Chapter 17 Motion

Introduction

This chapter covers the topic of motion control through the PLC to stepper and servo motor devices. It is not exhaustive in the sense that all motion subjects will be discussed but rather that the student will be given experiences with two of the more popular single axis control concepts, stepper and servo control.

Stepper motors are used traditionally for low torque applications with no feedback. The servo on the other hand can handle higher torque applications and requires a feedback device.

Mechanical Conversions and Moving a Device

A review of mechanical devices follows with some basic formulas for conversion of energy into rotating or linear motion, in this case, controlled motion. Both applications in this chapter involve motion projects that are considered controlled motion. The device does not need to coordinate with any other axis but must produce a controlled motion to specification. In general, motors do not provide this level of control. That is, dc motors and ac motors do not provide acceleration and constant speed control at a designated level repetitively. They do not hold a position at zero speed unless there is no torque on the motor shaft.

In general, motors have the following characteristics:

For rotating objects:

$$HP = \frac{T * N}{5252}$$

Eq. 17.1

where: T = Torque (lbft)
N = Speed (rpm)

For objects in linear motion:

$$HP = \frac{F * V}{33000}$$

Eq. 17.2

where: F = Force (lbs)
V = Velocity (ft/min)

Objects such as pumps, fans, blowers and conveyors require HP ratings for running based on the physical characteristics of the device. General formulas for torque and the relationship between torque and horse power are given in the following:

Torque Formulas:

$$T = \frac{HP * 5252}{N}$$

Eq. 17.3

where

T = Torque(LbFt)
HP = Horsepower
N = Speed (rpm)

$$T = F * R$$

Eq. 17.4

where

T = Torque(LbFt)
F = Force(Lbs)
R = Radius(Ft)

$$Ta(accelerating) = \frac{WK * \text{Change in RPM}}{308 * t(sec)}$$

Eq. 17.5

where

Ta = Torque(LbFt)
WK = Inertia at Motor Shaft (LbFt)
t = Time to Accelerate (sec)

Torque also has been defined in vector notation. If a force F is not perpendicular to the rotating arm, the component perpendicular produces torque.

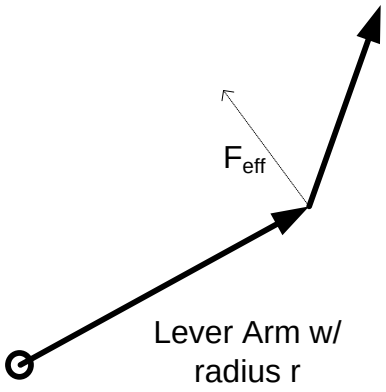


Fig. 17.1

A lever arm rotates with a force F is applied. The torque $T = r \times F$ has magnitude $T = |r|\,|F_\perp| = |r|\,|F| \sin\theta$ with direction out of the page.

The cross product is an example of the right-hand rule in which the fingers curl in the direction of the force from the lever arm. Then the thumb points in the direction of the torque. This can be expressed in terms of:

$$T = rF\sin\theta \quad \text{or} \quad T = rF_{\perp} \qquad \text{Eq. 17.6}$$

Torque is also related to angular momentum *L* via the following equation:

$$T = \frac{dL}{dt} \qquad \text{Eq. 17.7}$$

where L is angular momentum and t is time.

Also, for angular ration about a fixed axis:

$$L = I\omega \qquad \text{Eq. 17.8}$$

where I is moment of inertia and ω is the angular velocity.

Also derived:

$$T = \frac{dL}{dt} = \frac{d(I\omega)}{dt} = I\frac{d(\omega)}{dt} = I\alpha \qquad \text{Eq. 17.9}$$

The term α is the angular acceleration of the body with units rad/s^2.

These formulas are not inclusive and are not to be committed to memory or used except after checking with a manufacturer to verify the accuracy of the specific formula with their equipment. Formulas developed from these basic formulas exist to size the motor for an application. Torque is especially valuable in sizing the servo or stepper motor applications. While progressing through the following applications, remember that the proper sizing of the motor and motor controller are an integral part of the overall process of configuring the motion application. Depend on a particular manufacturer's data sheets when applying the manufacturer's equipment. They have done much more than the novice to correctly specify a motor or motor drive for an application (or at least hopefully so).

Motion Control Products

There is a decision to be made when specifying motion products. A servo-drive, stepper motor or a variable frequency AC drive are the choice for most motion applications. There is little reason to consider DC motors and their motor control.

Regarding precision, a servo drive is usually better than an AC drive. A servo drive can control position or speed under extreme conditions and be very precise. The servo drive requires higher performance from the electric motor than the AC drive. Servo drives and servo motors are inter-dependent so insist that they be supplied from the same manufacturer. On the other hand, if an AC drive is the better choice, it is easier to select the motor and controller from different suppliers. In most applications, the AC drive is less expensive than the comparable servo drive. Speed control and torque control have historically been the chief advantages of dc motors and drives. However, newer technologies have allowed ac motors to provide these functions and ac motors do not have brushes that periodically need maintenance. Application and cost, therefore, become the deciding factors when choosing between AC control or servo/stepper control.

The stepper drive may be used instead of servo control for low torque applications. If a position is to be held at zero speed, servos or steppers are the logical choice since torque drops to zero for AC induction motors at zero speed.

The servo motor is specialized for high-response, high-precision positioning. As a motor capable of accurate rotation angle and speed control, it can be used for a variety of types of equipment.

Closed Loop Control

A rotation detector (encoder) is mounted on the motor and feeds the rotation position/speed of the motor shaft back to the driver. The driver calculates the error of the pulse signal or analog voltage (position command/speed command) from the controller and the feedback signal (current position/speed) and controls the motor rotation so the error becomes zero. The closed loop control method is achieved with a driver, motor and encoder, so the motor can carry out highly accurate positioning operations. A PLC or other controller may generate a pulse (PTO) or analog signal to specify movement of the servo or stepper. Pictured below are a PLC with pulse output and a servo controller. The next figure shows the internal design of the drive. The pulse input or analog input is used to provide velocity or position control for the drive..

The controller inputs the pulse signal. The speed and stop position are then controlled according to the pulse number.

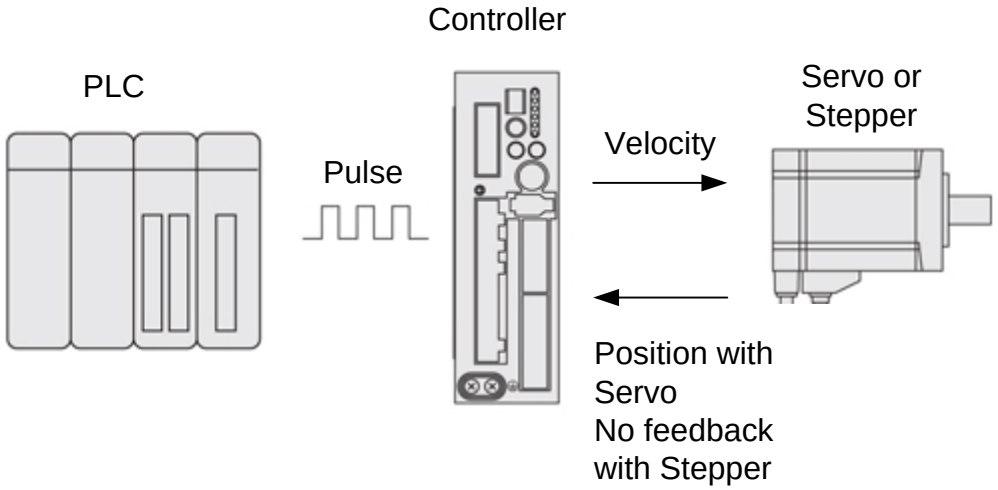


Fig. 17-2

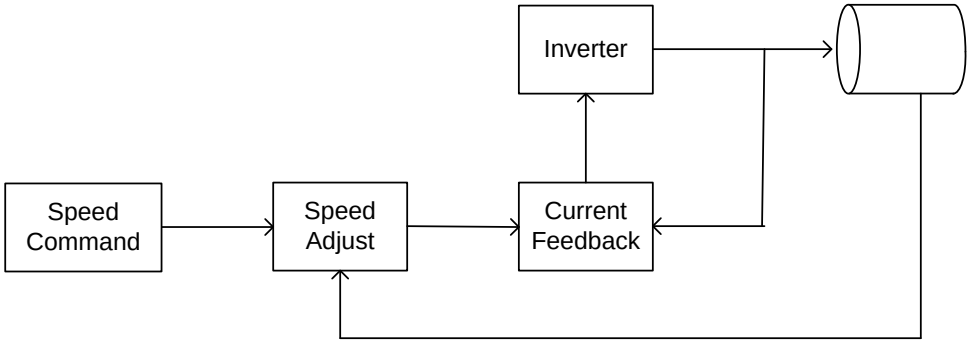


Fig. 17-3

Positioning

Positioning is one of the most frequently used motion functions. It is used when moving material from point A to point B along a pre-defined track, then on to point C and so on.

Positioning can also be divided into linear and roll-over positioning. Roll-over positioning means position calculation within one revolution.

Absolute Positioning

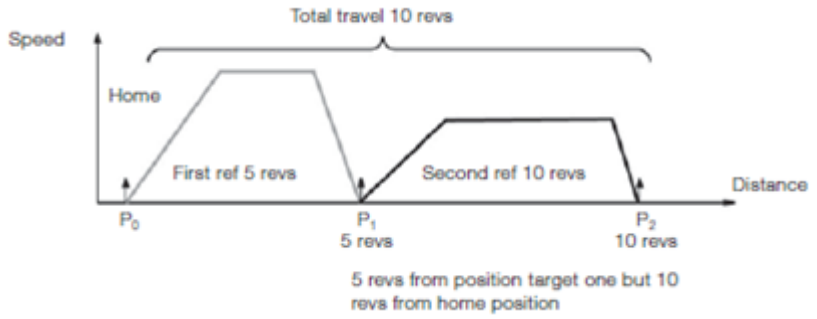


Fig. 17-4

Relative Positioning

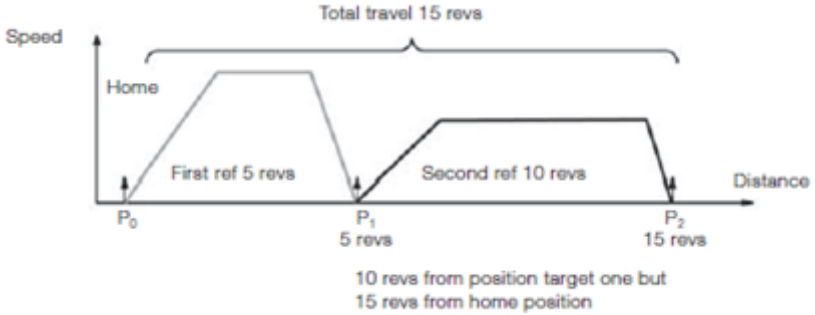


Fig. 17-5

Synchronization

Synchronization means that a follower drive reads speed and position reference from an external encoder or from the other drives. The gear ratio can normally be adjusted to suit the application. Synchronization can be absolute or relative and works with linear/rollover axes.

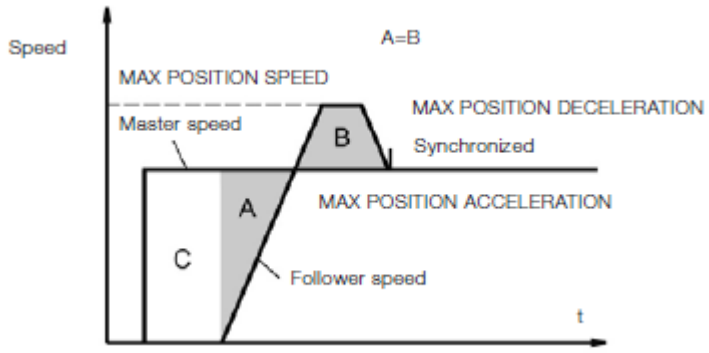


Fig. 17-6

The follower drive starts to accelerate and continues to increase the speed to catch up with the speed of the master. When areas A and B are equal, the follower has caught up.

