Interpolated Motion

This document further details the interpolated move commands:

- Interpolated Move Start (IMS)
- Interpolated Move Queue Clear (IMQ)
- Interpolated Move Write Queue (IMW)

This is an advanced topic. It is assumed the reader fully understand SilverLode programming. Please see SilverLode User Manual for more background.

Overview

Interpolated movement allows the servo to use Data Registers that cycle through Time, Position, Acceleration, and Velocity data to control motion. These registers define constant acceleration segments that ramp up or down to a desired velocity, or move at a constant velocity over a period of time. By streaming data through these registers, the Interpolated Move Start (IMS) command can interpolate the points in each velocity segment as well as between velocity segments to create a continuous, complex motion profile. The cycling of data can be accomplished by either a host that streams the data to the servo via serial communication or by a standalone program.

Registers Used with Interpolated Motion

There can be up to eleven data registers used by the Interpolated Movement operation at any time: three operational registers used to control the Interpolated Movement process, four user defined registers where velocity segment data is cycled, and four Profile Move registers (#20 to #23) where current velocity segment parameters are copied to just prior to execution of that segment. Although the user has no control over the final four registers, and one of the operational registers (Register 18), it should be noted that these registers should not be used by any other part of a program that plans to incorporate Interpolated Movement; these registers are written to automatically by the interpolated move process. Any use of these registers (18 and 20-23) while in an IMS operation could cause a program or movement error.

Registers 17, 18, and 19 are the operational registers. When using the register based queue, the upper word of Register 17 indicates whether the data in the user registers has been used; when the IMS operation copies data from the user defined registers to the Profile Move registers, it writes a “1” here, indicating the velocity segment parameters are now *stale* thus providing a handshake to the user indicating fresh data should be supplied. The lower word of Register 17 contains the register reference number of the first of four user registers if using the register based queue; or, a “0” if using the Interpolated Move Write Queue (see IMQ/IMW Commands) via a host through the serial interface. This location of the next set of data will have to be written to Register 17 with a Write Register command in a loop after the IMS operation transfers the data and writes a “1” to the upper word (register based queue only). Register 18 is where the segment time (the first data parameter) is copied just before execution of that segment. Register 18 is decremented every servo cycle (120 usec) until it reaches “0”. If there is a still a “1” in the upper word of register 17 when the timer is

exhausted, it indicates that fresh data has not been loaded and causes the IMS operation to shut down using the value in Register 19 as a default deceleration value to stop current motion and exit IMS. (Note, when using the IMQ command, the IMS operation checks to see that the queue is not empty, and draws the next set of 4 data words from the queue; if the queue is empty, then the IMS operation uses the value in Register 19 as a default deceleration value to end the IMS operation.) This checking for the next data set is needed to prevent the previous position information, typically a significant distance from the current position, from being used as a final target in the case of a communications breakdown. With loss of communications, the IMS operation will simply shut down with the user specified deceleration.

