Drum Position Optical Encoder Driven Level Wind Controller

Introduction

This paper describes the operation of a level wind system for a glider winch that emulates a reversing screw behavior, such as is used in fishing reels, using a recirculating ball screw stage driven by a stepper motor. Such level wind systems have been implemented using such reversing screws driven by a chain or cogged belt from the drum shaft. This approach is an alternative approach to accomplishing this same behavior with uniformly spaced interrupts at a 100 kHz rate. In this approach, the interrupts are generated from an optical encoder that is sensing the drum position, generally at a point further up the drive system to realize higher position resolution and encoder pulse rates than if directly sensing the drum shaft position. First some preliminaries are covered before a block diagram illustrating the processing involved. Then some topics deferred in the main body are addressed. There is also an appendix that contains an involved mathematical analysis needed to determine one of the key parameters for this system.

Preliminaries

Readers are assumed to be familiar with the prototype winch system in development. Later editions of this paper will attempt to provide more details for readers not so intimately involved with this development. The prototype will use two (initially one) Siemens 4-pole ac motors controlled by a iFOC controller to control their output torque. The base speed for these motors is around 3,000 RPM but we will use them in some phases of the launch to slightly over 6,000 RPM. The speed of these motors is reduced by a two-stage reduction system that will take the 3,000 RPM motor base speed down to about 500 RPM drum speed for a cable speed of about 20 m/s.

Respecting the 6,000 RPM limit of the encoder on hand, I would propose that the encoder be coupled to the shaft that drives the driveshaft. It will be reduced in speed by about a factor of 1.5:1 from the motor speed. This also reduces the encoder edge rates into what I believe are slightly more favorable ranges. The drive motor speed at 6,000 RPM would be geared to the drum such that the rope speed would be about 40 m/s. This intermediate shaft would be turning about 4,000 RPM here. For this discussion I will assume the encoder I have providing 360 PPR. If only one encoder edge is used to generate interrupts, the interrupt rate near maximum speed would be 24,000 interrupts per second. If 2 edges, 48,000. This rate is less than half that being considered for the time-based approach.

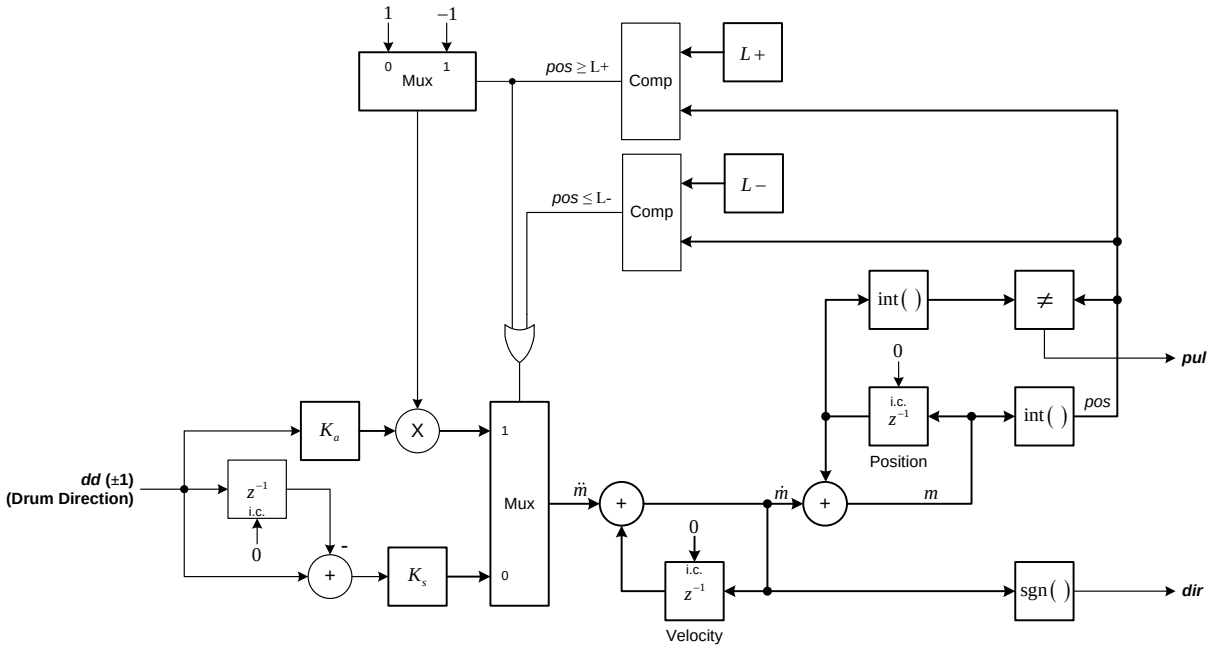
For our 2X working level wind factor, it was previously established that the sweep speed would be about 100 mm/s for a cable speed of 20 m/s. For a 40 m/s maximum rope speed, this would be about 200 mm/s. One rotation of the stepper moves the level wind 20 mm. I will assume we are using the 2000 microsteps per stepper revolution driver settings. (Going forward, I will use steps and microsteps synonymously.) At this setting, one step corresponds to 10 um of level wind motion. The number of steps per second for 200 mm/s then is about 20,000 steps per second. A requirement is the number of steps per interrupt must be no greater than one as the ISR can only issue at most 1 step command per interrupt. This requires the maximum step rate be less than the corresponding maximum interrupt rate. That is just satisfied for 1 edge and well satisfied by 2 edge operation. If the level wind factor needs to be increased above the current 2X baseline very much, this requirement could be violated for 1 edge. There is also some reduction in jitter realized by using a higher interrupt rate. So for the discussion to

