CONTROL TECHNOLOGY CORPORATION

Model 5200 Motion Module Applications Guide

Model 5200 Motion Module Applications Guide

Blank

WARNING: Use of CTC Controllers and software is to be done only by experienced and qualified personnel who are responsible for the application and use of control equipment like the CTC controllers. These individuals must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and/or standards. The information in this document is given as a general guide and all examples are for illustrative purposes only and are not intended for use in the actual application of CTC product. CTC products are not designed, sold, or marketed for use in any particular application or installation; this responsibility resides solely with the user. CTC does not assume any responsibility or liability, intellectual or otherwise for the use of CTC products.

The information in this document is current as of the following Hardware and Firmware revision levels. Some features may not be supported in earlier revisions. See www.ctc-control.com for the availability of firmware updates or contact CTC Technical Support.

Model Number	Hardware Revision	Firmware Revision
5200	All Revisions	>= 5.00.31

TABLE OF CONTENTS

Blank

CHAPTER 1

Introduction

This document provides details about programming Quickstep programs for 5200 series controllers containing M1-40A Dual Servo Motor Controllers and/or M1-50A Dual Stepper Motor Controllers.

Servo Motors

A servo motor is used in a closed loop control system, where the controller has information about both the actual position (and velocity) of the motor as well as the desired position (or velocity), and then controls the motor (adjusts the output) to remove the difference between the actual and desired values. Because this system has information about the error, and the output (which is usually proportional to the motor power) increases as the error increases, it can require very little power when the error is small. This means that the average power needed for a high performance application may be considerably less than the peak power, so smaller motors and drives may be used.

There is usually a Servo Drive module between the motion controller and the motor that accepts the control signal from motion controller (a low current signal in the range -10 Volts to +10 Volts) and converts it into the high power (depending on the motor, several amps of current at 24V to 200V) signals required by the motor. The Servo Drive must usually be configured to match the Motor (or is designed specifically for the motor) and are often made by the same manufacturer.

The motor may be a simple brush type DC motor (which is low cost but requires periodic replacement of brushes) or a Brushless DC or AC motor which requires additional circuitry in the Servo Drive to handle electronic commutation and will generally require additional sensors and signals from the motor to the driver.

The M1-40A module is a dual axis Servo controller that can control 1 or 2 servo motor systems with Analog Torque or Velocity control and Quadrature encoder feedback. The analog output is used, via a servo amplifier, to control the current in a DC motor generating torque at the shaft. The amplifier may also handle other functions such as commutation for a brushless motor or it may use the analog input to control the velocity. From the motor, an encoder (usually an optical encoder) generates two pulse signals that

are used by the M1-40A module to track the motor position. The M1-40A module can also accept a third encoder channel (the Z axis or Index) that can be used to identify a specific point in the motor rotation.

The M1-40A module also has an additional input, at standard I/O voltage levels, that can be used to modify the motion to synchronize with external devices. This operation is described below as Registration.

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 5200 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 M1-40A module at TB2-1.

For some Brushless Servo system the Servo Drive also uses an encoder for information about the position and will provide a set of suitable Encoder outputs for connection to the Servo Controller. In this case the power for the encoder is usually provided by the Servo Drive and it is not necessary to connect power for the encoder, however 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 over voltage.

 *When the encoder output is provided by the Servo Drive, case must be taken that the signals are actually encoder signals and are not a simulated encoder generated by the Servo Drive from other signals. When the outputs are simulated encoder signals there is generally a delay between the movement of the motor and the encoder signal being generated. Where this delay is small this is not a concern, but since the M1-40A updates the servo command at a rate of over 2 kHz, delays as small as 200 µs can be significant.*

Command Output

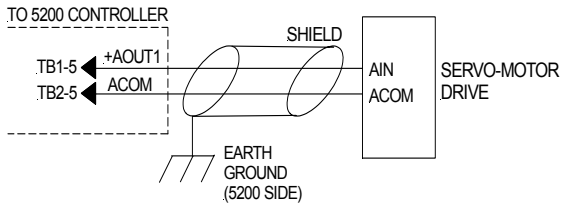
The output of the Servo Controller is an Analog signal that can vary from -10V to 10V with 16 bit resolution. Care must be taken in the wiring to minimize the possibility of errors being introduced into the signal by noise induced from any high power switching circuitry near to the system, since this will directly affect the quality of the control.

Shielded cabling must be used between the Servo Controller and the Servo Drive and the distance between them should be minimized.

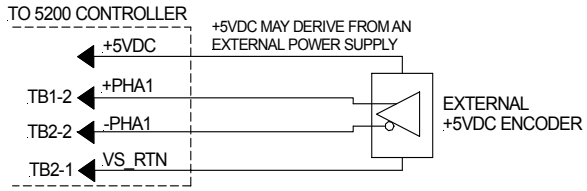
Connections: M1-40A Dual Axis Servo Module

Application Information

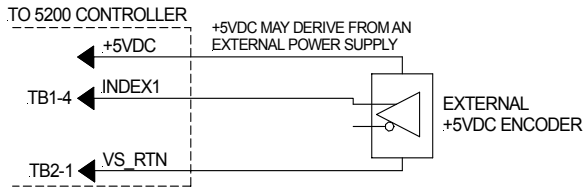
Typical Command Output Application



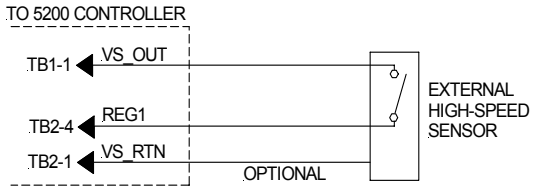
Typical Encoder Input Application



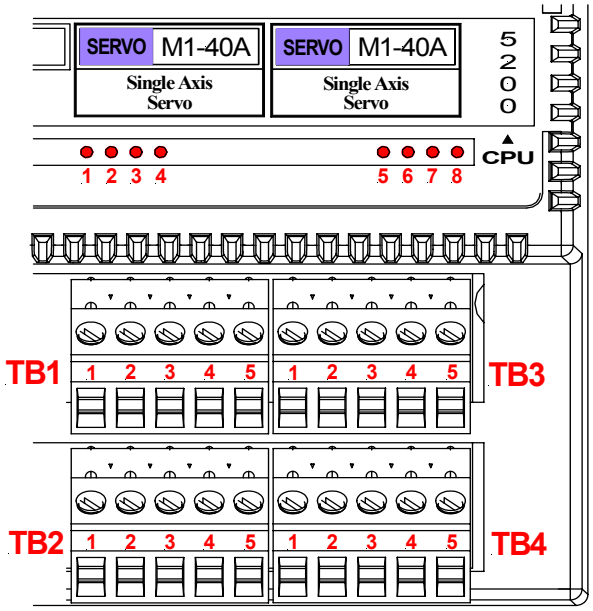
Typical Index Input Application



Typical Registration Input Application



Module Identification



I/O Terminations

Axis 1			Axis 2		
TB1-1	VS_OUT		TB3-1	VS_OUT	
TB1-2	+PHA1	LED1	TB3-2	+PHA2	LED5
TB1-3	+PHB1	LED2	TB3-3	+PHB2	LED6
TB1-4	INDEX1	LED3	TB3-4	INDEX2	LED7
TB1-5	+AOUT1		TB3-5	+AOUT	
TB2-1	VS_RTN		TB4-1	VS_RTN	
TB2-2	-PHA1		TB4-2	-PHA2	
TB2-3	-PHB1		TB4-3	-PHB2	
TB2-4	REG1	LED4	TB4-4	REG2	LED8
TB2-5	ACOM		TB4-5	ACOM	

Notes

Shields must be terminated on the controller side of the cable.
External power supply commons must be tied to the controller's supply voltage return (VS_RET) and/or analog common (ACOM).
VS refers to the voltage supply of the 5200 controller.

