

The following is a description of a microprocessor-based position control system for the CM-320 motor/encoder. The position control loop resolves 1 part in 262,144 (2^{18}), using the 1024 c/r sine wave output of the encoder. The described techniques are useful in any type application where fast and accurate positioning of an object is desired.

The servo was designed to control the position of a shaft-mounted slitwheel. Each slit is used to set one of the measurement characteristics of an optical spectrum analyzer. The overall product design required that each slit be rotated in position quickly and with zero steady-state position error.

Conventional solutions such as a digital, low resolution encoder with a gear reduction system, were not practical due to friction, backlash, gear wear and response speed. Using an encoder with sine/cosine outputs rather than the conventional square wave outputs, in conjunction with interpolation, the desired resolution could be obtained from a physically small unit with moderate line count. As the slitwheel can be directly mounted on the motor/encoder, the friction was reduced to only two small, instrument grade ball bearings.

The four main functions needed in the motion control portion are

- communication with the rest of the system,
- position sensing including interpolation,
- feedback control including loop compensation,
- motor drive

The specifics of communication are particular to the application and include the passing of the desired position and loop filter parameters from the system to the control processor. Conversely, information such as actual position and error condition can be passed to the main system controller.

Since feedback control is well covered in the literature, not much will be said except to highlight some of the problems encountered and how they were solved. The processor chosen for this application was a 68HC11. The reasons for this choice were the low cost and a built-in A to D with an 8 to 1 multiplexer ahead so multiple analog signals can be converted to digital information. Since the 68HC11 is only an 8 bit processor, the well known PID control algorithm was used to minimize its work load. Other processors could be used and in fact designs starting today would be wise to use one of the low-cost digital signal processors.

Position sensing:

In order to obtain the needed resolution, the encoder with "sine wave" outputs was chosen. Instead of the more conventional quadrature square wave encoders, the outputs are a sine and a cosine, each channel undergoing a complete cycle for each line of the encoder disk. By properly processing the analog information in the sine and cosine terms, resolution far beyond the normal "x4" linecount can be achieved. For absolute position control, an index pulse is also supplied to establish an absolute reference.

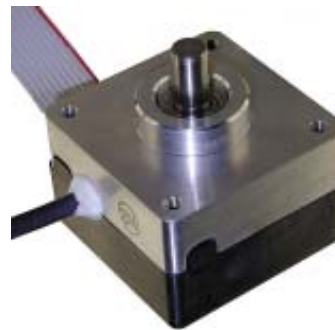
The CP-320 encoder has internal electronics that provide 4 V peak-to-peak outputs on the quadrature signals, centered on a user supplied bias voltage.

The bias was set to 2.5 V to provide suitable levels for use with the 8 bit A to D converter in the 68HC11, which in this design works over the range of 0 to 5 V. Driven this way, the output of the D to A is an offset binary representation of the selected sine or cosine channel.

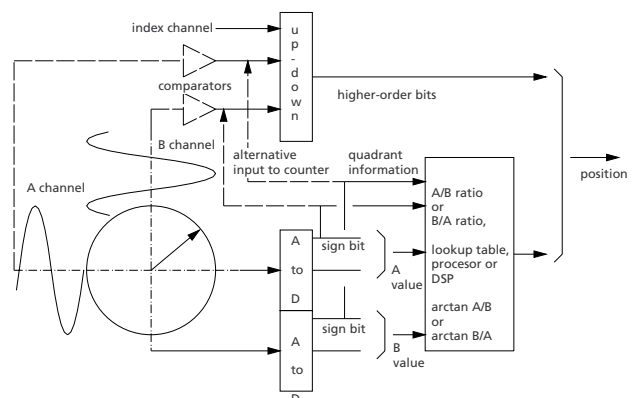
Coarse position information is obtained by squaring up the sine and cosine outputs to produce two quadrature square waves that change state on the zero-crossings of their respective inputs. These can be processed in the usual way to control an up-down counter which can be read by the processor. In this design, an HCTL-2020 quadrature decoder/counter from Hewlett Packard was used to provide the processing and counting function in a small package. This was done because the 68HC11 could not process the encoder sine and cosine fast enough to keep up with the rate that can occur at high slew rate position changes. The HCTL-2020 provides a 16 bit up/down counter with its outputs multiplexed 2:1 onto a three-state 8 bit bus driver. With a 1024 line count encoder, there are only 4096 zero-crossings per revolution so a 16 bit counter is more than adequate.

