# Specification for ASIC Motor Control

This motor controller operates a two phase stepper motor which runs on a modified sine current waveform which has the same symmetry as a sine wave. Each motor has two phases which are 90 degrees out of phase. In order to simplify calculations all position moves are began from zero velocity. The ASIC design will contain two controllers one for pan and one for tilt. The system operates with adjustable values of jerk and acceleration to produce velocity and position. The output of the controller is a analog voltage signal which is then converted to winding current. At this time the voltage to current conversion is not covered.

## 1.0 Initialization

Maximum acceleration is set on powerup. Acceleration is limited to this value by hardware.

Maximum Velocity is set on powerup. Velocity is limited to this value by hardware.

Position is a 56 bit unsigned number with 24 bits of full steps, 8 bits of micro steps, and 24 bits of fraction. The 8 bits of micro steps index the lookup table to produce the PWM.

## 2.0 Position,Velocity, Acceleration, and Jerk Ranges

Position is a positive number which has a programmable wrap around point. If the system has 65536 micro steps in one revolution of the camera, then position will start at 0 and then go to 65535 and then back to 0 with positive velocity. Negative velocity will go move from 0 to 65535 and decrement next to 65534 and then down to zero. The wrap around point will match one full revolution of the camera.

Jerk, Acceleration, and Velocity are signed values, clockwise being positive and counterclockwise negative. Maximum velocity and maximum acceleration are contained in registers and absolute value of velocity and acceleration are limited to these maximums without any processor action except at the start when the registers are loaded.

## 2.1 Register formats

| | | | |
|---|---|---|---|
| **Position** | **xxxxxxxx xxxxxxxx xxxxxxxx** | **xxxxxxxx** | **xxxxxxxx xxxxxxxx xxxxxxxx 0** |
| | **24 bits full steps** | **8 bits of micro step** | **24 bits of fractional micro step** |
| **Velocity** | | **00xxxxxx** | **xxxxxxxx xxxxxxxx xxxxxxxx x** |
| | | **6 bits of micro step** | **25 bits of fractional micro step** |
| **Acceleration** | | | **00xxxxxx xxxxxxxx xxxxxxxx x** |
| | | | **23 bits of fractional micro step** |
| **Jerk** | | | **00xxxxxx xxxxxxxx xxxxxxxx x** |
| | | | **23 bits of fractional micro step** |

A value in the jerk register will move from jerk to acceleration to velocity to position in one cycle. Example jerk = 100 hex in one cycle acceleration=100 hex, velocity=100 hex and position=80 hex.

Absolute maximum velocity is 64 micro steps/cycle which is a value of 3FFFFFFF hex (1 sign bit 31 bits of magnitude, this is actually one count less than 64 micro steps/cycle but close enough.) Acceleration maximum value is 7FFFFF hex (1 sign bit 23 bits of magnitude). Jerk maximum value is 7FFFFF hex (1 sign bit 23 bits of magnitude).

**2.2 Cycle Time and Maximum Velocity**
A 0.9 degree/step Motor should be able to run at 1000 degrees/second with a gear ratio of 20 to 1, therefore the maximum speed should be (1000*20)/0.9 steps/second = 22,222 steps/second = 5,688,832 micro steps/second . Maximum velocity is 64 micro steps/cycle, therefore our cycle speed must be at least 88,888 cycles/second. With a 108 Megahertz clock we can output 10 bits of resolution (1024 clocks) at 105,468 cycles per second. Which the makes a maximum cycle speed of 6,750,000 micro steps/second (at 64 micro steps/cycle).

Velocity has 1 sign bit, 6 micro step bits and 25 bits of fraction. The least significant bit of the fraction is ignored when adding velocity to position. This makes the position fraction equal to 24 bits. Minimum velocity is (6,750,000 micro steps/second )/(31 bits magnitude) = 0.00314 micro steps/second

**2.3 Acceleration and Jerk**
Maximum acceleration is ¼ micro step/cycle/cycle = ¼*105468*105468(1/256) = 10,862,792 steps/second/second Minimum acceleration is maximum acceleration/(23 bits) = 165 step/second*second

Maximum jerk is equal to maximum acceleration in one cycle. A jerk value larger than acceleration makes no sense. Minimum jerk is 165 step/second*second/cycle = 17,481,642 step/second*second*second.