Care should be taken when connecting multiple Servo drives to ensure that there are no potential ground loops. Some drives may not have any isolation between their encoder signals and the analog input. Connecting to this type of drive will connect the isolated analog output of the M1-40A to the system ground and could provide a path for noise and generate unwanted transients on the Analog command.

When multiple drives with a common Analog and Encoder ground are connected, the common system grounding point must be at the 5200 system, since the encoder signals are already connected to the 5200 system 24V Return within the M1-40A module.

Using Servo Filters

A servo filter is a high speed calculation that continuously updates the servo system's output (the Analog output in the case of the M1-40A module) to control the servo motor. The 5200 offers a variety of filters that perform this function. The filter you choose depends on the type of servo drive used in your application and on the desired characteristics of the application.

The servo filter uses the difference between the actual value of the servo parameter (in the case of the M1-40A module, this is the motor position) and the intended value. This is called the servo error and for position, it is measured in Encoder Counts. The filter uses the current value and some previous values of the servo error and the filter coefficients (also known as the gains) to calculate a new value for the output. For the M1-40A module, this calculation is updated every 500 µs. Before the M1-40A module can be used to move a motor, the Servo filter type and gains must be specified for each axis using the PROFILE SERVO command.

PID Filter

The 5200'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 the requested torque (corresponding to 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. The M1-40A module 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 5200'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 5200'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 5200 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 5200 into direct mode for applications where a servo loop is not desired but you wish to command a velocity output (when the servo drive input is a velocity signal) or a torque output (where the servo drive input is the current). Set a register 14501, the Velocity-Feedforward register, to a value between 0 and 32767 to command a 0 to 10 VDC output.

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.

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

Stepper Motors

A stepper motor is generally an open loop system where the motor has a number of stable positions within each revolution. Depending on the type of motor and drive used, this can range from 50 to several hundred positions (steps) per revolution for a full or half step motor and several thousand steps per revolution for a micro-stepping system (which is a more sophisticated driver). The controller generates a series of pulses to the driver to control the motor position, but because the system is open loop, there is generally no feedback to the controller as to the actual motor position (although it is possible to add such feedback). Because of this, the motor needs to be large enough to overcome the largest possible load and is generally run at this power continuously, requiring larger motor/drive combinations than an equivalent servo system, but at a significantly reduced cost.

The M1-50A module is a dual axis Stepper motor controller combined with an 8 input Digital Input card. It generates that can be input to a Stepper Drive that will control the position of a stepper motor.

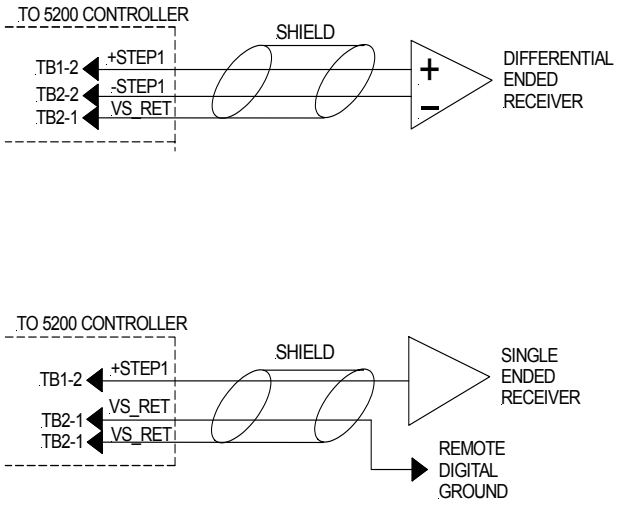
Step and Direction Output

The M1-50A module generates independent Step and Direction output signals for each axis. These are high speed digital signals at relatively low voltages for industrial control (to ensure compatibility with Stepper Drivers). To ensure signal integrity the outputs are differential drivers, but the wiring between the controller and Stepper driver should be kept as short as possible and use shielded twisted pairs to reduce the possibility of noise from nearby high power switching circuits causing false motion commands.

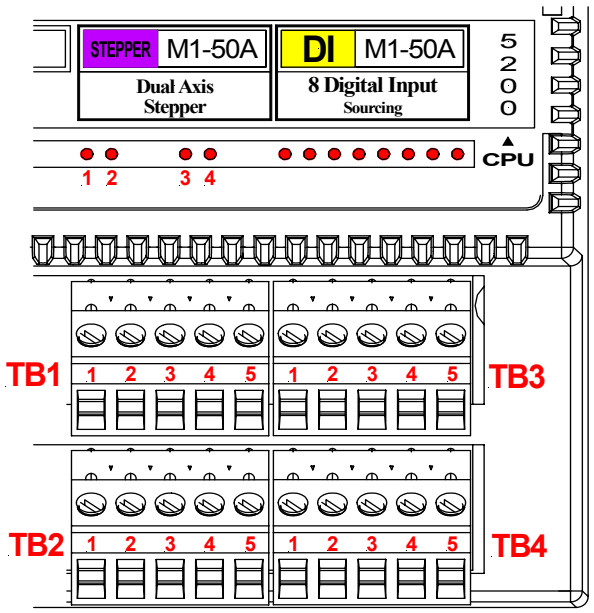
Connections: M1-50A Dual Axis Stepper Module

Application Information

Typical Output Application



Module Identification



I/O Terminations

TB1-1		VS_OUT
TB1-2	LED1	+STEP1
TB1-3	LED2	+DIR1
TB1-4	LED3	+STEP2
TB1-5	LED4	+DIR2
TB2-1		VS_RTN
TB2-2		-STEP1
TB2-3		-DIR1
TB2-4		-STEP2
TB2-5		-DIR2

Notes

If an output is used to drive transistor loads, proper current limiting must be observed.
When a digital device is powered via an external power source, it may be necessary to tie the ground of this power source
to the controller's voltage supply return (VS_RTN)
VS refers to the voltage supply of the 5200 controller.

Quickstep Motion Programming

To control motion from a Quickstep program there are several dedicated Quickstep commands and multiple special purpose registers.

Quickstep Commands

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

- `Profile Servo` – This configures a Servo or Stepper axis.
- `Monitor Servo` – This is used to determine when the motor is busy.
- `Turn Servo` – This starts a motor moving. There are several variations on this command for different types and directions of motion.
- `Zero Servo` – This sets the current motor position to zero.
- `Search and Zero Servo` – This starts a sequence of moves during which the motor will search for an input defining the zero position and will stop based on the input with a high accuracy and repeatability.
- `Stop Servo` – This will stop the current motion.