follow I will assume we get interrupts based on 2 edge encoder processing. It should be very easy to change this during testing if we want to explore 1 edge (or 4 edge) processing.

With this choice, roughly the ratio of microsteps per encoder step is around 20,000/48,000 or about 5/12 or 0.417 steps per encoder edge/interrupt during the sweep period. (Super precision here is not needed.) The reversals take place over a relatively short distance. A typical value might be 2.5 mm. 2.5 mm corresponds to 250 microsteps. For comparison, the drum width is planned to be 165 mm. That distance corresponds to 16,500 microsteps. This concludes the preliminaries that needed to be covered. One of the integrators employed holds the current level wind system position in integral microsteps plus a fairly fine fractional component. The integral component must support the number of microsteps across the operating range so the 16,500 value here says it could possibly be supported by a 14-bit value. But the range may not be well centered so 15 bits would be better. Since this is so close to 16 bits and the processors we are using are generally 32 bit, the starting plan is to use the upper half of the 32-bit register for the integral component and the lower half for the fractional component.

Main Body

Below is a block diagram for the processing required by this encoder triggered interrupt approach.



This discussion will initially assume no abnormal conditions, e.g., loss of synchronization, and defers discussion of initialization until the normal operational mode is covered. I am depicting this system using z^{-1} elements which in DSP processing normally represent a unit sample delay at a fixed rate. Here they represent a one interrupt delay but the interrupt rate can vary from zero to the maximum rates discussed above.

The rightmost digital integrator's register, termed the Position integrator, holds the current level wind system position in integral microsteps plus a fine fractional component. This value is termed m , m

referring to microsteps. The integral component must support the number of microsteps across the operating range so the 16,500 value developed above says it could possibly be supported by a 14-bit value. But the range may not be well centered so 15 bits would be better. Since this is so close to 16 bits and the processors we are using are generally 32-bit, the starting plan is to use the upper half of the 32-bit register for the integral component and the lower half for the fractional component. The preceding integrator, termed the Velocity integrator will generally be less than 2^{16} in magnitude but it needs a sign bit so it will employ a 32-bit word size also.

Along with each encoder edge interrupt, a binary indication of the winch drum's direction of travel is required. Drum travel in the positive direction will be considered that which pulls in cable but that will not be important for this discussion. (When odometer processing is considered it will be important. The drum direction input, *dd*, will be a binary value but with values +1 for a forward/positive direction and -1 for travel in the reverse/negative direction. In the drawing, the lighter/finer lines and blocks are associated with binary (not necessarily 0 and 1 as just indicated) values and operations. The heavier lines represent numerical values represented with more than 1 bit.

There are 3 regions the level wind may be in. Most of the time the level wind rollers will be sweeping across the face of the drum at a speed proportional to the drum speed. That can be in either direction. When the sweep gets near an inner drum flange, the travel direction must reverse and the rollers start sweeping in the opposite direction. Most of the time the drum speed will be approximately constant during this reversal action (and even across a sweep). In this case, the scheme to be described will reverse the direction of travel over a short, fixed distance at a constant acceleration rate. As will be discussed, this reversal distance is constant independent of the drum speed or even if it varies during the reversal period. The acceleration magnitude will increase (quadratically) with higher drum speed magnitudes. This is essentially the same behavior as that of a reversing screw mechanism as used in fishing reels. This system will even work just as such a reversing screw fishing reel if the drum is operating near zero speed and between positive and negative velocities, e.g., if the drum were held by hand and moved back and forth.

