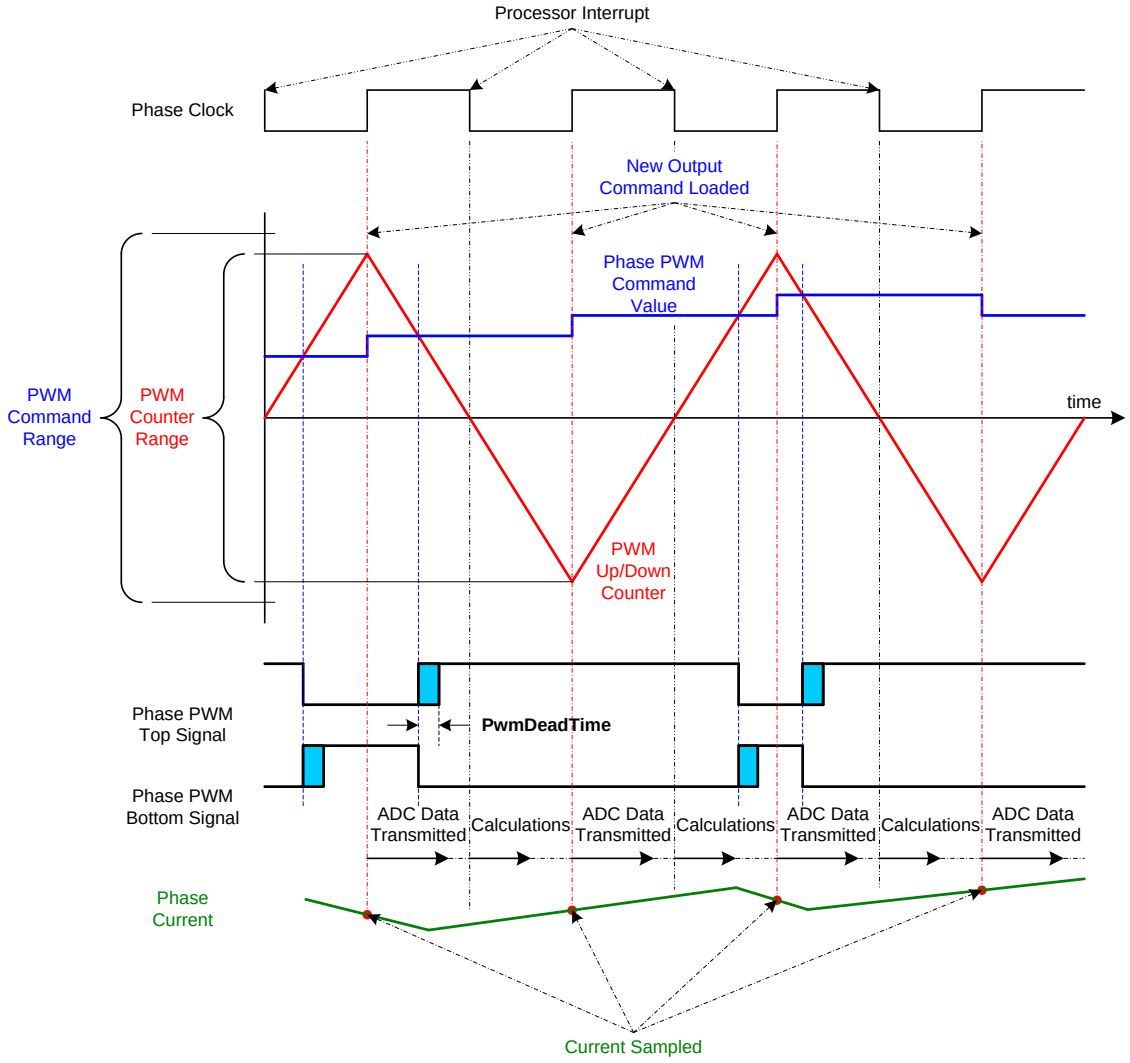
The most commonly used Delta Tau controllers that use these connectors are the Clipper and PC/104 versions of the PMAC controllers.

All of the signals sent in both directions are transmitted as single-ended CMOS signals operating at 5V levels.

The pin-out of the 50-pin stack connector is given at the back of this document. Note that there is only one ADC clock signal and one ADC strobe signal, which must be used by all four axes.

Direct PWM Hardware/Software Timing

The direct-PWM technique provides tightly coupled hardware and software operation for minimum delay in loop closure, and so maximum performance. It is important to understand how this coupling works. The following diagram shows key internal and external signals for the most common case where there are two phase clock cycles per PWM cycle:



Direct PWM Hardware and Software Timing

Phase Clock Signal

The process is fundamentally driven by the "phase clock" signal generated in the Servo IC. Its frequency is programmable in very small increments, but it is usually set in the range of 8 to 24 kHz. On the rising edge of the phase clock, the current ADCs are strobed to sample the current, and the output command values from the previous cycle are loaded into the IC's PWM generation registers. On the falling edge of the phase clock, the processor is interrupted to close the current loops based on the current feedback values sampled half a clock cycle before, and the command values it calculates are loaded for active use a half clock cycle afterwards.

PWM Up-Down Counter

Synchronized to the phase clock signal is an internal PWM up/down counter. Its value when plotted creates a "sawtooth" wave. It reaches its peak positive and negative values at the rising edges of the phase clock, and it crosses zero at the falling edges of the phase clock. The counter value is updated at a very high frequency (~120 MHz in the PMAC2-style IC, 300 MHz in the PMAC3-style IC) and compared to the PWM command value for each phase at this same frequency so the output command lines can be turned on and off with very high time resolution (~8 nsec in the PMAC2-style IC, ~3 nsec in the PMAC3-style IC).

Current Sampling

By sampling the actual current value on the rising edge of the phase clock, the "sawtooth" current waveform is sampled at the middle of a sawtooth edge. This samples it at its average value, and as far away from the PWM switching transients as possible.

The drive designer should be very careful in introducing techniques such as significant analog filtering, oversampling and averaging, and long sampling windows. While these may be effective in many applications, because of the sawtooth waveform and the switching transients, these can easily lead to worse performance than a single fast sample at the clock edge.

Relationship of Hardware PWM Frequency to Software Sampling Frequency

While the relationship between the hardware PWM frequency and the controller's software sampling frequency requires tight coupling, a variety of choices can be made as to what this relationship is. The decision is made by the user in the controller setup, but the amplifier designer must understand the relationship so any necessary constraints or limitations on the user's choices can be properly documented.

At most, there can be two software samples (control cycles) per PWM period, because there can ultimately be only two decision points in a PWM period – when to turn the signal on, and when to turn it off. Expressed in terms of frequencies, the minimum PWM frequency is one-half of the software sampling frequency. (Alternately stated, the maximum software sampling frequency is twice the PWM frequency.) This is the case shown in the above diagram.

However, several other choices are possible. The valid relationships are expressed by the equation:

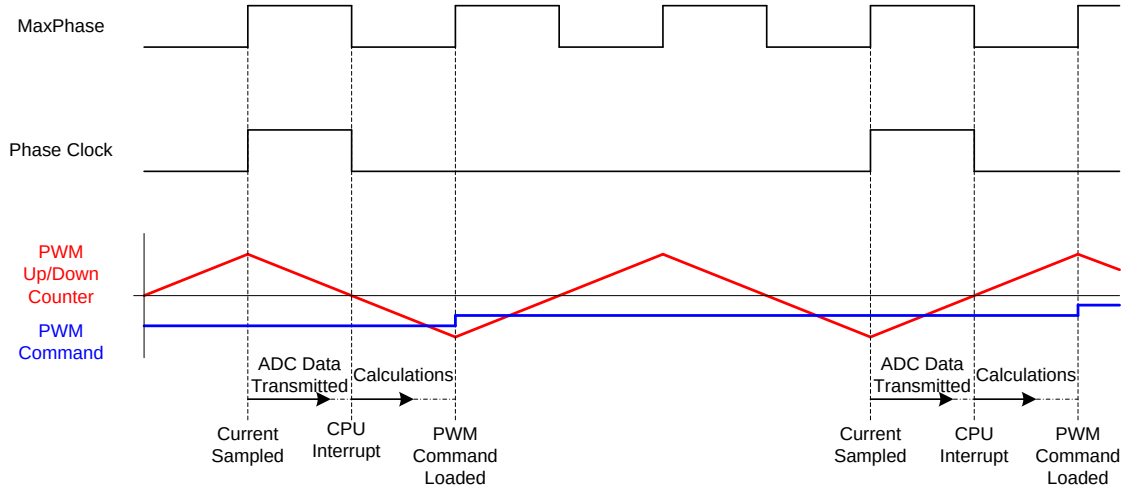
$$f_{PWM} = \frac{n+1}{2} f_{Phase}$$

where f_{Phase} is the software update frequency, and n is a non-negative integer.

While the maximum permitted PWM frequency for an amplifier is often determined by the thermal limits of switching the power transistors, there may also be limitations due to calculation and/or signal propagation delays. The exact nature of these limitations is slightly different with PMAC2 and PMAC3 Servo ICs. (PMAC2 Servo ICs can be used in both Turbo PMAC and Power PMAC systems, but PMAC3 Servo ICs can only be used in Power PMAC systems.)