In addition, the Quickstep commands that manipulate registers (such as `Store` and tests like `Monitor` register and `If`) can be used to control motion.

The M1-50A Stepper module is programmed as if it were a M1-40A Servo module with the only difference that the M1-50A module does not need to have PID parameters specified in the `PROFILE SERVO` command. This enables the M1-50A module to take advantage of Quickstep's expanded servo instruction set.

M1-50A motor axes are defined as servos in the Quickstep Symbol Browser.

Servo and Motion Registers

The registers in the range 14000 to 17499 are reserved for use by Servo controllers, although not all these registers are currently defined or implemented. The registers used by the Servo and Stepper controllers are listed in Appendix A.

Blank

CHAPTER 2

Motion Controller Setup

The Motion Control Modules must be configured before the 5200 can use them to turn a motor. The most important setting is the Motor Operating Parameters, which is mandatory. Other settings are used to take advantage of some of the additional features of the M1-40A and M1-50A Modules.

Setting the Motor Operating Parameters

The 5200 must provide a set of operating parameters using the PROFILE SERVO instruction for each motor before it can be used. Until the information has been provided, the motor is in the Un-initialized state (special register 15003/14301 is 0), and sending any command to the motor will result in a Software Fault. The stepper 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 normal deceleration rate is the same as the acceleration rate. The deceleration rate may be set using special register 15006 as described below.
- **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. The P parameter is not required for a M1-50A module and is ignored if it is specified.
- **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. The I parameter is not required for a M1-50A module and is ignored if it is specified.

- **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. The D parameter is not required for a M1-50A module and is ignored if it is specified.

- **Holding Mode —** This parameter must be specified, but has no effect on operation for the M1-50A module. For the M1-40A module, this parameter specifies the operation of the servo when stopped. It can be set to one of the following values:

  – *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 M1-40A module will generate an output signal and attempt to correct the perceived error.

  – *Deadband of x 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. Must be specified but has no effect on stepper operation.

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 motor to take on new settings during a motor motion, is possible. To re-profile the motor, 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 motor 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 motor is stopped or is turning at maximum speed. You can view the status of the motor by checking the appropriate special registers. For example, check register number 15003/14301 for the current status of the first motor.

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 axis_1 servo at position maxspeed=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.

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

Dedicated and Assigned Inputs

There are some inputs that are used in many motion control applications, and some motion controllers have dedicated input signals that can be used for these functions. However, very few applications use all these inputs and some systems may need to have the same input signal connected to multiple inputs so the M1-40A and M1-50A modules were designed to be able to use the standard inputs of the 5200 controller for their dedicated inputs.

The M1-40A and M1-50A modules allow the Quickstep program to assign any of the 5200 controller's first 16 general purpose inputs to a dedicated input. The same signal may be used by multiple axes and by the Quickstep program without affecting the operation. If the application does not need a particular dedicated input then there is no need to assign a signal to it and no input channels are used.

For the M1-40A module, the Index (or registration) and the Z-axis input signals do have dedicated inputs because they require a significantly faster response and need to be configured both as a High speed signal and wired directly to the M1-40A module to provide the necessary resolution.

You cannot use the digital inputs on the stepper module as assigned inputs unless they are among the first 16 digital inputs in the 5200 controller.

You can assign the following functions to any of the 5200 controller's first 16 general purpose inputs:

- **Start Input:** Begins motion from a turn "on start" Quickstep instruction.
- **Reset Input:** Uninitializes the motor axis and sets the command output to 0V.
- **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 (CW) move.
- **Reverse Limit:** Prevents motion in the reverse (-) direction; hard stops motion if activated during a reverse (CCW) move.

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

Table 1. **Motor 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

Group access register 14701 or individual access register 15007 can return a bit pattern that indicates if any of the dedicated inputs are active. A binary representation of the dedicated input number is stored in the register. Each input has a unique binary value; refer to the *Model 5200 Quick Reference Register Guide (Document #951-520006)* for more information on the data in the register.

If Register 14701 is found to have a value of 18, then that indicates that home and reverse limit inputs are active. Register 14701 returns a value of 18 because the respective weights of the inputs are 2 and 16.

To test any individual input, use the bit wise AND instruction 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 default active state of the inputs is On (i.e. when the LED associated with the input is illuminated), and the appropriate bit in register 14701/15007 will be 1 when the input is On. If no input has been assigned then the value of the bit will remain zero.

The Index and Z-axis inputs are always assigned to the physical inputs on the M1-40A module and the information in the register will always reflect the current state of the input.

To reverse the polarity on selected inputs, add the values for the appropriate inputs together and store to register 17002. 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.

Status Information

Position

The motor position is available in register 14001 (alternate in 15000). This position is set to zero when the M1-40A or M1-50A card is initialized and when either a ZERO SERVO or a SEARCH AND ZERO SERVO command is executed.

For the M1-40A module, the position is the current shaft position as determined by the encoder signals and will change if the motor shaft is turned even if the axis has not yet been configured. The desired motor position can be calculated from the position and the error information (Register 14101/15001).

For the M1-50A module, the position is the commanded position and, since the controller is open loop, the error remains zero and the position will only change when the axis is commanded to move.

Status

The motor status is available in register 14301/15003 and should be checked before issuing any motion commands. The Quickstep commands MONITOR SERVO READY and MONITOR SERVO BUSY check if this register is 1 or anything except 1 respectively. Other values in this register indicate whether the axis is accelerating, decelerating/stopping, etc. Refer to the *Model 5200 Quick Reference Register Guide (Document #951-520006)* for more information on meanings of the values.

Since the MONITOR SERVO BUSY command checks for any value except 1, an axis that becomes un-initialized (either from the Reset input or because of a problem with the motion such as position error exceeded) will continue to appear busy. The Quickstep program should check for this condition and respond appropriately.

Software Limits

As well as the inputs limiting motion (Forward and Reverse Limit inputs) the motion control module has some internal limits on some operational parameters.

Position Limits

The system software position limits are in registers 17008 (Maximum allowed position) and 17009 (Minimum allowed position). These limits are active when the Maximum position is larger than the Minimum position. Since the default values for these registers are zero, the software position limits are disabled.

If the software position limits are configured and enabled, then when the motor position becomes larger than the Maximum Position while the motor is moving Clockwise, or becomes smaller than the Minimum Position when the motor is moving Counterclockwise, then the controller will execute an automatic SOFT STOP command.

In addition, when the motor position is outside the software limits, the controller will not start any motion that will move further outside the limits, although it will accept a command that brings the position back within the limits.

Speed Limit

The PROFILE command is used to specify a maximum speed for each motion. Register 17007 can be used to set the largest axis speed allowed for all motion on that axis. This can be useful in applications where the speed is limited externally (e.g. a long lead screw may vibrate when operated too fast) and where the motion maximum speed is calculated for each move.

If the maximum speed in the PROFILE instruction is larger than the Speed Limit value in register 17007, then the controller will respond as if the maximum speed value was equal to the Speed Limit value.

The default value of this register is equal to the maximum speed accepted by the controller.

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) then 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 30000, but it can be set to any positive integer value.