Two parameters determine when to start the reversal, one for each side of the drum. The first parameter is L+ and it defines the level wind position (in integral microsteps) where the reversal action begins for the positive side of the drum. (The side that is in the direction of positive level wind travel.) L- similarly indicates the start of the reversal region going in the negative direction. This integral value, the upper 16 bits of the Position integrator register, is named *pos*. There are two comparators employed to determine which region the level wind is currently in. One compares the current integral position value *pos* to L+ and the other to L-. The comparator output for the comparison against L+ is true if pos≥ L+. This signifies the level wind is in the positive reversal region. Similarly, the one comparing against L- outputs true if *pos* ≤ L- when in the negative reversal region. These are ORed together to signify that the level wind is in either of the reversal regions. When in either of these regions, the lower multiplexor selects its upper input. When in the sweep region, its lower input is used.

First the normal sweeping behavior is discussed. This is when the lower input signal is selected by the lower mux. Assume the drum is and has been drawing in rope at some reasonable and fixed speed. Then *dd* will be and has been for a while positive and its signal line is and has been at a value of +1 for some time (at least one prior interrupt). The digital differentiator on that channel will be outputting a value of 0 as *dd* has not changed recently. So the value output from the mux will be 0. This value is the velocity increment $\dot{m}$ input into the Velocity integrator so its value will not be changing. (The first integrator is referred to as the Velocity integrator. But technically that is a misnomer. Velocity is the rate of change of

position with respect to time. Here it is not with respect to time but to encoder edges. If the drum speed is constant, the encoder edge rate is constant and this is indeed proportion to the sweep velocity so I use this simple nomenclature. Similarly, its input, $\ddot{m}$, is not technically an acceleration unless the drum speed is constant. In both such cases, these are only proportional to velocity and acceleration scaled by the current edge rate (and direction).) To be further elaborated on later, the velocity integrator output value $\dot{m}$ is and has been a positive value K_s, (*s* for sweep) while it has been sweeping in the positive direction. This positive parameter is the number of microsteps per encoder edge value previously developed above expressed as a signed 32-bit integer with the binary point just above the 16th lsb. So, its value would be about $5/7*2^{16}$ = 27,307. (Some further restrictions on actual allowable values will be added shortly but they are not relevant yet.) Every time there is another interrupt, this value increments the Position integrator. The Position integrator employs a 32-bit signed register with the same binary point location. For the assumed value of K_s, every 2 or 3 interrupts, the fractional part will overflow and increase the integer *pos* value by 1. When such an overflow occurs, a stepper micropulse is generated. This results in a pulse on the *pul* output line. This is effected in the block diagram by looking for a change in the *pos* value between the current *pos* value and its previous value. (For travel in the negative direction, this function also detects a borrow action associated with a decrease in the *pos* value.) The positive sign of the velocity input $\dot{m}$ indicates to the stepper driver the direction this step should be in. The direction input to the stepper is name *dir*. Discussion of setup requirements for the *dir* signal is deferred until later. This continues so long as the drum turns in the positive direction and the level wind does not enter the positive reversal region. If the drum speed increases/decreases, the interrupt speed increases/decreases proportionally, and the sweep speed increases/decreases proportionally with the scale factor set by K_s. This includes the drum slowing to a complete stop. Then the encoder interrupts stop and the level wind stops.

Now considered is when the drum speed drops to zero and then continues to decrease—meaning the drum is now rotating in the negative direction. When the drum first starts turning backward, *dd* goes to -1. The first time this happens, the current *dd* value into the differentiator is -1 and the value in the delay element is +1. But this delayed value is subtracted from the current value resulting in a value of -2. This is multiplied by the scaling factor K_s resulting in the value out of the multiplexor being $-2K_s$. This is added to the previous value, K_s, in the Velocity integrator so the velocity value $\dot{m}$ changes from K_s to $-K_s$. This negative value is integrated by the Position integrator resulting in its value decreasing. When such a subtraction results in the integral position value *pos* decreasing (by 1), the carry/borrow detector generates a stepper pulse command, *pul*. The Direction value is the sign of the velocity output, now negative, so the stepper moves in the negative direction. On the next interrupt with the drum velocity still negative, the current and prior *dd* values are both -1 and so the input differentiator produces a 0 output and $\dot{m}$ remains $-K_s$. This value remains fixed indefinitely as long as *dd* remains negative. Should the velocity return to 0 and go positive, on that transition the differentiator will output a value $2K_s$ and the velocity integrator value will toggle back to K_s. So the only two values the velocity output can take on are $\pm K_s$ so long as the level wind remains in the sweep region. As previously noted, if the drum were moved back and forth by hand through zero, the level wind would move back and forth proportionally with the drum deflections. This concludes the description of operation in the sweep region.