PMAC2 Servo ICs

In a PMAC2 Servo IC, setting the PWM frequency (for all four channels simultaneously) automatically sets the frequency of an internal clock signal called "MaxPhase" to twice the PWM frequency. The "phase clock" signal itself, which specifies the actual software sampling frequency, is divided down by a programmable integer factor from MaxPhase. When the phase clock has a lower frequency than MaxPhase, the relationship between hardware signals and software samples is shown in the following diagram:



PMAC2 IC Direct PWM Timing

In this diagram, the phase clock is divided down in frequency by a factor of 3 from the internal MaxPhase signal. The PWM frequency is therefore 3/2 that of the phase clock, so the current is sampled every 1.5 PWM cycles.

Note that the current is sampled one-half of a MaxPhase clock cycle before the CPU interrupt that starts the current loop calculations, so the resulting digital data must be fully transferred to the controller in one-half of a MaxPhase clock cycle, which is one-quarter of a PWM cycle. Otherwise, the processor will not read valid data. Since this transfer takes 25 ADC clock cycles and must occur in one-quarter of a PWM cycle, the PWM frequency cannot be greater than 1% of the ADC clock frequency. For more details, refer to the section on the ADC signal interface, below.

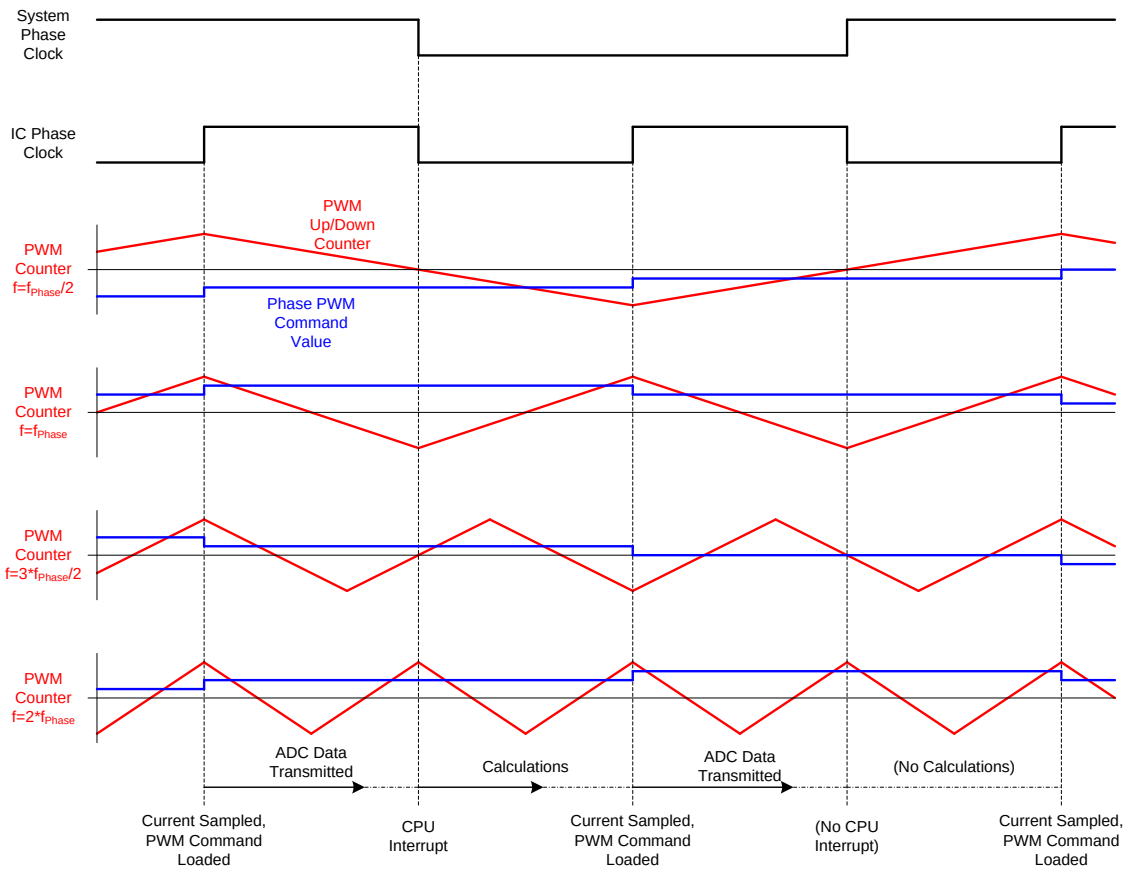
Note also that the current-loop and commutation calculations by the processor should be completed within one-half cycle of MaxPhase, which is one-quarter of a PWM cycle. If this does not happen, the commanded PWM output values will not be used for another full MaxPhase cycle, which is another one-half PWM cycle. While not disastrous, this delay can lead to a significant reduction in performance.

PMAC3 Servo ICs

In a PMAC3 Servo IC, there is a further level of flexibility possible. The frequency of the phase clock signal internal to the IC can be higher (by a factor of 2, 3, or 4) than the frequency of the system phase clock signal that interrupts the CPU. Additionally, the frequency of the PWM cycle for each of the four channels in the IC can be set (independently) according to the equation:

$$f_{PWM} = \frac{n+1}{2} f_{Phase}$$

where *n* can range from 0 to 7, so the PWM frequency can range from ½ to 4 times the IC's internal phase clock frequency. These relationships are shown in the following diagram:



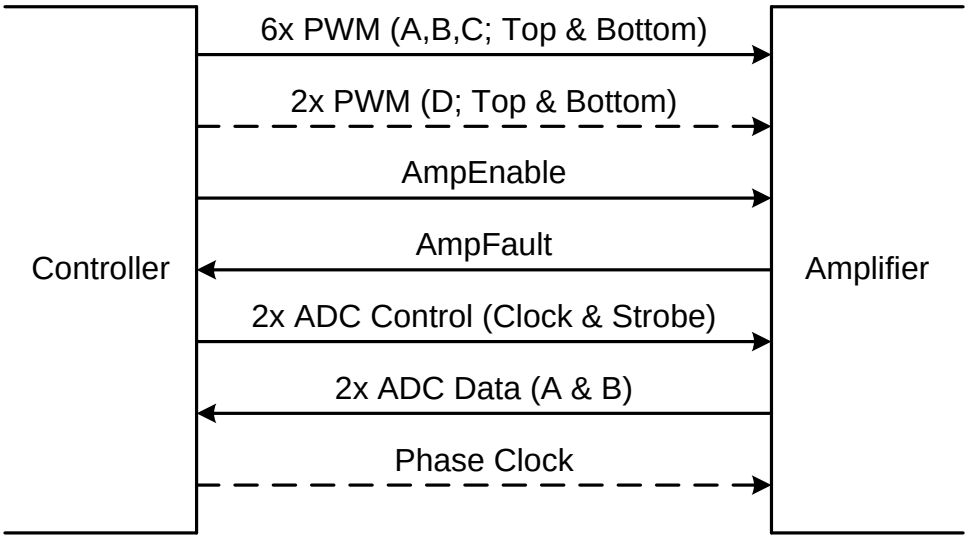
PMAC3 IC Direct PWM Timing

In the example shown in the diagram, the IC's internal phase clock has twice the frequency of the system phase clock signal that interrupts the CPU. The PWM frequencies shown for the four channels of the IC are set to 0.5, 1, 1.5, and 2 times the IC phase clock frequency.

One of the advantages of this system compared to that of the PMAC2 IC is that it is not always necessary for the data from the current ADCs to be transferred in one-quarter of a PWM cycle (but it must be transferred within half of the IC's internal phase clock cycle). This permits greater flexibility in supporting very high PWM frequencies, which can be important, especially with MOSFET power devices. For more details, refer to the section on the ADC interface, below.

Interface Signals

The direct PWM interface consists of 12 signals for an axis, with optional expansion to 15. In the cable interface, each signal is transmitted as a differential pair. Of these signals, 9 (optionally 12) are outputs from the controller, and 3 are outputs from the amplifier. The following diagram shows the basic signal interface. The dashed lines represent the optional signals for an enhanced interface, supported only by the newer PMAC3-style ICs.



Direct PWM Interface Signal Overview

Cable Signal Characteristics

All signals in both directions in the cable interface are transmitted as differential signal pairs at 5V RS-422 levels. It is strongly recommended that these be sent through twisted pairs in the cable. Standard cables conforming to the IEEE-1284C standard will provide the proper twisted pairs.

The controller provides a +5V supply and GND return on the cable, each with two pins. This can be used to power the "outer" side of an optically isolated interface in the amplifier.

Controller End

On the controller end, all signal pairs are transmitted with MC3487 or equivalent differential line drivers. Each output line has a 33-ohm in-line resistor.

On the controller end, all signal pairs are received with MC3486 or equivalent differential line receivers. There is a 220-ohm termination resistor between the two lines of a signal pair. There are no pull-up or pull-down resistors.