 *Since this uses the error value, which is always zero for a M1-50A module, this parameter is only useful on M1-40A modules.*

Gain Scaling

With some Motor-Drive combinations the standard range of gains in the controller may not be enough. In particular, with high performance drives it is possible that the closed loop gain (which includes the drive gain) is so large that the only way that the system can be stable is with very small gains (less than 10, some less than 5). Under these circumstances it is difficult to tune the system optimally since, if a gain is, say, 5 then the smallest change possible is a change of 20%, which may be more than is needed.

A similar problem may be found when the loop gain is very low (often a sign of an undersized motor/drive combination) and in this situation even a gain of 255 (the maximum allowed value) may not be enough to get acceptable performance.

To deal with these situations, there is a Gain Scaling register available for each Servo axis, Register 16169 (for Servo Channel 1) is used as a gain scaling constant when executing the `PROFILE` command.

The default value of this register is 8, it can be set to between 2 and 14 and increasing the value makes the gains more sensitive and decreasing it will make the gains less sensitive. Increasing the register value by 1 will double the scaling when converting from the gains specified in the `PROFILE` command to internal controller gains. This means that to keep the same performance the gains specified in the `PROFILE` command can be divided by 2, allowing lower values to be used.

Similarly, when the gains are found to be very small, the value of the gain Scaling register can be decreased by 1, which halves the scaling when converting from the gains specified in the `PROFILE` command to internal controller gains. This means that to keep the same performance the gains specified in the `PROFILE` command can be multiplied by 2, allowing higher values to be used and making them less sensitive to small changes in gains and therefore easier to tune.

Since the `PROFILE` command uses the gain scaling register to set the internal values of the gains when the P, I and D parameters are specified, any changes to the Gain Scaling Register will not affect operation until a new `PROFILE` command is executed. The Quickstep program should make sure that any changes to the Gain Scaling Register are made before a new `PROFILE` command is executed.

Blank

CHAPTER 3

Starting and Stopping Motion

Starting Motion

There are three modes of turning the motor:

1. **Absolute Positioning** — In this mode, the motion control module 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 axis_1 to 50000
   ```

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

2. **Relative Positioning** — In this mode, the direction of the turn (clockwise or counterclockwise) is specified in the turn instruction along with a defined number of steps to turn. For example, the following instruction

   ```
   turn axis_1 cw 12340 steps
   ```

   will turn the motor 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 axis_1 cw
   ```

   starts the motor turning clockwise at its current maximum speed and acceleration. The motor continues to turn until the 5200 issues a STOP SERVO instruction or until a Limit or Stop input is activated.

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

The motion control module tracks the position of the motor with all three modes and allows you to use all three types of positioning and control in the same program.

 *Quickstep instructions specifying clockwise or counterclockwise operation assume that the motor is wired according to the manufacturer's recommendations and that the logical sense of the direction output of the motion control module agrees with the logical sense expected by the motor's drive.*

Stopping the Motor

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

- STOP (SOFT) SERVO causes the motor to stop at the deceleration rate specified in the last profile instruction.
- STOP (HARD) SERVO causes the 5200's stepper board to try to stop the motor instantly. However, because of the momentum, the motor may not stop instantly.

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

Searching for Home

Each motor 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 motor stops and the position is set to zero.

The M1-40A and module also supports a highly accurate method of finding the home position. In addition to providing direct support for a two-stage homing routine, the M1-40A module also makes use of the Z-axis signal available on many encoders to further increase the consistency of the home position. There is an additional input for encoders that provide the additional output for each axis on the M1-40A module to accept the Z axis signal.

Specifying the Home Direction and Operation

The default initial direction for the Home operation is in a counterclockwise direction and the motion control module uses both the Home input and the Z-axis signal to determine the home position. You can reverse the direction of the homing motions described below and instruct the module to ignore either the Home input or the Z-axis signal by changing the value of the special purpose register 17003. You can restore the default setting mentioned by storing 0 or –1 to the register.

The acceptable range of values that may be stored in register 17003 and the effect of these on the Homing operation are defined in the *Model 5200 Quick Reference Register Guide (Document #951-520006)*. For more information on these and other special registers, refer to this document.

If there is no Home input assigned (register 15162 is zero) then the system operates as if the setting in register 17003 has disabled the Home input.

Home Operation

The homing sequence is as follows:

When the 5200 system executes a SEARCH AND ZERO SERVO instruction, the motor begins turning in a counterclockwise direction at the acceleration rate and maximum speed specified in the most recent PROFILE instruction. The motion control module will monitor the Home input signal. When the home input becomes active (based on the setting of register 17002 as above), the motor stops at the profiled deceleration rate.

If the Home input is active when the SEARCH AND ZERO SERVO instruction is executed, then the above step is skipped.

The motor then automatically begins searching in the opposite direction (default clockwise) at a fixed speed of 950 steps per second. When the home input turns on again, the speed decreases to 192 steps per second. If the home input is still on when the motor stops before reversing direction, then the initial speed will be 192 steps per second.

When the home input opens (turns off), the motor will be stopped with no ramp (hard stop). If the Z-axis input is disabled, then the Home operation is complete and the current motor position is set to zero.

If there is no Home input or the Home Input has been disabled as described above, then the steps above are skipped.

If the Z-axis input is enabled, the motor will once again reverse direction and move counterclockwise (default) until the Z-axis input becomes active, at which point the motor will be stopped, again with no ramp (hard stop) and the current motor position will be set to zero, completing the Home operation.

Blank

CHAPTER 4

Setting Up Electronic Following

The 5200 can perform a function called electronic following, or ratioing. In electronic following, one servo axis is commanded to match its motions to a leader or encoder based on a specific ratio. You can create this ratioing function with two special registers and two simple Quickstep instructions. You can even adjust the ratios on the fly within your Quickstep program.

Dual Servo Axis-to-Axis Following

You can configure the 5200 for axis-to-axis following (Figure 2–2). The first axis is always the follower and the second axis is always the leader. There are two types of following modes:

- **Trajectory Following** - This mode is the default mode. It creates a ratio mode that is based on the leader servo's theoretical (calculated) position, not its actual position. This mode causes the follower axis to be in phase with the leader axis, which results in a closer match that is based on the defined ratio.

- **Encoder Following** - This mode causes the follower to use the leader's encoder information to perform its ratio.

Configuring Electronic Following

Configure an axis for following by setting two special purpose registers for the follower axis as if you were defining a fraction. The first register specifies the numerator and represents the follower axis. The second register specifies the denominator and represents the leader axis.

You must decide how to describe the fraction since both a 1/1 fraction and a 10,000/10,000 fraction define a 1:1 ratio. However, to achieve better resolution in your application, you may want to use more decimal places in the fraction. For example, defining a fraction of 9,978/10,000 causes the follower to be geared slightly lower than the leader.

Special register 16005 specifies the numerator and register 16006 specifies the denominator. You should store the denominator value before the numerator value because storing the numerator activates electronic following.

You must define a complete PROFILE SERVO instruction with working tuning parameters for the follower axis before 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 it is engaged, the follower's status register (register 14301 or 15003) contains the number 10, which indicates that it is following its leader.