Now the reversal behavior is described. Assume the drum is traveling in the forward direction at a constant speed. Eventually, the stepper position will increase to and become equal to L+. When this

happens, the lower mux switches to the upper input. The level wind is now in the positive reversal region with the upper comparator asserting true. That gates the -1 value through the upper mux. *dd* is still +1 and that is multiplied times a positive parameter K_a, the *a* is for acceleration. This parameter determines the acceleration (actually deceleration) magnitude during a reversal. This parameter will generally be much smaller than K_s. The number of encoder interrupts between L+ and the farthest travel point is now shown to be K_s / K_a, call this ratio N_r. Not certain yet this is actually required but for now, assume K_s is an exact multiple of K_a so that N_r is an integer. Returning to the description of the behavior during the reversal, since the drum's direction of travel is positive, the value into the multiplier is $+K_a$. The value out of the upper mux was -1 so the value into the lower mux and gated through to become the acceleration increment $\ddot{m}$ is $-K_a$. When this value is subtracted in the Velocity integrator, this reduces the value of $\dot{m}$ slightly below K_s. On subsequent interrupts, this value continues to drop linearly, with interrupt edges, until it reaches 0. This will occur N_r interrupts after the reversal begins. Remember the drum speed is assumed to remain relatively constant during this reversal so the rate of encoder edge interrupts remains essentially constant. But the decreasing value of $\dot{m}$ results in the stepper pulse rate (again with respect to encoder interrupts) dropping until the stepper velocity value reaches 0. At this time, the level wind will have reached zero speed and the level wind position is at its greatest value. That value will be determined shortly. But *dd* remains +1 and the level wind position is still greater than $L+$ so the velocity increment value $\ddot{m}$ remains $-K_a$. The encoder interrupts keep occurring at the same rate so the velocity output of the first integrator continues to fall linearly but now the *dir* output has gone negative so step pulse commands, at an ever increasing rate now, are now in the reverse direction. *pos* now begins to fall and the stepper rate continues to increase until the velocity value $\dot{m}$ reaches $-K_s$. By symmetry, the distance traveled in the positive direction as $\dot{m}$ fell to zero will now have been traversed in the negative direction so the Position value will once again reach $L+$ but now traveling in the reverse direction at speed $-K_s$. The upper comparator will deassert and the system is now in the sweep region traveling at speed $-K_s$. When the sweep reaches position $L-$, the dual reversal process ensues. Here the drum travel direction is still positive but the upper mux will be passing the +1 value so the acceleration $\ddot{m}$ employed for the reversal will be $+K_a$. The velocity input $\dot{m}$ on entry was $-K_s$ so it will increase linearly reaching zero N_r interrupts later when the travel in the negative direction is at its extrema value and then become positive finally reaching $+K_s$ as the level wind exits the negative side reversal region and is now sweeping forward again. Should the drum be rotating in the negative direction, the behavior is the same except the directions are reversed. It is the product of the *dd* and the upper multiplexor output that selects the correct sign for the acceleration value $\ddot{m}$ for all situations.

The remaining question is what is the distance traveled during the reversal period? More precisely, how much does the position value change after reaching $L+$ or $L-$ until the velocity reaches zero. This value has been determined by analysis, included as an Appendix, to be

$$\Delta P = \frac{N_r - 1}{2} K_s$$

(This is closely related to the distance traveled by an object under constant acceleration being $0.5At^2$. The ½ factor comes from this being the area under a triangle associated with the linearly increasing (or

decreasing) velocity.) ΔP is properly in units of steps. So this distance is fixed regardless of drum speed and proportional to K_s which determines the level wind factor. Previous analysis shows the actual acceleration employed here increases quadratically with increasing drum, hence sweep, speeds. With a little algebra, we can choose K_a to set this distance as given by

$$K_a = \frac{K_s^2}{2\left(\Delta P + K_s\right)} \approx \frac{K_s^2}{2\,\Delta P} \qquad N_r \gg 1$$

N_r, the number of interrupts required to take the stepper "velocity" to zero is given by

$$N_r = 2\left(\frac{\Delta P}{K_s} + 1\right) \approx 2\frac{\Delta P}{K_s} \qquad \frac{\Delta P}{K_s} \gg 1$$

Note that the reversing period, in interrupts is actually $2\,N_r + 1$ as N_r interrupts only get the velocity to 0, then another N_r interrupts are needed to get the speed back up to the reverse sweep speed.

Just to emphasize, these computations are not done in real time during the interrupts. They will selected and adjusted during field trials and will likely be hard coded. (It is possible there may be some way provided to set $L+$ and $L-$ for adjusting the level wind travel for field maintenance and allowing K_s and K_a to be similarly adjusted would be simple to include.)

