Control Technology Corporation

Installation and Applications Guide

Model 2219 Servo Motor Control Module

This document is current as of the following revision levels:

- Controller Firmware – R 2.10
- Board Firmware – R 2.6
- Board Hardware – A

This guide shows how to install your model 2219 Servo Motor Control Module. This module provides fully programmable control over servo motor functions. It uses an independent on-board microprocessor that provides superior performance in multi-axis systems, without degrading the response of the controller's main processor.

All motion parameters for the servo motors are programmable, providing flexibility when initially installing the system and during operation. The 2219 Servo Motor Control Module provides dedicated inputs for many commonly-used functions, such as registration or forward and reverse end limits.

This guide is divided into the following sections:

25 South Street, Hopkinton, MA 01748

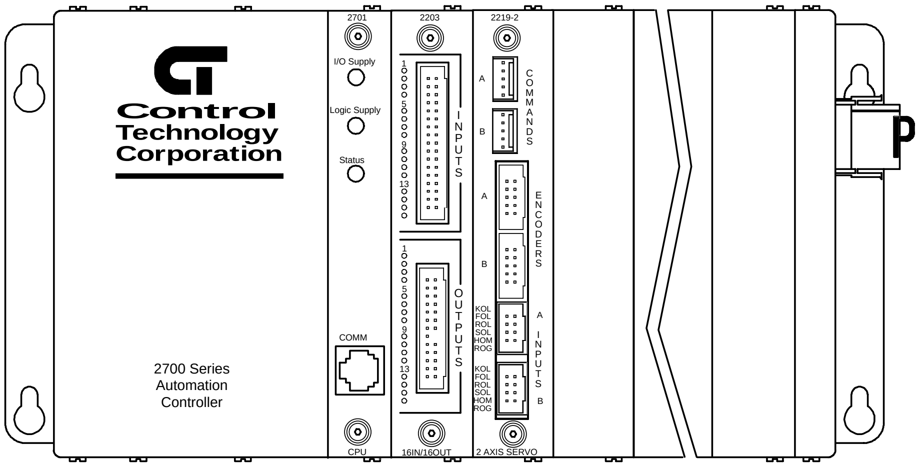
March 30, 2006

Inserting the 2219 Module into Your Controller

The model 2219 Servo Motor Module must be inserted into one of the slots of your automation controller. Any combination of modules may be inserted into the controller, subject to system limits. You may insert them in any order; the controller's CPU dynamically assigns motor numbers, input numbers, output numbers, etc., each time power is reapplied to the controller. These numbers are assigned from left to right across the controller. For example, from slot No. 1 to slot No. 5 in the controller shown on the next page.

To install a module into the automation controller:

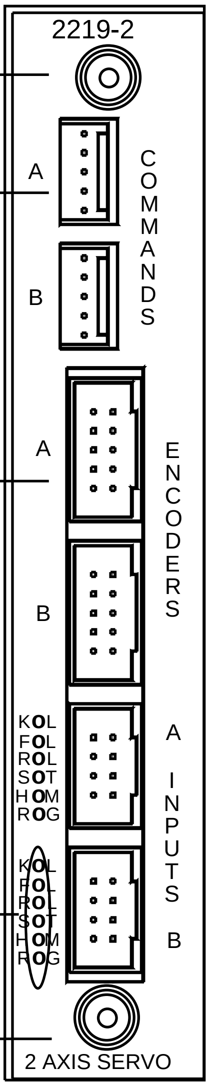
1. Insure that all A.C. and D.C. power to the controller has been removed. This includes any external supplies which may be connected to the controller.
2. Remove the retaining screws from the top and bottom of the cover plate in the position to be used for the new module. Save these screws to re-install the new module.
3. Slide the module into the slot, insuring that the circuit board slides into the nylon guides at top and bottom, and that the card is oriented properly (labelling should read right-side-up).
4. Press the module firmly into the controller; when properly seated, the faceplate of the module should be flush with the adjacent sheet metal surface.
5. Reinstall the retaining screws in the top and bottom of the new module.



Model 2700 Automation Controller shown with a model 2203 Combination I/O Module in the first position, and a model 2219 Servo Motor Control Module in the second position.

Connecting the Model 2219 to External Devices

Front View Model 2219 Servo Motor Control Module



Module Extraction Handle – Do Not Detach!

Command Connector – This connector gives you access to the analog output for commanding a servo drive and the relay outputs for enabling a servo drive. Use a 2289C Pigtail Cable to connect to the servo drive. You can create your own cable using a Molex #22-01-3057 housing with #08-55-0102 contacts.

Encoder Connectors – These 10 pin connectors allow you to connect a quadrature encoder. To connect these inputs use a model 2289E pigtail cable. You can also create your own cable using an AMP #87922-1 housing with #1-87309-4 contacts.

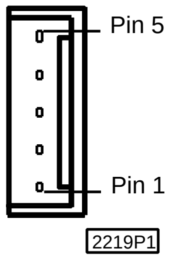
Auxiliary (Dedicated) Input Connectors – These 10 pin connectors provide access to the 6 dedicated inputs (per axis) of the model 2219, as well as providing +24 for powering external sensors. To connect to the inputs use a model 2289L pigtail cable which has a mating connector and 6-foot unterminated wires. You can also create your own cable with an AMP #87631-4 housing and #1-87309-4 contacts.

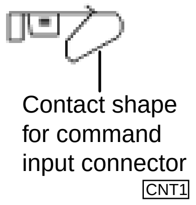
Sensor Indicators – Six LED indicators (per axis) are provided to indicate the status of the home, start, kill command, reverse limit, forward limit, and registration index dedicated inputs.

Module Extraction Handle – Do Not Detach!

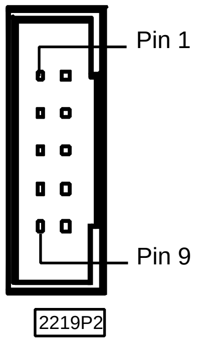
A 2269-1 or 2269-2 connector kit supplies pins and connectors for the entire module.

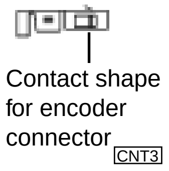
Command Input Pin Connections

Command Connector	Pin #	Signal
	1	Shield
	2	Drive Kill Relay (Common)
	3	Drive Kill Relay (N.O.)
	4	Analog Command Return
	5	Analog Command Output

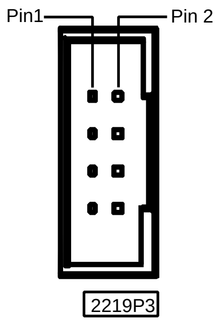


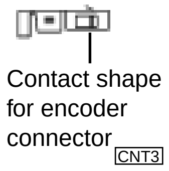
Encoder Input Pin Connections

Encoder Connector	Pin #	Signal
	1	Phase A (+)
	2	Phase A (-)
	3	NC
	4	NC
	5	Index (-)
	6	5 Volt Return
	7	+5 VDC (for encoder)
	8	Phase B (+)
	9	Phase B (-)
	10	Index (+)



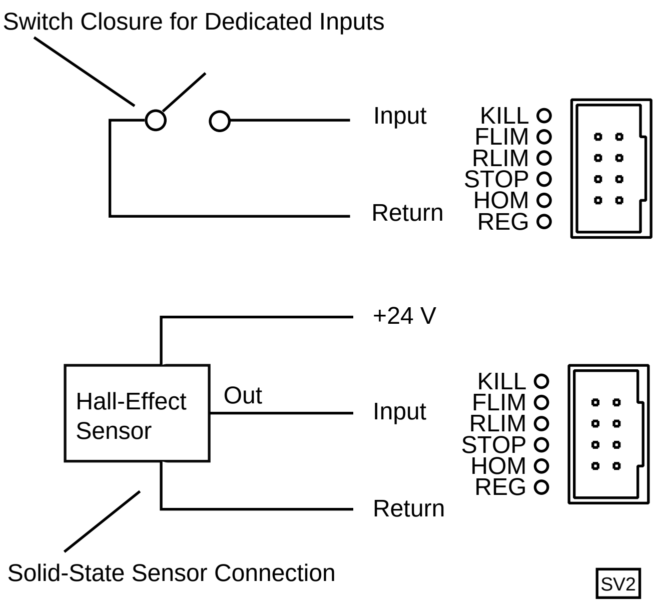
Dedicated Input Pin Connections

Dedicated Input Connector	Pin #	Signal
	1	Start
	2	Kill
	3	Forward Limit
	4	Reverse Limit
	5	Home
	6	Registration Input
	7	+24 VDC
	8	24 Volt Return



Connecting the Model 2219 Dedicated Inputs

The six dedicated inputs on the model 2219 require only a switch closure to return (the common for the controller's 24 volt supply) to be actuated. Each input is internally self-powered from the controller's 24 volt power supply through a current limiting resistor and is opto-isolated from the controller's logic. The following illustration is an example of the switch closure for the dedicated inputs.



Using Solid State Sensors

You can connect many types of electronic sensors to the dedicated inputs. You can connect three wire Hall-effect sensors, proximity sensors, and phototransistors with out any additional circuitry. These devices should be specified as having sinking-type open-collector outputs (NPN) and must be capable of withstanding at least +24 volts on their output terminals when in the off state. The sensor must also be able to sink the required input current, i.e., 2.4 mA, when on.

NOTE: Do not use two-wire solid state sensors.