When you activate the follower axis and you are encoder following, the servo board automatically resets the leader's position to zero. You cannot reset the leader's position with a Quickstep instruction.

Notes

1. *The maximum range of 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 servo board automatically accumulates and adjusts for fractional remainders to maintain synchronization between the follower and the leader.*

Ending Electronic Following

You can disengage the axis from following the leader by storing a 0 to the numerator. This causes the axis to decelerate to a stop at the profiled deceleration. You can also 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 and later comes to a stop.

Reading Leader Position and Velocity

Special register 16007 specifies the current leader position and special register 16008 specifies the current leader velocity registers. Leader position is expressed in encoder pulses and leader velocity is expressed as encoder pulse-per-second. These registers are read-only. You can use them to monitor real-time leader activities from within your Quickstep program.

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

Specifying Encoder Following

Store the number 128 plus the servo filter type code to the servo filter register (17001) to select the encoder following mode within axis-to-axis following. For example, the command `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.

```
[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 10000 to Denominator_r16006
store 5000 to Numerator_r16005
[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 10000 to Denominator_r16006
store 1000 to Numerator_r16005
```

Blank

CHAPTER 5

Configuring Servo Registration

The M1-40A servo controller 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 that requires automatic synchronization of a product from one cycle to another, the registration input and the special registers provide a method of recording real-time position information. Both axes have the ability to record the absolute position of the servo when the registration input is activated.

The M1-40A motion control module also has the ability to change the current servo motion and adjust the end position of the move for reliable synchronization. Registration on the 5200 is processed in with dedicated hardware latches, so the servo's absolute position is captured with a resolution of ±1 encoder count (step) regardless of the servo's velocity.

Configuring and Monitoring Registration

Registration is controlled by a set of registers associated with each axis. The registration is enabled by storing a value of 0 to the Registration Status register (16004 for axis 1). When the M1-40A module has detected a Registration input it will set the value of this register to 1, at which point the data in the Registration Position register (16002) is valid, and is set to the position at which the input was detected. The Registration Position is only valid while the Registration Status is 1.

To reset the registration latch, a 0 must be stored to this register. To initialize this latch for the first time, a 0 must be stored after the window has been set up, even though the value in this register may be read as a zero. The act of initially writing to this register arms the registration latch.

The value stored in the Registration Offset register (16003) is used by the M1-40A module to change the motion that is in progress when the Registration Input is detected. If this value is 0 then no move modification will occur. However, the position of the registration input will still be stored in the Registration Position register. This is considered Registration measurement.

If there is a non zero value stored in this register, then the M1-40A Module will set the destination point of the current motion to be this distance (in steps) from the registration position. The distance is measured in the current direction of motion, so a positive

Registration Offset should be used regardless of the direction of motion. Altering an existing move is considered Registration with move modification.

Care should be taken when using Registration with move modification that the timing of the Registration input within the move does not cause the M1-40A module to generate an unwanted motion to achieve the modification. In general the Registration Offset should be larger than the distance required to decelerate from the maximum speed at the programmed Deceleration rate, and the Registration Input should not occur during the acceleration or deceleration ramps.

Designating a Predefined Registration Window

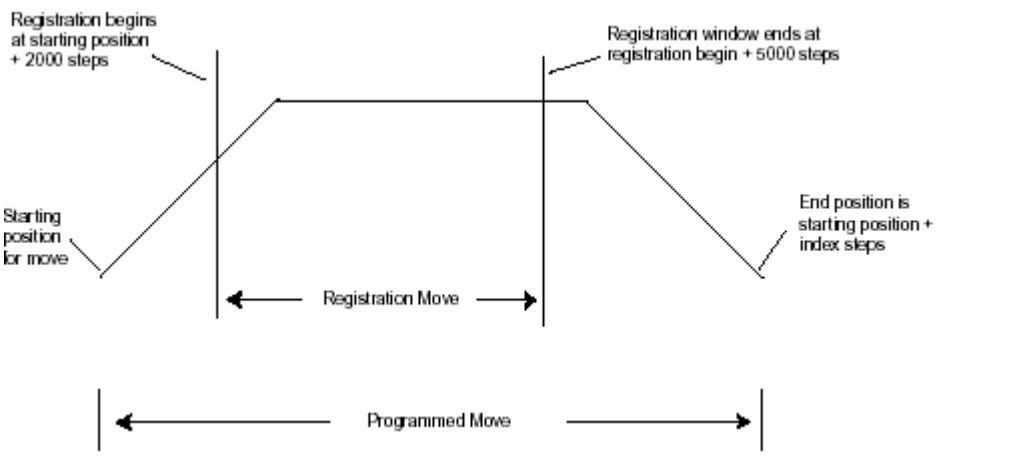
The registration input is usually connected to some type of electronic sensor or photo eye. To make use of the registration feature, you need to define the window where the servo expects the electronic sensor to activate the registration input.

A predefined registration window (Figure 5.1) tells the servo where to look in its move for the sensor's input. The servo's position is only captured if the sensor triggered the registration input in this window. The servo module does not record the servo's position if the sensor triggers the registration input outside of this window. This prevents other events from triggering registration.

You define the size and range of the registration window with the special registers set up for this purpose. In the following example, the servo is programmed to move a specific distance (labeled Index). The registration window for servo 1 is defined as 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, enter the following values in registers 16000 and 16001:

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

Figure 5.1, Predefined Registration Window



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

If the registration window is set to zero then the module will always treat the registration input as valid (i.e., the window will cover the entire possible position range).

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 servo control module allows you to program an offset position that you can add to the captured registration position. It uses this new position to redefine the stopping point for the current motion and overwrites the original programmed destination.

This allows for precise correction to your motion based on when registration was sensed.

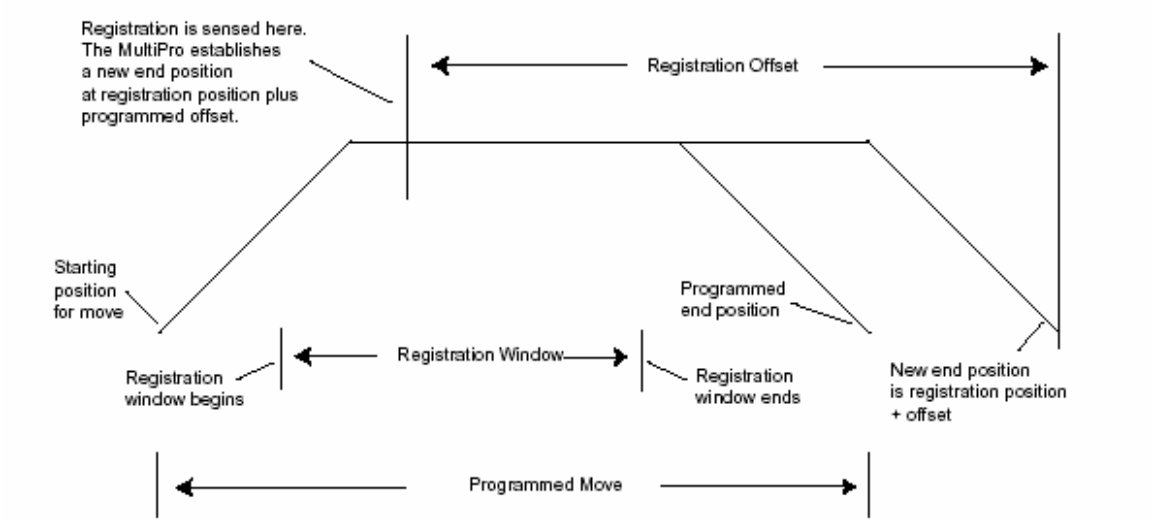
```
[1] REGISTRATION_EXAMPLE
;;;_____________________________________
   <NO CHANGE IN DIGITAL OUTPUTS>
store 5000 to Reg_16001
goto Next
[2] REGISTRATION_MOVE
;;;
;;;_____________________________________
   <NO CHANGE IN DIGITAL OUTPUTS>
profile Servo_1 servo at position maxspeed=Reg_501 accel=
Reg_502 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] GOTO_REGISTRATION
;;;_____________________________________
   <NO CHANGE IN DIGITAL OUTPUTS>
store Reg_16002 to Reg_10
```