Amplifier End

On the amplifier end, if optical isolation is not desired at the interface, the interface circuitry can mirror that on the controller end, with MC3487 or equivalent differential line drivers and MC3486 or equivalent differential line receivers.

If optical isolation is desired at the interface, the amplifier can receive each signal pair across the anode and cathode of the LED of an opto-coupler. The opto-coupler should be "fast" for all PWM and ADC signals (HCPL-0631 or equivalent work well), and an in-line current-limiting resistor is required (330 ohms is often used). To match turn-on and turn-off times well, a load resistor in parallel with the LED is strongly recommended so that current also flows even when the LED is reverse-biased.

Note that the choice of optical isolation directly at the interface is a separate decision from whether these signal-level circuits in the amplifier are isolated from the power stage or not.

Inter-Board Stack Signal Characteristics

All signals in the inter-board stack interface in both directions are single-ended 5V CMOS signals. On the controller end, there are no pull-up or pull-down resistors on the inputs.

PWM Command Signals

The PWM command signals from the controllers are intended to be used directly as the on/off commands for the power transistors in the amplifier. (This is why the interface is called "direct PWM".) Each signal (sent as a differential signal pair in the cable interface) controls a separate power transistor.

The standard PWM interface provides 6 PWM signals for 3 top-and-bottom pairs of power transistors, each in a totem-pole half-bridge topology. Each top-and-bottom pair is labeled as a "phase", with the phases named as A, B, and C for a standard 3-phase motor. (In many motors, the phases will be named as U, V, and W.)

In the newest version of the interface, employing the PMAC3-style IC, signals for a fourth (D) phase top-and-bottom pair are provided. It is not necessary to use these signals. However, they can be used for driving two completely independent phases, each with a full H-bridge, or for other purposes such as smart software-based shunt control. Check to ensure that software support for a given use of the D-phase PWM signals is possible before implementing the use of this phase.

PWM Signal Generation

In Delta Tau controllers, the PWM signals for a phase are generated fully digitally in the Servo IC by numerical comparison of the computed signed phase command value to the instantaneous value of the up-down sawtooth counter. This comparison is performed at a very high frequency, almost 120 MHz in the PMAC2-style IC, providing an edge timing resolution of about 8 nanoseconds, and at 300 MHz in the PMAC3-style IC, providing an edge timing resolution of about 3 nanoseconds. Because the counter counts up and down (rather than up and resetting), it produces a "centered" PWM signal.

Switching and Deadtime

The process for a phase is shown graphically in the diagram **Direct PWM Hardware and Software Timing**, above. When the value of the counter passes the value of the phase command in the positive direction, the top transistor is commanded to turn off instantly, and the process to turn on the bottom transistor is started. However, the bottom transistor is not commanded to turn on until after a programmable deadtime.

When the value of the counter passes the value of the phase command in the negative direction, the bottom transistor is commanded to turn off instantly, and the process to turn off the top transistor is started. However, the top transistor is not commanded to turn on until after a programmable deadtime.

Some deadtime is required because the power transistors do not turn off and on instantaneously. Without deadtime, each switching cycle there would be periods in which both top and bottom transistors are at least partially on simultaneously, providing a direct path from bus power to return. Even if only for a fraction of a microsecond each PWM cycle, this can quickly lead to dangerous heating.

With a PMAC2-style IC, the deadtime is programmable in increments of 135 nanoseconds ranging from 0 to 34.4 microseconds, common for all four channels of the IC. The default deadtime is 2.0 microseconds. With a PMAC3-style IC, the deadtime is programmable in increments of 53.3 nanoseconds ranging from 0 to 13.6 microseconds. The default deadtime is 0.8 microseconds.

While it is essential that a direct-PWM amplifier provide its own protection enforcing a minimum deadtime, this should be used only as a protective feature. Instructions for use of the amplifier should instruct the user to set a slightly larger deadtime than this protective minimum in the controller IC so that this IC actively sets the deadtime itself. Doing this maintains the high time-resolution of signal edges (3 – 8 nsec) and keeps the edges fully synchronous, eliminating dither.

Signed Command Value and Signal Duty Cycles

Note that a zero numerical command for a phase produces PWM duty cycles of 50% (minus deadtime) for both the top and bottom signals, for a net signed duty cycle for the phase of 0%. This means that there is no special condition for the signed 0% command, minimizing zero-crossing distortion.

As the command to the phase increases in the positive direction, the on-time to the top transistor grows symmetrically about the center, and the on-time to the bottom transistor shrinks symmetrically about its center. This is shown in the diagram below.

At a 20% positive phase command, the top transistor is commanded on for 60% of the PWM cycle (minus the programmed deadtime), and the bottom transistor is commanded on for 40% (minus the programmed deadtime).

At a 40% positive phase command, the top transistor is commanded on for 70% of the PWM cycle (minus the programmed deadtime), and the bottom transistor is commanded on for 30% (minus the programmed deadtime).

At a 60% positive phase command, the top transistor is commanded on for 80% of the PWM cycle (minus the programmed deadtime), and the bottom transistor is commanded on for 20% (minus the programmed deadtime).

At an 80% positive phase command, the top transistor is commanded on for 90% of the PWM cycle (minus the programmed deadtime), and the bottom transistor is commanded on for 10% (minus the programmed deadtime).

As a 100% positive phase command is approached, the bottom transistor on-time is reduced to zero. At a 100% positive phase command, the top transistor is commanded on for 100% of the PWM cycle, and the bottom transistor is always off.

For negative phase commands, the algorithm works as described above, except with the roles of top and bottom transistors reversed.

PWM Frequency Specification

The PWM frequency is set by parameters in the Servo IC, which specify the period of the cycle of the PWM up-down counter. The IC can specify a very broad range of frequencies with very small increments between settings, so it will not be the limiting factor in setting the range of frequencies.

The manual for a direct-PWM amplifier should specify the minimum and maximum frequencies at which it can operate. Most likely, the maximum frequency will be set by thermal limits of the power transistors, but there can be a limit based on the ADC clock frequency and how fast the ADC information is transferred to the controller.

PWM Maximum Duty Cycle Specification

In PMAC controllers, the maximum output net duty cycle is set by the ratio of the software parameter that scales the numerical PWM output commands (the "PWM scale factor") to the limit of the PWM up/down counter. In PMAC2 Servo ICs, this limit is proportional to the PWM period, and so inversely proportional to the PWM frequency; in PMAC3 Servo ICs, this limit is fixed at 16,384.

If the PWM scale factor is set exactly equal to the counter limit value, a ratio of 1.0 (100%), then it is possible to saturate the PWM outputs of the phases, full-on or full-off, although this would only happen at the peak of the commutation sine wave for an AC motor.

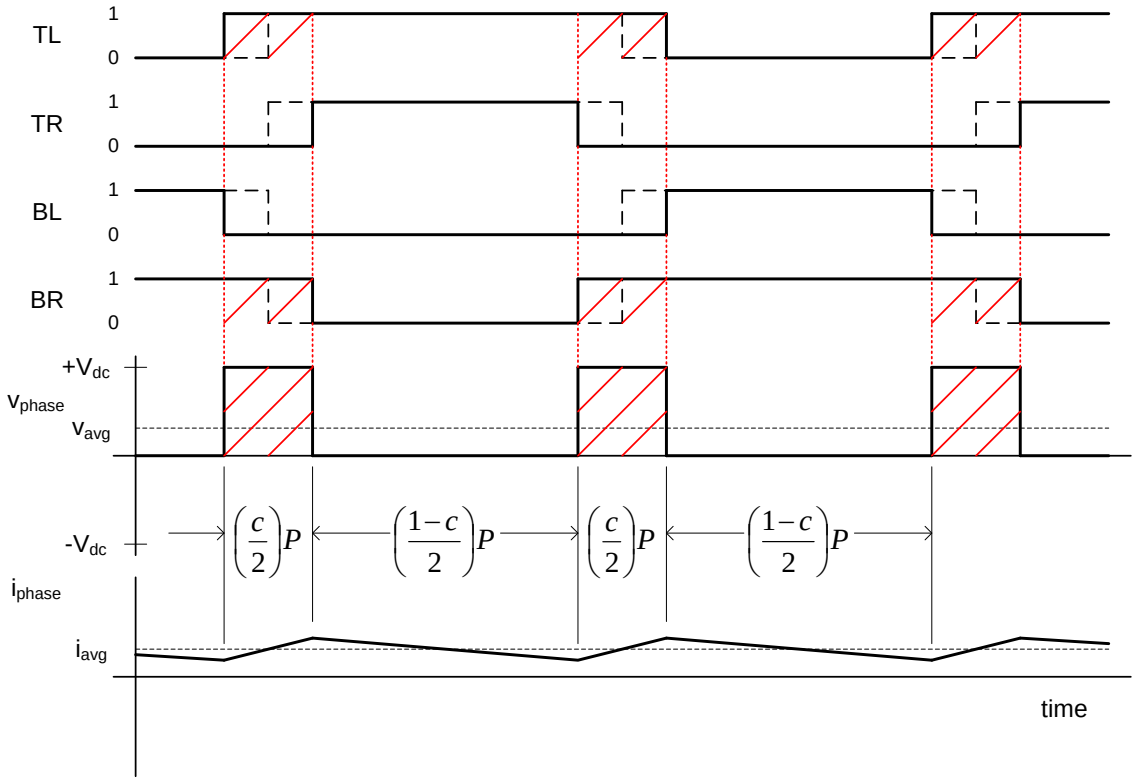
Some amplifiers may require that the PWM waveforms never saturate, limiting signal duty cycles, say, to the 5% - 95% range, in order to drive a charge pump for gate driver circuitry. Users of these drives should be instructed in how to limit their PWM scale factors so that these duty cycle values are never exceeded. Of course, it is essential that no damage occur to the amplifier if the user accidentally violates these constraints.