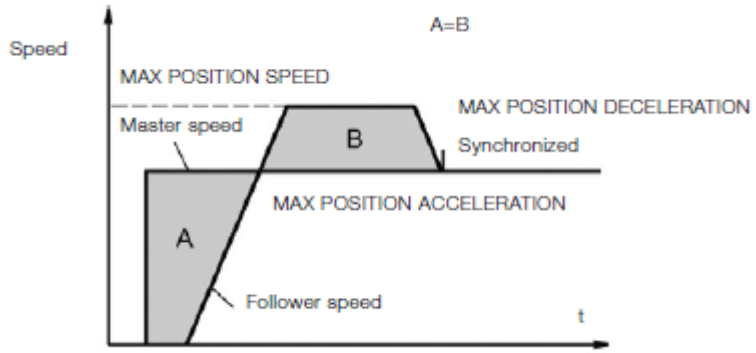


Fig. 17-7

Linear axis, absolute synchronization: In this case, the reference is the total travel distance the master drive has to complete. The follower drive will run at a higher speed for long enough to catch up with the position of the master drive.

Rollover Axis

Rollover axis mode is such that only one revolution is calculated and then calculation starts all over again.

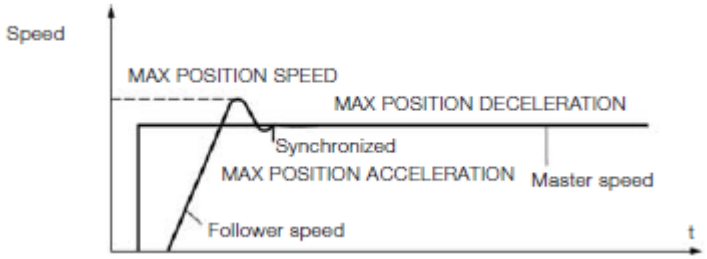


Fig. 17-8

The illustration shows how the follower drive catches up with the master drive's position.

Homing

Homing is required at startup and if position is lost due to power loss of the system. If an absolute encoder is used, the real position is known as soon as power comes back. One way around is to use an auxiliary power supply.

Whenever the system starts up, the home position must be determined. If there is only a homing limit switch, the software checks the status of this switch. I the switch is on, the load must move in a positive direction until the switch turns off. This is the position defined as home.

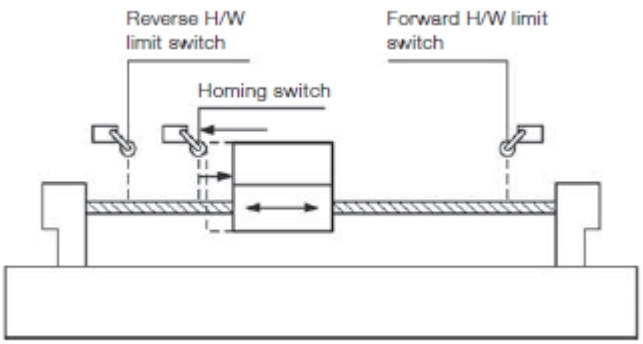


Fig. 17-9

Encoder Gear Functions

Motion control applications usually need feedback. The feedback can be connected to the motor, the load, or both.

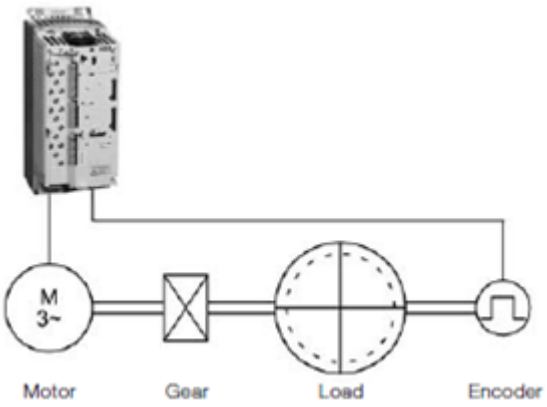


Fig. 17-10
Encoder attached to Load

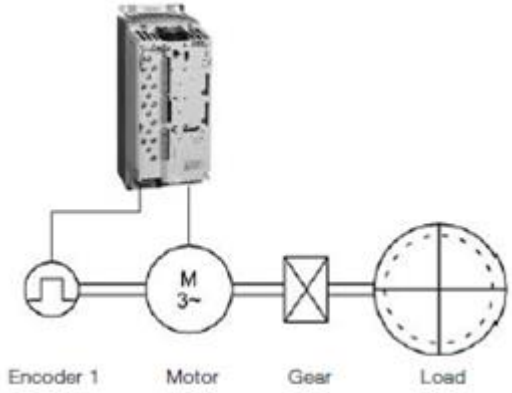


Fig. 17-11
Encoder attached to Motor

If there is no encoder on the load side, the load gear ratio has to be set up according to the gear ratio. In the following set-up, an encoder is attached to both motor and load.

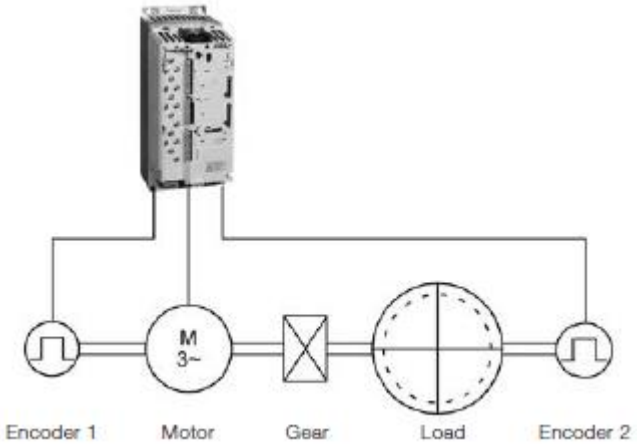


Fig. 17-12

How stepper or servo motors are used in applications is discussed next. Several of these applications require coordination between multiple axes. Some don't. As you look at the various applications in this next section, use the instructions discussed later in the chapter to program the motion of the drives in the automatic mode.

Applications: Constant Gap Maintaining

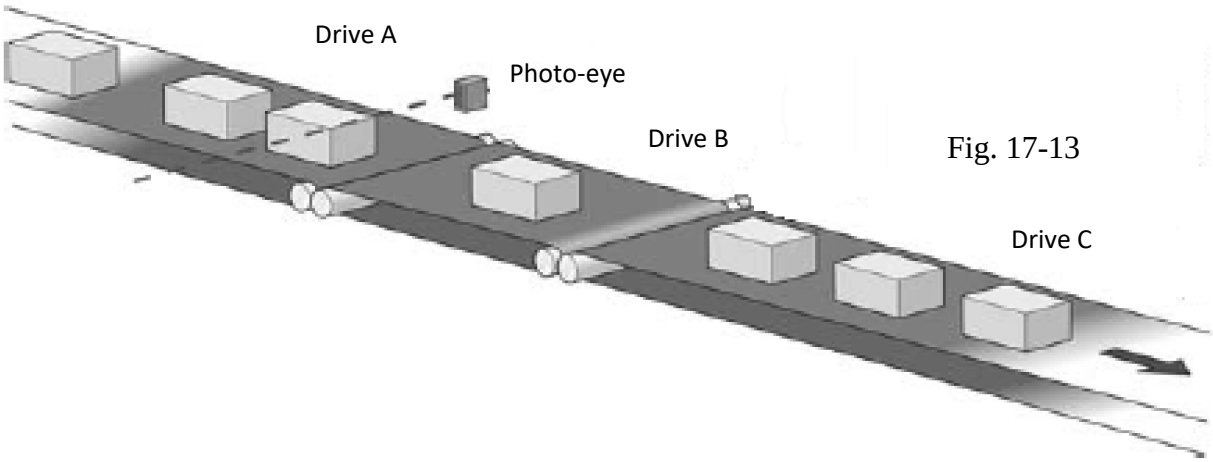


Fig. 17-13

This conveyor has a feed belt (A), an adjusting belt (B) and a receiving belt (C). The boxes arrive with random spacing. Each drive receives the line speed reference from the encoder. The sensor follows the presence of the boxes. When the sensor detects a box, it follows the top edge of the box until length of the box is determined. The dropping edge is seen by the sensor and the distance to the next rising edge is the actual gap between boxes. This is compared to the required gap and the software makes the necessary correction by altering the speed of the adjusting belt. Each conveyor can be stopped to control the spacing of the boxes. If the next box does not arrive in time to maintain spacing, Drive C may be stopped or slowed to a near-zero speed. This control provides precise spacing between boxes.

A simpler control would require a minimum spacing between boxes. In this scenario, if boxes arrive spaced greater than the desired spacing, the boxes may continue uninterrupted. The spacing may increase, just never falling below a minimum spacing.

Applications: Cut to Length

There are many methods to cut different materials to the required length. Here are some examples. In applications where the line is stopped to make the cut, both axes use the positioning feature of the drive. The drive that is fed the material first runs a determined number of revolutions corresponding to the required length of the material. When the target position has been reached, the drive signals the PLC that it is at the required position. The cutting motor runs the required number of revolutions to execute the guillotine operation. Its drive then gives the feed motor permission to again run. As in other applications, the dynamic performance requirements of the system must guide the motor selection.

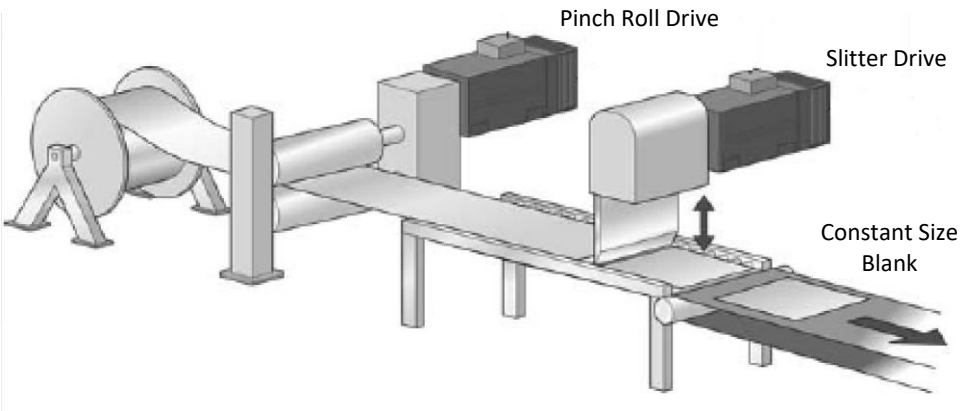


Fig. 17-14

Applications: Rotary Knife

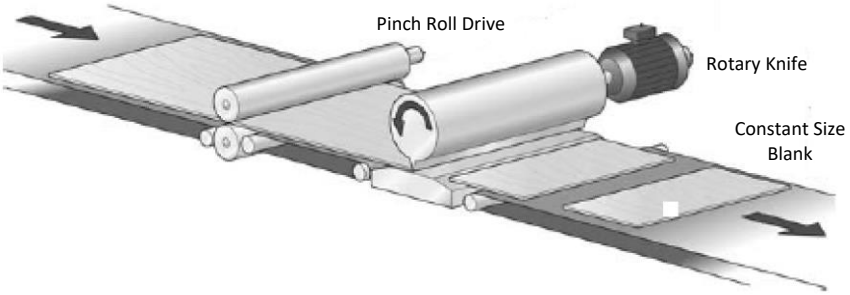


Fig. 17-15

A rotary knife is used to cut material into required length or cut off of unwanted material. The simplest rotary knives are synchronized to the line speed using an electrical gear. However, in many applications, this will not give satisfactory performance.