Electronic sensors typically require an external power source for powering their internal circuitry. If the sensor chosen requires a power supply voltage equal to the controller's built-in auxiliary supply (24 volts), it can be powered directly by the controller, eliminating the need for an additional external supply. The illustration above shows how to connect a solid state sensor

2219 Servo Motor Control Module Specifications

Absolute Maximum ratings	Min	Typ	Max	
Command Load Resistance	2			kΩ
Encoder Input voltage	0.0		+5.0	V DC
Encoder (+ 5 V) Supply Output Current (total - both axes)			500	mA
Ambient Temperature (operating)		0	50	°C

Specification	Min	Typ	Max	
Command Outputs				
Nominal Voltage Range	-10.0		+10.0	V DC
Differential Encoder Inputs				
Nominal Input Range	0.0		+5.0	V DC
Open-circuit Voltage (I_I = 0 mA)		5.0	5.38	V DC
Logic-low Current (I_I = 0 V)		1.1	1.2	mA
Auxiliary Inputs (Except Registration)				
Off Voltage[2] (I_I = 0 mA)		24.0	26.4	V DC
On Current (I_I = 0 V)		2.12		mA
Threshold				
low-to-high		14.0		V DC
high-to-low		12.5		V DC
Registration Auxiliary Input				
Off Voltage[2] (I_I = 0 mA)		24.0	26.4	V DC
On Current (I_I = 0 V)		2.28		mA
Threshold				
low-to-high		5.1		V DC
high-to-low		4.9		V DC

Performance Specifications	Min	Typ	Max	
Maximum Velocity Setting	1		4,000,000	steps/sec
Resolution of Maximum Velocity		1		steps/sec
Acceleration and Deceleration Settings	0		130,000,000	steps/sec^2
Resolution of Acceleration and Deceleration Settings		1		steps/sec^2
Position range	-2,147,483,648		2,147,483,647	steps
Relative Motion Command Range	-2,147,483,648		2,147,483,647	steps
Position Registration Accuracy		$\pm$1		count
Dedicated Input Accuracy (repeatability)		$\leq$ 4.0		ms

Power Supply Requirements (from controller)

Logic Supply (5 V)	260	mA
Auxiliary Supply (24 V)	175	mA

Notes:

1. Specifications shown above are at 25 °C, unless otherwise noted.
2. Dependent on controller auxiliary supply voltage (24 V typ)
3. PID parameters are programmed as relative values in the range of 0 to 255. Acceleration (A_{ff}) and velocity feed forward (V_{ff}) range from 0 to 65535.
4. In Performance Specifications, the term step refers to one edge transition on either encoder input for that axis.
5. Ratio range for both axis following and ratio control is $\pm$1 to 32767 minimum and $\pm$32767 to 1 maximum. Depending on the applications, high ratios may result in instability.

Special Purpose Registers

Group access special purpose registers display the same parameters for all of the 16 axes together. Individual access special purpose registers display all the parameters for a single axis.

NOTE: R indicates that the controller can read the register.
W indicates that he controller can write to the register.

Group Access

Leader On-Start Feature

Registers 13801 - 13816	R/W	Leader on-start enable 0 = disabled 1 = enabled
Registers 13901 - 13916	R/W	Leader position set point for triggering armed axis

Axis Status and Feed Forward Parameters

Registers 14001 - 14016	R/W	Actual position
Registers 14101 - 14116	R only	Position error
Registers 14201 - 14216	R only	Theoretical velocity
Registers 14301 - 14316	R only	Status 0 = Un-initialized 1 = Stopped 2 = Waiting 3 = Accelerating 4 = At speed 5 = Deceleration speed 6 = Decelerating to stop 7 = Commence soft stop 8 = Commence registration move 9 = Searching for home 10 = Following (ratioed from leader) 12 = Command accepted 128-255 = Errors
Registers 14401 - 14416	R only	Cumulative (integrated) position error
Registers 14501 - 14516	R/W	Velocity feed forward constant. Normal values 0 to 65535
Registers 14601 - 14616	R/W	Deceleration rate. Normal values 1 to 130,000,000 pulses/sec^2.
Registers 14701 - 14716	R only	Monitoring dedicated (auxiliary) inputs using a bit map. The default is normally open inputs. bit 0 = Not used bit 1 = Home bit 2 = Start bit 3 = Kill command bit 4 = Reverse limit bit 5 = Forward limit bit 6 = Index bit 7 = Not used
Registers 14801 - 14816	R/W	Acceleration feed forward constant. Normal values 0 to 65535

Individual Access

Axis Status and Feed Forward Parameters

Axis No. 1

Register 15000	R/W	Actual position
Register 15001	R only	Position error
Register 15002	R only	Theoretical velocity
Register 15003	R only	Status 0 = Un-initialized 1 = Stopped 2 = Waiting 3 = Accelerating 4 = At speed 5 = Deceleration speed 6 = Decelerating to stop 7 = Commence soft stop 8 = Commence registration move 9 = Searching for home 10 = Following (ratioed from leader) 12 = Command accepted 128-255 = Errors
Register 15004	R only	Cumulative (integrated) position error
Register 15005	R/W	Velocity feed forward constant. Normal values 0 to 65535
Register 15006	R/W	Deceleration rate. Normal values 1 to 130,000,000 pulses/sec^2.
Register 15007	R only	Monitoring dedicated (auxiliary) inputs using a bit map. The default is normally open inputs. bit 0 = Not used bit 1 = Home bit 2 = Start bit 3 = Kill command bit 4 = Reverse limit bit 5 = Forward limit bit 6 = Index bit 7 = Not used
Register 15008	R/W	Acceleration feed forward constant. Normal values 0 to 65535

Axis No. 2	Register 15010 - 15018
Axis No. 3	Register 15020 - 15028
Axis No. 4	Register 15030 - 15038
Axis No. 5	Register 15040 - 15048
Axis No. 6	Register 15050 - 15058
Axis No. 7	Register 15060 - 15068
Axis No. 8	Register 15070 - 15078
Axis No. 9	Register 15080 - 15088
Axis No. 10	Register 15090 - 15098
Axis No. 11	Register 15100 - 15108
Axis No. 12	Register 15110 - 15118
Axis No. 13	Register 15120 - 15128
Axis No. 14	Register 15130 - 15138
Axis No. 15	Register 15140 - 15148
Axis No. 16	Register 15150 - 15158

Registration Feature

Axis No. 1

Register 16000	R/W	Specifies the position where the registration window begins. Absolute position, specified as the number of steps from the servo's home position. Normal values -2,147,483,648 to 2,147,483,647.
Register 16001	R/W	Specifies the position where the registration window ends. Relative position, specified as the number of steps from the beginning of the registration window. Normal values -2,147,483,648 to 2,147,483,647.
Register 16002	R only	Indicates the position where registration occurred. Absolute position, and is the number of steps from the servo's home position. Normal values -2,147,483,648 to 2,147,483,647.
Register 16003	R/W	Specifies an offset to be added to the location where registration occurred. Relative position, specified as the number of steps from registration position. Normal values -2,147,483,648 to 2,147,483,647.
Register 16004	R/W	Indicates whether or not registration occurred. 1 indicates registration occurred. 0 indicates that the servo is ready for registration move.

Axis No. 2	Register 16010 - 16014
Axis No. 3	Register 16020 - 16024
Axis No. 4	Register 16030 - 16034
Axis No. 5	Register 16040 - 16044
Axis No. 6	Register 16050 - 16054
Axis No. 7	Register 16060 - 16064
Axis No. 8	Register 16070 - 16074
Axis No. 9	Register 16080 - 16084
Axis No. 10	Register 16090 - 16094
Axis No. 11	Register 16100 - 16104
Axis No. 12	Register 16110 - 16114
Axis No. 13	Register 16120 - 16124
Axis No. 14	Register 16130 - 16134
Axis No. 15	Register 16140 - 16144
Axis No. 16	Register 16150 - 16154

Axis Following Feature

Axis No. 1

Register 16005	R/W	Ratio numerator. Normal values ±32767.
Register 16006	R/W	Ratio denominator. Normal values 1 to 32767.
Register 16007	R only	Leader position. Normal values -2,147,483,648 to 2,147,483,647.
Register 16008	R only	Leader velocity. Normal values ±4,000,000.
Register 16009	W access	Transfers data-table row to corresponding cam-table row. Available in the Model 2219-1CF only.
	R access	Reads current cam-table row being executed. Available in the Model 2219-1CF only.

Axis No. 2	Register 16015 - 16019
Axis No. 3	Register 16025 - 16029
Axis No. 4	Register 16035 - 16039
Axis No. 5	Register 16045 - 16049

Axis No. 6	Register 16055 - 16059
Axis No. 7	Register 16065 - 16069
Axis No. 8	Register 16075 - 16079
Axis No. 9	Register 16085 - 16089
Axis No. 10	Register 16095 - 16099
Axis No. 11	Register 16105 - 16109
Axis No. 12	Register 16115 - 16119
Axis No. 13	Register 16125 - 16129
Axis No. 14	Register 16135 - 16139
Axis No. 15	Register 16145 - 16149
Axis No. 16	Register 16155 - 16159

Special Registers for Additional Features

Axis No. 1

Register 17000	R only	Firmware version number
Register 17001*	R/W	Servo filter selection, using a bit map. 0 = default (PID) 1 = Direct CCW 2 = Direct CW 3 = PID 4 = PIVAFF 5 = PAV 7 = Virtual master
Register 17002 *Must be set prior to initial profile instruction.	R/W	Reverses input polarity so that the default is normally closed. Uses a bit map. bit 0 = Not used bit 1 = Home bit 2 = Start bit 3 = Kill command. Cannot be changed, remains open. bit 4 = Reverse limit bit 5 = Forward limit bit 6 = Index. Cannot be changed, remains open. bit 7 = Not used
Register 17003	R/W	Direction of home 0 = default (CCW) +1 = CW -1 = CCW
Register 17005	R only	Number of bytes per row in cam-table

Axis No. 2	Register 17010 - 17015
Axis No. 3	Register 17020 - 17025
Axis No. 4	Register 17030 - 17035
Axis No. 5	Register 17040 - 17045
Axis No. 6	Register 17050 - 17055
Axis No. 7	Register 17060 - 17065
Axis No. 8	Register 17070 - 17075
Axis No. 9	Register 17080 - 17085
Axis No. 10	Register 17090 - 17095
Axis No. 11	Register 17100 - 17105
Axis No. 12	Register 17110 - 17115
Axis No. 13	Register 17120 - 17125
Axis No. 14	Register 17130 - 17135
Axis No. 15	Register 17140 - 17145
Axis No. 16	Register 17150 - 17155