If it is permissible to saturate the PWM output signals, it can be useful when driving an AC motor (brushless servo motor or AC induction motor) to use a PWM scale factor somewhat greater than the counter limit. This permits the phase voltage waveform to saturate over an angle range on both sides of the peak, effectively turning the sinusoidal waveform into a trapezoidal waveform when the highest voltages are commanded. This in turn permits higher average voltage commands over the full cycle. If the ratio of PWM scale factor to counter limit is 1.1 (110%), saturation over about 1/6 of the commutation cycle around both peaks can be obtained.

Coordination of Multiple Amplifier Phases

In the Delta Tau controller ICs, the same digital sawtooth waveform is used in the comparison to the command values of all phases. Because of this common comparison waveform, the center of the "top on" period and the center of the "bottom on" period are the same for all phases. This produces what is often called "in-phase" PWM signal generation, which produces much lower current ripple, and therefore power dissipation, than the simpler "anti-phase" PWM scheme.

The following diagram shows the centered in-phase PWM waveforms on two amplifier phases (labeled L and R for left and right) driving a single motor phase. Below that, it shows the net voltage applied between the amplifier phases, and the resulting current waveform in the motor windings, whose rate of change is limited by the inductance of the windings. In the diagram, *P* is the time period of the PWM cycle (inversely proportional to the frequency), and *c* is the net signed duty cycle. For simplicity, the deadtime is not shown in this diagram.



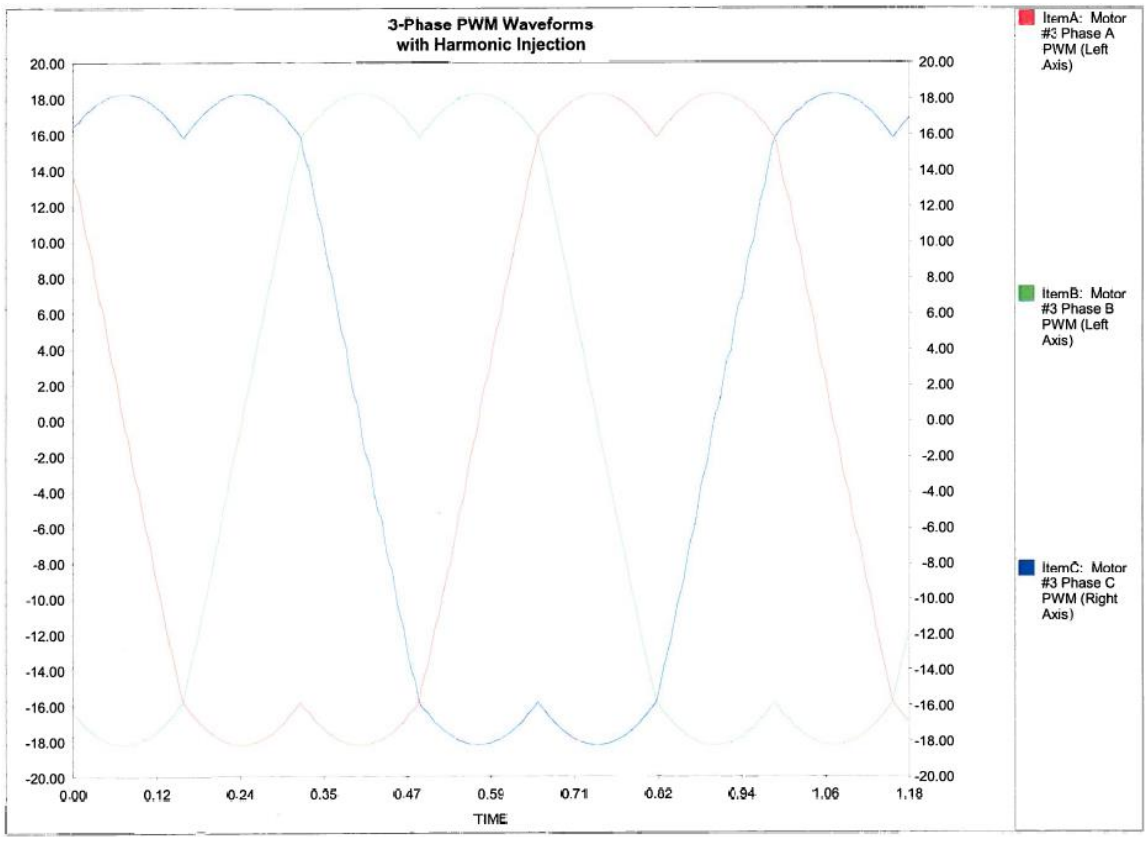
Centered In-Phase PWM Command, Voltage, and Current Waveforms

In this example, the top left (TL) on time has been expanded from its 50% base case (shown with the dashed line) symmetrically about the center. The same is true for the bottom right (BR). Now, twice per PWM cycle, these on times overlap and the bus voltage is applied across the motor winding in the "positive" direction (left to right). During the rest of the PWM cycle, there is zero net voltage applied across the motor winding. The top right (TR) and bottom left (BL) on times never overlap in this case, so the bus voltage is not applied across the motor winding in the "negative" direction (right to left) in any part of the PWM cycle.

Modified Sinusoidal Waveforms

By default for 3-phase motors, PMAC controllers compute waveforms that are modified sinusoids over the commutation cycle. These waveforms, with significant 3rd-harmonic components, permit 15% greater phase-to-phase voltages for the same DC bus voltage compared to unmodified sinusoidal waveforms. This technique is sometimes called "space-vector PWM". This modification can be turned off with a software "switch" in the PMAC.

The shape of these modified waveforms for the three phases can be seen in the following plot of steady-state operation from the numerical PWM phase commands computed by the PMAC controller:



The values plotted are effectively the phase-to-neutral voltage commands. Phase-to-phase voltages are purely sinusoidal.

Nothing in the amplifier design needs to be modified to accommodate these waveforms, but in characterizing the performance of the amplifier, the 15% increase in the effective available performance can be important to understand.

PWM Command Resolution

The effective resolution of the PWM output command is determined by the ratio of the clock frequency implementing the internal PWM up/down counter to the frequency of the PWM signal itself. Edges of the PWM output occur twice per PWM cycle, so the resolution of the output is determined by the number of possible time locations an edge could occur in in half of a PWM cycle. The effective bit resolution of the output is the log (base 2) of this number:

$$Res(bits) = \log_2\left(\frac{PwmClockFreq}{2 * PwmOutFreq}\right)$$

For example, using a PMAC2-style IC, with an internal PWM clock frequency of 118 MHz, the effective bit resolution of a PWM output of 20 kHz is:

$$Res(bits) = \log_2\left(\frac{118{,}000}{2*20}\right) = \log_2(2950) = 11.5bits$$

Using a PMAC3-style IC, with an internal PWM clock frequency of 300 MHz, the effective bit resolution of a PWM output of 20 kHz is:

$$Res(bits) = \log_2\left(\frac{300{,}000}{2*20}\right) = \log_2(7500) = 12.9bits$$

The amplifier designer should not process the PWM command signals through any "clocked" logic such as flip-flops before sending them on to the gate driver circuitry. Clocked logic in the amplifier would likely be of a lower frequency than those in the controller ICs generating the signals, reducing the effective bit resolution, and it will not be synchronous with the controller IC clock signals, introducing jitter.

It is acceptable to have protective logic in the drive, such as reset/disable inhibit signals and shoot-through/deadtime inhibit signals, be clocked (and at a lower frequency), but the final combination with the actual PWM command signal from the controller should not be clocked.

Amplifier Enable and Fault Signals

The direct-PWM interface supports amplifier-enable/amplifier-fault handshaking between the controller and the drive. While operation is technically possible without using this handshaking, operation without using these signals in their intended manner violates basic safety standards.

Amplifier-Enable Command Signal

The direct-PWM interface provides a digital "amplifier-enable" command signal (AENA) for each motor. The polarity of this signal cannot be set in software in the controller for safety reasons, and so is fixed. A high state on the AENA line for the single-ended interface, or a high state on the AENA+ and a low state on the AENA- line in the differential cable interface, is necessary for the amplifier to be enabled.