**3.0 Wait Register Comparisons**
When comparing with values of position the fraction part can be ignored (less than one micro step can be ignored). When the motor is stopped the fraction part should be flushed so that a new move start with zero fraction.

When comparing position, velocity and acceleration to targets, if value is ramping up the compare should be greater than or equal rather than just equal because the equal value may not be hit exactly. Ramping down should be less than or equal.

**4.0 Command Implementation**
3.1 Reading Values
All registers ( Position, Velocity, Acceleration, Jerk) can be read at any time by the processor.

**4.1 Hitting the targets**
All position moves will be done from zero velocity. Therefore if a position is commanded while the motor is moving the velocity is commanded to zero and then the move is made.

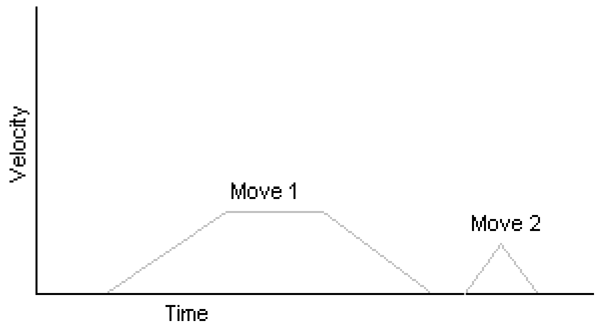
To command a target velocity the processor will set a wait on velocity compare. This compared velocity will be reached before the target velocity is reached. When the compared velocity is reached, jerk is set to a value which will drive the acceleration to zero. When zero acceleration is reached the target value of velocity will be reached. (If the processor has computed the wait velocity and jerk value correctly.)

To command a target position the processor will set to wait on a position which will be reached before the target position is reached. When the wait position is reached, jerk is set to a value which will drive the velocity to zero. When zero acceleration is reached the target value of velocity will be reached. (If the processor has computed the wait velocity and jerk values correctly.)

In order to insure that the target velocity or target position are reached it may be required to calculate target velocity or target position that have a small amount of velocity left when the target values are reached and to clamp the motor on the target. This may mean slightly overshooting the target instead of computing zero velocity at the exact target point. If the target position is undershot and the velocity will reverse before the target is reached. It would be better to slightly overshoot the target and clamp the motor to exactly the target when it is reached. As long as the velocity is small this will not be problem.

The processor will calculate distances and velocities in non-real time. Therefore it could have several milliseconds to make the calculations necessary for a position move or velocity changes.

Note that the end of the ramp up point is determined distance required to reach zero velocity when traveling the maximum velocity ( see move 1 below). For short moves the end of ramp up is equal to start of ramp down and maximum velocity is never reached (see move 2 below). (Note that the example is shown with very little or no jerk effect. The effect of jerk is to round off velocity changes.)



**4.2 Overriding exist commands in sequence buffer:**
A command in the sequence buffer can be overwritten. Example: The system is waiting on position to reach a value before changing the jerk input. A velocity command comes in and immediately interrupts the wait with a new jerk value and sets a new wait.

**4.3 Velocity command duration**
Velocity commands produce velocities which last until a new command changes the velocity, therefore if no new command comes in the velocity will keep the motor running indefinitely.

**4.4 Setting Position**

Position can be immediately set to a value by command. This will only be done by the processor during calibration when motion is stopped. It is used to establish the home position.

## 5.0 PWM Motor Output:

The output of the motion control system is two phases of PWM for each axis. A 10 bit value is converted to PWM by a digital accumulating comparator which produces the closest possible PWM state at all times. For example: values of 50% would be produced as a 101010101….. rather than series of 512 ones and then 512 zeros. The frequency contained in the output is therefore higher and easier to low pass filter. The low pass filter converts the PWM to analog voltage signal.