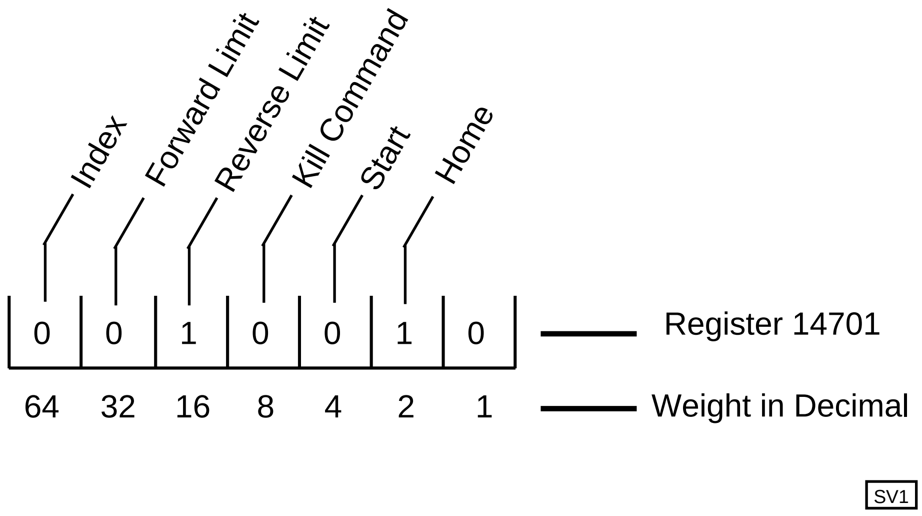
Error limit

If the motor is commanded to operate beyond the physical limits (e.g., to accelerate faster than it has torque available to accelerate the load, or when there is a load greater than the maximum torque — an external jam), the error will increase. This will, for a Servo system, also cause the output command to increase and could cause the motor to overheat.

If the error is larger than the value in the Error Limit register, register 17006, then the servo axis will become un-initialized, any motion will stop, and the output will be set to zero. The default value for this register is 32767, but it can be set to any positive integer value.

Monitoring Dedicated Inputs

Group access registers 14701 - 14716 and individual access registers 15007, 15017, 15027, etc. can return a bit pattern that indicates which, if any, of the dedicated inputs are active. The number of the dedicated input is stored as a binary representation (of the input number) in the register. Each input has its own binary value.



In the illustration above, the home and reverse limit inputs are shown as active. Register 14701 (for axis No. 1) would return a value of 18, because the respective weights of the inputs are 2 and 16. To test any individual input, the bitwise AND instruction may be used to apply a mask to the register. The following instruction applies a bit mask that tests to see if the Home input is active:

```
[1]  TEST_FOR_HOME_AXIS1
    ;;; Home = 2
    ─────────────────────────-──────────────────────

        <NO CHANGE IN DIGITAL OUTPUTS>

    ────────────────────────────────────────────────

    store reg#14701 and 2 to reg#10
    if reg#10 = 2 goto FOUND_HOME
    goto TEST_FOR_HOME_AXIS1
```

The Index bit (64) is inverted. If you are not using the index marker on your encoder, then the controller sets this bit continuously. If you are using the index marker, the controller only set this bit when the encoder position reaches the index position. This occurs only once per revolution of the encoder.

Programming Your 2219 Servo Motor Module

The following Quickstep instructions are used to program your model 2219 Servo Motor Module:

- Profile Servo
- Turn Servo
- Monitor Servo
- Zero Servo
- Search and Zero Servo
- Stop Servo
- If Servo
- Store Servo

Setting Up Servo Motor Operating Parameters

Before the controller can turn your servo motor, it must have a set of operating parameters. These parameters must be specified using the PROFILE SERVO instruction. Operating parameters for the motor are as follows:

- Max Speed – Establishes the maximum speed of the motor.
- Accel Rate – Specifies the acceleration rate of the motor. The deceleration rate is the same as the acceleration rate. See the section on *Setting Acceleration and Deceleration Parameters* for information on setting a different deceleration rate.
- P Parameter — This factor is called the system gain. It specifies the factor applied to the sensed position error to create a correction signal. The gain factor is highly depended of the gain of any external amplifier being used to drive the actuator. Possible values range from 1 to 255.
- I Parameter — The I (integral) factor is used to obtain increased accuracy at low frequencies. It integrates, or builds up, a corrective signal in response to steady-state error. A greater I factor will cause the filter to build up a corrective signal for even small amounts of error, greatly increasing the terminal accuracy of each move. Possible values range from 0 to 255.
- D Parameter — The D (derivative) factor senses and responds to rapidly changing rates of error. It is therefore most useful in increasing system response to varying loads and frictions at high speeds. Possible values range from 0 to 255.
- Holding Mode — Specifies the status of the servo when stopped, using one of the following parameters:
  - — Servo at position — Once the servo reaches the desired position, the actuator will continuously seek this position. If the actuator is forced from its position, the 2219 module sends a correction signal to attempt to correct the perceived error.
  - — Deadband of __ at position — The senses position errors but does not correct them unless the error is out of the range of the Deadband.
  - — Off at position — Once the servo reaches position no further corrective action occurs. This allows manual adjustment or another external force to change the position of the servo.

The maximum speed is expressed in units of steps-per-second (steps/sec). Your programmed maximum speed has a resolution of about 1 steps/sec. The acceleration and deceleration is expressed in units of steps-per-second-per-second (steps/sec^2) with a granularity of about 1 steps/sec^2.

The Profile Servo instruction must appear before the first Turn Servo instruction in your Quickstep program. If it is not executed prior to the first Turn Servo instruction, a software fault stating, “Servo not ready,” results. Additional Profile Servo instructions are only necessary when you want to change the motor’s operating parameters.

Re-profiling on-the-fly, which allows the servo to take on new settings during a motor motion, is possible. To re-profile the servo, program another Profile Servo instruction with a new maximum speed or acceleration value. You do not have to re-specify a value that does not change.

IMPORTANT! Adjustments to the ramping (acceleration and deceleration) parameters while the servo is accelerating or decelerating causes an instantaneous change in the ramp that may be undesirable. To avoid this, make changes to the ramping parameters when the servo is stopped or turning at maximum speed. You can view the status of the servo by checking the appropriate special registers. For example, check register number 14301 for the current status of the first servo. Refer to the list of special registers for additional information.

Using Servo Filters

A servo filter is a high speed calculation used to continuously command an output to a servo system. The 2219 offers a variety of filters that perform this function. The choice of which filter to use is based on the type of servo drive used in your application. If other than the default filter is used (PID), you must set the filter register, associated with each axis, prior to the initial profile.

PID Filter

The 2219 filter setting defaults to a calculation called PID (Proportional, Integral, Derivative). It is intended to be used with drives that are configured for **Torque** mode, sometimes also called **Current** mode. In this case, the command output of the 2219 (0 to ±10VDC) represents zero to full current of your servo drive’s output. The polarity of the command output governs the direction of travel of your servo.

The difference between the actual position of a servo and the intended position is called servo error. Here, it represented in encoder counts. At a rate of 2048 times per second, the 2219 uses the following equation to command the servo:

Servo Output = (position_error * User_Proportional) +
[(position_error - last_position_error) * User_Differential)] + (cumulative_error * User_Integral)

The result of this calculation is scaled into the span of the 2219’s analog output in the form of a new command signal. The 2219 then adds the servo error to the cumulative error and records the servo error in preparation of the next calculation.

PAV Filter

Storing a value of five (5) to the 2219 filter register (Register 17001) prior to the initial profile instruction causes the 2219 to use the PAV (Proportional,Acceleration-Feedforward, Velocity-FeedForward) filter. This filter is intended for use with drives configured for velocity mode. Here, the analog command output of the 2219 (0 to ±10 VDC) represents zero to full velocity of the servo drive’s and motor’s capabilities (or configuration). The polarity of the command output governs the direction of travel of your motor.

The 2219 uses the following calculation when you specify the PAV filter:

Servo Output = (position_error * User_Proportional) +
(change_in_velocity * User_AccelFF) + (current_velocity * User_VelocityFF)

The final result of this calculation is scaled into the span of the analog output of the 2219 in the form of a new command signal.

In this mode, the controller ignores the I gain and the D gain in the profile instruction. However, You must specify some value for them when writing your Quickstep program, for the compiler and proper program operation. Control Tech recommends setting them to zero. The Feedfoward parameters are set using special-purpose registers. See the register description for registers 14501 and 14801 on page 8.

Direct Mode

In applications where a servo-loop is not desired but you wish to command a velocity output, you can set each axis of the 2219 into direct mode. In this case, you can set a register to a value between 0 and 32767 to command a 0 to 10 VDC output. The Velocity-Feedforward register is used for this purpose. See the register description for register 14501 on page 8.

To configure a servo axis into direct mode, you must store a value of one (1) for counter clockwise direction (negative command signal) or a value of two (2) for clockwise direction (positive command signal) prior to profiling the axis. For additional information on specifying servo direction using Direct Mode, see the descriptions for registers 17001 (axis 1), 17011 (axis 2), and so forth.

You must program a complete profile instruction, using the `servo at position` parameter (holding mode) in your Quickstep program, and it must be executed for this feature to be active.

PIVAFF

This filter is for CTC testing only and should not be used.

Virtual Leader

This filter is intended for use in conjunction with the 2219-1CF CAM follower module. The CAM follower concept allows the user to define the amount of distance (encoder counts) traveled based on the amount distance a leader has traveled, in a ratioed fashion. Each relationship is known as a segment. The 2219-1CF allows up to 255 unique segments. This filter allows a second axis to be present on the 2219-1CF but is intended to be used to provide leader trajectory information for use by the follower. The CAM table containing these segments will follow the leaders trajectory using theoretical velocity and position information.