If the amplifier-enable line(s) is(are) in the disabled state (AENA low in the single-ended case, AENA+ low and AENA- high in the differential case), the amplifier should not be able to drive the motor no matter what the other command signals are. It is strongly recommended that none of the power transistors can be turned on at all if the amplifier-enable signal is in the disabled state. Note that it is possible that PWM signals can still be sent from the controller (probably at net 0% duty cycle) when the amplifier-enable signal is in the disable state.

Amplifier-Fault Feedback Signal

The direct-PWM interface provides a digital "amplifier-fault" feedback signal (FAULT) for each motor. The polarity of this signal in the controller is software programmable. In the factory default setting, a low level on the FAULT signal in the single-ended case, or a low level on the FAULT+ signal and a high level on the FAULT- signal in the differential case indicates that the amplifier is in a fault condition, and the controller will immediately disable the amplifier.

The amplifier should automatically disable itself in the event of a detected fault. The purpose of the amplifier-fault feedback to the controller is to permit the controller to take appropriate action

with regard to other aspects of the machine, such as other axes and process outputs, and to prepare the system for an orderly re-enabling.

Optional Fault Codes

The fault signal itself does not indicate the reason for the fault. In the direct-PWM interface, many amplifier designers use parts of one of the ADC data feedback words (low bits that are masked off from use as actual current feedback) for fault-code information. The direct-PWM standard does not define how this should be done, so the amplifier manufacturer has freedom to define the fault codes. The fault codes could be a set of individual bits, or the bits could be combined into a numerical value representing the code value.

Note

Some early direct-PWM amplifiers used four separate signal lines (pins 1, 2, 18, and 19) on the 36-pin connector for fault-code information. No present controllers support these signals.

Re-Enabling After a Fault

An explicit command is always required to re-enable the amplifier after a fault. The controller will not automatically re-enable the amplifier if and when the fault signal indicates that the fault condition is no longer present. Many amplifiers always indicate a fault whenever they are not enabled. This condition is satisfactory, but upon enabling, the fault signal must go away within about 100 µsec of enabling, or the controller will conclude there is a real fault on its next software scan, and immediately disable the drive again.

Current Feedback Signal Interface

The current feedback signal interface is designed so that standard analog-to-digital converters with Serial Peripheral Interface (SPI) communications can be used in the amplifier directly linked to the controller, but with the capability for enhancements through straightforward logic. In the speed and resolution range required by the digital current loops that are closed in the controller, successive-approximation ADCs are most commonly used.

The SPI interface consists of separate clock, strobe and data lines. Each is covered below, followed by the exact relationship between the signals for each type of Servo IC. The two data lines will provide numeric values for the currents in the A and B phases of the motors. These must properly match the A and B PWM command phases.

Interface Clock Signal

The clock signal is always active in the SPI interface, not just when active data transfers are taking place.

There are several considerations for specifying the serial clock frequency for the operation of this SPI interface. One bit of data is transmitted per ADC per clock cycle.

The “ADC clock” frequencies in the Delta Tau ICs are programmable, permitting the setting of frequencies in increments that are a factor of 2 from each of the neighboring frequencies. When using a PMAC2-style IC, the relevant frequencies that can be used are 2.46 MHz and 4.92 MHz. When using a PMAC3-style IC, the relevant frequencies that can be used are 3.125 MHz and 6.25 MHz. While both ICs can be programmed for higher frequencies, the driver and receiver circuits

cannot reliably support these, especially for the cable interfaces. Lower frequencies are possible as well, but typically provide for too slow of a data transfer, limiting performance.

With the PMAC2-style IC, the transmission of a set of data, which takes 25 ADC clock cycles, must be completed in one-quarter of a PWM cycle, so a full PWM cycle must take at least 100 ADC clock cycles. This means that the maximum permitted PWM frequency is 1% of the ADC clock frequency. For example, if the 2.46 MHz ADC clock frequency is used, the maximum PWM frequency that can be used is 24 kHz.

With the PMAC3-style IC, this transmission of data must be completed in one-half of a phase clock cycle for the IC. Typically, this is also one-quarter of a PWM cycle, but the IC does provide additional flexibility as discussed above in the section *Relationship of Hardware PWM Frequency to Software Sampling Frequency*.

In the PMAC2-style IC, if the transmission is not completed in time, the processor will read erroneous data values. In the PMAC3-style IC, if the transmission is not completed in time, the processor will read valid data from the previous sample cycle. (This can be harder to detect as a problem in operation.)

Interface Strobe Signal

The "strobe" signal initiates the data transfer. In the Delta Tau Servo ICs, the strobe signal is created by clocking out a programmable 24-bit "strobe word" through a shift register, one bit per clock cycle, most significant bit (MSB) first. For the current feedback data, this occurs immediately following the rising edge of the phase clock signal.

The initial rising edge of the strobe signal, set by the first "1" in the strobe word, starts the data transfer process. With most ADCs, it directly triggers the "sample-and-hold" function in the ADC, freezing the level of the analog signal input at that moment.

The rest of the programmable strobe word determines the level of the strobe signal for the next 23 ADC clock cycles. Many ADCs do not care what happens to the strobe signal once the conversion has started. If this is the case (or if intervening logic can provide a continually high signal to the ADC during the conversion), then intermediate bits of the strobe word can be set and cleared to send other information or commands to the drive. This functionality is commonly used, and the specifications of what those bits mean can be unique to an amplifier.

Remember, however, that the strobe word is common to all 4 axes driven by the IC. When using the inter-board stack connection for multiple amplifiers in a single module, it is easy for each amplifier to "know" which axis it is, and pick out particular bits of the strobe word for special commands. However, this is more difficult for individual amplifiers on a cable connection.

Interface Data Signals

The digital data values from the amplifier are returned as two serial words, each on their own signal line, returned simultaneously. These are shifted, most significant bit first, into registers in the Servo ICs that are latched on the falling edge of the phase clock for immediate use by processor software in closing the current loops.

The values sent should be signed numerical binary (two's complement) values proportional to the current measured in the A and B phases. The values must be transferred so that the MSB of the

current-feedback data (the sign bit) for the phase ends up in the MSB of the register in the IC. The timing required to produce this result is discussed below for each IC.

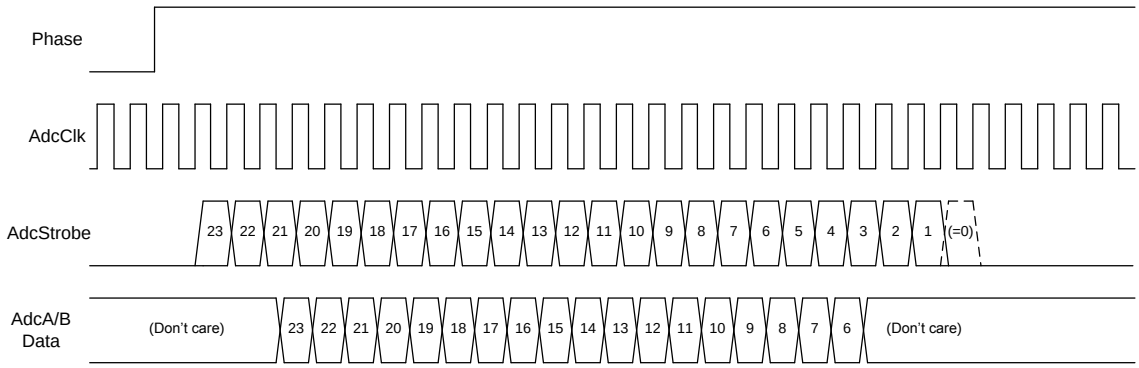
The polarity of these numerical values can be in either sense, that current "into" the phase can produce either a positive or negative numerical value, but the sense for both phases must be the same. Software setup of a parameter in the controller will be different depending on the polarity of these phase current readings.

The current readings must be scaled so that the full useful range of measured currents, including possible momentary overcurrent values, can be reported within the numerical range of the data word. The number of bits of current data reported is up to the discretion of the amplifier designer. It is usually either 12 or 14 bits of data, but the interface supports up to 20 bits. (Note that the PWM output resolution at typical frequencies means that feedback resolutions greater than 14 bits are usually not valuable.) The user of the amplifier will need to know the amperage of the full-range but unsaturated current measurement, as this provides the critical scaling value for all of the current-loop parameters.

The software reading these data words for current feedback permits the masking off of parts of the register that do not contain numerical current data. This means that the less significant parts of these registers can be used for sending other data from the amplifier at the discretion of the designer. This other data can be discrete status and error bits, or it can be other numerical data such as temperature and bus voltage.

Basic PMAC2-Generation Interface

The simplest (but least flexible) interface was provided in the original PMAC2-style "DSPGATE1" Servo IC, introduced in 1994. Standard SPI ADCs with signed data output and no "header" bits can be connected to the interface with no intervening logic. The timing diagram for this interface is shown here:



Original (Basic) PMAC2-Generation Interface ADC Timing Diagram

There is a continually running clock output (AdcClk) from the controller. It is of programmable frequency, but typically run at 2.46 or 4.92 MHz. This is the bit clock for the serial data transfers.