There are a number of considerations to take into account for rotary knife operations. First, if the cutting length varies, it must be decided whether the tool should be at standstill or move continuously. Secondly, when the tool hits the material, it will in most cases, need to have the same speed as the line. Thirdly, it is important to determine where to place the cut.

For more sophisticated applications, the knife must form a motion profile during the cycle. When the knife is at a standstill and a cut command is given, it has to accelerate to reach the position and then decelerate to cutting speed. After cutting, the tool should return to the home position as fast as possible to be ready for the next cut.

In some cases, the tool may not be able to stop but has to start another cut "on the fly". This means using two profiles that are added together. CAM profiles with flexible parameter settings are normally used in these situations.

Applications: Cyclic Correction, Packing Application

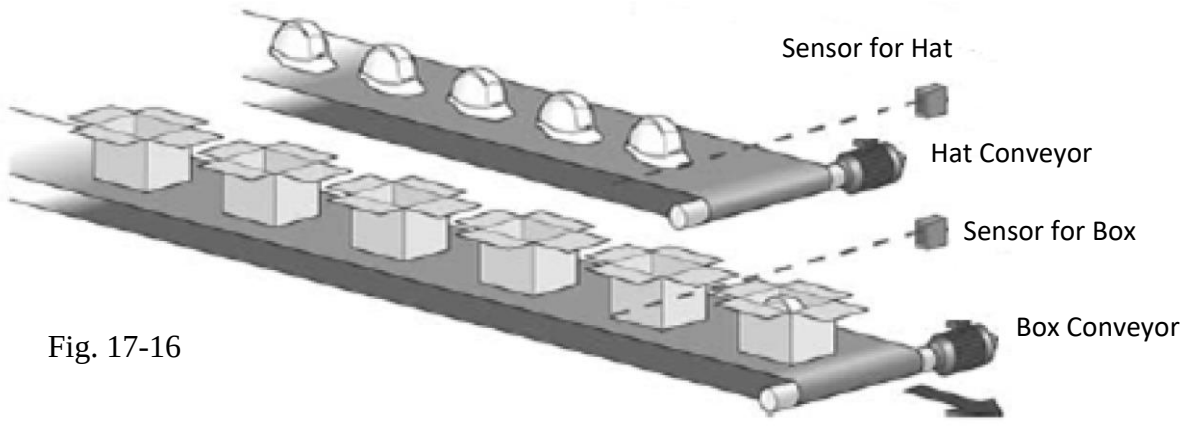


Fig. 17-16

This operation may be performed with both conveyors constantly moving or with the box conveyor stopped while the hat conveyor moves to drop the hat in the box. Boxes and hats cannot be assumed to be equidistant from each other in either case.

Applications: Flying Shear, Angled

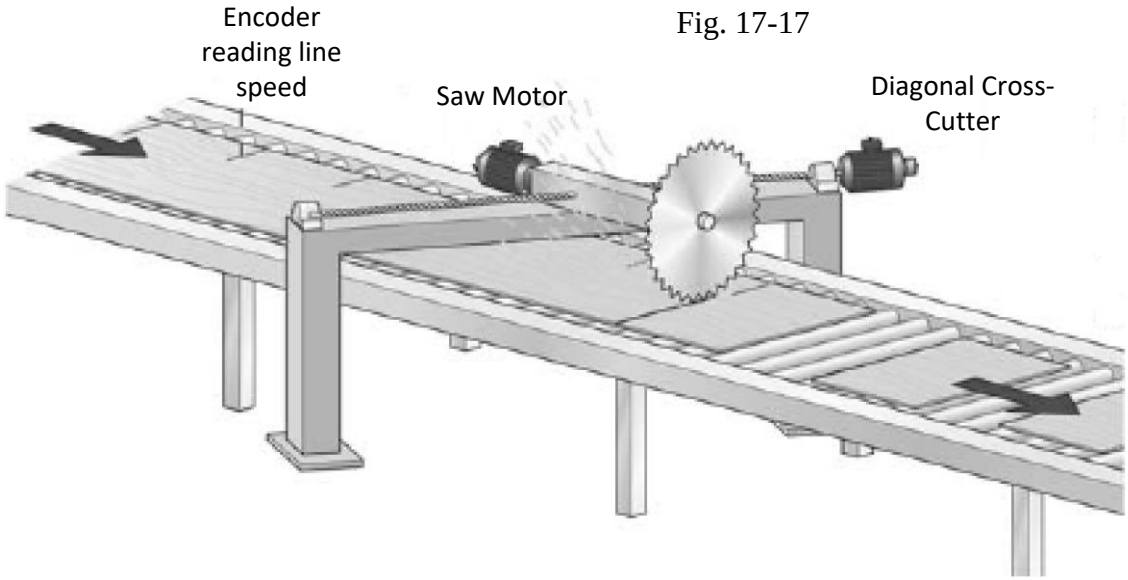


Fig. 17-17

A flying shear is a cutting machine that allows constant material flow during cutting. It is based on right-angle trigonometry. When the speed of the line and the speed of the saw are known, the angle of the cut can be calculated and adjusted accordingly. In this illustration, the angle means that the blade moves in the direction of the line when the saw operates. Saw speed control is not critical; even an ac single phase motor can be used.

The cutting point can be indicated by a mark on the material or through rotational measurement by encoder. Typically, synchronizing or CAM functions are used. This setup is often used in applications where the material must be cut by a saw rather than a knife/guillotine. A similar application is found in the diagonal cross-scoring of glass in the flat glass manufacturing process.

Applications: Plywood Machine

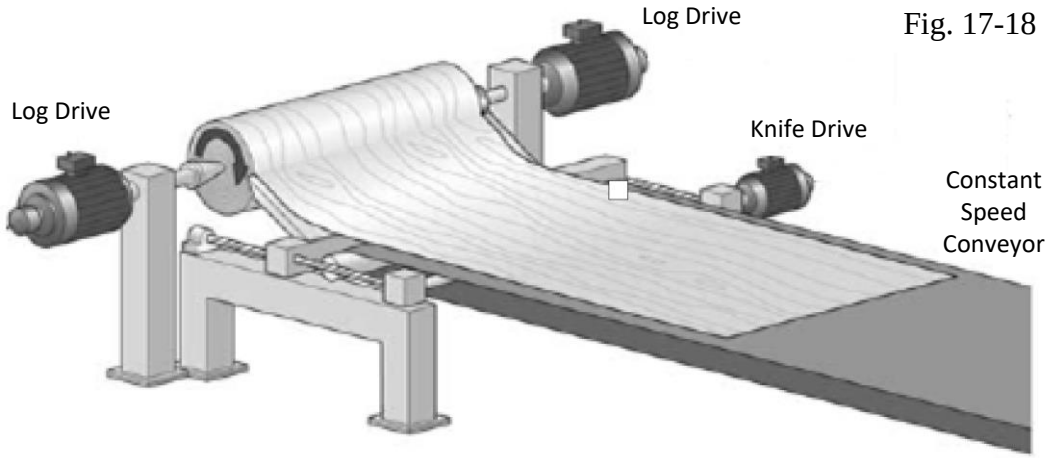


Fig. 17-18

This machine peels plywood from a log. The two log drive motors turn and the knife cuts a constant thickness of wood from the log. The speed of the conveyor is assumed to be constant. The speed of the log drives must increase in speed as the log diameter decreases. The knife speed must increase as well to keep the thickness constant.

Applications: Material Filling

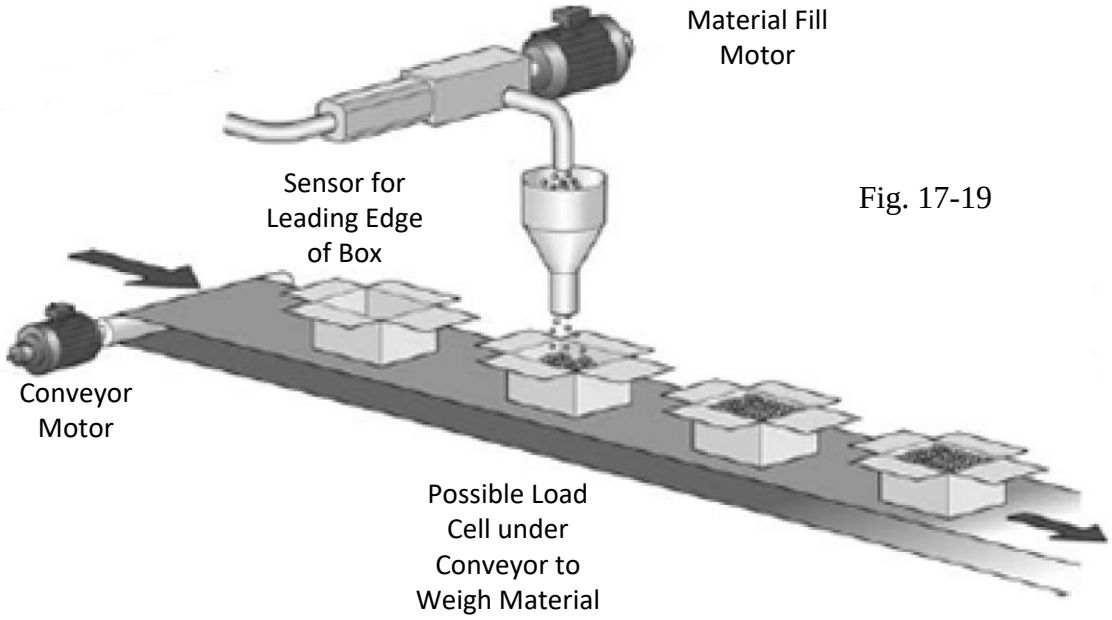


Fig. 17-19

This application is very similar to the one described above with the Hat to Box. Similar configurations are used to fill bottles, boxes, and other containers. Usually an inspection station is located downstream to validate the fill.