Sample Servo Motor Tuning Program

The following program is a sample program for tuning a servo motor. It consists of two tasks, SERVO_ERROR and RUN_SERVO. SERVO_ERROR monitors the servo error. If the error exceeds the value specified, it turns off the output to the servo driver and stops the servo. RUN_SERVO tunes the servo with the P, I, and D parameters. It turns the servo clockwise and counter clockwise, allowing for a technician to adjust the three turning factors.

```
[1] NEW_SERVO_PROGRAM
    ;;;
    ;;; This is program tunes a servo.  It consists of two
    ;;; tasks: SERVO_ERROR and RUN SERV. If your servo drive
    ;;; must be enabled by turning on an output from the con
    ;;; troller, you must specify and turn on the output in
    ;;; this step. We recommended that you configure your
    ;;; servo amplifier for torque or current mode.
    ————————————————————————————————————-
    OUT_1_ON
    ————————————————————————————————————-
    profile servo_1 servo at position maxspeed=reg_501
      accel=reg_502 P=reg_503 I=reg_504 D=reg_505
    zero servo_1
    monitor   in_1A goto Next

[2] START_TASKS
    ;;;
    ————————————————————————————————————-
    <NO CHANGE IN DIGITAL OUTPUTS>
    ————————————————————————————————————-
    do ( RUN_SERVO SERVO_ERROR) goto NEW_SERVO_PROGRAM

[3] SERVO_ERROR
    ;;;
    ;;; This task monitors servo error and shuts down the
    ;;; drive if servo error is too great.  For tuning pur
    ;;; poses, we use an error of 4000 encoder counts.  For a
    ;;; 500 line encoder, this equates to two revolutions.
    ;;; After the servo is tuned, you may wish to reduce this
    ;;; servo error, if you include such a task in your
    ;;; program.
    ————————————————————————————————————-
    <NO CHANGE IN DIGITAL OUTPUTS>
    ————————————————————————————————————-
    store servo_1:error to reg_515
    store servo_1:position to reg_516
    if servo_1:error > 4000 goto STOP_SERVO
    if servo_1:error < -4000 goto STOP_SERVO
    goto SERVO_ERROR

[4] STOP_SERVO
    ;;;
    ;;; In this program, we assume your servo drive must be
    ;;; enabled by turning on an output. This step stops the
    ;;; servo by sending a hard stop command and by turning
    ;;; off the output that enabled the drive.
    ————————————————————————————————————-
    OUT_1_OFF
    ————————————————————————————————————-
    stop (hard) servo_1
    cancel other tasks
    monitor servo_1:stopped goto NEW_SERVO_PROGRAM

[5] RUN_SERVO
    ;;;
    ;;; This step turns the servo clockwise. While the servo
    ;;; is in motion, it can tuned by programming the tuning
    ;;; parameters to access registers. The program executes
    ;;; a clockwise turn followed by a counter clockwise
    ;;; return. The tuning process is as follows:
    ;;;
    ;;; 1. Set the P parameter to 1, the I parameter to 0, and
    ;;; the D parameter to 0.
    ;;;
    ;;; 2. Set switch 1 and watch/listen to the servo. It
    ;;; should turn, but it will be “mushy”.
    ;;;
```

```
        ;;; 3. While it is turning, increase the D parameter in
        ;;; increments of 10 up to a maximum of 255, until the
        ;;; servo stabilizes.
        ;;;
        ;;; 4. Increase the P parameter until the servo becomes
        ;;; unstable, then reduce it until the servo becomes
        ;;; stabilized again.
        ;;;
        ;;; 5. While monitoring the servo error, increase the I
        ;;; parameter to minimize the servo error to the point
        ;;; where the servo becomes unstable, then reduce it until
        ;;; the servo stabilizes again. Your servo is now tuned!
        ;;;
        ;;; NOTE:  Please insure you have loaded the appropriate
        ;;;        registers with valid values before switch No. 1
        ;;;        is set.
        --------------------------------------
           <NO CHANGE IN DIGITAL OUTPUTS>
        --------------------------------------
        profile servo_1 maxspeed=reg_501 accel=reg_502 P=reg_503
          I=reg_504 D=reg_505
        turn servo_1 to 4000
        monitor servo_1:stopped goto Next

    [6] DELAY_STEP
        ;;;
        --------------------------------------
        <NO CHANGE IN DIGITAL OUTPUTS>
        --------------------------------------
        delay 1 sec goto next

    [7] REVERSE_DIRECTION
        ;;; This step reverses direction of the servo and returns
        ;;; it to the starting position.
        --------------------------------------
        <NO CHANGE IN DIGITAL OUTPUTS>
        --------------------------------------
        profile servo_1 maxspeed=reg_501 accel=reg_502 P=reg_503
          I=reg_504 D=reg_505
        turn servo_1 to 0
        monitor servo_1:stopped goto next

    [10] DELAY_2
        ;;;
        ;;; After the delay, the program returns to the
        ;;; RUN_SERVO step and starts the cycle over.  Since
        ;;; RUN_SERVO step specifies the servo profile, we can
        ;;; change the P, I, D parameters to optimize motor
        ;;; performance.
        --------------------------------------
        <NO CHANGE IN DIGITAL OUTPUTS>
        --------------------------------------
        delay 1 sec goto RUN_SERVO
```

Setting Acceleration and Deceleration Values

The PROFILE SERVO instruction acceleration parameter sets both the acceleration and deceleration values. If you want the acceleration and deceleration values to be different, use one of the group or individual access special purpose registers to set a different deceleration value. For example:

```
profile servo_1 max=50000 accel=100000
store 20000 to reg_15006 (axis No. 1 deceleration register)
```

sets the acceleration equal to 100000 steps/sec^2 and the deceleration equal to 20000 steps/sec^2. Refer to the list of special purpose registers for the appropriate register number for each axis.

NOTE: Setting Up RegistrationIf you specify a new acceleration rate, the deceleration rate is overwritten by the new acceleration rate. You must re-specify the deceleration rate.

Searching for Home

Each servo axis has a dedicated home input. This is most often used in conjunction with the instruction to set a home position for the axis. When home is sensed, the servo stops and the position is set to zero.

The model 2219 Servo Control Module supports a highly accurate method of finding the home position. In addition to providing direct support for a two-stage homing routine, the model 2219 can also make use of the index signal available on many encoders to further increase the consistency of the home position. An additional input is provided for each axis on this module to accept the index signal.

The homing sequence for the model 2219 follows. The directions of travel specified are the default (counter clockwise). It is possible to reverse these directions using a special purpose register.

1. When the controller executes a Search and Zero Servo instruction, the servo begins searching in a counterclockwise direction at the acceleration rate and maxspeed specified in the most recent Profile instruction.
2. When the home input closes (turns on), the servo stops at the profiled deceleration rate.
3. The servo then automatically begins searching in a clockwise direction at a fixed speed of 950 steps per second.
4. When the home input turns on again, the speed decreases to 192 steps per second.
5. When the home input opens (turns off), the servo hard stops.
6. If the encoder's index marker signal is connected to the index input on the module (this is automatically sensed), the servo begins searching in the counterclockwise direction at a speed of 192 steps per second.
7. When the index marker is sensed, the servo hard stops and the position is set to zero.

Specifying the Homing Direction for the Model 2219

The direction of the homing motions described above can be reversed by storing the number 1 to a special purpose register. A different register is used for each axis, as follows:

- Servo axis 1 – register 17003
- Servo axis 2 – register 17013
- Servo axis 3 – register 17023, up to
- Servo axis 16 – register 17153

You can restore operation to the default directions shown above by storing 0 or –1 to these registers.

Turning a Servo

There are three modes of turning the servo:

1. Absolute Positioning – In Absolute Positioning the 2219 Servo Motor Module always references the home (or zero) position in a turn instruction and moves a

specified distance from home position. For example, the following instruction

```
turn servo_12 to 50000 instruction
```

causes the servo to position itself 50000 steps from home. The servo automatically turns in the correct direction to reach the new position.

2. Relative Positioning – In Relative Positioning, the direction of the turn, either clockwise or counter clockwise, is specified in the turn instruction along with a defined number of steps to turn. For example, the following instruction

   ```
   turn servo_1 cw 12340 steps
   ```

   turns the servo 12340 steps clockwise from its current position.

3. Velocity Control – In this case, you establish a direction and begin continuous operation . The maximum speed and acceleration are based on the current profile instruction and can be changed. For example, the following instruction

   ```
   turn servo_1 cw
   ```

   starts the servo turning clockwise at its current maximum speed and acceleration. The servo will continue to turn until the controller issues a STOP SERVO instruction or until a Limit or Stop input is activated.

Once a servo is in motion do not initiate another turn or zero instruction until the motion is complete, or a software fault, “servo not ready,” results. Use the MONITOR SERVO instruction to check the current status (running/stopped) of the servo.

The model 2219 Servo Motor Module tracks the position of the servo through all three types of instructions, allowing you to use all three types of positioning and control in the same program.

NOTE: Quickstep instructions specifying clockwise or counter clockwise operations assume the following:

- The servo is wired according to manufacturer's recommendations.
- The logical sense of the direction output of the 2219 Servo Motor Module agrees with the logical sense expected by the servo's drive.

Stopping the Servo

There are two instructions that terminate the motion of a servo already in motion:

- STOP (SOFT) SERVO causes the servo to stop at the deceleration rate specified in the last profile instruction.
- STOP (HARD) SERVO causes the 2219 Servo Motor Module to try to stop the servo instantly. However, because of momentum, the servo may not stop instantly.

In either case, you should use a MONITOR SERVO STOPPED instruction before issuing another turn instruction.

Monitoring and Changing Other Servo Parameters

There are a number of special purpose registers available that allow you to monitor and change the servo parameters. For more information, refer to the list of special purpose registers for the 2219 Servo Motor Module and your automation controller.

Setting Up Registration

The 2219 Servo Motor Module is set up with a special registration input. The registration input works in conjunction with a series of special registers. If you have an application requiring automatic synchronization of a product from one cycle to another, the registration input in conjunction with the special registers provides a method of recording real-time position information. Each axis on the 2219 has the ability to record the absolute position of the servo when the registration input is activated. The 2219 also has the ability to change the current servo motion and adjust the end position of the move for reliable synchronization. No mater what your application is, registration on the 2219 is so accurate that the servo's absolute position is captured with a resolution of $\pm$ 1 encoder count (step) regardless of the servo's velocity.

Designating a Predefined Registration Window

The registration input on the 2219 module is usually connected to some type of electronic sensor or photo eye. To make use of the registration feature, you need to define the window in which the servo should expect the electronic sensor to activate the registration input.