Positions between the zero-crossings of the sine and cosine are resolved using interpolation. In its simplest conceptual form, interpolation of a sinewave encoder is simply a matter of converting the

sine and cosine into digital form with an A to D converter. The ratio of the sine to cosine is then computed and the result used as an address of an arc-tangent lookup table. The output of the table is a fraction of rotation between the zero crossings of the two encoder channels that provide the coarse information.



The real world of course is not quite so simple. An attempt at directly implementing this conceptual scheme will run into a problem: when the value of the cosine approaches zero, the division will be a large number that will cause overflow and will eventually "blow up" when divide by zero is attempted.



A solution to this problem is to compare both channels and always divide the smaller value by the larger. The result is a fraction that ranges between zero and one as the electrical angle of the sine and cosine terms ranges from 0 to $\pi/4$ and from one to zero from $\pi/4$ to $\pi/2$, when the next coarse transition occurs. The computed result is then the input into the arc-tangent table and the output is an angle. In the first segment (sine less than cosine), the output of the table is the desired fraction of rotation between coarse position increments. Scaling of the angle is such that (since it is a fraction of rotation between coarse position increments) it ranges between 0 and 127/256. In the second segment (sine greater than cosine), the fraction is subtracted from 1, to yield a result that ranges between 128/256 and 255/256.

A final problem is that the comparators that drive the coarse position counter almost certainly will not switch states at exactly the same voltage that the A to D converters use as mid-range (= where sine and cosine are zero). If external A to D converters are used, this problem can be avoided by biasing the analog sine and cosine to operate about the midpoint of the A to D converters input range.

The most significant bit of each A to D converter ("sign bit") can then be used to drive the coarse position counter instead of comparators. In this case, the fractional position information is simply added to the integer (coarse) position information to produce a high resolution value.

Using the 68HC11 where the A to D converter is internal to the processor, another solution had to be found: there is always a fixed relation between the two least significant bits of the actual position counter and the sign of the sine and cosine signals. If this relation is known, it can be used to correct the reading from the coarse position counter "on the fly".

The coarse position counter can be in error by at most only ± 1 LSB from the true position, and only in the region of the zero-crossings of the sine and cosine encoder signals. Knowing the correct relation between the two LSB's of the integer position value and the signs of the sine and cosine allows the value read from the position counter to be corrected as necessary.

Once the correct value for the integer position is known, the fractional part is simply added to it to produce a high resolution position value which is then used for control in the usual way. All this may seem complex, and it does take a bit of effort to "get it right" but in case of the 68HC11 code it only requires 60 lines of assembly language code and only 15 lines of code execute on any given pass through the algorithm.

With an 8 bit A to D converter, the interpolation algorithm provides 8 bits of position resolution beyond that available by counting the zero-crossings of the two channels. Unfortunately, not all 256 values are resolvable. This is because the sine and cosine signals have some harmonic distortion, their peak-to-peak amplitudes do not perfectly match the range of the A to D converter and there are roundoff errors in the division process and arc-tangent table. These effects combine such that there will be some "missing" positions if use of the full 8 bits is attempted. However, the 6 most significant bits do provide information without missing positions and the two least significant bits are useful in reducing the amount of digital "hunting" that invariably takes place about the target position.

Processor controlled commutation:

Once position information is available, it becomes possible to have the control processor provide commutation of the phases of a brushless DC (BDC) motor. Historically, commutation information has been provided by Hall sensors contained in the motor. The Hall sensors crudely detect the position of the permanent magnets in the motor and allow the appropriate winding to be driven. This scheme has the disadvantages that only one winding at a time is usually driven.

This causes the effective torque constant of the motor to vary with position and the switching from one winding to another usually takes place at a location where the torque constants do not exactly match, which causes a step in the magnitude of the torque applied to the shaft. With position information from the incremental encoder available, and if the position of the magnet poles relative to the encoder can be established, then the processor can take over the commutation function. The advantages are: no Hall switches, the motor drive is simplified and the processor can synthesize commutation appropriate to the motor.

In case of the CM-320, the motor has a two phase sinusoidal output voltage when driven as a generator so the desired commutation function is sinusoidal. By exciting each phase one at a time, and noting each position where the motor stops, the processor can obtain enough information to determine how to commutate the drive to the windings. This can obviously be done for motors with three or more phases as well. Once the commutation reference positions are determined, it is a simple matter to multiply the desired control effort calculated by the appropriate sine and cosine terms and apply the results to the winding drivers. The result is an applied torque with much reduced ripple, compared with that achieved using Hall sensors. In addition, the torque ripple varies in a smooth fashion without the stepped behavior that occurs with Hall derived commutation.