Applications: Slitter

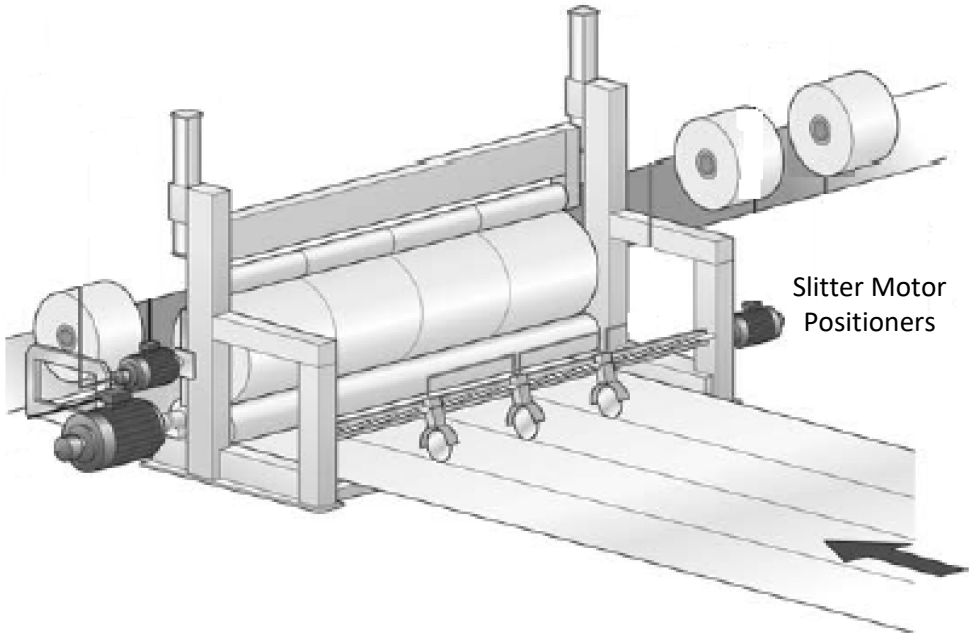


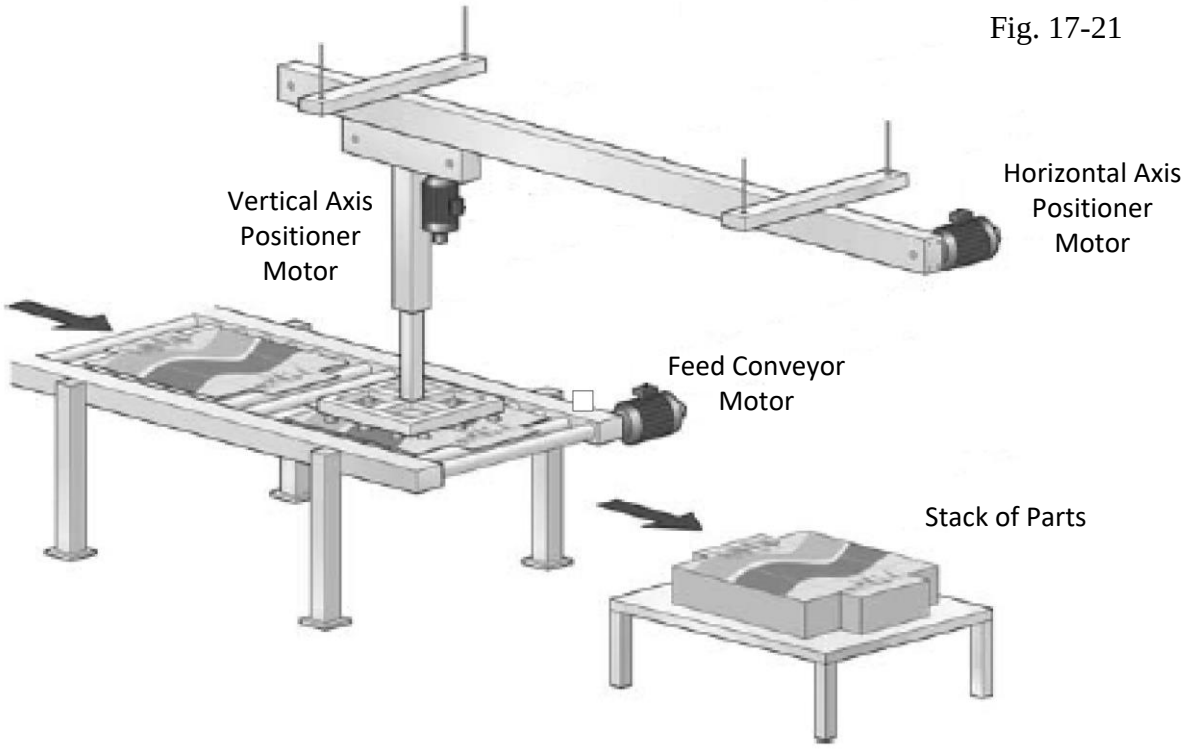
Fig. 17-20

This illustration describes the cutting and take-up of a winder machine. Each tool is individually connected to the screw. When a tool is engaged, the position of each individual slitter is set.

Applications: Picking and Stacking

This application uses distributed control in three axes. The main controller gives commands to each axis to make the material flow of the plates occur. The plates are picked up with the picking tool using position control. The plate, still in position control, is moved forward to the stacking location. Finally the plate is positioned down to build up the stack. The plates feeding conveyor can run in continuous speed or position control mode.

Fig. 17-21



This action is possible with either a robot or a frame with servo controllers as shown here. Either a robot or the servos can accomplish the same function.

Applications: Warehouse Automation

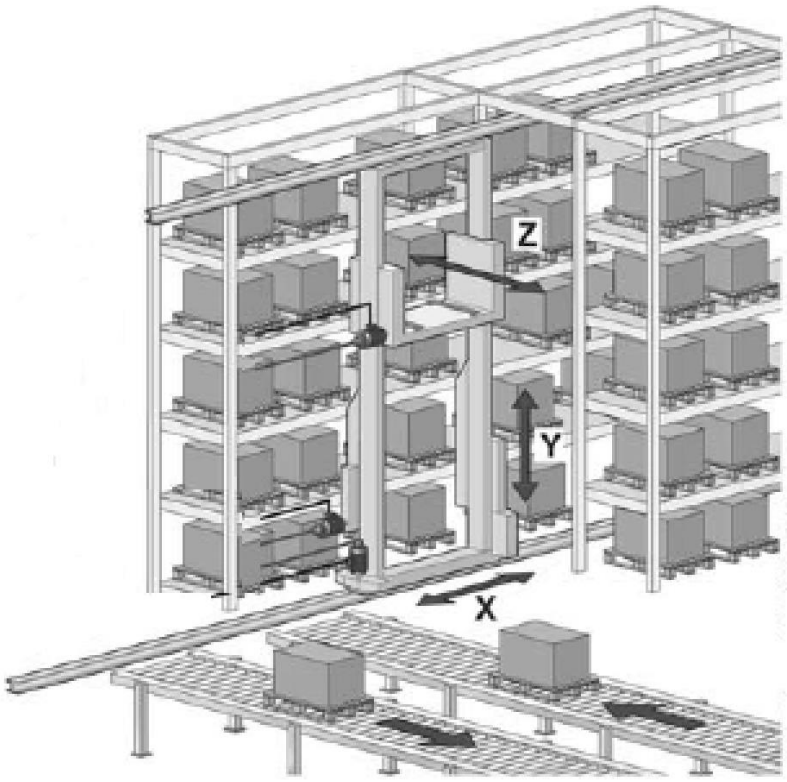
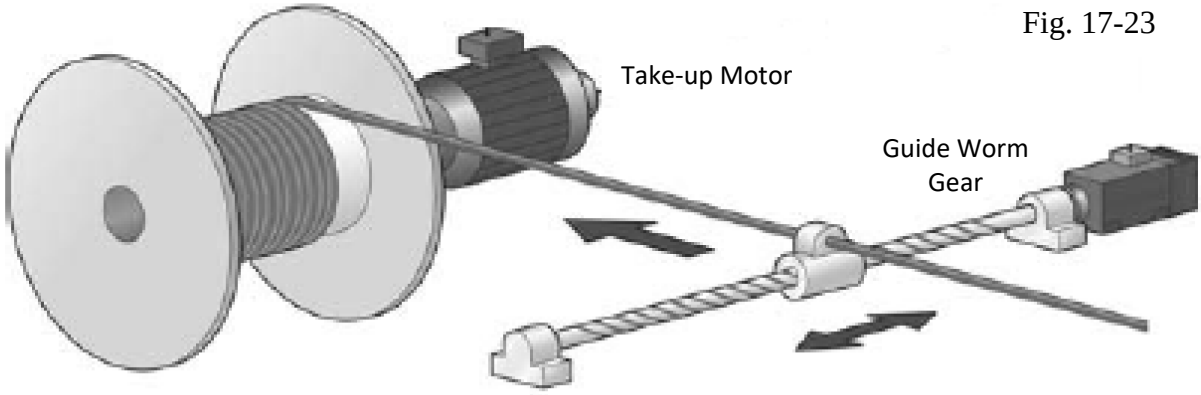


Fig. 17-22

Warehouse automation applications can be configured very cost-effectively using the PLC. The control system is part of the full factory automation system and knows where the pallets need to go.

Applications: Winding

Fig. 17-23



This application pictured here is of traverse control. Traverse control is an electronic gear function where the gear ratio is setup so that the traverse linear movement is locked to the build-up of the material. The illustration does not show the limit switches that typically control the turning point action.

Winding and unwinding are well-established applications and there are many dedicated software packages commercially available.

Applications: Wrapping

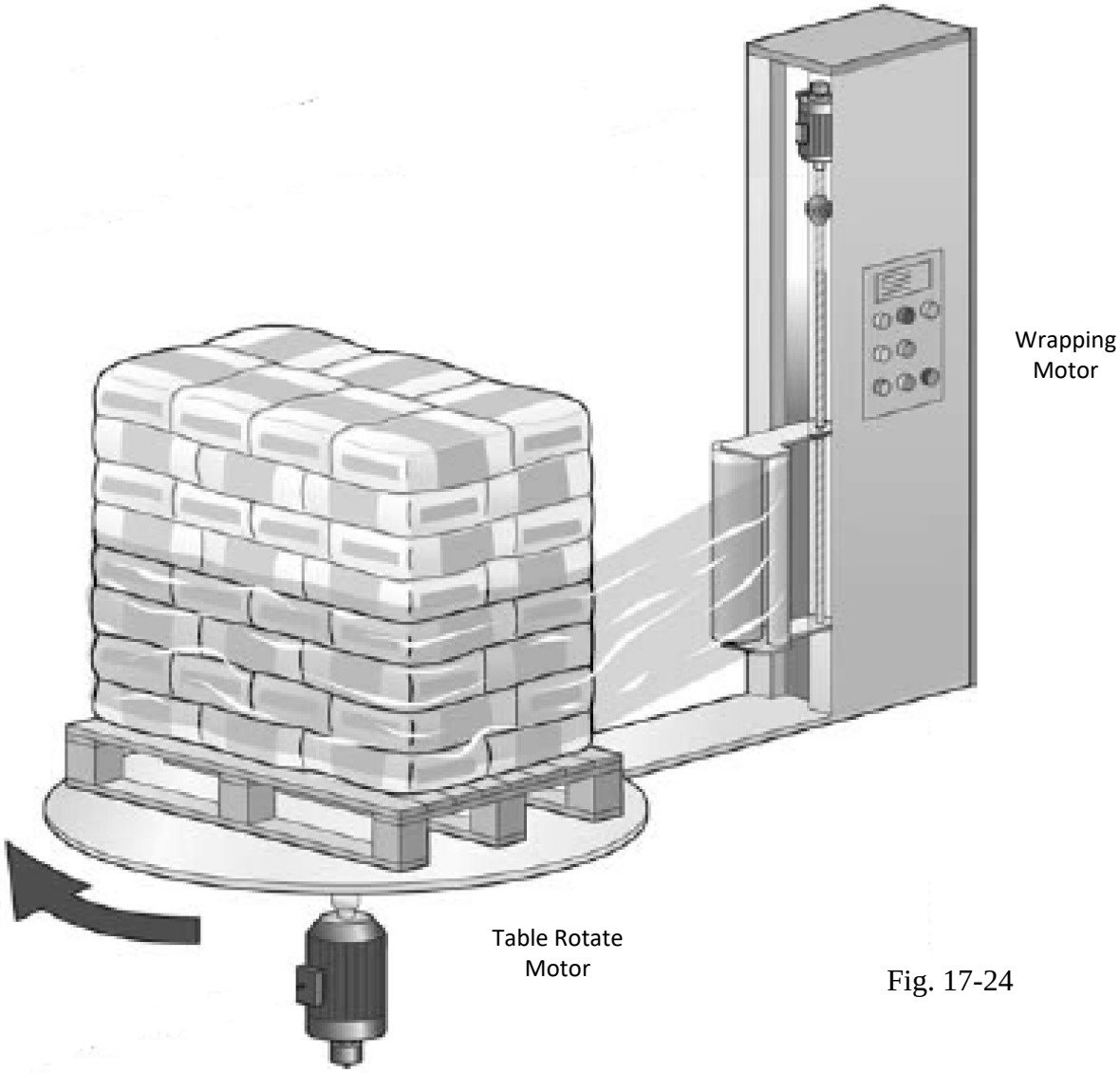


Fig. 17-24

This illustration shows a simple packaging application. The electrical gear is formed between two motors.