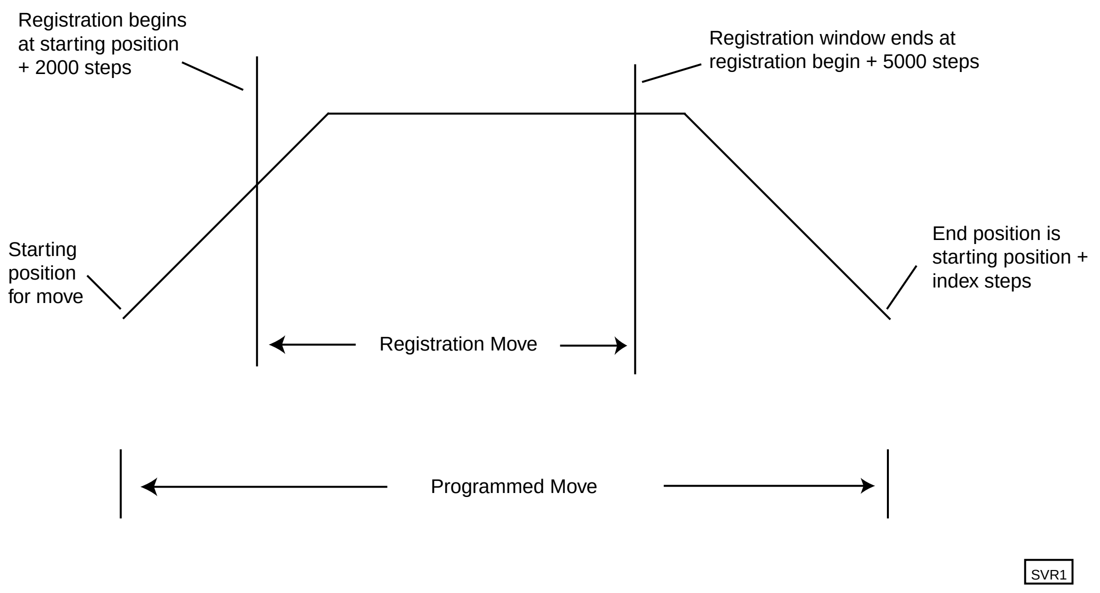
A predefined registration window tells the servo module which part of its move to look for an input from the sensor. The position of the servo is only captured if the sensor triggered the registration input in this window. The 2219 does not record the position of the servo if the sensor triggers the registration input outside of this window. This way no other event can trigger registration. You define the size and range of the registration window using the special registers set up for this purpose. In the following example, the servo is programmed to move a specific distance (labeled Index). This example defines the registration window for servo 1. The registration window is 5000 steps long. The servo begins looking for a registration input when its absolute position is at 2000 steps and ends when its absolute position is at 7000 steps. To define this window, we must enter the following values in registers 16000 and 16001.

- Register 16000 is set to 2000, meaning that the registration window begins at 2000 steps from the beginning of servo 1's move.
- Register 16001 is set 5000, meaning that after the servo travels another 5000 steps the registration window ends.

If the sensor is triggered during the registration window, the controller records the absolute position of the servo in register 16002. If the sensor is triggered outside of the registration window, the controller does not record the servo's position.

At the same time the controller records the servo's absolute position, it also sets the value in register 16004 to 1. As long as the value in register 16004 is 1, the absolute position of the servo where registration occurred is locked into register 16002. The controller does not change the value in register 16002 until the value in register 16004 is reset to zero by your Quickstep program. Resetting register 16004 rearms registration for the next move.

For a listing of special registers, see *Special Purpose Registers*.


Registration begins at starting position + 2000 steps
Registration window ends at registration begin + 5000 steps
Starting position for move
End position is starting position + index steps
Registration Move
Programmed Move
SVR1

The following program shows how to set up and use a predefined registration window:

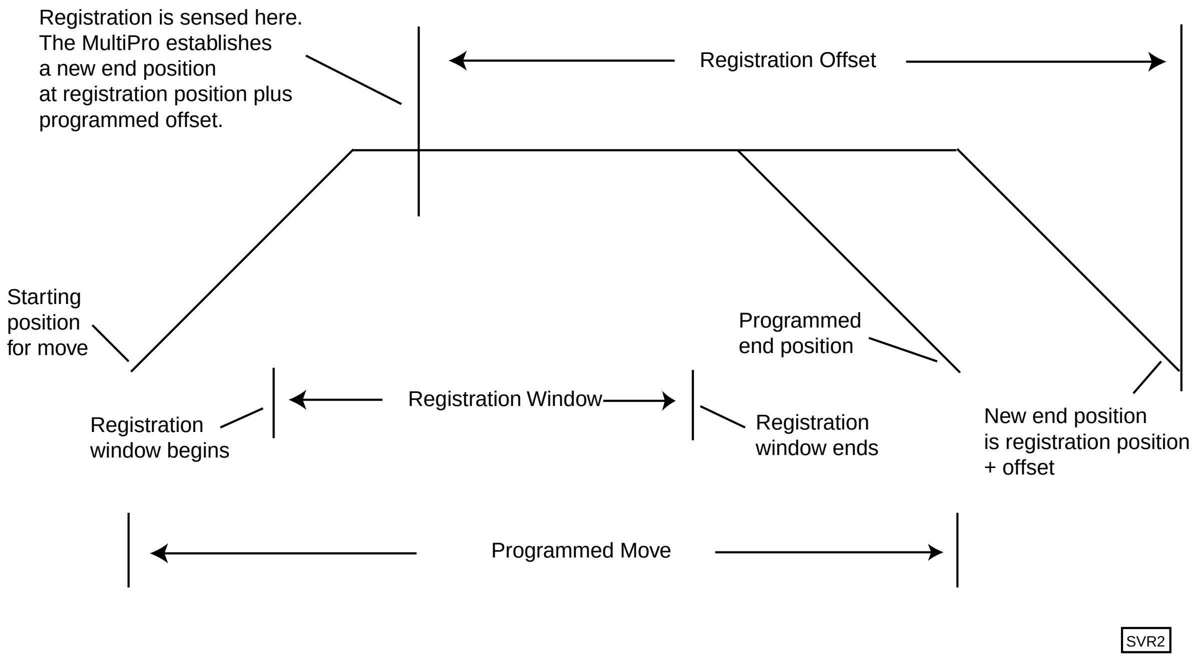
```
[1] REGISTRATION_EXAMPLE
    ;;;
    ____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    ____________________________________
    store 5000 to Reg_16001
    goto Next

[2] REGISTRATION_MOVE
    ;;;
    ;;;
    ____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    ____________________________________
    profile Servo_1 servo at position maxspeed=Reg_501 accel=
      Reg_502 P=Reg_503 I=Reg_504 D=Reg_505
    store Servo_1:position + 2000 to Reg_16000
    store 0 to Reg_16004
    turn Servo_1 ccw Index steps
    monitor Servo_1:stopped goto Next

[3] REGISTRATION_CHECK
    ;;;
    ____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    ____________________________________
    if Reg_16004=1 goto GOT_REGISTRATION
    delay Reg_100 sec goto REGISTRATION_MOVE

[4] GOT_REGISTRATION
    ;;;
    ____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    ____________________________________
    store Reg_16002 to Reg_10
```

Using Registration to Change the End Position of a Move

Some applications require registration to change the end target position of the servo's move while the servo is still in motion. The 2219 module allows you to program an offset position which can be added to the captured registration position. The 2219 module uses this new position to redefine the stopping point for the current motion and overwrites the original destination programmed. This allows for precise correction to your motion based on when registration was sensed.

In the following example, if registration is sensed within the registration window, the servo defines a new end position by adding the number of steps defined in the offset register (register 16003) to the position where the sensor was triggered.

- Register 16000 is set to 2000, meaning that the registration window begins at 2000 steps from the beginning of servo 1's move.
- Register 16001 is set 5000, meaning that after the servo travels another 5000 steps the registration window ends.
- Register 16003 is set to 7500, meaning that the new end position for the move is 7500 steps after the 2219 senses the registration input.



When the sensor is triggered during the registration window, the controller and 2219 does the following:

- Record the absolute position of the servo in register 16002 (for axis 1).
- Calculate a new end position for the servo by adding the position where the registration sensor was triggered (stored in register 16002) and the offset position in register 16003.

- Set the value in register 16004 to 1. As long as the value in register 16004 is 1, the absolute position of the servo where registration occurred is locked into register 16002.

The controller does not change the value in register 16002 until the value in register 16004 is reset to zero by your Quickstep program. Resetting register 16004 rearms registration for the next move.

During registration, The deceleration rate for the servo is always the rate you specified.

The following program shows how to set up and use a registration offset window:

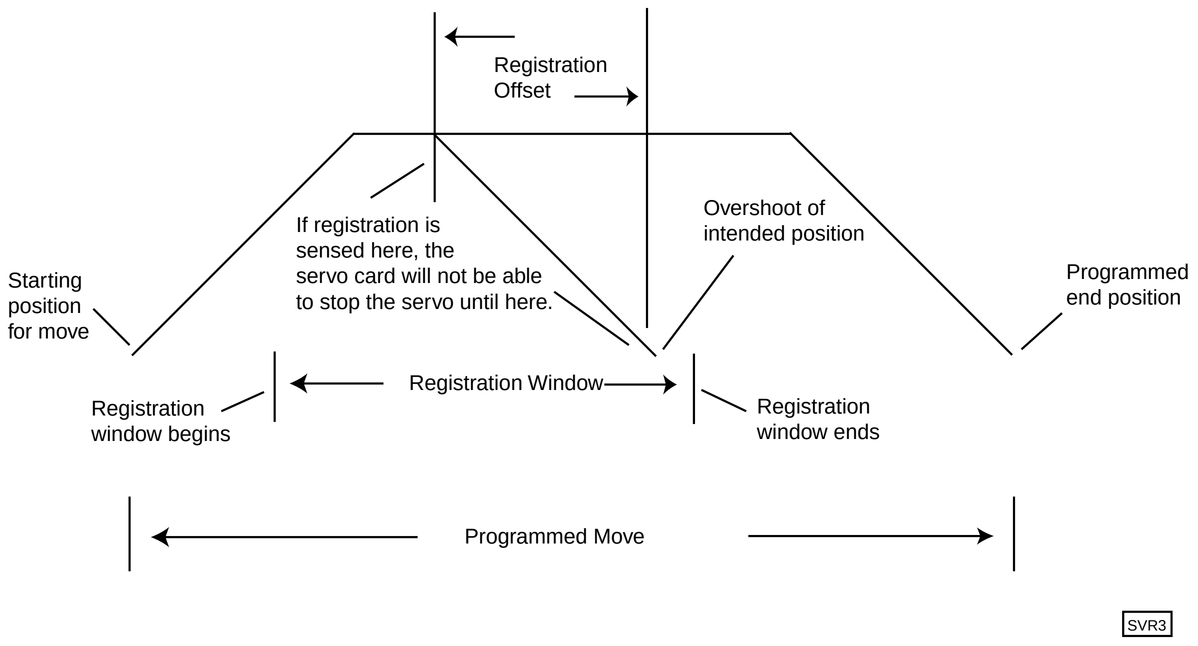
```
[1] REGISTRATION_EXAMPLE
    ;;;
    ;;; Here, we program the registration window for 5000
    ;;; steps and program the registration offset for 7500
    ;;; steps
    ____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    ____________________________________
    store 5000 to Reg_16001
    store 7500 to Reg_16003
    goto Next

[2] REGISTRATION_MOVE
    ;;;
    ____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    ____________________________________
    profile Servo_1 servo at position maxspeed=Reg_501 accel=
      Reg_502 P=Reg_503 I=Reg_504 D=Reg_505
    store Servo_1:position + 2000 to Reg_16000
    store 0 to Reg_16004
    turn Servo_1 ccw Index steps
    monitor Servo_1:stopped goto Next

[3] REGISTRATION_CHECK
    ;;;
    ____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    ____________________________________
    if Reg_16004=1 goto GOT_REGISTRATION
    delay Reg_100 min goto REGISTRATION_MOVE

[4] GOT_REGISTRATION
    ;;;
    ____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    ____________________________________
    store Reg_16002 to Reg_10
```

Overshooting the End Position

When you program a registration offset, the deceleration rate is always the rate you programmed. This makes it possible to write a Quickstep program so that the servo can overshoot the intended end position once the offset is taken into account. The illustration on the following page shows a case where the servo is unable to decelerate in time to stop at the new end position.

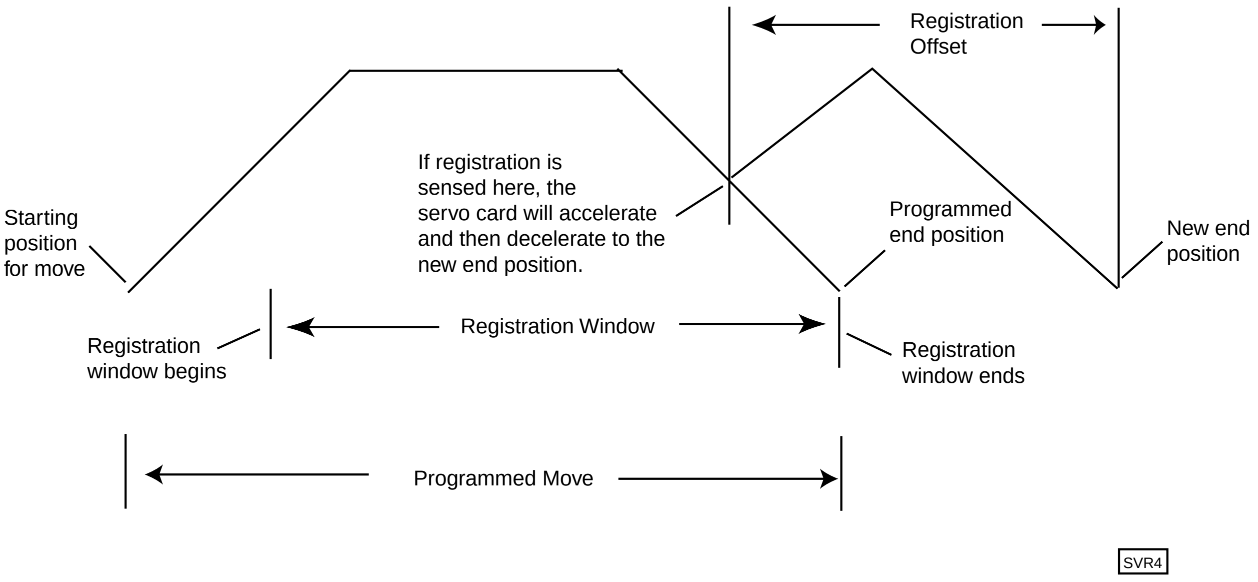
You should take care that this does not happen. Two possible methods of avoiding this problem are as follows:

- Raise the deceleration rate so that the servo can reach the desired offset position
- Lengthen the registration offset value.


Registration
Offset
If registration is sensed here, the servo card will not be able to stop the servo until here.
Overshoot of intended position
Starting position for move
Programmed end position
Registration Window
Registration window begins
Registration window ends
Programmed Move
SVR3

Registration During Deceleration

In cases where the 2219 module senses registration during the deceleration portion of a programmed move. In the following example, the servo re-accelerates when it senses registration and then decelerate to a stop a the new end position. If there is sufficent distance available before the new end position is reached, the servo either accelerates to the maximum speed or accelerates until it is time to decelerate before decelerating to a stop at the new end position. This is an automatic function of the 2219 module


Registration Offset
If registration is sensed here, the servo card will accelerate and then decelerate to the new end position.
Starting position for move
Programmed end position
New end position
Registration window begins
Registration Window
Registration window ends
Programmed Move
SVR4

Guidelines and Rules for Setting up Registration

Following these guidelines and rules will make it easier to program your 2219 module for accurate registration:

- Make sure the registration offset value reflects the direction the servo is traveling. A positive value represents a clockwise direction and a negative value represents a counter clockwise direction. Failure to take the direction into account, results in your servo becoming uninitialized at the point where registration is triggered.
- To inhibit the registration offset function, use a Store instruction to set it to zero.
- The 2219 module senses registration when the state of your sensor changes to the opposite state.

Setting Up Electronic Following

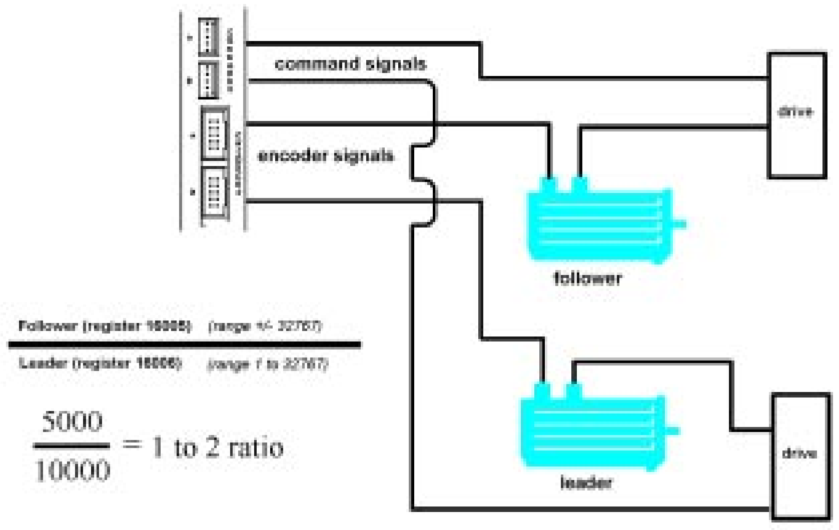
The 2219 Servo Motor Control Module can perform an automatic function called electronic following or ratioing. In electronic following, one servo axis is commanded to match its motions to another axis or encoder based on a specific ratio. You can create this ratioing function using two special registers and two simple Quickstep instructions. You can even adjust the ratios on the fly within your Quickstep program.

Model 2219 Dual Servo Axis-to-Axis Following

Each model 2219 dual servo (2219-2) module can be configured for axis to axis following. The first axis on a 2219-2 module is always the follower and the second axis on the module is always the leader. There are two types of following modes:

- **Trajectory following:** Trajectory following is the default and creates a ratio command to the follower based on theoretical position information of the leader. This mode causes the follower axis to be in phase with the leader axis causing a closer match based on the defined ratio. This mode does not follow encoder pulses coming from the leader but rather its intended (calculated) position.
- **Encoder following:** Encoder following causes the follower to use the leader's encoder information to perform its ratio.

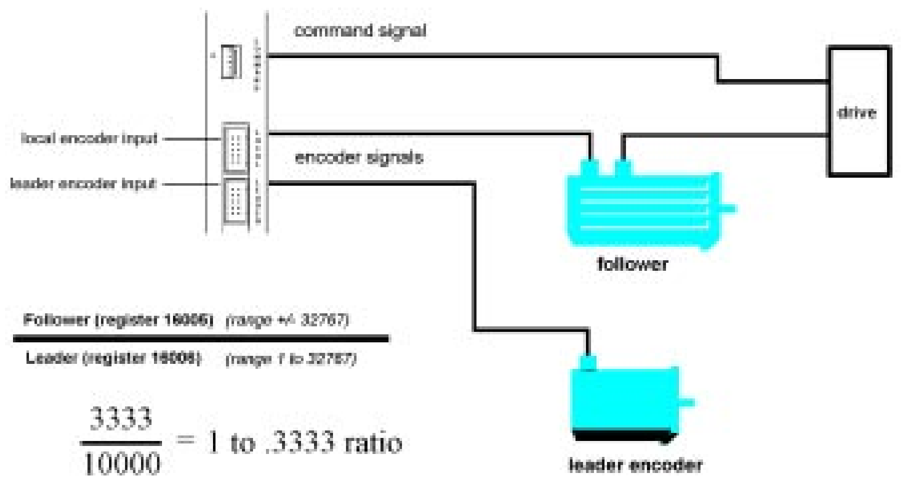
The illustration below shows axis-to-axis following


command signals
encoder signals
drive
follower
Follower (register 16005) (range +/- 32767)
Leader (register 16006) (range 1 to 32767)
5000 / 10000 = 1 to 2 ratio
leader
drive

Model 2219 1.5 Servo Axis Encoder Following

A model 2219 1.5 servo axis (2219-1.5) module can be configured for encoder following. The 2219-1.5 supports a local encoder port, for connection to the servo motor, and a leader encoder port, for connection to your external encoder. The 2219-1.5 defaults to encoder following when placed into ratio mode.