There is a much lower-frequency "phase clock" signal (internal to the ASIC) that triggers the ADC sampling process. It is also of programmable frequency, typically in the range of 10 – 20 kHz. The rising edge of the phase clock signal starts the sampling. Note that this edge is coincident with the middle of the length of the current "sawtooth waveform" so the sampling is at the average value of the current, and away from the switching transients.

On the second rising edge of the high-frequency AdcClk signal after Phase goes high, the IC starts outputting the programmable 24-bit "strobe word" through a shift register on the AdcStrobe signal, most significant bit (MSB) first. The timing diagram shows how bits 23 through 0 of the 24-bit word are output.

Most commonly, bits 23 through 1 of the strobe word are 1, and bit 0 is 0. This makes the strobe word value FFFFFE(hex). The rising edge caused by the bit 23 value of 1 starts the conversion, and the bit 0 value of 0 resets the line for the next cycle.

On the rising edge of AdcClk that outputs the 4th bit of the strobe word (bit 20), the IC will latch in the MSB of each of the serial data words from the two phase-current ADCs. These bits should have been introduced to the signal line on the previous falling edge of the AdcClk signal, so they have one-half of a clock cycle to stabilize. These data bit values will end up in the most significant bit of the ADC data registers in the IC, where they will be treated as the sign bits for the current values.

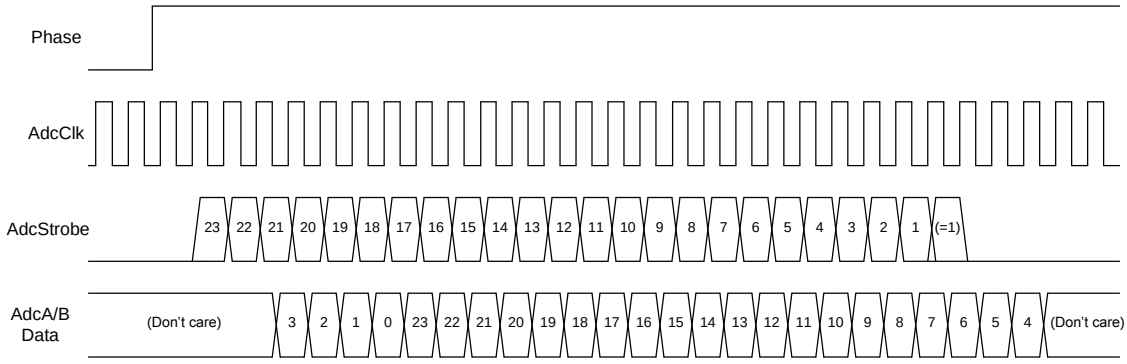
For the next 17 AdcClk cycles, this process is repeated, so up to 18 bits of current feedback data can be returned in two's-complement binary format. It is more typical that only 12 or 14 bits of current feedback is used. The current-loop closure algorithm permits the masking out of low bits of the returned word, so low bits can used to provide status and fault information.

Because ICs limited to this basic interface (revisions A, B, and C of the DSPGATE1 IC) are no longer sold, it is not necessary to limit the interface to just this specification. However, all newer IC interfaces can be set up to be compatible with this basic interface.

Enhanced PMAC2-Generation Interface

Revision D of the PMAC2-style "DSPGATE1" Servo IC, introduced in 2002, provides an enhanced ADC interface with more flexibility. If the LSB (bit 0) of the programmable strobe word is set to 0, it operates identically to earlier revisions of the IC, as described above.

However, if bit 0 of the programmable strobe word is set to 1, the timing of the interface is changed to support "header bits" from the ADCs and/or clock cycles of delay through intervening logic, permitting more flexibility and enhancements in the amplifier's interface. The timing diagram for this interface is shown here:



Modern (Enhanced) PMAC2-Generation Interface ADC Timing Diagram

The difference in this interface is that the first four data bits latched from each ADC line are "rolled over" to the low four bits of the 24-bit register in a "barrel shift" operation. The next 20

bits are latched into the high 20 bits of the register, so all 24 bits of the register can potentially be used for some data.

By delaying the time for latching in the MSB of real ADC data by four clock cycles, this interface mode supports a combination of header bits from the ADC and clock-cycle delays from intermediate logic totaling up to four. If there are less than four of these, the beginning of the strobe signal is delayed by setting one or more of the MSBs of the strobe word to 0.

For example, if the ADCs provide one header bit, but there are no added delays, there is a one clock cycle delay in returning the real ADC data. In this case, the start of the ADC strobe signal must be delayed by 3 clock cycles, so the strobe word is set to 1FFFFF(hex).

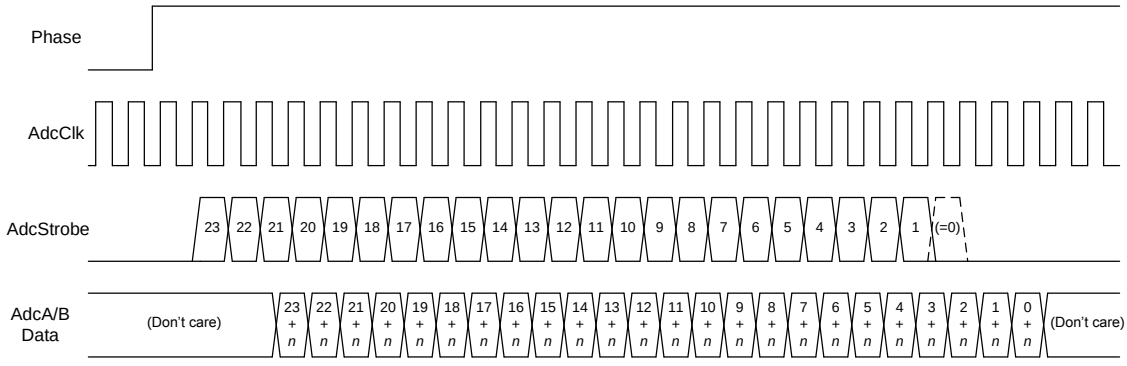
If the ADCs provide one header bit, and the intermediate logic delays the returned data by one clock cycle, there is a net two clock cycle delay in returning the real ADC data. In this case, the start of the ADC strobe signal must be delayed by 2 clock cycles, so the strobe word is set to 3FFFFF(hex).

If the ADCs do not provide any header bits, and there are no intermediate logic delays, the start of the ADC strobe signal must be delayed by 4 clock cycles in this mode, so the strobe word is set to 0FFFFF(hex). This setting produces identical results to a setting of FFFFFE(hex) in the basic mode (see above).

Note that in this mode of operation, even though the last bit of the ADC strobe word is one, the strobe signal is automatically cleared to zero (low) after this clock cycle.

PMAC3 Generation Interface

The PMAC3-style "DSPGATE3" IC, introduced in 2009 with the Power PMAC controller (it is not compatible with the older PMAC and Turbo PMAC controllers), provides additional flexibility and enhancements in the ADC interface, but can be used in a backward-compatible way. The timing diagram for this interface is shown here:



PMAC3-Generation Interface Main ADC Timing Diagram

There is a continually running clock output (AdcClk) from the controller. It is of programmable frequency, but typically run at 3.125 or 6.25 MHz. This is the bit clock for the serial data transfers.

In this interface, the number of header bits and clock cycle delays (*n* in this diagram) can be specified by the user in a separate IC software parameter as a value from 0 to 7. The first *n* data

bits latched by the IC are rolled over to the bottom of the 24-bit register in a "barrel shift" operation. (In the diagram, if the sum of the number and *n* is greater than 23, subtract 24 from the sum to obtain the bit number in the register where this rollover puts the value.)

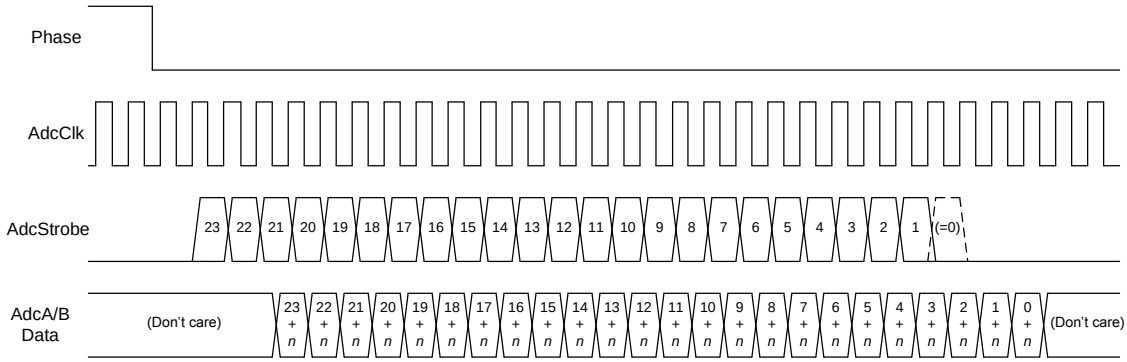
In virtually all cases, the MSB (bit 23) of the 24-bit ADC strobe word should be set to 1 to start the ADC conversion with the timing correct to get the MSB of the actual ADC data in to the MSB of the data register.