Applications: Grinding

Fig. 17-25

Grinder Spindle Motor Runs Constantly

Manual Grinder

The X, Y and Z Axes may be turned automatically as well as manually

This application shows a manual grinder. The same grinder could be automated by adding servo motors to the X, Y and Z axes. The grinder would then be able to remove metal in an automated process. The grinder motor itself does not necessarily need to be controlled from a servo or stepper motor. However, the axes that are used for position control must be configured with these motors and appropriate drive electronics.

Mechanical Components

Almost all mechanical motion manufacturers have a website with instructions for calculating the size of their equipment for various operations. One such company is Nook Industries of Cleveland, Ohio. The following is from one of their manuals:

Manual Lathe Application Description

Given the following requirements, select an acme screw for an application which uses precision Acme screw systems for an automatic part feeder on a machine.

Step 1: Determine Specifications

- 5,000 lb load supported and guided on linear bearings moving horizontally
- 36" travel
- Complete 36" travel in 10 seconds
- Bearing Support Undecided
- Positioning accuracy +/- ¼"

Step 2: Analysis

Find the Axial Force Required to Move the Load

The axial force is determined by multiplying the coefficient of friction of the guidance system by the load.

- $F = \mu \times N$ (μ = Coefficient of Friction of the guidance system)

Using Nook linear bearings in this application we can determine:

- μ = Coefficient of Friction for lubricated Nook Linear Bearings = .0013
- N = Load = 5000 pounds
- $F = \mu \times N$
- $F = .0013 \times 5000$ lb
- F = 6.5 lb

Therefore, we can definitively state that the axial force the screw must produce to move the load is 6.5 lb.

and so it goes…

The purpose of this example is not to convince you to buy a particular product but to show the methods used to size a simple device such as a ball screw for a lathe. While it is not expected that you as a PLC programmer will need to size a ball screw but you may and if you do, these are the places to look and the procedures to follow.

The study of motion control requires many parts and the above discussions are just a few. First is the understanding of angular or rotating mechanical motion. Definitions for horsepower, force, velocity, and torque are some of the equations needed. Then there is the requirement to size the mechanical equipment. While we are not going to size the equipment in this chapter, know that the steps necessary to properly size the equipment are available. Whether you are going to build a complete machine or only program the motion portion, the concepts of mechanical motion and the rules for sizing components will be necessary for you to understand.

Faceplates for Motion Equipment

We next look at the control of an axis or multiple axes of motion from the point of the operator. What specifically will we allow the operator to do?

The machine will dictate much of the answer but we should explore what is a standard faceplate for motion control. The figures below are of some typical multi-axis machine controls that are found on the plant floor. The faceplate may be entirely created on a computer screen and have only a few external buttons or may be totally created with hard-wired buttons. For sake of simplicity, the lab experiences in this chapter will be created totally with faceplates.

The one button that is always programmed with hard wire is the E-Stop. The figure below has the E-Stop pictured on the screen but this is never done with equipment that can hurt. We will allow the E-Stop button to be programmed on the faceplate only because the equipment does not have the torque or capability to hurt an individual.

A sample of a two different systems below in Figure 17-25 and later in Figures 17-26a,b show some common elements.

There should be an on-off enable of some kind for each axis. The enable switch usually is not on the face of the panel but there usually one, either hard-wired or set in software.

There should be an on-off button. This button or buttons shows the present state of the machine. There may be a home switch that either directs the machine to find a hard-wired home position or set the present position as the home position. There should be a jog button that manually increments each axis in a manner that slowly moves the axis as long as the button is held down. The jog button(s) should allow the machine to turn either in the forward direction or the reverse. These buttons should be active only in the manual mode.

There should be an auto-manual switch. In auto, there is the ability to run a sequence of automated moves that would make a part or help in the making of a product. It may have several components such as a move forward, dwell, move reverse, or other sequence of selected moves. There should be a button to run the automatic sequence as well as stop the sequence. There should also be a button to halt or pause the sequence and resume the sequence to completion.

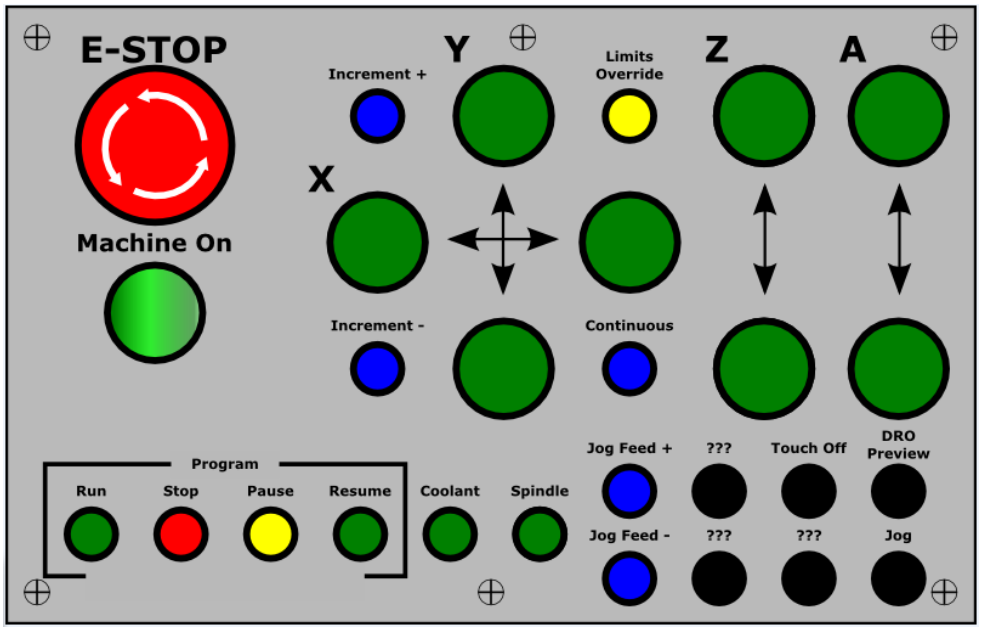


Fig. 17-26a HMI Screen for X-Y Axis Machine

Fig. 17-26b HMI Panel for Series-9 A-B Multi-Axis Motion System (rt side)

Notice in the HMI above Fig 17-26a, the terms such as spindle. The spindle is a motor that turns an axis that may or may not be controlled for position. Spindles may reference a lathe in which the motor just turns very fast or a motor that may turn very fast but also be used for threading, a controlled action. Also, notice the very large emergency stop and the e-stop reset just above the E-stop button.

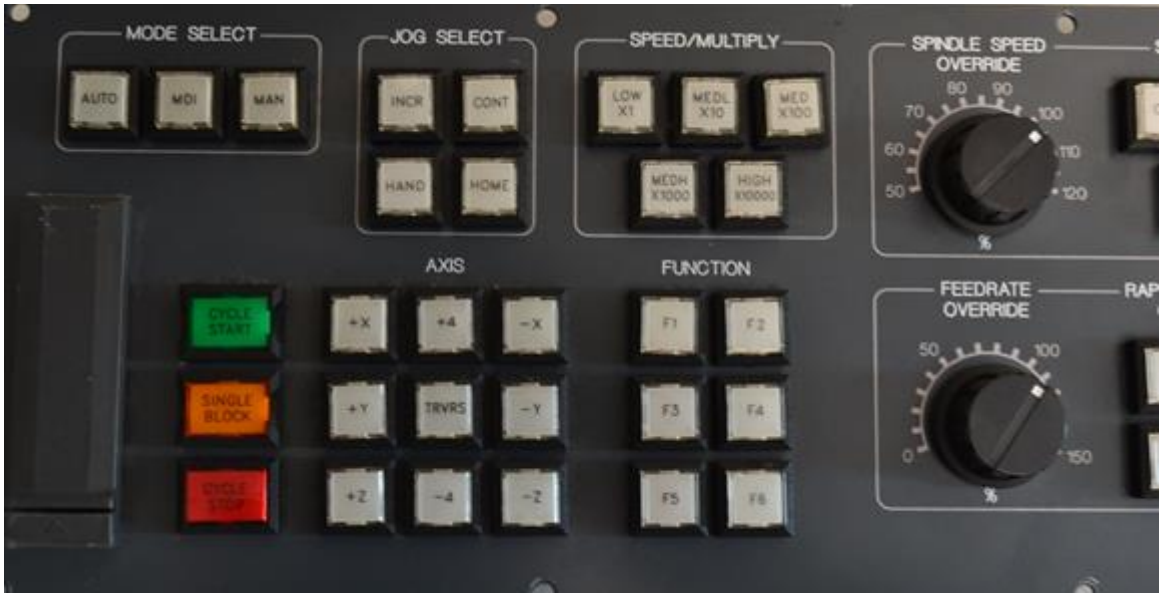


Fig. 17-26c HMI Panel for Series-9 A-B Multi-Axis Motion System (left side)

Notice the above figure controls three axes, X, Y, and Z. Any axis can be jogged or moved in an auto, MDI or manual mode.

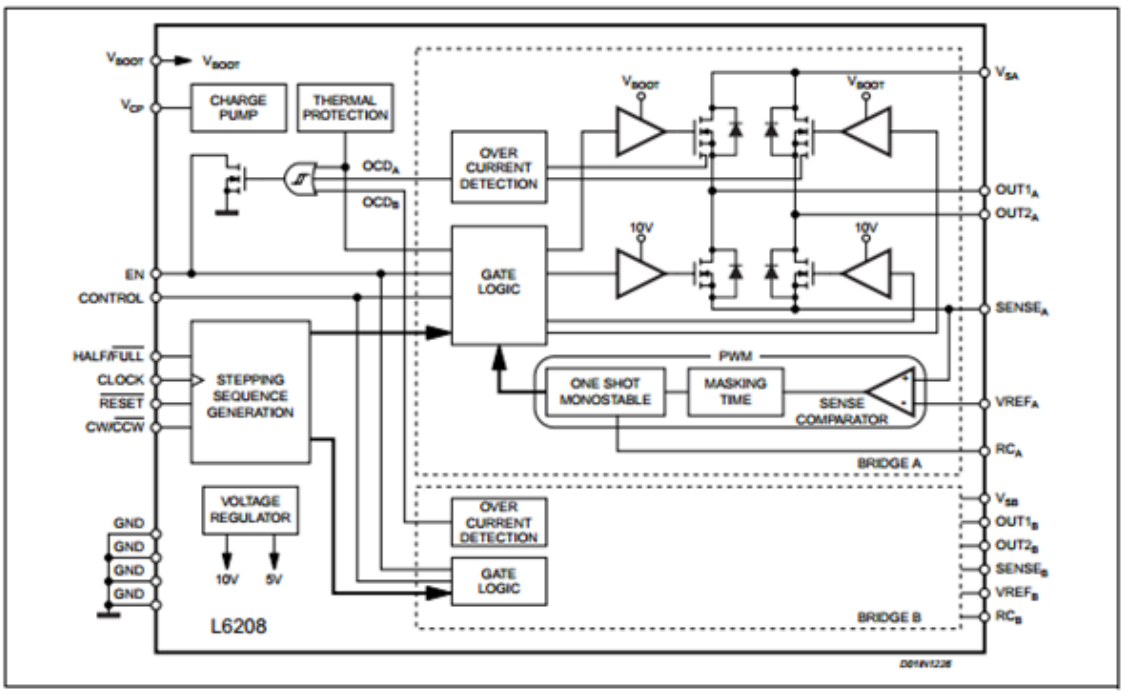
These HMI screens may provide more information and go further than a simple single-axis motion HMI for classroom use. They do, however, include most of the motion functions required for a real milling machine or lathe in industry. If the task was to program a factory-floor motion application, most of the functions on these panels would at least need to be considered prior to commissioning.