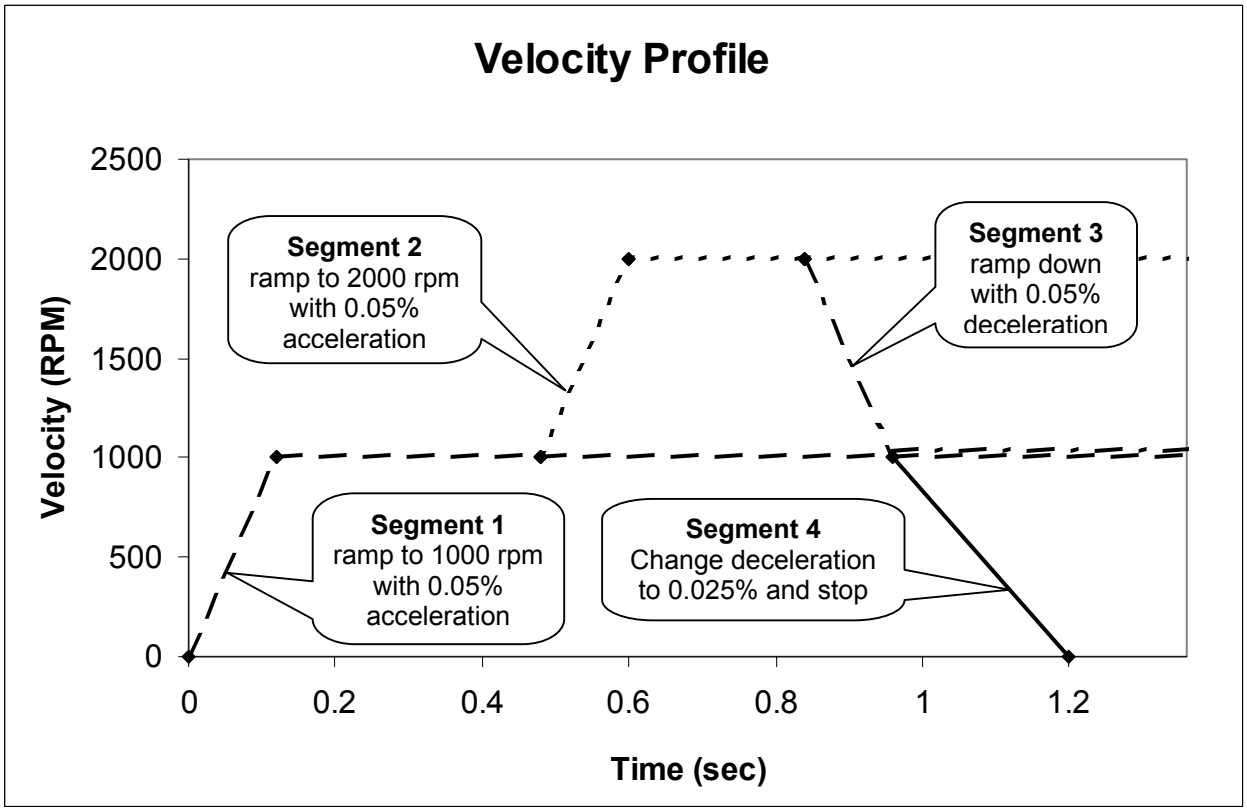
The four user defined registers can be any four sequential user registers: these four registers make up the register-based queue. They must be loaded with Time, Position, Acceleration, and Velocity data (in that order) of the next velocity segment to execute. There are two options to successfully stream data to IMS with the register-based queue. Either store all data in successive blocks of four user registers (if the data profile is small enough) and construct a loop that iterates the appropriate register reference number into the lower word of register 17; or, import data from a register file array into the same set of user registers and leave the lower word of register 17 alone. The register file array method allows for much larger data tables, and higher profiling resolution.

Time: 0 to 2,147,483,647
A value within this range indicates the number of servo cycles (120 usec time slices) to count down before loading the next set of data. Note that this time parameter does not influence the motion profile; it acts as a time delay, giving the profile enough time to develop based on the acceleration, velocity, and position data sent with this parameter. The next set of move parameters must be loaded by the register-base queue or the Interpolated Move queue before this counter decrements to zero, or the deceleration value in register 19 will be used to stop the profile. A “0” indicates the last set of data to be transferred and tells the servo to end IMS operation.

Position: -2,147,483,648 to +2,147,483,647
Unless the segment is intended to come to a halt at a given location, the position parameter is only used to project the direction of motion. It should be noted that position calculations incorporate register wrap-around. For a general move use the present position plus or minus 1,073,741,824. This value is large enough to project moves properly, while remaining small enough to never wrap around the register range and project movement in the opposite direction. If the segment comes to a halt (decelerates to zero velocity), give the desired ending position of the move. In this case, the position parameter is used to direct the motion profile to a specific location.

Acceleration: 1 to 2,147,483,647 (Scaled from a 0 – 2000RPM/120usec value)
Acceleration or deceleration used in reaching the ending velocity of the segment. Since the acceleration corresponding to the full-scale value is physically impossible to achieve, limit acceleration values to approximately ½ of this scale (1073741823). A “0” indicates a constant velocity segment.

Velocity: 0 to 2,147,483,647 (Scaled from a 0 – 4000RPM value)
Unless the segment is the last segment of the profile, this should be the desired ending velocity of the segment. When specifying the last segment of a profile, set velocity to the starting velocity of the segment. Do not set as the ending velocity of the segment to (zero); this will cause the IMS to undershoot the desired ending position.



Example of Interpolated Move:

In the following move there are four velocity segments: two segments that ramp up to speed, and two decelerating segments. Note that each segment is projected off to an unattainable position (1,073,741,824) because the time parameter determines where the next segment starts.