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 (Figure 5.2).

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

- Register 16003 is set to 7500, which means that the new end position for the move is 7500 steps after the servo senses the registration input.

Figure 5.2. Changing the End Position of a Move



When the sensor is triggered during the registration window, the servo control module operates as follows:

- Records the absolute position of the servo in register 16002 (for axis 1).
- Calculates 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.
- 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 servo control module 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 re-arms 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:

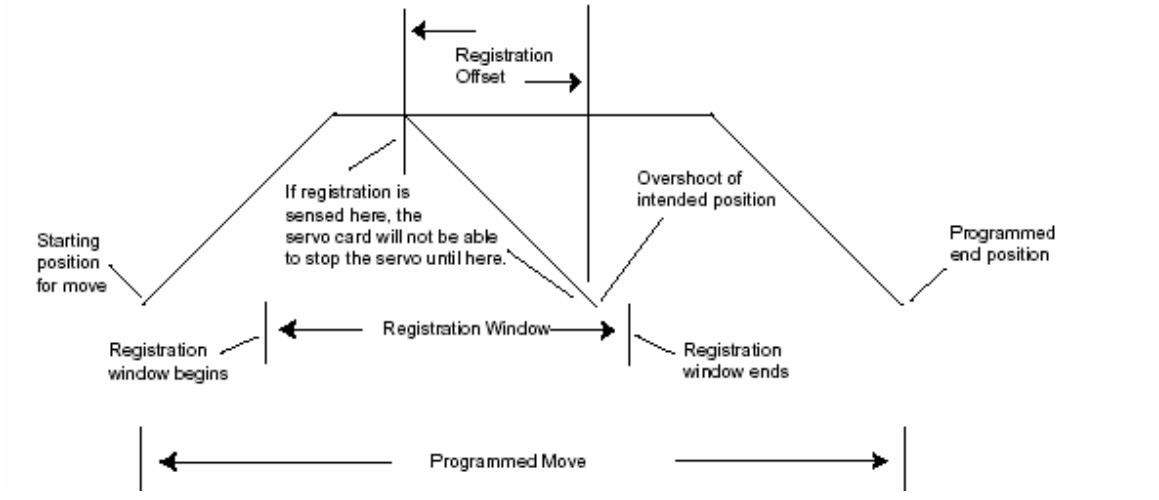
Overshooting the End Position

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. Figure 5.3 shows a case where the servo is unable to decelerate in time to stop at the new end position. Avoid this problem by using one of the following methods:

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

```
[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
```

Figure 5.3, Overshooting the End Position



Registration during Deceleration

There are cases where the 5200 senses registration during the deceleration portion of a programmed move. In this situation the motion module will attempt to stop the motion at the desired position, i.e. at the registration position plus the registration offset. However, depending on the motor speed at the time the registration is detected, this will not generally be possible while keeping the programmed deceleration rate.

Since most applications will have a registration offset that is larger than the distance required to decelerate, this means that the motor will generally have to travel further than it would have if the registration had not occurred. The controller will calculate a trajectory that stops at the desired target position in the normal deceleration time (i.e. the time needed to decelerate from the maximum move speed to a stop at the specified deceleration). This trajectory follows a parabolic velocity-time path, starting at the registration position and velocity and ending at the target position with zero velocity and the programmed deceleration. This may mean however that the motor will increase speed during this time (although it should not exceed the maximum move speed).

Registration Measurement

The M1-40A motion module can also be used to make a measurement when it becomes necessary to read the distance between two points. These two points need to use the same reference in order to establish a difference. Since the registration input uses the feedback encoder for its associated axis as the reference for its captured position, the M1-40A module provides a second input using that same reference.

Each axis has an additional input normally used for an encoder identified as the Index input. This input is designed to be connected to the *mark* or *index* or *Z channel* on an encoder to capture a second position based on the marker pulse. This 5 volt input can be manipulated for use as a second registration latch. Since most proximity switches or potential sensors used in triggering the registration input are 24 volt devices, it is important **NOT** to connect them directly to this Index input. Installing a 4.7K Ohm resistor in series with this input limits the current enough to safely allow the 24 volt sensor to be connected.

As for the registration input discussed previously, the M1-40A module provides special registers associated with this index input. Since the Index input cannot modify any motion, there is no Offset register and there are also no window control registers, so the Index input will be processed whenever the Index Status register (16163) is set to 0. Like the registration status, this register is 0 when the 5200 is reset but a 0 must be stored to enable the Index input. The position at which the Index input is detected is stored in the Index Position register and is only valid when the Index Status is 1.

Using Registration for Measurement

The following program can be used to measure a distance using both Index and Registration inputs:

```
[1] one
<TURN OFF ALL DIGITAL OUTPUTS>
store 0 to Reg_Window_Begin_R16000
store 0 to Reg_Window_Open_R16001
store 0 to Reg_Window_Offset_R16003
store 0 to Reg_Status_R16004
store 0 to Reg_Index_Status_R16164
goto Next

[2] two
 <NO CHANGE IN DIGITAL OUTPUTS>
profile First_Axis servo at position
maxspeed=Velocity_R501 accel=Accel_R502 P=P_Gain_R503
I=I_Gain_R504 D=D_Gain_R505
zero First_Axis
turn First_Axis cw
if Reg_Status_R16004=1 goto Triggered

[3] Triggered
 <NO CHANGE IN DIGITAL OUTPUTS>
 if Reg_Status_R16164=1 goto Measure

[4] Measure
 <NO CHANGE IN DIGITAL OUTPUTS>
store  Reg_Index_Position_R16162 - Reg_Position_R16002 to
      Registration_Length_R125
stop soft servo First_Axis
goto Next

[5] Finished
 <NO CHANGE IN DIGITAL OUTPUTS>
 done
```

The registration window is disabled (registration always enabled, i.e., “wide open”) by storing a 0 to Window_Open. The offset is defined as 0 so a move modification will not occur but simply a position will be captured. The first registration latch is armed by storing a 0 to register 16004. The second position latch is armed by storing a 0 to register 16164. In step #2 the servo axis is profiled and its position is set to zero. Notice a move is commanded to run CW at just a velocity. At the same time the move commences the step waits for the registration status to change from a 0 to a 1 on the first latch. This will indicate a capture has occurred. Next, the program waits for the registration status to change from a 0 to a 1 on the second latch. When this occurs, a math operation subtracts the second latched position from the first latched position and stores it into a general purpose register to be used later. The servo is commanded to stop and the measurement task is done.