Hardware will produce a signal which indicates the direction of the current flow in each phase. Note that the two phases are 90 degrees out of phase.

To minimize the table size symmetry in a sine wave ( or sine wave like waveforms) can be use to reduce the table size by ¼ . The table contains the waveform from 0 to 90 degrees. 90 to 180 is gotten by indexing the table from 90 to 0 or in reverse. The value for 180 to 360 are gotten by reversing the current direction line and repeating the values form 0 to 180 degrees. The table size is therefore 256 * 10 bits = 2560 per table. Each motor needs one table.

## 6.0 Motion Equations

acceleration = jerk * time

velocity = jerk * ( time * time + time) / 2 )

position = jerk * ( ( time*time + time) / 2 + (time*time*time - time) / 6 )

| Time | Jerk | Accel | Velocity | Calc Vel | Position | Calc Position | H | I |
|---|---|---|---|---|---|---|---|---|
| t | jerk | acc +=jerk | vel +=acc | H * jerk | pos += vel | (I + H)*jerk | (t*t + t)/2 | (t*t*t - t)/6 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 2 | 1 | 2 | 3 | 3 | 4 | 4 | 3 | 1 |
| 3 | 1 | 3 | 6 | 6 | 10 | 10 | 6 | 4 |
| 4 | 1 | 4 | 10 | 10 | 20 | 20 | 10 | 10 |
| 5 | 1 | 5 | 15 | 15 | 35 | 35 | 15 | 20 |
| 6 | 1 | 6 | 21 | 21 | 56 | 56 | 21 | 35 |
| 7 | 1 | 7 | 28 | 28 | 84 | 84 | 28 | 56 |
| 8 | 1 | 8 | 36 | 36 | 120 | 120 | 36 | 84 |
| 9 | 1 | 9 | 45 | 45 | 165 | 165 | 45 | 120 |
| 10 | 1 | 10 | 55 | 55 | 220 | 220 | 55 | 165 |
| 11 | 1 | 11 | 66 | 66 | 286 | 286 | 66 | 220 |
| 12 | 1 | 12 | 78 | 78 | 364 | 364 | 78 | 286 |
| 13 | 1 | 13 | 91 | 91 | 455 | 455 | 91 | 364 |
| 14 | 1 | 14 | 105 | 105 | 560 | 560 | 105 | 455 |
| 15 | 1 | 15 | 120 | 120 | 680 | 680 | 120 | 560 |
| 16 | 1 | 16 | 136 | 136 | 816 | 816 | 136 | 680 |
| 17 | 1 | 17 | 153 | 153 | 969 | 969 | 153 | 816 |
| 18 | 1 | 18 | 171 | 171 | 1140 | 1140 | 171 | 969 |
| 19 | 1 | 19 | 190 | 190 | 1330 | 1330 | 190 | 1140 |
| 20 | 1 | 20 | 210 | 210 | 1540 | 1540 | 210 | 1330 |
| 21 | 1 | 21 | 231 | 231 | 1771 | 1771 | 231 | 1540 |
| 22 | 1 | 22 | 253 | 253 | 2024 | 2024 | 253 | 1771 |
| 23 | 1 | 23 | 276 | 276 | 2300 | 2300 | 276 | 2024 |
| 24 | 1 | 24 | 300 | 300 | 2600 | 2600 | 300 | 2300 |
| 25 | 1 | 25 | 325 | 325 | 2925 | 2925 | 325 | 2600 |
| 26 | 1 | 26 | 351 | 351 | 3276 | 3276 | 351 | 2925 |
| 27 | 1 | 27 | 378 | 378 | 3654 | 3654 | 378 | 3276 |
| 28 | 1 | 28 | 406 | 406 | 4060 | 4060 | 406 | 3654 |
| 29 | 1 | 29 | 435 | 435 | 4495 | 4495 | 435 | 4060 |
| 30 | 1 | 30 | 465 | 465 | 4960 | 4960 | 465 | 4495 |
| 31 | 1 | 31 | 496 | 496 | 5456 | 5456 | 496 | 4960 |
| 32 | 1 | 32 | 528 | 528 | 5984 | 5984 | 528 | 5456 |
| 33 | 1 | 33 | 561 | 561 | 6545 | 6545 | 561 | 5984 |
| 34 | 1 | 34 | 595 | 595 | 7140 | 7140 | 595 | 6545 |
| 35 | 1 | 35 | 630 | 630 | 7770 | 7770 | 630 | 7140 |
| 36 | 1 | 36 | 666 | 666 | 8436 | 8436 | 666 | 7770 |
| 37 | 1 | 37 | 703 | 703 | 9139 | 9139 | 703 | 8436 |
| 38 | 1 | 38 | 741 | 741 | 9880 | 9880 | 741 | 9139 |
| 39 | 1 | 39 | 780 | 780 | 10660 | 10660 | 780 | 9880 |
| 40 | 1 | 40 | 820 | 820 | 11480 | 11480 | 820 | 10660 |
| 41 | 1 | 41 | 861 | 861 | 12341 | 12341 | 861 | 11480 |
| 42 | 1 | 42 | 903 | 903 | 13244 | 13244 | 903 | 12341 |
| 43 | 1 | 43 | 946 | 946 | 14190 | 14190 | 946 | 13244 |
| 44 | 1 | 44 | 990 | 990 | 15180 | 15180 | 990 | 14190 |
| 45 | 1 | 45 | 1035 | 1035 | 16215 | 16215 | 1035 | 15180 |
| 46 | 1 | 46 | 1081 | 1081 | 17296 | 17296 | 1081 | 16215 |
| 47 | 1 | 47 | 1128 | 1128 | 18424 | 18424 | 1128 | 17296 |
| 48 | 1 | 48 | 1176 | 1176 | 19600 | 19600 | 1176 | 18424 |