As with the modern PMAC2-style interface, the value of *n* should be the sum of the number of header bits in the data stream and the number of clock-cycle delays in intermediate logic. However, with the PMAC3-style interface, because the value of *n* is directly programmable, there is no need to delay the start of the strobe signal based on the value of *n*. A value for *n* of 0 (no header bits or delays) is compatible with the original "basic" PMAC2 IC interface.

Alternate Data

While the PMAC2-style interface leaves the ADC interface inactive while the phase clock is low, the PMAC3-style interface performs another sampling on the falling edge of the phase clock, bring the received data into alternate registers for the channel. The intended purpose of this feature is to support the gathering of additional information from the amplifier in what would otherwise be unused time.

The following diagram shows the timing for this interface, identical to the one above, except the process starts on the falling edge of the phase clock instead of the rising edge.



PMAC3-Generation Interface Alternate ADC Timing Diagram

Common data obtained with this feature from amplifier ADCs include bus voltage and power-module temperature. Other data sources, not necessarily of analog origin, are also possible. There is no standard as to the meaning of the data obtained with this feature.

Note that the falling edge of the phase clock interrupts the processor so it will start its commutation and current-loop software update. The data sampled here will not be ready in time for this cycle's software, so it should not be used for time-critical data.

For the amplifier to use the two half-cycles of the phase clock for different data sources, it must use the phase-clock output signal from the controller to determine what is sampled. The phase current values must be sampled on the rising edge of phase clock, and the alternate values must be sampled on the falling edge of phase clock.

If the phase clock signal is not used (as with interfaces designed for the PMAC2-style IC), the phase current values will be sampled on both the rising edge and the falling edge of phase clock. Only the values sampled on the rising edge will be used as current feedback. The values sampled on the falling edge will typically not be used at all (although they could provide an indication of how big the PWM-induced current ripple is, because they will be sampled near the peaks of the current ripple).

Common Drive Configurations

The direct-PWM interface can support several configurations of amplifier phases and motor windings.

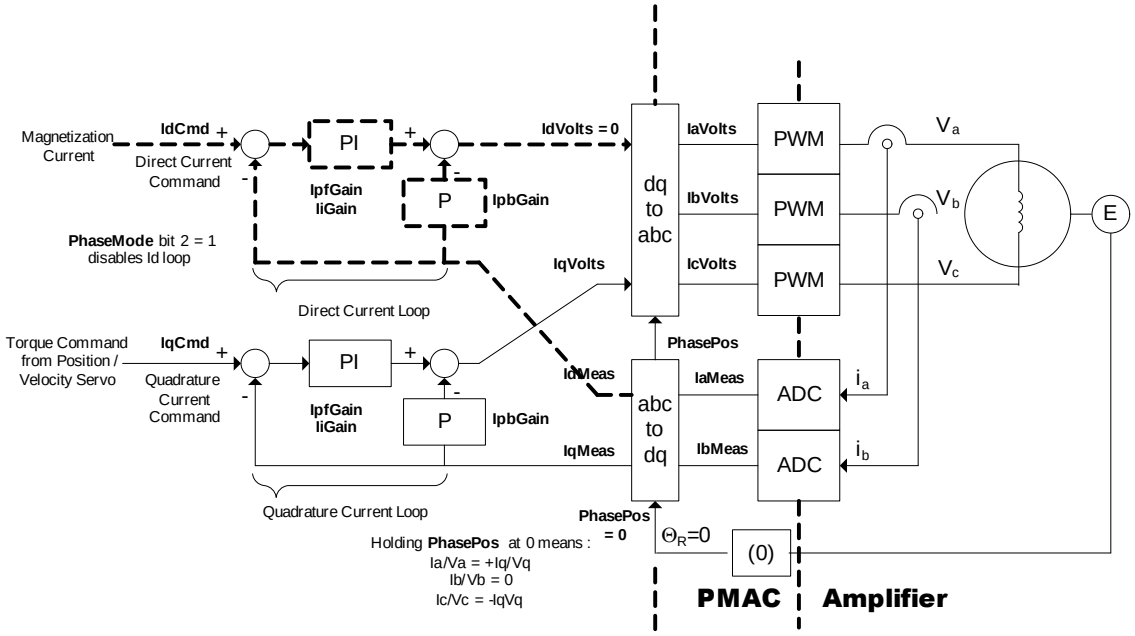
Three-Phase Motors

The most common configuration for using direct PWM amplifiers has 3 half bridges driving a 3-phase brushless motor. The 3 phases in the motor can be connected in either a Y (star) configuration or a Delta configuration. The motor can be a permanent-magnet brushless servo motor or an AC induction motor. No hardware changes to the amplifier are necessary to control any of these types of motors, as all changes are done in software in the controller.

DC Brush Motors

Direct-PWM amplifiers, even if designed for 3-phase motors, can be used to drive permanent-magnet DC brush motors. The motor armature should be connected between the A and C phases of the amplifier, and use the A-phase current feedback. The B-phase PWM command and current feedback are not used in this mode of operation, even if the hardware is physically present.

The following block diagram shows how the PMAC accomplishes this direct-PWM control of a DC brush motor:



Direct PWM Control of DC Brush Motors

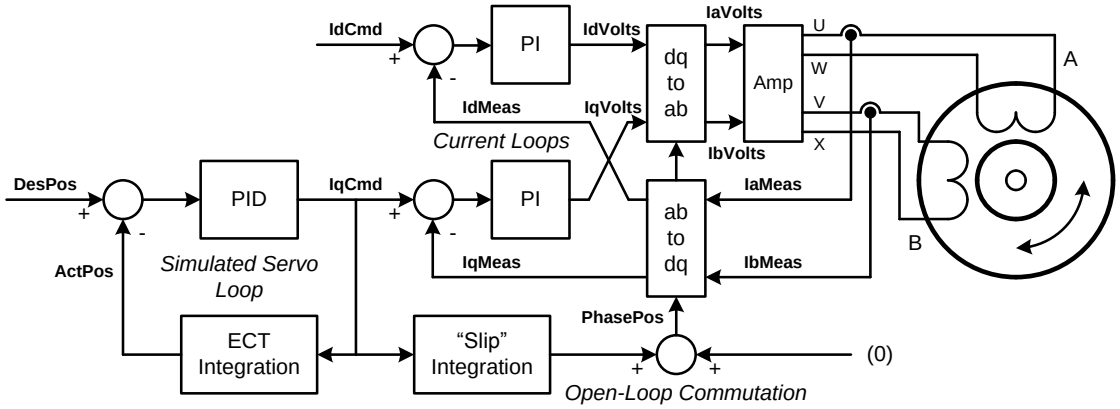
Two-Phase Motors

Direct PWM control of two-phase motors is also possible. The most common motors of this type are stepper motors with two independent phases. This requires four half bridges in the power-block amplifier. Since most direct-PWM interfaces do not provide four independent phase commands due to either hardware or software limitations, some of the PWM commands must be synthesized in the amplifier logic.

Most schemes for this PWM signal synthesis use the A and B phase top and bottom signals directly for the "left" half-bridge on each motor phase. To synthesize the PWM signals for the "right" half-bridge of a phase, the best technique is to delay the matching "left" signal by one-half PWM cycle. This maintains the "in-phase" PWM scheme that minimizes current ripple and resulting heating.

A simpler technique is to use the "left" half-bridge signals inverted for the "right" half-bridge – that is, the top left PWM on/off signal is also used for the bottom right transistor, and the bottom left signal is also used for the top right transistor. However, this technique provides "anti-phase" PWM, which leads to much higher current ripple and resulting heating.

The following block diagram shows how the PMAC accomplishes this direct-PWM control of a 2-phase stepper motor without position feedback:



Direct PWM Control of 2-Phase Stepper Motor

Protective Logic and Circuitry

Because several of the settings for the direct-PWM interface are programmable in software in the controller, it is of course possible that they can be set incorrectly at times. It is therefore essential that the amplifier have protective logic and/or circuitry that prevents the amplifier from being damaged by any incorrect settings.

Each top and bottom power-transistor pair must be protected by lockout circuitry that prevents shoot-through and insufficient deadtime in transitions. Typically a minimum deadtime is enforced each transition (but if the programmed deadtime is greater than the minimum, the commands from the controller should be used directly).

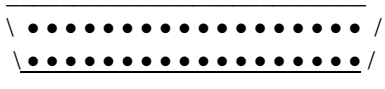
Since it is possible to set the PWM frequency in the controller too high for the amplifier, the amplifier should enforce a maximum PWM frequency, shutting down on a fault if the provided frequency is too high.

Standard power-circuit protections that come with any amplifier should be provided in a direct-PWM amplifier, including over-voltage, over-current, short-circuit, and over-temperature detection. It is strongly recommended that some sort of integrated current detection and limiting be provided in the amplifier. While the PMAC controllers provide "I^2T" protection for direct-PWM amplifiers, it cannot be guaranteed that this protection will be set properly for the particular amplifier, especially during initial setup.

