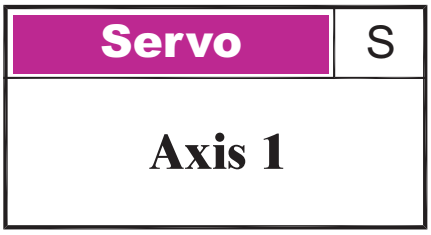
Dual Servo Motion Module Applications Guide

Contents



Doc. No. 5140AG
Revision A
June 2002

The following are trademarks of Control Technology Corporation:

- Quickstep
- CTC Monitor
- CTC Utilities

Windows is a trademark of Microsoft Corporation.

Programming a Servo

The following Quickstep instructions are used to program a servo motor:

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

Note

The registers mentioned in this section only apply to Axis 1. Refer to *Special Purpose Registers for Servos* for more information on other special registers.

Setting Up Servo Motor Operating Parameters

The 5100 must have a set of operating parameters before it can turn a servo motor. You must specify these parameters with the PROFILE SERVO instruction. The servo motor operating parameters 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. Refer to *Setting Acceleration and Deceleration Values* for information on setting a different deceleration rate.
- **P Parameter** — The P parameter is the system gain. It specifies the factor applied to the sensed position error to create a correction signal. The gain factor is highly dependent on the gain of any external amplifier that is 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 a steady-state error. A greater I factor causes the filter to build up a corrective signal for even small amounts of error and greatly increases 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 and is most useful in increasing the system response to varying loads and friction 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 5100 sends a correction signal and attempts to correct the perceived error.

  – *Deadband of __ at position* — The servo senses position errors but does not correct them unless the error is out of the range of the deadband. This parameter is specified with encoder counts.

  – *Off at position* — Once the servo reaches its position, no further corrective action occurs. This allows manual adjustment or another external force to change the servo's position.

Notes

1. The maximum speed is expressed in units of steps-per-second (steps/s). The programmed maximum speed has a resolution of 1 step/s. Acceleration and deceleration are expressed in units of steps-per-second-per-second (steps/s^2) with a granularity of 1 step/s^2.

2. The PROFILE SERVO instruction must appear before the first TURN SERVO instruction in your Quickstep program. If it is not executed before 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.

3. 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.

4. 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 is 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.

Using Servo Filters

A servo filter is a high speed calculation that continuously commands a servo system's output. The 5100 offers a variety of filters that perform this function. The filter you choose depends on the type of servo drive used in your application. If the default filter (PID) is not used, you must set the filter register associated with each axis before the initial profile instruction.

PID Filter

The 5100's default filter setting is a calculation called PID (Proportional, Integral, Derivative). It is generally used with drives configured for **Torque,** or **Current**, mode. In this case, the command output (0 to ±10 VDC) represents zero to full current of your servo drive's output. The polarity of the command output governs your servo's direction of travel.

The difference between the actual position of a servo and the intended position is called servo error. This error is represented by encoder counts. At a rate of 2,048 times per second, the 5100's servo board 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 servo board's analog output in the form of a new command signal. The 5100's servo board then adds the servo error to the cumulative error and records the servo error in preparation for the next calculation.

PAV Filter

The PAV (Proportional, Acceleration-Feedforward, Velocity-Feedforward) filter is selected by storing a value of 5 to the filter register (Register 17001) before the initial profile instruction. This filter is generally used with drives configured for velocity mode. The servo board's analog command output (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 your motor's direction of travel.

The 5100's servo board 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 servo board's analog output in the form of a new command signal.

In this mode, the 5100 ignores the I gain and the D gain in the profile instruction. However, you must assign values to these parameters when you write your Quickstep program for the compiler and for proper program operation. CTC recommends that you set these values to 0. The Feedforward parameters are set with special-purpose registers 14501 and 14801.