The illustration on the following page shows 1.5 axis encoder following


command signal
drive
local encoder input
leader encoder input
encoder signals
follower
Follower (register 16005) (range +/- 32767)
Leader (register 16006) (range 1 to 32767)
3333 / 10000 = 1 to .3333 ratio
leader encoder

Configuring Electronic Following

To configure an axis for following, you set two special purpose registers for the follower axis as if defining a fraction. The first register specifies the numerator and represents the follower axis, and the second register specifies the denominator and represents the leader axis. It is up to the programmer to decide how the fraction is described; meaning a 1/1 fraction and a 10,000/10,000 fraction both define a one-to-one ratio. However to achieve better resolution in your application you may want to use more places in the fraction. For example, defining a fraction of 9,978/10,000 will cause the follower to be geared slightly lower than leader.

Special registers 16005 - 16155 specify the numerator and registers 16006 - 16156 specify the denominator. Registers 16005 and 16006 are for the first follower servo axis, registers 16015 and 16016 are for the second follower servo axis, and so on.

You must define a complete PROFILE SERVO instruction with working tuning parameters for the follower axis prior to storing the values to the follower's special purpose registers for ratioing. Once the values are stored, the follower is engaged and begins following the leader. While engaged, the follower's status register (register 14301 or 15003 for the first axis) contains the number 10, indicating it is following its leader.

When you activate the follower axis, the servo board automatically resets the leaders postion to zero. When electronic following is active, you cannot reset the leaders position using a Quickstep instruction.

Notes:

1. The maximum values for the fraction is ±32767 / 32767. The sign of the numerator represents the direction the follower will travel with respect to the leader.
2. The 2219 automatically accumulates and adjusts for fractional remainders to maintain synchronization between the follower and the leader. This will account for ratios like 1/3, which calculate a continuous remainder.

Ending Electronic Following

To disengage the axis from following the leader you can store a 0 to the numerator, causing the axis to decelerate to a stop at the profiled deceleration, or you can execute a STOP SERVO (soft) or (hard) instruction. If you have also programmed your servo for a registration move, a valid registration input with an offset value causes the follower axis to depart from the leader. The follower axis then begins the offset move, later coming to a stop.

Reading Current Position and Velocity

Special registers 16007 - 16157 (grouped by ten) specify the current leader position, and special registers 16008 - 16158 (also grouped by ten) specify the current leader velocity registers. Registers 16007 and 16008 are the leader information for the first follower, while registers 16017 and 16018 are the leader information for the second follower axis, and so on. Leader position is expressed in encoder pulses, and leader velocity is expressed as encoder pulse-per-second.

These registers are read-only and can be used to monitor real-time leader activities from within your Quickstep program. They are also updated approximately every 250 ms.

The following examples show how to set up axis-to-axis and encoder following.

```
[11] AXIS_TO_AXIS_FOLLOWING
    ;;;
    ;;; Example Axis-to-Axis following program.
    ;;;
    ;;; Follower_Servo_1 is the follower axis
    ;;; Leader_Servo_2 is the leader axis
    ;;; In this example, the follower will follow
    ;;; the leader at a 1:2 ratio
    ;;;
    ----------------------------------------------------------
    <NO CHANGE IN DIGITAL OUTPUTS>
    ----------------------------------------------------------
    profile Follower_Servo_1 servo at position maxspeed=
     Max accel=Accel P=Pval I=Ival D=Dval
    profile Leader_Servo_2 servo at position maxspeed=
     Max accel=Accel P=Pval I=Ival D=Dval
    store 5000 to Numerator_r16005
    store 10000 to Denominator_r16006
[15] ENCODER_FOLLOWING
    ;;;
    ;;; Example encoder following program.
    ;;;
    ;;; Follower_Servo_1 is the follower axis
    ;;; In this example, the follower will follow
    ;;; the leader at a 1:10 ratio
    ;;;
    ----------------------------------------------------------
    <NO CHANGE IN DIGITAL OUTPUTS>
    ----------------------------------------------------------
    profile Follower_Servo_1 servo at position maxspeed=
     Max accel=Accel P=Pval I=Ival D=Dval
    store 1000 to Numerator_r16005
    store 10000 to Denominator_r16006
```

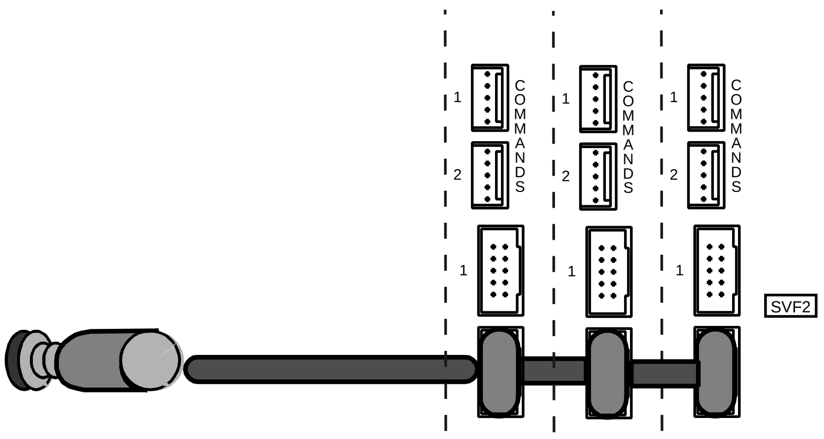
Specifying Encoder Following for a Model 2219-2 module

Storing the number 128 plus the servo filter type code to the servo filter register (register 17001 for the first axis) selects the encoder following mode within axis-to-axis following. For example, `store 133 to register 17001` specifies the PAV filter mode for the first servo axis (5 + 128) and tells it to follow its leader in the encoder following mode.

Interconnecting Multiple Followers with the same Leader:

If your application requires one leader with multiple followers you can daisy-chain the leader encoder signals across up to three 2219-1.5 modules. To daisy-chain more followers, you need an encoder expander board to distribute the encoder signals.

Partial view of encoder sections interconnected as multiple followers



Hardware Considerations

Dedicated Inputs

All dedicated inputs on the 2219 Servo Motor Module are internally pulled-up to 24 VDC, requiring a contact closure to 24 V Return to actuate. The contact closure can be supplied by a mechanical switch or by certain types of solid state open-collector outputs.

- Forward Limit – If the servo is moving clockwise, disables all clockwise movement, attempting to stop the servo instantaneously.[1]
- Reverse Limit – If the servo is moving counter clockwise, disables all counter clockwise movement, attempting to stop the servo instantaneously.[1]
- Kill – places the 2219 in an uninitialized state which sets the command output to 0 V DC. The contact output for this axis opens to disable your servo drive. Refer to *Enabling a Servo Drive* for an explanation about the kill dedicated input.

  In critical or dangerous applications, external means should be used to implement an E-STOP function!
- Home – used to establish a home (zero) reference point for absolute positioning. Refer to *Searching for Home*.
- Start – any motion may, optionally, be programmed to wait for this input.

  To use the Start dedicated input you must program your TURN SERVO instruction using the ON START parameter. The 2219 checks the On Start input every 4 ms, which improves the repeatability of the start motion. The motion does not begin until the Start switch is activated.
- Registration – accurately captures the servo position and, if desired, can alter the move. Refer to *Using the Registration Input* for more information.

NOTE: If a servo has been stopped by any of the above stop inputs, the programmed instruction MONITOR SERVO STOPPED becomes true and your program proceeds to the step specified.

Not all of the dedicated inputs are active at the same time. The following list describes when the dedicated inputs are active and inactive.

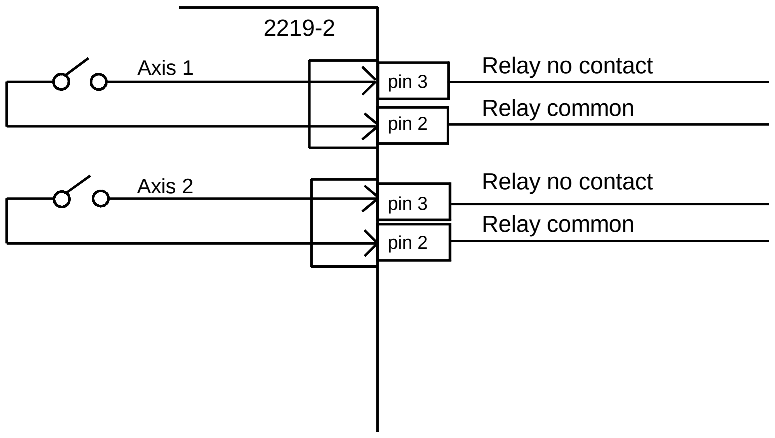
- Forward Limit input – This input is active only when the servo is moving in the clockwise direction. This allows the servo to back off a limit switch, because you can still move counter clockwise.

  For example, if the Forward Limit input is connected to a limit switch for a linear table and the table hits the limit switch, the table can still move counter clockwise and return to the home position.
- Reverse Limit input – This input is active only when the servo is moving in the counter clockwise direction.
- Kill input – This input is active at all times.
- Start input – This input is active when you have programmed a TURN SERVO instruction with an ON START parameter.
- Registration – This input is active only when the registration feature is active.

1. For these inputs to take effect, the 2219 must have full closed loop control. The forward and reverse limits do not stop a motor that has lost its encoder feedback.

Enabling Servo Drives

The following diagram shows how to connect the command output to a servo amplifier enable input. This is Control Tech's recommended method of inhibiting an external servo drive.



The enable relay closes when the controller executes a PROFILE SERVO instruction containing a SERVO AT POSITION parameter for the holding mode. The enable relay remains closed unless one of the following actions occur:

- The servo error exceeds ±32768.[2]
- The kill dedicated input closes.[2]
- The controller executes a PROFILE SERVO instruction containing a MOTOR OFF parameter for the holding mode and stops the servo.