Cable Connection Pinout

36-pin Mini-D Connector

PIN#	SYMBOL	FUNCTION	DESCRIPTION	NOTES
1	PHASE+	COMMAND	PHASE CLOCK SIGNAL	SYNCHRONIZING*
2	*RESERVED*			
3	ADC_CLK+	COMMAND	A/D CONVERTER CLOCK	
4	ADC_STB+	COMMAND	A/D CONVERTER STROBE	
5	CURRENTA+	FEEDBACK	PHASE A ACTUAL CURRENT DATA	SERIAL DIGITAL
6	CURRENTB+	FEEDBACK	PHASE B ACTUAL CURRENT DATA	SERIAL DIGITAL
7	AENA+	COMMAND	AMPLIFIER ENABLE	HIGH IS ENABLE
8	FAULT+	FEEDBACK	AMPLIFIER FAULT	HIGH IS FAULT
9	PWMATOP+	COMMAND	PHASE A TOP CMD	HIGH IS ON COMMAND
10	PWMABOT+	COMMAND	PHASE A BOTTOM CMD	HIGH IS ON COMMAND
11	PWMBTOP+	COMMAND	PHASE B TOP CMD	HIGH IS ON COMMAND
12	PWMBBOT+	COMMAND	PHASE B BOTTOM CMD	HIGH IS ON COMMAND
13	PWMCTOP+	COMMAND	PHASE C TOP CMD	HIGH IS ON COMMAND
14	PWMCBOT+	COMMAND	PHASE C BOTTOM CMD	HIGH IS ON COMMAND
15	GND	COMMON	REFERENCE VOLTAGE	
16	+5V	POWER	+5V POWER	FROM CONTROLLER
17	PWMDTOP+	COMMAND	PHASE D TOP CMD	HIGH IS ON COMMAND
18	PWMDBOT+	COMMAND	PHASE D BOTTOM CMD	HIGH IS ON COMMAND
19	PHASE-	COMMAND	PHASE CLOCK SIGNAL	SYNCHRONIZING*
20	*RESERVED*			
21	ADC_CLK-	COMMAND	A/D CONVERTER CLOCK	
22	ADC_STB-	COMMAND	A/D CONVERTER STROBE	
23	CURRENTA-	FEEDBACK	PHASE A ACTUAL CURRENT DATA	SERIAL DIGITAL
24	CURRENTB-	FEEDBACK	PHASE B ACTUAL CURRENT DATA	SERIAL DIGITAL
25	AENA-	COMMAND	AMPLIFIER ENABLE	LOW IS ENABLE
26	FAULT-	FEEDBACK	AMPLIFIER FAULT	LOW IS FAULT
27	PWMATOP-	COMMAND	PHASE A TOP CMD	LOW IS ON COMMAND
28	PWMABOT-	COMMAND	PHASE A BOTTOM CMD	LOW IS ON COMMAND
29	PWMBTOP-	COMMAND	PHASE B TOP CMD	LOW IS ON COMMAND
30	PWMBBOT-	COMMAND	PHASE B BOTTOM CMD	LOW IS ON COMMAND
31	PWMCTOP-	COMMAND	PHASE C TOP CMD	LOW IS ON COMMAND
32	PWMCBOT-	COMMAND	PHASE C BOTTOM CMD	LOW IS ON COMMAND
33	GND	COMMON	REFERENCE VOLTAGE	
34	+5V	POWER	+5V POWER	FROM CONTROLLER
35	PWMDTOP-	COMMAND	PHASE D TOP CMD	LOW IS ON COMMAND*
36	PWMDBOT-	COMMAND	PHASE D BOTTOM CMD	LOW IS ON COMMAND*



36-pin Mini-D Receptacle

Suggested Part Numbers:

3M: 10236-62x2VC (straight-through board mount)

3M 10136-52x2VC (right-angle board mount)

* Optional signal for enhanced interface

Inter-Board Stack Connection Pinout

50-pin Stack Connector

PIN#	SYMBOL	FUNCTION	DESCRIPTION	NOTES
1	PWMATOP1	COMMAND	AXIS 1 PHASE A TOP CMD	HIGH IS ON COMMAND
2	PWMBTOP1	COMMAND	AXIS 1 PHASE B TOP CMD	HIGH IS ON COMMAND
3	PWMABOT1	COMMAND	AXIS 1 PHASE A BOTTOM CMD	HIGH IS ON COMMAND
4	PWMBBOT1	COMMAND	AXIS 1 PHASE B BOTTOM CMD	HIGH IS ON COMMAND
5	PWMATOP2	COMMAND	AXIS 2 PHASE A TOP CMD	HIGH IS ON COMMAND
6	PWMBTOP2	COMMAND	AXIS 2 PHASE B TOP CMD	HIGH IS ON COMMAND
7	PWMABOT2	COMMAND	AXIS 2 PHASE A BOTTOM CMD	HIGH IS ON COMMAND
8	PWMBBOT2	COMMAND	AXIS 2 PHASE B BOTTOM CMD	HIGH IS ON COMMAND
9	PWMATOP3	COMMAND	AXIS 3 PHASE A TOP CMD	HIGH IS ON COMMAND
10	PWMBTOP3	COMMAND	AXIS 3 PHASE B TOP CMD	HIGH IS ON COMMAND
11	PWMABOT3	COMMAND	AXIS 3 PHASE A BOTTOM CMD	HIGH IS ON COMMAND
12	PWMBBOT3	COMMAND	AXIS 3 PHASE B BOTTOM CMD	HIGH IS ON COMMAND
13	PWMATOP4	COMMAND	AXIS 4 PHASE A TOP CMD	HIGH IS ON COMMAND
14	PWMBTOP4	COMMAND	AXIS 4 PHASE B TOP CMD	HIGH IS ON COMMAND
15	PWMABOT4	COMMAND	AXIS 4 PHASE A BOTTOM CMD	HIGH IS ON COMMAND
16	PWMBBOT4	COMMAND	AXIS 4 PHASE B BOTTOM CMD	HIGH IS ON COMMAND
17	PWMCTOP1	COMMAND	AXIS 1 PHASE C TOP CMD	HIGH IS ON COMMAND
18	ADC_A1	FEEDBACK	AXIS 1 PHASE A CURRENT	SERIAL DIGITAL DATA
19	PWMCBOT1	COMMAND	AXIS 1 PHASE C BOTTOM CMD	HIGH IS ON COMMAND
20	ADC_A2	FEEDBACK	AXIS 2 PHASE A CURRENT	SERIAL DIGITAL DATA
21	PWMCTOP2	COMMAND	AXIS 2 PHASE C TOP CMD	HIGH IS ON COMMAND
22	ADC_A3	FEEDBACK	AXIS 3 PHASE A CURRENT	SERIAL DIGITAL DATA
23	PWMCBOT2	COMMAND	AXIS 2 PHASE C BOTTOM CMD	HIGH IS ON COMMAND
24	ADC_A4	FEEDBACK	AXIS 4 PHASE A CURRENT	SERIAL DIGITAL DATA
25	PWMCTOP3	COMMAND	AXIS 3 PHASE C TOP CMD	HIGH IS ON COMMAND
26	ADC_B1	FEEDBACK	AXIS 1 PHASE B CURRENT	SERIAL DIGITAL DATA
27	PWMCBOT3	COMMAND	AXIS 3 PHASE C BOTTOM CMD	HIGH IS ON COMMAND
28	ADC_B2	FEEDBACK	AXIS 2 PHASE B CURRENT	SERIAL DIGITAL DATA
29	PWMCTOP4	COMMAND	AXIS 4 PHASE C TOP CMD	HIGH IS ON COMMAND
30	ADC_B3	FEEDBACK	AXIS 3 PHASE B CURRENT	SERIAL DIGITAL DATA
31	PWMCBOT4	COMMAND	AXIS 4 PHASE C BOTTOM CMD	HIGH IS ON COMMAND
32	ADC_B4	FEEDBACK	AXIS 4 PHASE B CURRENT	SERIAL DIGITAL DATA
33	ADC_CLK	COMMAND	A/D CONVERTER CLOCK	FOR ALL AXES
34	ADC_STB	COMMAND	A/D CONVERTER STROBE	FOR ALL AXES
35	AENA1	COMMAND	AXIS 1 AMPLIFIER ENABLE	HIGH IS ENABLE
36	FAULT1	FEEDBACK	AXIS 1 AMPLIFIER FAULT	USER SET POLARITY
37	AENA2	COMMAND	AXIS 2 AMPLIFIER ENABLE	HIGH IS ENABLE
38	FAULT2	FEEDBACK	AXIS 2 AMPLIFIER FAULT	USER SET POLARITY
39	AENA3	COMMAND	AXIS 3 AMPLIFIER ENABLE	HIGH IS ENABLE
40	FAULT3	FEEDBACK	AXIS 3 AMPLIFIER FAULT	USER SET POLARITY
41	AENA4	COMMAND	AXIS 4 AMPLIFIER ENABLE	HIGH IS ENABLE
42	FAULT4	FEEDBACK	AXIS 4 AMPLIFIER FAULT	USER SET POLARITY
43 - 50	(OTHER SIGNALS)	--	--	NOT PART OF DIRECT PWM INTERFACE