The designs of motion controls below are for Siemens and Allen-Bradley single-axis machines. The processors and motion equipment do not support multiple axis machines that are coordinated. The A-B equipment does support multi-axis coordinated machines but not with the drive equipment provided. Both Siemens and Allen-Bradley provide both single and multiple axis controls for machine tool applications. The costs rise as accuracy and coordination between axes increase.

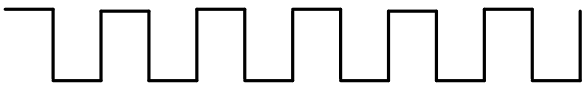
Siemens Single Axis Stepper Control

The L6208 chip is designed specifically for the control of stepper motors. It may be used in a bread-board application or in conjunction with external components in a pre-packaged circuit board. We will use it in conjunction with a pre-packaged circuit board.

L6208 block diagram. Fig. 17-27



The external clock input is shown below. The Siemens S7-1200 outputs a waveform (PTO) switched between 0-24 VDC that creates this waveform.



From oscillatory circuit
(the PLC in our case)

To Clock Input (Stepping Sequence Generation)

Fig. 17-28

Other logic pins such as CW/CCW, CONTROL, RESET, HALF/FULL are similar to the CLOCK signal in that they are CMOS/TTL compatible.

The EN pin is a bidirectional input and must be pulled less than a trigger value to turn the PowerDMOS off. This value is attained with a 2.2K resistor at 5 Vdc.

The EVAL6208N board was used for this application. It is pictured below. It was powered with a drive voltage of 12Vdc and a signal voltage of 5Vdc. The board seemed to do a good job of powering the NIDEC H17ET120434 stepper motor. The motor also has been successfully powered from an Arduino stepper controller card. For purposes of this lab, however, the decision was made to use the PLC along with an interface card and the stepper to turn a dial or simple bolt pointer. Also in the project is a HMI used to set the parameters of the rotation such as speed, number of turns and direction.

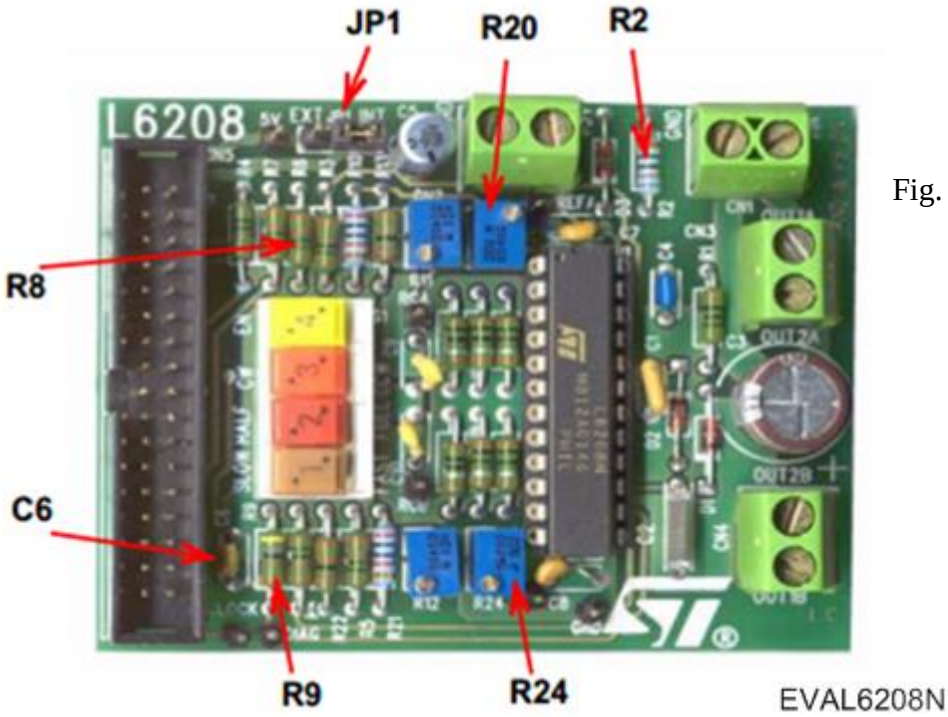


Fig. 17-29

The board used is powered by 12 V and 5 V or by 12 VDC only with 5 VDC converted on the board. We will use it in the latter mode with only the 12 VDC power to the board. Signals for stepper pulse and direction will be included through the ribbon connector at the left of the board. Pin-outs for the ribbon cable include all signal levels for the board but only the two (direction and pulse) are used.

The stepper purchased was the one pictured below from Hurst (Nidec).

Series H17ET Hybrid Stepper Motor

Mounting Flange:	NEMA 17
Step angle:	1.8°
Positional Accuracy:	+ 5% max.
Number of Phases:	2 or 4
Temperature Rise:	70°C max
Insulation Resistance:	100M ohms at 500VDC for 1 minute
Dielectric Strength:	500VAC for 1 minute
Insulation Class:	Class B
Number of lead wires:	4, 5 or 6
Lead wire:	UL3265 AWG#26
Operation Ambient Temp:	-10°C ~ +50°C
Radial Play:	0.03 mm max at 0.4 kg load
Axial Play:	0.08 mm max at 0.5 kg load

Stepper Motor Used in Lab

Fig. 17-30a

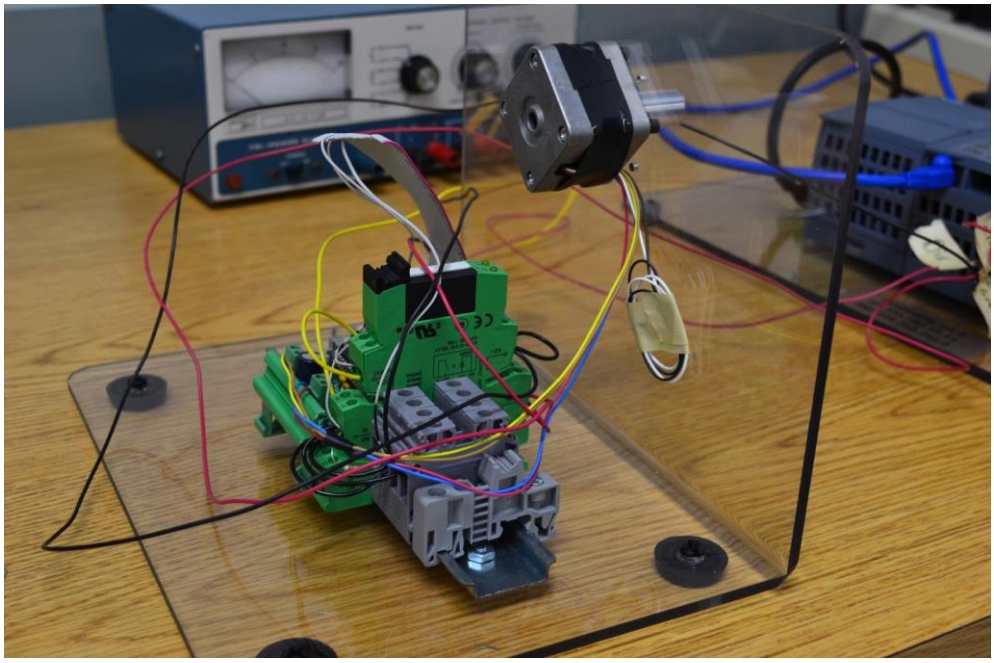


Fig. 17-30b

Above is shown the stepper motor and Eval board with interface connections.

The following two modules have changed from green to gray but are the same modules as those shown in the EVAL board design. The first is a relay board and is used to determine direction of the motor. A closed relay will allow the motor to rotate in one direction while an open relay allows the motor to rotate in the opposite direction.

Relay Module - PLC-RSC- 24DC/21 - 2966171

PLC relay, consisting of base terminal block PLC-BSC.../21 with screw connection and pluggable miniature relay with power contact, for assembly on DIN rail NS 35/7.5, 1 PDT, input voltage 24 V DC

Generate product PDT | 3D

Fig. 17-30c

The optocoupler is designed to step the voltage down from 24 VDC to 5 VDC as required by the EVAL board for the pulse input driving the stepper. The output from the PLC is 24 VDC. While cost of these modules was over $100, the price has dropped to $30-$50 per module.

Optocoupler - PLC-OSC- 24DC/ 24DC/ 2/C1D2 - 5603260

Please be informed that the data shown in this PDF Document is generated from our Online Catalog. Please find the complete data in the user's documentation. Our General Terms of Use for Downloads are valid (http://phoenixcontact.com/download)

PLC-INTERFACE, consisting of DIN rail-mountable basic terminal block with screw connection and plug-in miniature solid-state relay, input: 24 V DC, output: 3 ... 33 V DC/3 A, UL/cUL: approved for use in Ex Zone Class I, Div. 2

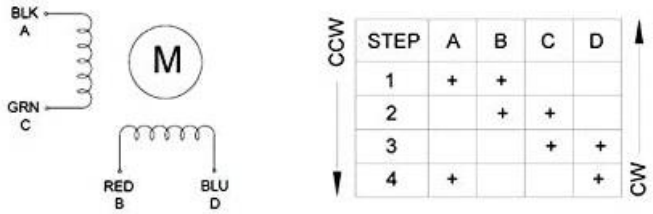
Fig. 17-30d

The lab of Fig. 17-30c is an upgrade to the lab above. It is much less expensive and uses only a single power supply (24 VDC). These components were made for the Arduino and 3D Printer. Distributors such as the Robot Shop or Amazon have these components and they are very economical. The signal interfaces for the EVAL board above cost more by far than the entire lab below.

Fig. 17-30e

Nema 17 Stepper Motor with Black L Stepper Motor Bracket and TB6600 4A 9-42V Stepper Motor Driver CNC Controller for 3D Printer
by ICROATO

Fig. 17-30f



STEP	A	B	C	D
1	+	+		
2		+	+	
3			+	+
4	+			+

From the manual for the stepper drive, we find the following wiring diagram which shows the wiring between the PLC and the stepper drive:

Common-Cathode Connection:

Connect PUL -, DIR - and EN - to the ground terminal of the control system. Pulse signal connects to PUL-; direction signal connects to Dir- ; Enable signal connects to EN-. As shown below:

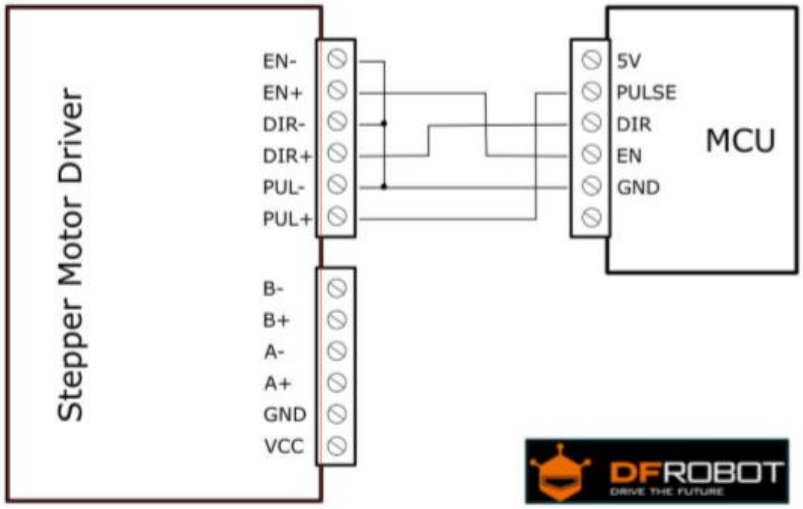


Fig. 17-30g

Wiring from the PLC to the Stepper Controller using the diagram above:

Pulse wiring from MCU:	Q0/0	from Siemens PLC
Dir wiring from MCU	Q0.1	from Siemens PLC
EN wiring from MCU	Q0.2	from Siemens PLC
GND wiring from MCU	M Terminal	from Siemens PLC (0 V)
VCC on drive	L Terminal	from Siemens PLC (24 V)