## 7.0 External Motor Controller Interfaces

$I^2C$ for Trinamic

SPI

## 8.0 Internal Current Control

Average and Peak Internal Current Control

## 9.0 Processor Inteface

For a position move, the sequencer would have loaded into it the following commands which would clamp (ie affectively stop) the accumulator from moving to a new position:

Instruction (i): wait(position, ramp_down) // Accumulator waits for position= ramp_down
Instruction (i+1): set(jerk, ramp_down_jerk, Done=1) // Set jerk to ramp_down_jerk (Which will cause velocity=0, acceleration=0 and position= target all to happen at the same point in time).

Instruction (i): wait(position, target) // Accumulator waits for position target to be reached,
Instruction (i+1): set(velocity,0) // Set velocity to zero, (executed after position X is reached)
Instruction (i+2): set(accel,0) // Set acceleration to zero
Instruction (i+3): set(jerk,0,Done=1) // Set jerk to zero, Done=1 indicates it is the last instruction.

For a velocity move:
Instruction (i): wait(velocity, velocity_target) // Accumulator waits for velocity_target to be reached,
Instruction (i+1): set(accel,0) // Set acceleration to zero, (executed after position X is reached)
Instruction (i+2): set(jerk,0,Done=1) // Set jerk to zero, Done=1 indicates it is the last instruction.

The Done=1 only indicates it is the last instruction, and software still
needs to toggle the
Go/Start bit to get the sequencer to execute a new sequence out of
memory.

For hardware to automatically reset the appropriate counters, it would decode the command, wait(position or velocity, X, Done=1): If the cmd=wait and done=1, reset accel, and jerk, and conditionally reset velocity if waiting for position (if waiting for velocity, then don't reset velocity).

## 10.0 Output Lines

The two output bits for each phase need be configured in two ways:

Configuration 1:
One bit for Direction 0=negative, 1=positive
One bit for On/Off (Configurable on/off = 0/1 or 1/0)

Configuration 2:

One bit Positive Direction
One bit Negative Direction
Off is both directions off Hardware protection so that both are never on at the same time

## 11.0 Block Diagram
**See PDF file (PTZ ASIC Motion)**

PTZ ASIC Motion pdf.pdf