	Time	Position	Velocity	Acceleration
Segment 1:	**4000**	**1073741824**	**1073742**	**536870912**

Ramp up to 1000 RPM (¼ velocity: (2147483647)/4=536870912) with .05% acceleration (.0005*2147483647=107342) and project that motion for 4000 ticks (.48 seconds) before looking to load a new segment or decelerate to a stop using register 19. Note that these parameters project the initial ramping and constant velocity segments and the position parameter for this segment is never realized. The position parameter is intended to define direction of motion while allowing acceleration and velocity parameters to define the shape of the profile.

The 4000 ticks of time data are copied to register 18 just before this segment's execution. This count is decremented every servo cycle (120 usec) until it reaches "0". Before the counter

decrements to “0”, the lower word of register 17 should contain the register reference number of the first register out of four user registers to load data from for the next segment. In addition, register 17 still has a “1” in the upper word, indicating new move parameters have not been loaded into registers 18 and 20-23. Clearing the upper word of register 17 (writing a “0”) must be done manually (ideally in a loop) or the IMS operation will use the value in register 19 to stop motion and exit IMS operation.

Segment 2: 3000 1073741824 1073742 1073741824
Ramp up to 2000 RPM (½ velocity: (2147483647)/2=1073741824) using .05% acceleration (.0005*2147483647=107342) for 3000 ticks (0.36seconds) before looking to load a new segment or decelerate to a stop using register 19. Note that these parameters project the second set of ramping and constant velocity segments and the position parameter for this segment is never realized.

Time decrements down in register 18 as described above.

Segment 3: 1000 1073741824 1073742 536870912
Ramp back down to 1000 RPM (¼ velocity) using .05% acceleration for 1000 ticks (0.12seconds) before looking to load a new segment or decelerate to a stop using register 19. Note that these parameters project a set of decelerating and constant velocity segments, but the time value is chosen so the next set of parameters are loaded before the constant velocity segment starts.

Time decrements down in register 18 as described above.

Segment 4: 2000 92000 536871 536870912
This is the last segment of the profile. Therefore, define position as the desired stopping position of the profile (92000) rather than a large value to simply project the segment. Since this segment decelerates to a stop, define velocity as the starting velocity of the segment (536870912) rather then the ending velocity. Note that the purpose of this segment is to decrease the deceleration to .025% (536871).

Time decrements down in register 18 as described above.

Final Segment: 0 92000 536871 536870912
Feed one additional segment to the IMS operation with a “0” for the time parameter prompting an exit from IMS operation. Repeat position, acceleration, and velocity parameters, as they should have no effect on the profile; the profile was physically completed with the previous segment (Segment 4). This segment takes effectively one servo cycle to execute: the “0” for time is copied to register 18 and is decremented, instantly exiting IMS operation.

Interpolated Motion - Host Based Control

Rather than using the register based queue and writing the reference number of the first of four registers to register 17, write a “0” to the lower word of register 17 indicating to the IMS operation to use the Interpolated Move Queue. This queue is a 4 stage (SilverDust Rev 06 increases this to 8 stages) software FIFO with each stage holding 4 coefficients for each “segment” of the IMS operation. A host can be programmed to control Interpolated Movement via this queue. The host must utilize two commands to accomplish this, Interpolated Move Queue Clear (IMQ) and Interpolated Move Write Queue (IMW).

The IMQ command is sent through the serial interface and clears any residual data that may have been left in the Interpolated Move Queue. The IMW command writes data to the Interpolated Move Queue through the serial interface. If the data is able to fit within the queue, it is accepted and the communication is positively acknowledged (ACK). If the queue is full, the request is answered with a NAK – Full response. This NAK is to be expected: it indicates the host is successfully keeping the queue filled. The same data should be sent repeatedly until it is positively acknowledged (ACK). (Note: the upper word of Register 17 is NOT written as each segment starts, rather the handshaking between the source of segment information and the IMS command is done via the Queue with the NAK – Full response indicating that the host is keeping up with the IMS command. If the Queue is ever empty when the previous segment timer counts to zero, this indicates a communications problem to the host (or other host failure), and the IMS operation is shut down using the value in Register 19 as a deceleration value to end this operation.