What is a typical value for this parameter? As noted in the preliminaries, a typical value for ΔP is 2.5 mm. But ΔP here is in unit of microsteps and since a microsteos is 10 um, ΔP here is about 250. So K_a is about 3.47E-4 as compared to about 0.417 for K_s. So N_r, the ratio of these, is about 1,200 indicating the reversal period in interrupts would span about 2,400 encoder interrupts. In terms of the 32-bit increment value for the Velocity integrator, the value used for K_a would be about 23. Comparison with the value of K_s, 21,800, demonstrates the assertion that $K_a \ll K_s$. This is pretty coarse but a lot of accuracy is not needed here. This just goes to the distance in steps from $L+$ to the farthest point traveled during the reversal. In practice this reversal distance will be determined by trial and error in the field as will be K_s, $L+$, and L-. If we should want finer resolution, we could shift the binary point up one, possibly 2, bits but this might complicate separating the fractional and integer parts of the Position integrator for determining when to emit a pulse. If it remains a requirement that K_s must be an integer multiple of K_a, that would require here that K_s must be a multiple of 23. But the fractional change in K_s around its nominal value of 21,800 is minuscule, about 0.1%. To emphasize that high precision here should not be important, we are debating whether the level wind factor should be 2 or 3 and K_s is proportional to the level wind factor.

Deferred Topics

Initialization

There are several steps required to get the system fully initialized. It is most simple if the drum is stopped or moving slowly during the initialization. That is common and making it a requirement will simplify the procedure. If for whatever reason that is not always possible, it can be accommodated. It would be similar to what will need to be done if synchronization is lost during a launch and an attempt to reacquire on the fly is desired. First the procedure for initialization when the drum is stopped or at very low speed corresponding to very low interrupt rates. Also assumed is the level wind rollers physical position is between L+ and L-. What 'very low speed' is can be analyzed or determined experimentally but will not be addressed at this time. How to gracefully deal with cases where the position may not be within these limits is briefly discussed later but this may involve considerations that will not be fully understood until the physical system is constructed so adjustments will likely be needed.

The first step in the initialization sequence involves getting the Velocity integrator's and its preceding differentiator's memory elements properly initialized. Assume we have disabled the encoder edge inputs. If the drum is stopped, that would technically be the case but very fine movements of the drum can generate many interrupts. Assume also that known values have been established for both K_s and K_a. These may be the operational value but might need to be different for the indexing operation to follow initialization. It was noted that the only two values for $\dot{m}$ that can exist during the sweep are $\pm K_s$. $+K_s$ was associated with sweep travel in the positive direction and $-K_s$ with negative sweeps. It generally does not matter which we establish so choose sweep travel in the positive direction. Under this assumption, the Velocity integrator value $\dot{m}$ would be $+K_s$. Assume we initialized $\dot{m}$ to $+K_s$. But sweep travel in the positive direction can be associated with drum movement in either direction. What would need to be done is to ensure the value of the differentiator on the first processed interrupt is either 0 or -2. Either of these will result in a valid result for the first computation of $\dot{m}, \pm V_s$. What cannot be allowed to happen is for the first differentiator output to be +2 as that would result in the first computed value being $3K_s$..We could complete this initialization by setting a flag to indicate we are initializing the system and, when we see what direction the encoder processing indicates, make the differentiator's memory element value such that we get one of these two values. If we forced the differentiator value to be 0, subsequent forward drum motion would result in the sweep moving in the positive direction. If -2, the sweep for forward drum motion would move in the negative direction. But there is a simpler solution. Set $\dot{m}$ and the differentiator's memory value both to 0. Technically the differentiator input values can only be ± 1 but it will actually be a signed binary word so this poses no problem. Now let interrupts begin. Assume the drum direction *dd* for the first interrupt is +1. The differentiator would produce an output value of +1 and the input to the mux, passed on to the Velocity integrator would be $+K_s$. It would add this value to 0 resulting in the first newly computed value of $\dot{m}$ being $+K_s$. The level wind would move in the positive direction from here on in response to drum motion in the positive direction. If, equally likely, the first interrupt was associated with a negative drum motion, then the differentiator would compute a first velocity increment of $-K_s$ and the first computed output of $\dot{m}$ would be $-K_s$. Subsequent forward drum motion would result in the level wind sweeping in the negative direction. Unless some additional requirement to set the direction of sweep for forward drum motion emerges, this is as good a behavior as any. To complete the initialization process, the Position integrator's initial value must be initialized. Since all we have assumed is that the level wind is

somewhere in the sweep region between L+ and L-, setting it to zero is as good a value as any. Getting it to its proper value is the subject of indexing that is considered next. So what is seen is that the system can be initialized for accepting the first interrupt by simply setting all three of these memory elements to 0 in preparation for processing the first interrupt.