Direct Mode

You can set each axis of the 5100 into direct mode for applications where a servo loop is not desired but you wish to command a velocity output. Set a register to a value between 0 and 32767 to command a 0 to 10 VDC output. Register 14501, which is the Velocity-Feedforward register, is used for this purpose.

To configure a servo axis into direct mode, you must store a value of 1 for counterclockwise direction (negative command signal) or a value of 2 for clockwise direction (positive command signal) before profiling the axis. Refer to the description of register 17001 in *Special Purpose Registers for Servos* for more information on specifying servo direction with direct mode.

You must program a complete profile instruction in your Quickstep program and it must be executed to activate this feature.

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 specified value, it turns off the servo driver's output and stops the servo. `Run_Servo` tunes the servo with the P, I, and D parameters. It turns the servo clockwise and counterclockwise and allows a technician to adjust the three tuning factors.

```
[1] New_Servo_Program
    ;;; This program tunes a servo for Torque mode operation.
    ;;; It consists of two tasks: Servo_Error and Run_Servo.
    ;;; If your servo drive must be enabled by turning on an
    ;;; output from the MultiPro, you must specify and turn
    ;;; on the output in this step.
    ------------------------------------------------------------------------
    <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
    ;;; purposes, 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 counterclockwise
    ;;; 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.
    ;;; 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. 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
    ;;; the Run_Servo step specifies the servo profile, we can
    ;;; change the P,I, and 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 100,000 steps/s^2 and the deceleration equal to 20,000 steps/s^2.

Note

If you specify a new acceleration rate, it overwrites the existing deceleration rate. Therefore, you must specify a new deceleration rate.

Searching for Home

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

The 5100 supports a highly accurate method of finding the home position. In addition to providing direct support for a two-stage homing routine, the 5100 also makes use of the index signal available on many encoders to further increase the consistency of the home position. The encoder has a connector that provides an additional input for each axis on the 5100 to accept the index signal.

The servo travels in a counterclockwise (default) direction unless otherwise specified. You can change this default setting with a special purpose register. The homing sequence is as follows:

1. When the 5100 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

You can reverse the direction of the homing motions described above by storing the number 1 to special purpose register 17003. You can restore the default setting mentioned above by storing 0 or –1 to the register. You can also skip searching for the home input and search only for the encoder index pulse. Store a 3 to search for the index in a CW direction and a -3 for the CCW direction.

Turning a Servo

There are three modes of turning the servo:

1. **Absolute Positioning** – In this mode, the 5100's servo board always references the home (or zero) position in a turn instruction and moves a specified distance from the home position. For example, the following instruction

   ```
   turn servo_1 to 50000
   ```

   causes the servo to position itself 50,000 steps from home. The servo automatically turns in the correct direction to reach the new position.

2. **Relative Positioning** – In this mode, the direction of the turn (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 12,340 steps clockwise from its current position.

3. **Velocity Control** – In this mode, 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 continues to turn until the 5100 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 the "servo not ready" software fault occurs. Use the MONITOR SERVO instruction to check the current status (running/stopped) of the servo.

The 5100's servo board tracks the position of the servo with all three modes and allows you to use all three types of positioning and control in the same program.

Note

Quickstep instructions specifying clockwise or counterclockwise operation assume that the servo is wired according to the manufacturer's recommendations and that the logical sense of the direction output of the 5100's servo board 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 5100's servo board 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.

Servo Hardware Considerations

This section discusses the 5140's encoder and limit switch inputs.

Encoder

The encoder inputs accept a quadrature differential signal for the A and B encoder channels. The index pulse, or Z channel, is single-ended. The direction is counted positive, or clockwise (CW) when the A encoder phase leads the B encoder phase.

The first three LEDs for each servo axis indicate the states of the A, B, and Z channels respectively. 5 VDC power is available from the main 5100 CPU power connector at position TB1-5. The 5 V return is common to the controller's 24 V return. You can connect the return to the main CPU power connector at TB2-5 or from either axis on the SS module at TB2-1.

It is usually not necessary to connect power for the encoder when you use the encoder outputs on a servo drive. In this case, it is recommended that you connect the controller's 24V return to the common or return for the servo drive's encoder outputs. This limits the common mode voltage between the drive and controller and helps protect the encoder input circuits from damage caused by overvoltage.

Limit Switch Inputs

You can assign the following servo functions to any of the 5100 controller's first 16 general purpose inputs:

- **Start Input**: Begins motion from a turn "on start" Quickstep instruction.
- **Reset Input**: Uninitializes the servo axis and sets the command output to 0 V.
- **Home Input:** Used during a search and zero instruction.
- **Forward Limit:** Prevents motion in the forward (+) direction; hard stops motion if activated during a forward move.
- **Reverse Limit**: Prevents motion in the reverse (-) direction; hard stops motion if activated during a reverse move.

These inputs are assigned using the special purpose registers in Table 1.

Table 1. Servo Functions and General Purpose Inputs

	Axis 1	Axis 2	Axis 3	Axis 4
Start	15160	15170	15180	15190
Reset	15161	15171	15181	15191
Home	15162	15172	15182	15192
Fwd Lim	15163	15173	15183	15193
Rev Lim	15164	15174	15184	15194

To reverse the polarity of any inputs, use register 17002 for axis 1, 17012 for axis 2, 17022 for axis 3, and 17032 for axis 4 as shown in Table 2.

Note

These registers use a bit mask.

Table 2. Reversing the Polarity of Inputs

Bit	0	1	2	3	4	5	6	7
Input	None	Home	Start	Reset	Rev Lim	Fwd Lim	Index	None
Value	1	2	4	8	16	32	64	128

To reverse the polarity on selected inputs, add the values for the appropriate inputs together and store to that register. For example, to change the polarity of the Start and Reverse Limit Inputs only, add the appropriate values (4+16 = 20) together and store the result to the register for the desired axis. So the Quickstep instruction