2. When the servo error exceeds the limit or the kill input closes, the servo axis becomes uninitialized. In this case, a new complete PROFILE SERVO instruction must be executed prior to turning the servo.

Sample Quickstep Programs

The following pages contain sample quickstep programs.

Example 1 – Absolute Move of One Servo Motor

This example shows a servo motor moving 100,000 steps from its home position. The `monitor servo_1:stopped` instruction causes the controller's program to remain in this step until the motor completes the move.

Maxspeed units are in steps per second. Acceleration units are in steps per second2.

```
[1] ONE_AXIS_ABSOLUTE_MOVE
    ;;; This program will commence an absolute move on axis one
    ;;; based on the parameters in the profile instruction.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=50000 accel=100000
    turn servo_1 to 100000
    monitor servo_1:stopped goto next
```

Example 2 – Relative Move of One Servo Motor

This example shows a servo motor moving clockwise 100,000 steps from its current position. The `monitor servo_1:stopped` instruction causes the controller's program to remain in this step until the motor completes the move.

Maxspeed units are in steps per second. Acceleration units are in steps per second2.

```
[1] ONE_AXIS_RELATIVE_MOVE
    ;;; This program will commence an relative move on axis
    ;;; one based on the parameters in the profile instruction.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=50000 accel=100000
    turn servo_1 cw 100000 steps
    monitor servo_1:stopped goto next
```

Example 3 – Velocity Move of One Servo Motor

This example shows a servo motor moving clockwise from its current position. The motor will turn until it receives a STOP SERVO instruction or until a stop input is activated. The `monitor servo_1:stopped` instruction causes the controller's program to remain in this step until the motor completes the move.

Maxspeed units are in steps per second. Acceleration units are in steps per second^2.

```
[1] ONE_AXIS_VELOCITY_MOVE
    ;;; This program will commence a velocity move on axis one
    ;;; based on the parameters in the profile instruction.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS
    ─────────────────────────────────────
    profile servo_1 maxspeed=50000 accel=100000
    turn servo_1 cw
    monitor servo_1:stopped goto next
```

Example 4 – Absolute Move of Two Servo Motors

This example shows two servo motors. The motor connected to the first axis moves 100,000 steps from its home position, and the motor connected to the second axis moves 50,000 steps from its home position. The monitor servo instruction, `monitor (and servo_1:stopped servo_2:stopped)` causes the controller's program to remain in this step until both motors complete their moves.

You can re-profile either motor at any time to establish a new velocity.

If you want to start two or more axes simultaneously, you can program the TURN SERVO instructions using the ON START parameter. The motors' motion will begin once the start input located on each axis is triggered. This will start all motions within one millisecond. Refer to the section on *Dedicated Inputs* for information and instructions on using the ON START parameter and the Start dedicated input.

```
[1] TWO_AXIS_ABSOLUTE_MOVE
    ;;; This program will commence an absolute move on two motor
    ;;; axes based on the parameters in the profile instructions.
    ;;;
    ;;; Maxspeed units are in steps per second.
    ;;; Acceleration units are in steps per second
    ;;; per second.
    _____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    _____________________________________
    profile servo_1 maxspeed=50000 accel=100000
    profile servo_2 maxspeed=25000 accel=50000
    turn servo_1 to 100000
    turn servo_2 to 50000
    monitor (and servo_1:stopped servo_2:stopped) goto next
```

Example 5 – Staggering the Motion of Two Servo Motors

This example shows two servo motors. The motor connected to the first axis moves 100,000 steps from its home position, and the motor connected to the second axis moves 25,000 steps from its home position.

The first motor is put in motion. When the position of the first motor reaches half the travel distance, the program moves on to the next step and begins the motion of the second motor. The `monitor (and servo_1:stopped servo_2:stopped)` instruction causes the controller's program to remain in the second step until both motors complete their moves.

Maxspeed units are in steps per second. Acceleration units are in steps per second^2. So, any ratio between multiple axes may achieved by applying the following formula:

- Velocity = Acceleration * Time
- Acceleration = Velocity / Time

In this example, we are running a 2 to 1 ratio between the two axes. These motions will ramp up and down simultaneously.

```
[1] TWO_AXIS_STAGERED_MOVE
    ;;; This program will commence an absolute move on two motor
    ;;; axes based on the parameters in the profile instructions.
    _____________________________________
    <NO CHANGE IN DIGITAL OUTPUTS>
    _____________________________________
    profile servo_1 maxspeed=50000 accel=100000
    profile servo_2 maxspeed=25000 accel=50000
    turn servo_1 to 100000
    if servo_1:position >= 50000 goto next

[2] TRIGGER_SECOND_AXIS
    ;;; Turn the second axis and wait for both axes to be
    ;;; complete before moving on the next part of the program.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    turn servo_2 to 25000
    monitor (and servo_1:stopped servo_2:stopped) goto next
```

Example 6 – Velocity Move of Two Servo Motors

This example shows two servo motors moving clockwise from their current positions. The motors will turn until it receives a STOP SERVO instruction or until a stop input is activated. The `monitor (and servo_1:stopped servo_2:stopped)` instruction causes the controller's program to remain in this step until both motors complete their moves.

You can re-profile either motor at any time to establish a new velocity.

If you want to start two or more axes simultaneously, you can program the TURN SERVO instructions using the ON START parameter. The motors' motion will begin once the start input located on each axis is triggered. This will start all motions within one millisecond. Refer to the section on *Dedicated Inputs* for information and instructions on using the ON START parameter and the start dedicated input.

```
[1] TWO_AXIS_VELOCITY_MOVE
    ;;; This program will commence a velocity move on two motor
    ;;; axes based on the parameters in the profile instructions.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=50000 accel=100000
    profile servo_2 maxspeed=25000 accel=50000
    turn servo_1 cw
    turn servo_2 cw
    monitor (and servo_1:stopped servo_2:stopped)
    goto next
```

Example 7 – Changing the Velocity of a Servo Motor During Motion

This sample program positions a servo motor while generating various velocity profiles throughout the move. After the initial parameters are set, the motor motion is started. When the position reaches 50000 steps, the program continues to next step. Each subsequent step changes the velocity and specifies the servo position where the program moves to the next step.

Maxspeed units are in steps per second. Acceleration units are in steps per second^2.

```
[1] COMPLEX_PROFILE
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=10000 accel=200000
    turn servo_1 to 500000
    if servo_1:position >= 50000 goto next
```

```
[2] SECOND_PROFILE
    ;;; Re-profile the motor for the a new velocity.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=20000
    if servo_1:position >= 70000 goto next

[3] THIRD_PROFILE
     ;;; Re-profile the servo for a new velocity.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=50000
    if servo_1:position >= 110000 goto next

[4] FOURTH_PROFILE
    ;;; Re-profile the servo for a new velocity.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=100000
    if servo_1:position >= 300000 goto next

[5] FIFTH_PROFILE
    ;;; Re-profile the servo for the next velocity.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=80000
    if servo_1:position >= 420000 goto next

[6] SIXTH_PROFILE
    ;;; Re-profile the servo for the final velocity and wait for
    ;;; the move to complete.
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 maxspeed=30000
    monitor servo_1:stopped goto PROFILE_COMPLETE
```

Example 8 – Ratio of two Servo Axes Follower to Leader

```
[1] RATIO_TWO_AXES
    ;;; This example sets up a ratio between two servo motors.
    ;;; Servo axis one (servo_1) is the follower and servo axis
    ;;; two (servo_2) is the leader.
    ;;;
    ;;; This step sets up the initial parameters for the servo
    ;;; motion and programs a ratio for servo_1.  The ratio is 2
    ;;; to 1. From this point on, servo_1 follows all leader
    ;;; positions and velocity activities using the ratio.
    ;;;
    ;;; When programming a follower to leader ratio on the 2219,
    ;;; the first servo axis must be the follower axis, and the
    ;;; second axis must be the leader axis.
    ;;;
    ─────────────────────────────────────
    <NO CHANGE IN DIGITAL OUTPUTS>
    ─────────────────────────────────────
    profile servo_1 servo at position maxspeed=reg_501
      accel=reg_502 P=reg_503 I=reg_504 D=reg_505
    profile servo_2 servo at position maxspeedmaxspeed=reg_501
      accel=reg_502 P=reg_503 I=reg_504 D=reg_505
    store 5000 to reg_16005
    store 10000 to reg_16006
    goto next

[2] LEADER_MOVE_FORWARD
    ;;; This step moves servo_2 (the leader) clockwise. The fol-
    ;;; lower (servo_1) follows axis two at the ratio.
    ------------------------------------
    <NO CHANGE IN DIGITAL OUTPUTS>
    ------------------------------------
    turn servo_2 cw reg_506 steps
    monitor servo_2:stopped goto next

[3] TIME_DELAY
    ------------------------------------
    <NO CHANGE IN DIGITAL OUTPUTS>
    ------------------------------------
    delay 500 ms goto next

[4] MASTER_MOVE_REVERSE
    ------------------------------------
    <NO CHANGE IN DIGITAL OUTPUTS>
    ------------------------------------
    turn servo_2 ccw reg_506 steps
    monitor servo_2:stopped goto RATIO_TWO_AXES
```

Example 9 – Ratio Axis to Leader Encoder

```
[1] RATIO_AXIS_TO LEADER_ENCODER
    ;;; This example sets up a ratio of a servo motor (axis 1) to
    ;;; an external encoder. Axis 1 is the follower axis and the
    ;;; external encoder is the leader axis.
    ;;;
    ;;; This step sets up the initial parameters for the servo
    ;;; motion and programs a ratio for axis 1. The ratio is 2
    ;;; to 1. From this point on, axis 1 follows all leader posi-
    ;;; tions and velocity activities using the ratio specified.
    ------------------------------------
    <NO CHANGE IN DIGITAL OUTPUTS>
    ------------------------------------
    profile servo_1 servo at position maxspeed
    store 5000 to reg_16005
    store 10000 to reg_16006
    goto next
```

Register 16005 contains the ratio numerator, and register 16006 contains the ratio denominator for the follower axis.