Indexing

Indexing is the process of aligning the Position integrator value m with the actual level wind physical position. This is the most readily done using a limit switch to indicate when the level wind rollers are at a known location. If that limit switch was physically located so as to trigger when the rollers were at the center of the sweep, conceptually what needs to be done is to sweep the rollers such that this position is passed and, when that event occurs, the Position integrator value m would be overwritten and set to 0. Conceptually this is fine but there are issues and other desired use for the limit switch that will require some accommodation. The first is that the trigger point of a limit switch is typically dependent on the speed and direction of travel of the level wind mechanical elements. The simplest way to deal with this is to always pass the limit switch trigger position moving in a specified direction and specified speed. Since all we have assumed is we know is that we are somewhere between where reversals would begin, it behooves us to locate the indexing limit switch trigger position somewhere very close to one of the reversal points. Let assume we have physically positioned the indexing limit switch near the point where we initiate a reversal associated with L+ and, to allow it to be used for detection of loss of synchronization (described shortly), slightly towards the sweep center from L+. This completes the setup for the indexing procedure.

What we need to do now is to sweep the system in a positive direction towards this limit switch at a fixed, and typically relatively low, speed. If after the initialization process, we actually used drum encoder interrupts and direction signals, we are not guaranteed a direction or speed. So for this indexing operation, the simplest solution would be to provide a timer producing interrupts at a known rate and force *dd* to +1. This would simulate the drum moving in the positive direction at a known speed that results in a known level wind sweep speed. With this value set to +1 and the system initialized as described above, the system would conceptually move towards the indexing limit switch at the desired known rate. To prevent the system from thinking it has entered the reversal region, L+ can be temporarily made much much greater than the actual L+ value. (Making it greater than 2 times the operational value will generally be sufficient.) When the level wind position reaches this position, the level wind limit switch would trigger and the Position integrator value m would be overwritten. The most obvious value for this initial value would be a value that corresponds to the known location of the limit switch. We assumed it would be close to L+ but somewhat towards the sweep center, so this value would be slightly lower than L+. If this is done, the *pos* value 0 would correspond to close to the center of the sweep. In reality it could be any value, including 0, as the parameters L+ and L- will be determined experimentally and the system doesn't care about arbitrary offsets. The only true requirement is that L- < L+. But choosing a value to make the center of the sweep correspond to 0 position is the most logical. A change in 1 microstep only corresponds to 10 um. So the fractional value employed for m is really not that important. Making it 0 is the most obvious choice.

As usual, there are some issues that can occur here. One is the desired sweep speed to accomplish this indexing operation quickly may be higher than the level wind system can start from 0 speed and reach without losing synchronization. If synchronization is broken for this reason, the stepper can commonly stall and not move at all. If this is the case, the scheme can be simply augmented to deal with this. Rather than try to start with a lurch to this sweep speed, the acceleration provisions already available

can be simply used. We simply set the Velocity integrator value to 0 which, assuming we just finished the above initialization process, it is already at, and set L- such that the system thinks it is at the negative-most value the negative region reversal process. All that is needed for this is to set the value of L- temporarily to N_r. The system now thinks it is in the negative reversal region and just passing through 0 velocity. The acceleration increment value will be $+K_a$ and the stepper velocity will ramp up to value $+K_s$ over a period of N_r interrupts and then exit this negative reversal region as *pos* passes N_r. $\dot{m}$ will now be K_s and the sweep will continue at the rate associated with K_s at the simulated encoder edge interrupt rate. This will continue until the indexing limit switch triggers and then the remaining processing is as described above.

The indexing is now essentially complete. What remains is to stop the level wind system smoothly where we want it to start when we start using the drum encoder interrupts. This is essentially the reverse of the process used to accelerate the level wind up to full sweep speed. Since we assumed the limit switch was physically positioned just short of the positive reversal region and we overwrote *pos* to its value, *pos* should be increasing and quickly approaching the start of the nearby positive reversal region. Now we simply set L+ to the correct value and when it reaches there, it will decelerate the stepper to 0 velocity and then ramp it down to $-K_s$. The system is now sweeping back in the negative direction heading towards where ever it is we want to stop, call that value pos_init. If we now set L- to $pos\_init + N_r$, it will start decelerating and reach zero velocity just as it reaches the target location. At this point, we disconnect the timer simulated drum encoder interrupts and indexing is complete. The final step is to set L- to its operational value. When we are ready to start and the drum speed is 0 or below an acceptable speed (again still to be determined), we can start accepting the drum encoder interrupts and the level wind will track drum motion normally from here on.