```
Store 20 to Register_17002
```

changes the polarity of the Start and Reverse Limit inputs for axis 1.

Sample Quickstep Programs

This section contains sample Quickstep programs for different servo applications.

Note

Maxspeed units are in steps/s. Acceleration units are in steps/s^2.

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 5100's program to remain in this step until the motor completes the move.

```
[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 5100's program to remain in this step until the motor completes the move.

```
[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 turns until it receives a STOP SERVO instruction or until a stop input is activated. The `monitor servo_1:stopped` instruction causes the 5100's program to remain in this step until the motor completes the move.

```
[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 — Changing the Velocity of a Servo Motor During Motion

This sample program positions a servo motor and generates various velocity profiles throughout the move. After the initial parameters are set, the motor motion is started. When the position reaches 50,000 steps, the program continues to the next step. Each subsequent step changes the velocity and specifies the servo position where the program moves to the next step.

```
[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 5 — Velocity Move of Two Servo Motors

This example shows two servo motors moving clockwise from their current positions. The motors will turn until they receive a STOP SERVO instruction or until a stop input is activated. The `monitor (and servo_1:stopped servo_2:stopped)` instruction causes the 5100'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 axes simultaneously, you can program the TURN SERVO instructions using the ON START parameter. The motion of each motor will begin once the start input located on each axis is triggered. This will start all motion within one millisecond. Refer to *Dedicated Inputs* for information 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 6 — 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 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 with the ON START parameter. Both motors begin moving once the start input located on each axis is triggered. This action starts all motions within one millisecond. Refer to *Dedicated Inputs* for more information 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 7 — 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 first motor's position reaches half the travel distance, the program moves on to the next step and starts the motion of the second motor. The instruction

```
monitor (and servo_1:stopped servo_2:stopped)
```

causes the controller's program to remain in the second step until both motors complete their moves.

Any ratio between multiple axes may achieved by applying the following formula:

- Velocity = Acceleration * Time

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

```
[1] TWO_AXIS_STAGGERED_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
```