Resistor value between PLC output and stepper drive input:

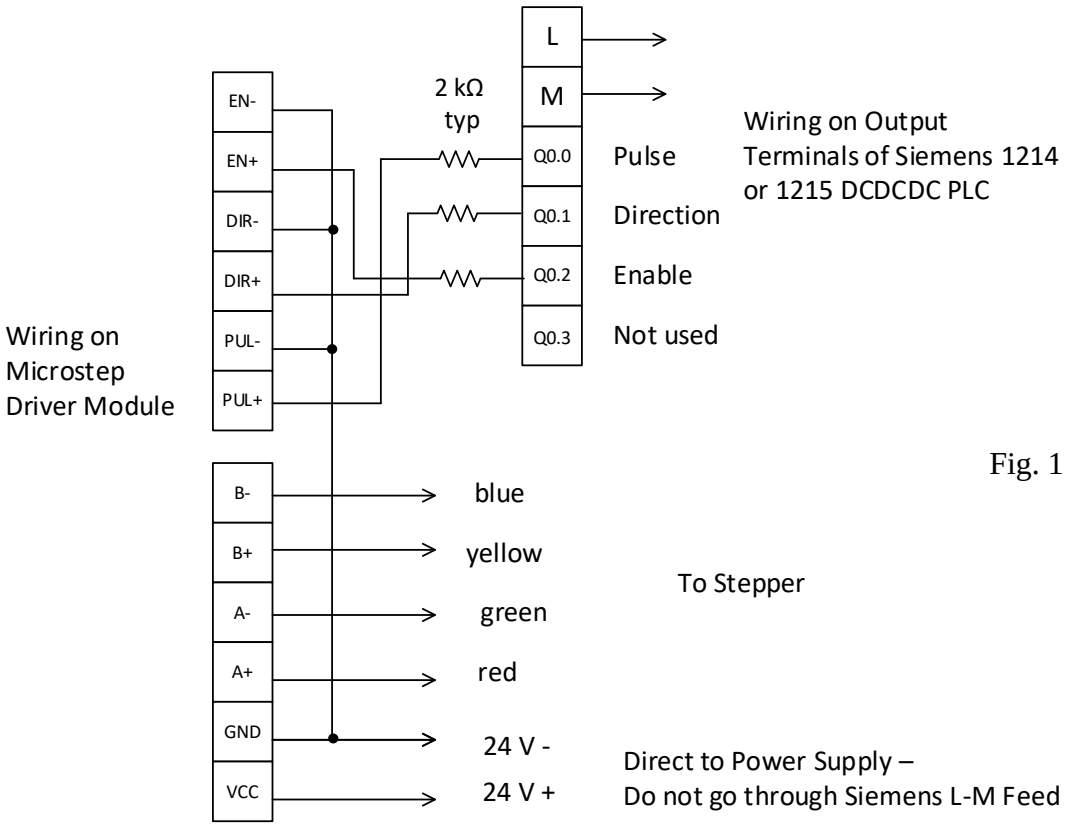


Fig. 17-30h

Programming the Stepper

The Siemens PLC is used to generate the pulses controlling the stepper motor through one of the two systems listed above. Software configuration is described in this section. Use this application to successfully start and control the stepper motor.

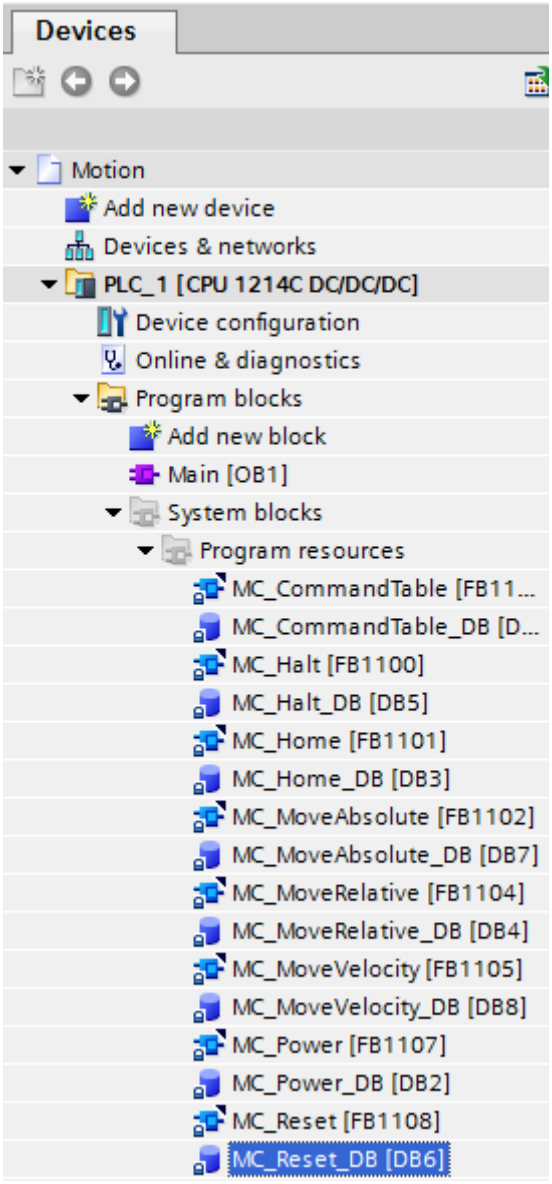


Fig. 17-31

The Project Tree for a single axis PTO drive. There are a number of blocks programmed in the example which allow the user to experiment with the various blocks using the inputs from the selector switches. The selector switches are useful but must be supplemented by the HMI panel in order to provide a complete project.

The various function and data blocks as well as tags are created in the steps following.

Follow the steps below in the order given to use motion control with the CPU S7-1200. The subject will be broadly divided into the following steps. The text will only cover portions of these:

1. Add technological object Axis
2. Working with the configuration dialog
3. Download to CPU
4. Function test of the axis in the commissioning window
5. Programming
6. Diagnostics of the axis control

The axis program given already has commissioned the drive and has several programming blocks present to test the operation of the stepper. The following gives the list of programming blocks available plus the Command Table blocks. The Command Table block allows the user to enter a number of commands in a sequence for execution as a block.

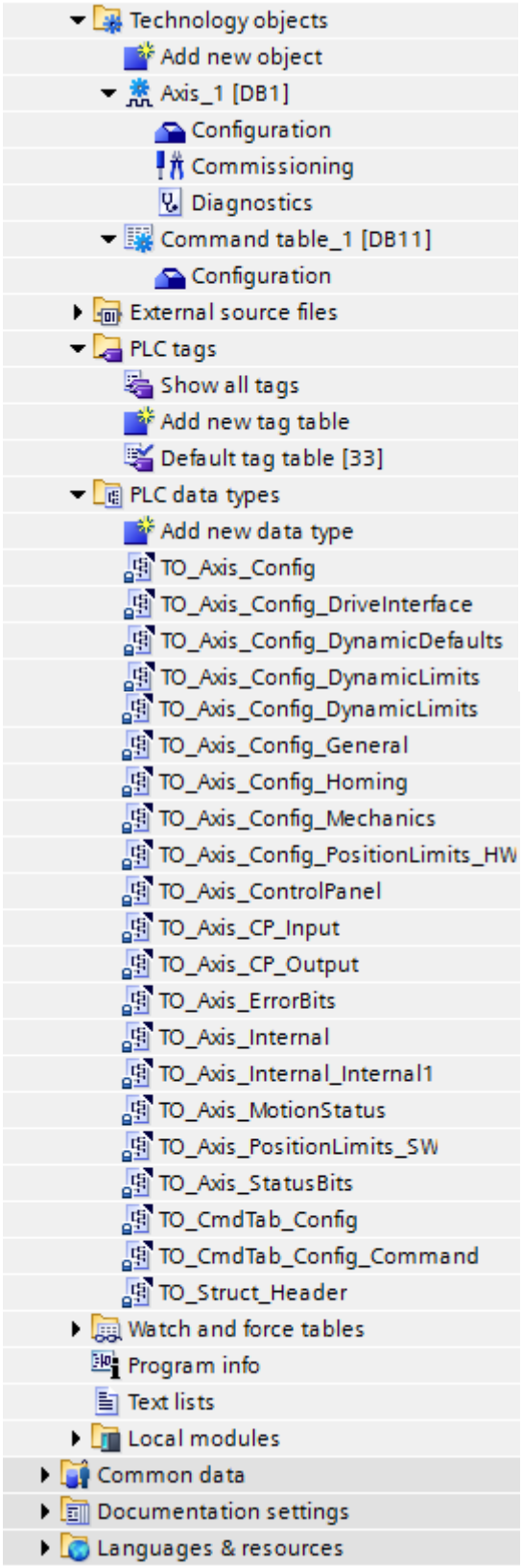


Fig. 17-32a
The Project Tree continued

Homing modes:

Active homing

In active homing mode, the motion control instruction "MC_Home" performs the required reference point approach. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are aborted.

Passive homing

During passive homing, the motion control instruction "MC_Home" does not carry out any homing motion. The traversing motion required for this step must be implemented by the user via other motion control instructions. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are not aborted upon start of passive homing.

Direct homing absolute

The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The value of input parameter "Position" of motion control instruction "MC_Home" is set immediately as the reference point of the axis.

Direct homing relative

The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The following statement applies to the axis position after homing:
New axis position = current axis position + value of parameter "Position" of instruction "MC_Home".

Overview of the Motion Control Statements:

You control the axis with the user program using motion control instructions. The instructions start Motion Control jobs that execute the desired functions.

The status of the motion control jobs and any errors that occur during their execution can be obtained from the output parameters of the motion control instructions. The following Motion Control instructions are available:

MC_Power:	Enable, disable axis
MC_Reset:	Acknowledge error
MC_Home:	Home axes, set home position
MC_Halt:	Halt axis
MC_MoveAbsolute:	Absolute positioning of axes
MC_MoveRelative:	Relative positioning of axes
MC_MoveVelocity:	Move axes at preset rotational speed
MC_Moveog:	Move axes in jogging mode
MC_CommandTable:	Run axis jobs as movement sequence
MC_ChangeDynamic:	Changing the dynamic settings for the axis

Creating a user program

Proceed as follows to create the user program:

1. In the project tree, double-click your code block (the code block must be called in the cyclic program). The code block is opened in the programming editor and all available instructions are displayed.

2. Open the “Technology” category and the “Motion Control” and “S7-1200 Motion Control” folders.

3. Use a drag-and-drop operation to move the “MC_Power” instruction to the desired network of the code block. The dialog box for defining the instance DB opens.

4. In the next dialog box, select from the following alternatives:
   Single instance
   Click “Single instance” and select whether you want to define the name and number of the instance DB automically or manually.
   Multi-instance
   Click “Multi-instance” and select whether you want to define the name of the multi-instance automatically or manually.

5. Click “ok”. The motion control instruction “MC_Power” is inserted into the network:

Fig. 17-32b

   Parameters marked with “<???>” must be initialized; all other parameters are assigned default values. Parameters displayed in black are required for use of the motion control instruction.

6. Select technology object in the project tree and drag-and-drop it on <???>.

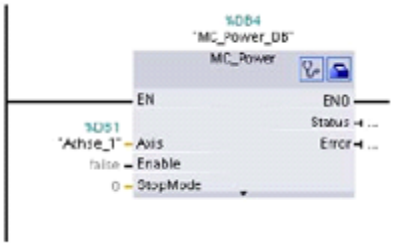


Fig. 17-32c

   Following selection of the technology object data block, the stethoscope button is available. Click the stethoscope if you want to open the diagnostics dialog for the technology object.