Loss of Synchronization and Recovery

The stepper can pull out of lock if the loads, either rope induced or inertial, are beyond the capability of the stepper to support. This is termed a loss of synchronization, LOS for short, event. Accelerating the inertial loads too fast is the reason the system limits the acceleration during reversals or when just accelerating the level wind inertial loads too quickly, such as for doing the indexing sweep. That was why for indexing, we employed the proposed systems intrinsic ramping velocity capability to limit acceleration. When a stepper pulls out of lock, typically it does not slip one stepper cog (about 1.8°) but many. In particular, if it loses lock due to excessive rope side loads, it will likely slip almost to the drum face center and then stall (stop) even though pulse commands continue. When that happens, the Position integrator will continue to emulate where it thinks the level wind mechanism is and even start the reversal process when it passes either L+ or L- (depending on the sweep direction). If a pair of limit switches are placed so as to trigger just before the mechanism reaches L+ or L-, then an indication that synchronization has been lost would be that the Position integrator value *pos* indicates reversal region has been entered but the associated limit switch did not trigger. Furthermore, whenever a limit switch triggers, the value of *pos* at the trigger time is checked to make sure it is within some tolerance of where it was expected. So a LOS event can be either 1), a limit switch did not trigger before reaching L+, when traveling in the forward direction, or L-, when traveling in the negative direction or 2) the value of *pos* when a limit switch triggered was beyond some tolerance of where it was expected to be. To simplify the description to follow, assume a type 1 LOS event occurred on the negative sweep, i.e., the pull-out occurred before the negative limit switch was passed and then L- comparator asserted true. For this discussion, we assume the mechanism was pulled back to the center and the motor stalled. The dual of

the recover behavior to be described would occur for loss during a positive sweep. Normally, to deal with a stalled motor, we would have to assume the motor mechanism is at zero speed and manually accelerate it back up to speed using simulated encoder inputs. But fortuitously, the standard reversal activity just initiated by *pos* passing L- will do this for us automatically. It ramps the Velocity integrator speed $\dot{m}$ towards 0 and then accelerates it to the positive sweep speed for the return sweep. But as $\dot{m}$ matches the physical stepper speed, assumed 0 now, the stepper will pull back into lock and start following $\dot{m}$ again. So that action neatly gets the stepper out of stall and moving in the proper direction but the system has lost index, i.e., the Position integrator position does not correspond the physical mechanism's position. As noted, under excessive side loads the mechanism will likely have slipped to near the drum's center. So the ISR algorithm thinks it is sweeping back near from the negative-most inner flange but, under the assumed scenario, the physical mechanism is starting to sweep from near the drum's center. What would happen now is the roller assembly will trigger the limit switch near the positive drum flange long before expected—based on the *pos* value. But this is simply a type 2 LOS event, the trigger occurred with pos not near where expected for the associated limit switch trigger point. When this occurs, we use the positive limit switch to re-index *pos*. Because the sweep speed is likely different than what is used for normal indexing, this won't be as accurate but hopefully it will be close enough to continue operations for a launch in progress. (Simple tests can be done to see how sensitive the re-indexing is to speed and possibly speed dependent corrections applied.) But at least we are traveling in the proper direction for indexing with this limit switch. So we actually knew we had likely lost synchronization when *pos* passed L- but we didn't have to do anything until the positive limit switch triggered. But when the L- passage occurred, a sticky bit indication that we had a LOS event would be set and the operator informed. Before a subsequent launch would be allowed to be initiated, the level wind system would require a normal indexing process. If the LOS occurred during a positive sweep, the dual of this recovery behavior would occur. Here we would approximately re-index when the L- limit switch triggers. So for LOS recovery, both limit switches are employed to detect LOS and approximately re-index the system. So, other than noting that an LOS has occurred, the only action that needs to be take is to re-index on either limit switch when the stepper is moving in the proper direction for indexing with that switch. Note the drum traveling in the negative direction needs to be handled properly here. The logic is simple, if $\dot{m}$ is positive, re-index on the positive positioned limit switch triggering on a positive stepper direction passage. If negative, re-index on the negative positioned limit switch on a negative stepper direction passage.

There is no reason to have a separate indexing limit switch. One of these LOS detection and recovery limit switches located near, but inside of, the physical reversal positions can serve as a normal indexing switch. Assuming indexing results in the Position integrator's value of 0 corresponding to near the drum face's center, the physical position should be able to be within a few percent of the reversal points, e.g., the positive limit switch might trigger around 0.98 L+. The tolerance for LOS detection here might be ±1%. Normal indexing is just a special case of LOS recovery. The only real difference is that we simulate the encoder edge interrupts to emulate a drum moving in a known direction at a known speed for precision indexing.