Guidelines and Rules for Setting up Registration

The following rules and guidelines make it easier to program your 5200 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 counterclockwise direction. Failure to take the direction into account results in your servo becoming uninitialized at the point where registration is triggered.

- Inhibit the registration offset function by using a STORE instruction to set it to zero.

- The servo control module senses registration when the state of your sensor changes to the active state.

CHAPTER 6

Sample Quickstep Programs

This section contains sample Quickstep programs for different motion applications.

Maxspeed units are in steps/s. Acceleration units are in steps/s2.

Example 1 — Absolute Move of One Motor

This example shows a motor moving 100,000 steps from its home position. The `monitor axis_1:stopped` instruction causes the 5200's program to remain in this step until the motor completes the move.

```
[2]   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 axis_1 servo at position maxspeed=50000 accel=100000
      turn axis_1 to 100000
      monitor axis_1:stopped goto next
```

Example 2 — Relative Move of One Motor

This example shows a motor moving clockwise 100,000 steps from its current position. The `monitor axis_1:stopped` instruction causes the 5200's program to remain in this step until the motor completes the move.

```
[1]   ONE_AXIS-RELATIVE_MOVE
      ;;; This program will commence a relative move on axis
      ;;; one based on the parameters in the profile
      ;;; instruction.
      <NO CHANGE IN DIGITAL OUTPUTS>
      profile axis_1 servo at position maxspeed=50000 accel=100000
      turn axis_1 cw 100000 steps
      monitor axis_1:stopped goto next
```

Example 3 — Velocity Move of One Motor

This example shows a 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 axis_1:stopped` instruction causes the 5200's program to remain in this step until the motor completes the move.

```
[2]   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 axis_1 servo at position maxspeed=50000 accel=100000
      turn axis_1 cw
      monitor axis_1:stopped goto next
```

Example 4 — Changing the Velocity of a Motor During Motion

This sample program positions a 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 stepper position where the program moves to the next step.

```
[3]   COMPLEX_PROFILE
      <NO CHANGE IN DIGITAL OUTPUTS>
      profile axis_1 servo at position maxspeed=10000 accel=200000
      turn axis_1 to 500000
      if axis_1:position >= 50000 goto next
[4]   SECOND_PROFILE
      ;;; Re-profile the motor for a new velocity.
      <NO CHANGE IN DIGITAL OUTPUTS>
      profile axis_1 servo at position maxspeed=20000
      if axis_1:position >= 70000 goto next
[5]   THIRD_PROFILE
      ;;; Re-profile the motor for a new velocity.
      <NO CHANGE IN DIGITAL OUTPUTS>
      profile axis_1 servo at position maxspeed=50000
      if axis_1:position >= 110000 goto next
[6]   FOURTH_PROFILE
      ;;; Re-profile the motor for a new velocity.
      <NO CHANGE IN DIGITAL OUTPUTS>
      profile axis_1 servo at position maxspeed=100000
      if axis_1:position >= 300000 goto next
[7]   FIFTH_PROFILE
      ;;; Re-profile the motor for a new velocity.
      <NO CHANGE IN DIGITAL OUTPUTS>
      profile axis_1 servo at position maxspeed=80000
      if axis_1:position >= 420000 goto next
[8]   SIXTH_PROFILE
```

```
;;; Re-profile the motor for the final velocity and wait
;;; for the move to complete.
<NO CHANGE IN DIGITAL OUTPUTS>
profile axis_1 servo at position maxspeed=30000
monitor axis_1:stopped goto PROFILE_COMPLETE
```

Example 5 — Velocity Move of Two Motors

This example shows two 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 `axis_1:stopped axis_2:stopped`) instruction causes the 5200'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 *Virtual Inputs* for more information on using the ON START parameter and the Start dedicated input.

```
[9]   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 axis_1 servo at position maxspeed=50000 accel=100000
      profile axis_2 servo at position maxspeed=25000 accel=50000
      turn axis_1 cw
      turn axis_2 cw
      monitor (and axis_1:stopped axis_2:stopped)
      goto next
```

Example 6 — Absolute Move of Two Motors

This example shows two 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 axis_1:stopped axis_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 *Virtual Inputs* for more information on using the ON START parameter and the Start dedicated input.

```
[10]  TWO_AXIS_ABSOLUTE_MODE
```

```
;;; 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 axis_1 servo at position maxspeed=50000 accel=100000
profile axis_2 servo at position maxspeed=25000 accel=50000
turn axis_1 to 100000
turn axis_2 to 50000
monitor (and axis_1:stopped axis_2:stopped) goto next
```

Example 7 — Staggering the Motion of Two Motors

This example shows two stepper 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 axis_1:stoppped axis_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 be 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.

```
[11]  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 axis_1 servo at position maxspeed=50000 accel=100000
      profile axis_2 servo at position maxspeed=25000 accel=50000
      turn axis_1 to 100000
      if axis_1:position >= 50000 goto next
[12]  TRIGGER_SECOND_AXIS
      ;;; Turn the second axis and wait for both axes to be
      ;;; complete before moving on to the next part of the
      ;;; program.
      <NO CHANGE IN DIGITAL OUTPUTS>
      turn axis_2 to 25000
      monitor (and axis_1:stopped axis_2:stopped) goto next
```

Example 8 — Ratio Axis to Leader Encoder

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

```
[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 servo
;;; board, 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 maxspeed=reg_501
accel=reg_502 P=reg_503 I=reg_504 D=reg_505
store 10000 to reg_16006
store 5000 to reg_16005
goto next
[2] LEADER_MOVE_FORWARD
;;; This step moves servo_2 (the leader) clockwise.
;;; The follower (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
```

Blank

Appendix A. Motion Register Summary

Motion Registers Grouped by function then axis

	The 5200 firmware is designed to access up to 16 axes. For the 14XXX register values below substitute the axis number for '**ax**' to get the correct register. **Axis #1 = 1**. For example, the position of axis #1 is stored in register 14001.
140ax	**Position** (counts), R only
141ax	**Error** (counts), R only
142ax	**Velocity** (counts / sec), R only
143ax	**Status**, R only:
144ax	**Integral Error** (count-seconds), R only
145ax	**Velocity Feedforward**
146ax	**Deceleration** (counts/sec^2)
147ax	**Dedicated Inputs**, R only:
148ax	**Acceleration Feedforward**
149ax	**Analog Output**, R only -32,767 = -10.000V; 32,767 = 10.000V;

Status	Description
0	Axis not initialized
1	Stopped and ready
2	Motion imminent: waiting for start,
3	Accelerating
4	At max speed.
5	Decelerating to new max speed
6	Decelerating to stop
7	Soft stop
8	Registration move
9	Home
10	Following
128-255	Error