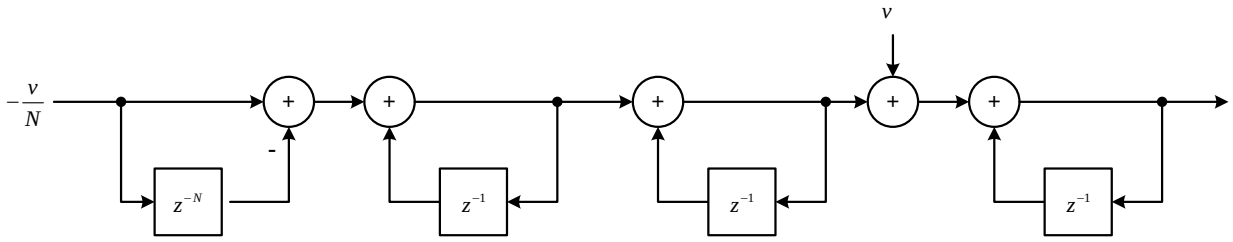
It has been asked why not use the limit switches' interrupts to initiate the reversals. There are several reasons for doing this. The first is that it would require the limit switches to be physically moved to adjust the reversal points. This would probably make desirable some sort a method to finely adjust the limit switch trigger position with something like a screw adjustment that could be then be locked down. But with the scheme proposed above, adjusting the reversal points only requires changing software parameter values and the positioning of the limit switches can be much cruder and the limit switches do not have to be precisely set and easily tweaked. The second, and much more important reason, is this

would make LOS detection and recovery much much more difficult. Once initialized and indexed, the system proposed runs essentially open loop. It would not need the limit switches at all except for indexing and LOS detection. Once indexed, it *thinks* it knows where the level wind physical mechanism is at all times and does the reversals at those emulated positions. The limit switches' primary use (other than for indexing) is for LOS detection and recovery. If the limit switches were used to initiate the normal reversals and the stepper had stalled, the limit switch trigger event would never occur. Then we would still have to monitor the simulated position to see if the limit switch triggered near where it was expected. There is a third reason but I can't remember it as I am writing this. I'll add it later when/if I recall what it was.

One thing that must be considered is limit switch debouncing. We don't want to get multiple interrupts from what is really a single transit of the switch. The switches planned to be used have both NO and NC contacts. We might want to implement a hardware switch debouncer using an RS flip-flop type scheme. Or software debouncing could be implemented. Another hardware dependent topic is that the stepper driver procured for the prototype has setup requirements for the *dir* input relative the *pul* signal. As *dir* is a level input, that only really comes into play when the sign of $\dot{m}$ changes. Further discussion of both of these issues is not really relevant to the purposes of this document but are very important to consider in the actual code and hardware development.

Why must Ka and Ks be integrally related
Stepper Pulse Jitter
Startup when physical mechanism is not interior to the limit switches.

Appendix



$$F(z) = v\left[\frac{z}{(z-1)}\frac{1}{\left(1-z^{-1}\right)} - \frac{1-z^{-N}}{N}\frac{1}{\left(1-z^{-1}\right)^3}\right]$$

$$= v\left[\frac{1}{1-z^{-1}}\frac{1}{1-z^{-1}} - \frac{1-z^{-N}}{N}\frac{1}{\left(1-z^{-1}\right)^3}\right]$$

$$= v\left[\frac{1-z^{-1}}{\left(1-z^{-1}\right)^3} - \frac{1-z^{-N}}{N}\frac{1}{\left(1-z^{-1}\right)^3}\right]$$

$$= v\left[\frac{1-z^{-1} - \frac{1}{N}\left(1-z^{-N}\right)}{\left(1-z^{-1}\right)^3}\right]$$

$$\lim_{k\to\infty} f(k) = \lim_{k\to 1}(z-1)F(z)$$

$$= v\lim_{k\to 1}(z-1)\frac{z^{-1}}{z^{-1}}\left[\frac{1-z^{-1}-\frac{1}{N}\left(1-z^{-N}\right)}{\left(1-z^{-1}\right)^3}\right]$$

$$= v\lim_{k\to 1}\frac{\left(1-z^{-1}\right)}{z^{-1}}\left[\frac{1-z^{-1}-\frac{1}{N}\left(1-z^{-N}\right)}{\left(1-z^{-1}\right)^3}\right]$$

$$= v\lim_{k\to 1}\left[\frac{1-z^{-1}-\frac{1}{N}\left(1-z^{-N}\right)}{z^{-1}\left(1-z^{-1}\right)^2}\right]$$

$$= v\lim_{k\to 1}\left[\frac{1-z^{-1}-\frac{1}{N}\left(1-z^{-N}\right)}{\left(1-z^{-1}\right)^2}\right]$$

$$= v\lim_{k\to 1}\left[\frac{z^{-2}-\frac{1}{N}\left(N\,z^{-(N+1)}\right)}{2\left(1-z^{-1}\right)z^{-2}}\right]$$

$$= v\lim_{k\to 1}\left[\frac{z^{-2}-z^{-(N+1)}}{2\left(1-z^{-1}\right)z^{-2}}\frac{z^2}{z^2}\right] = v\lim_{k\to 1}\left[\frac{1-z^{-(N-1)}}{2\left(1-z^{-1}\right)}\right]$$

$$= \frac{v}{2}\lim_{k\to 1}\left[\frac{(N-1)z^{-N}}{z^{-2}}\right]$$

$$= \frac{(N-1)}{2}v$$