This is a bit map of the input signals

Bit Number	Description	Bit Number	Description
0 (lsb)	Z-axis	**4**	Rev EOT
1	Home	**5**	Fwd EOT
2	Start	**6**	Index
3	Kill	**7**	Not Used

Motion Registers Grouped by axis then function

	For the 15xxx, 16xxx, 17xxx register values below, substitute the axis number for '**bx**' to get the correct register. Axis **#1 = 0**. For example the position of axis #1 is stored in register 15000.
15bx0	**Position** (counts), R only
15bx1	**Error** (counts), R only
15bx2	**Velocity** (counts / sec), R only
15bx3	**Status**, R only:

Value	Description	Value	Description
0	Axis not initialized	**6**	Decelerating to stop
1	Home	**7**	Soft stop
2	Motion imminent: waiting for start	**8**	Registration move
3	Accelerating	**9**	Home
4	At MAX speed	**10**	Following
5	Decelerating to new MAX speed	**128-255**	Error

15bx4	**Integral Error** (count-seconds), R only
15bx5	**Velocity Feedforward, also used to specify the Output in Direct mode**
15bx6	**Deceleration** (counts/sec^2), R only
15bx7	**Dedicated Inputs**, R only**:** **This is a bit map of the input signals**

Bit Number	Description	Bit Number	Description
0 (lsb)	Z-axis	**4**	Rev EOT
1	Home	**5**	Fwd EOT
2	Start	**6**	Index
3	Kill	**7**	Not Used

15bx8	**Acceleration Feedforward**
15bx9	**Analog Output**, R only -32,767 = -10.000V; 32,767 = 10.000V;
16bx0	**Reg. Start, R/W** – Position at which the registration will be enabled
16bx1	**Reg. Window, R/W** – The range that the registration will be enabled
16bx2	**Reg. Position**, R only – The position at which the registration was detected, when Reg status is 1
16bx3	**Reg. Offset, R/W** – This distance to be moved after the registration input
16bx4	**Reg. Status** – 0 = Armed, 1 = Detected, can only set to 0
16bx5	**Numerator**, R/W **–** For following the master axis
16bx6	**Denominator, R/W –** For following the master axis
16bx7	**Leader Position**, R only – Only valid when following a master axis
16bx8	**Leader Velocity**, R only – Only valid when following a master axis
16bx9	Reserved
17bx0	**Firmware Revision**, R only

<table>
<tr><td></td><td></td></tr>
<tr><td>17bx1</td><td>Filter & Mode, R/W:
In Direct mode the Feedforward Velocity gain specifies the output value (0 to 32767) with a value of 32767 = 10V (sign depends on the Filter type).
<table>
<tr><th>Description</th><th>Value</th></tr>
<tr><td>Lower 3 bits (0x07)
Filter type

Bits 4 & 5
Accel/Decel Type
Bit 7 (0x80)</td><td>0 or 3 = PID
1 = + Direct (CW) 2 = - Direct (CCW)
4 = PAVff 5 = PAV
0 = Linear 1 = S Curve
2 = Parabolic 3 = Inverse Parabolic
0=Trajectory Following
1 (value 128) = Encoder Following</td></tr>
</table></td></tr>
<tr><td>17bx2</td><td>Input Polarity, R/W:
This is a bit map that controls the active level of the input signals, when the bit is 0 then the input is active when it is On, if the bit is 0 then the input is active Off
<table>
<tr><th>Bit Number</th><th>Description</th><th>Bit Number</th><th>Description</th></tr>
<tr><td>0 (lsb)</td><td>Z-axis</td><td>4</td><td>Rev EOT</td></tr>
<tr><td>1</td><td>Home</td><td>5</td><td>Fwd EOT</td></tr>
<tr><td>2</td><td>Start</td><td>6</td><td>Index</td></tr>
<tr><td>3</td><td>Kill</td><td>7</td><td>Not Used</td></tr>
</table></td></tr>
<tr><td>17bx3</td><td>Home Direction, R/W
<table>
<tr><th>Direction</th><th>Description</th></tr>
<tr><td>CCW

CW</td><td>0 or –1 = Home & Index
-2 = Home Only
-3 = Index Only
1 = Home & Index
2 = Home Only
3 = Index Only</td></tr>
</table></td></tr>
<tr><td>17bx4</td><td>Options, R only</td></tr>
<tr><td>17bx5</td><td>Reserved</td></tr>
<tr><td>17bx6</td><td>Maximum Following Error, R/W</td></tr>
<tr><td>17bx7</td><td>Speed Limit, R/W – overrides maximum velocity, default = 4194303 steps/sec</td></tr>
<tr><td>17bx8</td><td>Maximum Position, R/W – Used as a Software EOT when it is larger than the Minimum Position</td></tr>
<tr><td>17bx9</td><td>Minimum Position, R/W – Used as a Software EOT when it is smaller than the Maximum Position</td></tr>
</table>

Extended Motion Registers Grouped by axis then function

	For the extended motion registers below the (15xxx, 16xxx, 17xxx) substitute the axis number for '**cx**' to get the correct register. In this series **Axis #1 =16**. For example, the Start Input of axis #1 is stored in register 15160.
15cx0	**Start Input, R/W – System input used as Start Input, 0 = disabled, 1-16 specifies input number**
15cx1	**Shutdown Input, R/W – System input used as Shutdown Input, 0 = disabled, 1-16 specifies input number**
15cx2	**Home Input, R/W – System input used as Home Input, 0 = disabled, 1-16 specifies input number**
15cx3	**Positive (CW) EOT Input, R/W – System input used as CW EOT Input, 0 = disabled, 1-16 specifies input number**
15cx4	**Negative (CCW) EOT Input, R/W – System input used as CCW EOT Input, 0 = disabled, 1-16 specifies input number**
15cx5	--------- Reserved ----------
15cx6	--------- Reserved ----------
15cx7	--------- Reserved ----------
15cx8	--------- Reserved ----------
15cx9	**Master Axis Select**
16cx0	--------- Reserved ----------
16cx1	--------- Reserved ----------
16cx2	**Index Position, R only – Latched when using Index input (5V level signal) as a second Registration input**
16cx3	--------- Reserved ----------
16cx4	**Index Status, can only write 0 – When using Index input (5V level signal) as a second Registration input**
16cx5	--------- Reserved ----------
16cx6	--------- Reserved ----------
16cx7	--------- Reserved ----------
16cx8	--------- Reserved ----------
16cx9	**Gain Scaling, default = 8** Used when converting PID gains from the PROFILE command to the internal values. Increasing by 1 allows the PROFILE command values to be divided by 2 without changing the actual performance. Suggested when the values used in the command are large (near 255) and need to be increased to get acceptable performance. Decreasing by 1 allows the PROFILE command values to be doubled without affecting performance. Suggested when the values used in the PROFILE command are small and don't allow fine enough control to get acceptable performance.
17cx0	--------- Reserved ----------
17cx1	--------- Reserved ----------
17cx2	--------- Reserved ----------
17cx3	--------- Reserved ----------
17cx4	--------- Reserved ----------
17cx5	--------- Reserved ----------
17cx6	--------- Reserved ----------
17cx7	--------- Reserved ----------
17cx8	**Module Serial Number**
17cx9	**Module Monitor Version**