Click the toolbox icon if you want to open the configuration view of the technology object:

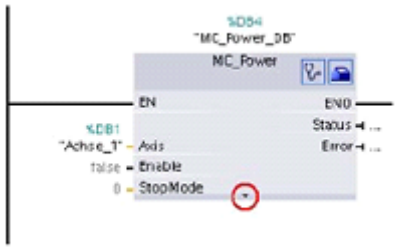


Fig. 17-32d

Click the toolbox icon if you want to open the configuration view of the technology object. Click the arrow down icon to view additional parameters of the motion control instruction.

7. Add your choice of motion control instructions from steps 3 to 6 above.

Programming notes:

When creating your user program, note the following:

- Cyclic call of utilized motion control instructions. The current status of command execution is available via the output parameters of the motion control instruction. The status is updated with every call of the motion control instruction. Therefore, make sure that the utilized motion control instructions are called cyclically.
- Transfer of parameter values of a motion control instruction. The parameter values pending for the input parameters are transferred with a positive edge at input parameter "execute" when the block is called. The motion control command is started with these parameter values. Parameter values that are subsequently changed for the motion control instruction are transferred until the next start of the motion control command. Exceptions to this are input parameters "StopMode" of motion control instruction "MC_Power" and "Velocity" of motion control instruction "MC_MoveJog". A change in the input parameter is also applied when "Enable" = true or "JogForward' and "JogBackward"…
- Programming under consideration of the status information. In a stepwise execution of motion control jobs, make sure to wait for the active command to finish before starting a new command. Use the status messages of the motion control instruction and the "StatusBits" tag of the technology object to check for completion of the active command.

  In the example below, observe the indicated sequence. Failure to observe the sequence will display an axis or command error.

  - Axis enable with motion control instruction "MC_Power"
    You must enable the axis before it can take on motion jobs. Use an AND operation of tag <Axis name>.StatusBits.Enable=TRUE with output parameter Status = TRUE of motion control instruction "MC_Power" to verify that the axis is enabled.
  - Acknowledge error with motion control instruction "MC_Reset". Prior to starting a motion control command, errors requiring acknowledgement must be acknowledged with "MC_Reset". Eliminate the cause of the error and acknowledge the error with motion control instruction "MC_Reset". Verify that the error has been successfully acknowledged before initiating a new command. For this purpose, use an AND operation of tag <Axis name>.StatusBits.Error = FALSE with output parameter Done = TRUE of motion control instruction "MC_Reset".

- Home axis with motion control instruction “MC_Home”
  Before you can start an MC_MoveAbsolute command, the axis must be homed. Use an AND operation of tag <Axis name>.StatusBits.HomingDone= TRUE with output parameter Done = TRUE of motion control instruction “MC_Home” to verify that the axis has been homed.

- Override of motion control command processing
  Motion control jobs for moving an axis can also be executed as overriding jobs. If a new motion control command is started for an axis while another motion control command is active, the active command is overridden by the new command before the existing command is completely executed. The overridden command signals this using CommandAborted = TRUE in the motion control instruction. It is possible to override an active MC_MoveRelative command with a MC_MoveAbsolute command.
- Avoiding multiple use of the same instance
  All relevant information of a motion control command is stored in its instance. Do not start a new command using this instance, if you want to track the status of the current command. Use different instances if you want to track the commands separately. If the same instance is used for multiple motion control commands, the status and error information of the individual commands will overwrite each other.
- Call of motion control instructions in different priority classes (run levels). Motion control instructions with the same instance may not be called in different priority classes without interlocking. To learn how to call locked motion control instructions, refer to “Tracking commands from higher priority classes”.

Monitoring active commands

There are three groups for tracking active motion control jobs, those with output parameter “Done”, instruction MC_MoveVelocity and instruction MC_MoveJog.

See the results of the instructions’ timing diagram with abort or error conditions. The WinCC book at about pg. 4000 shows these results.

Create a Motion Technology Object:
Add new object> Select Axis> Axis_1 (define the axis)

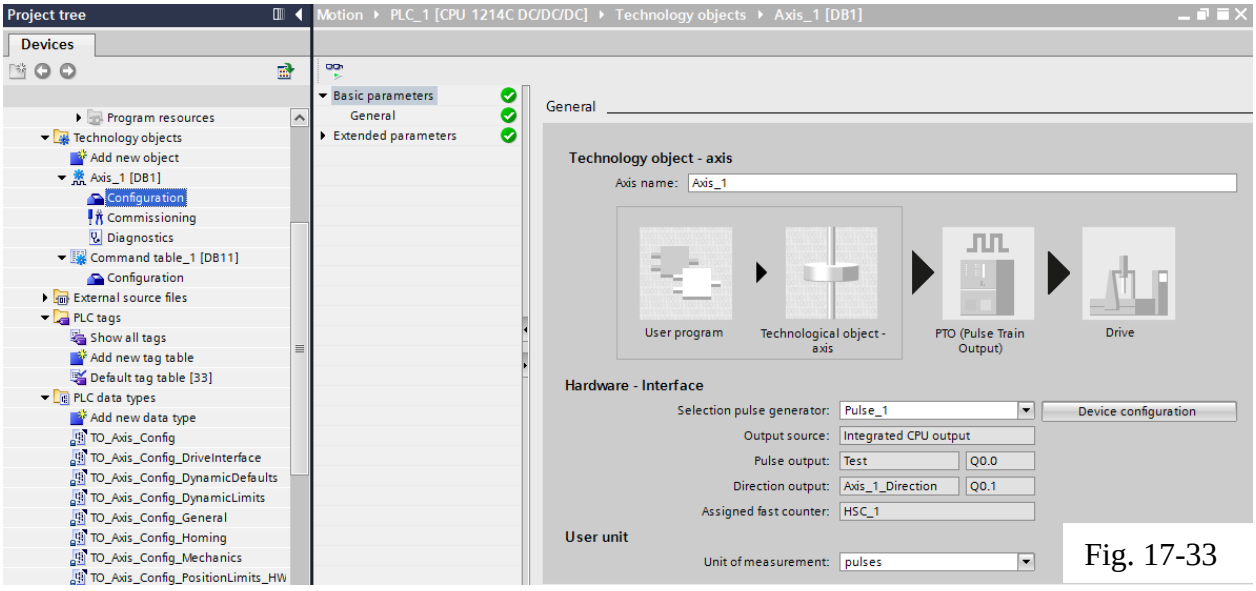


Fig. 17-33

Define the mechanical parameters of the machine:
Steps per revolution
Distance per Motor Revolution
Invert Direction Signal

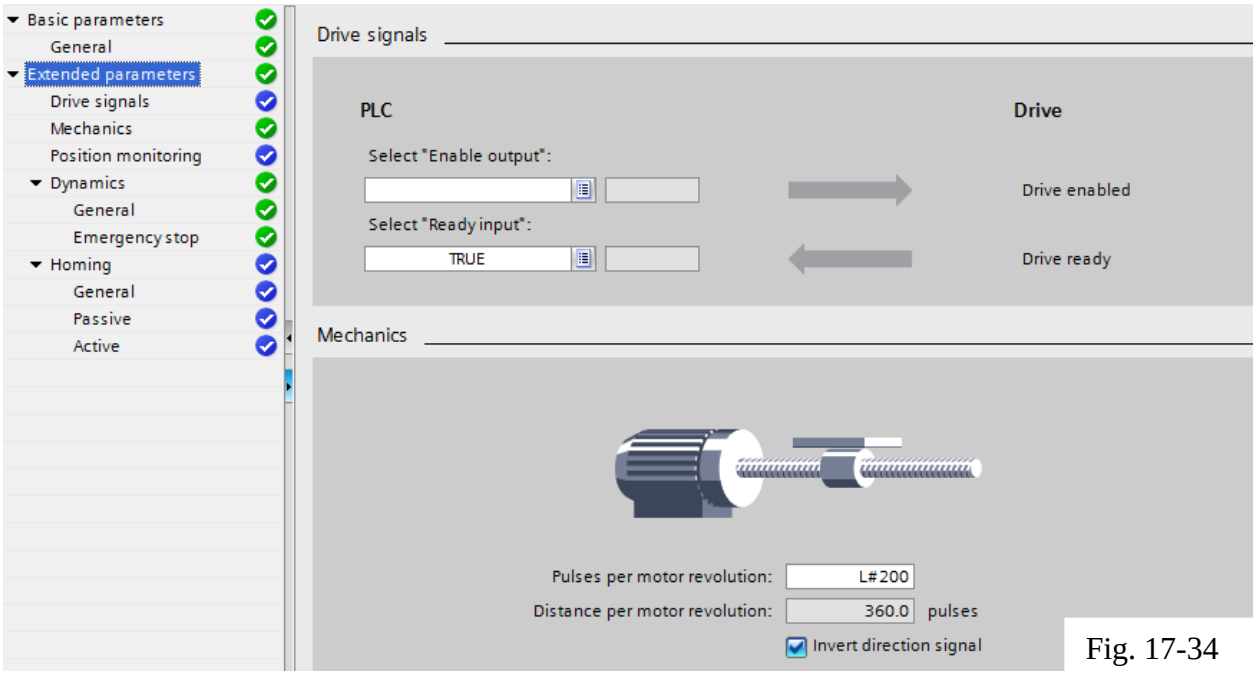


Fig. 17-34

Leave hardware limits not enabled

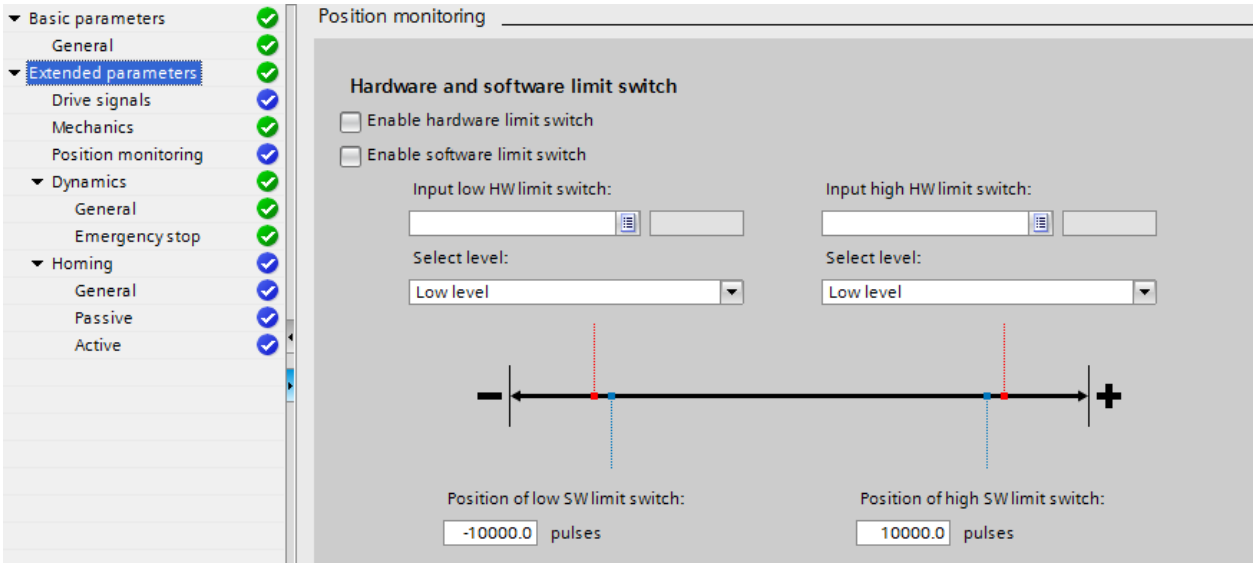


Fig. 17-35

Determine the acceleration/deceleration for the axis:
Select units for pulses/s
Set maximum Velocity
Set Start/Stop Velocity

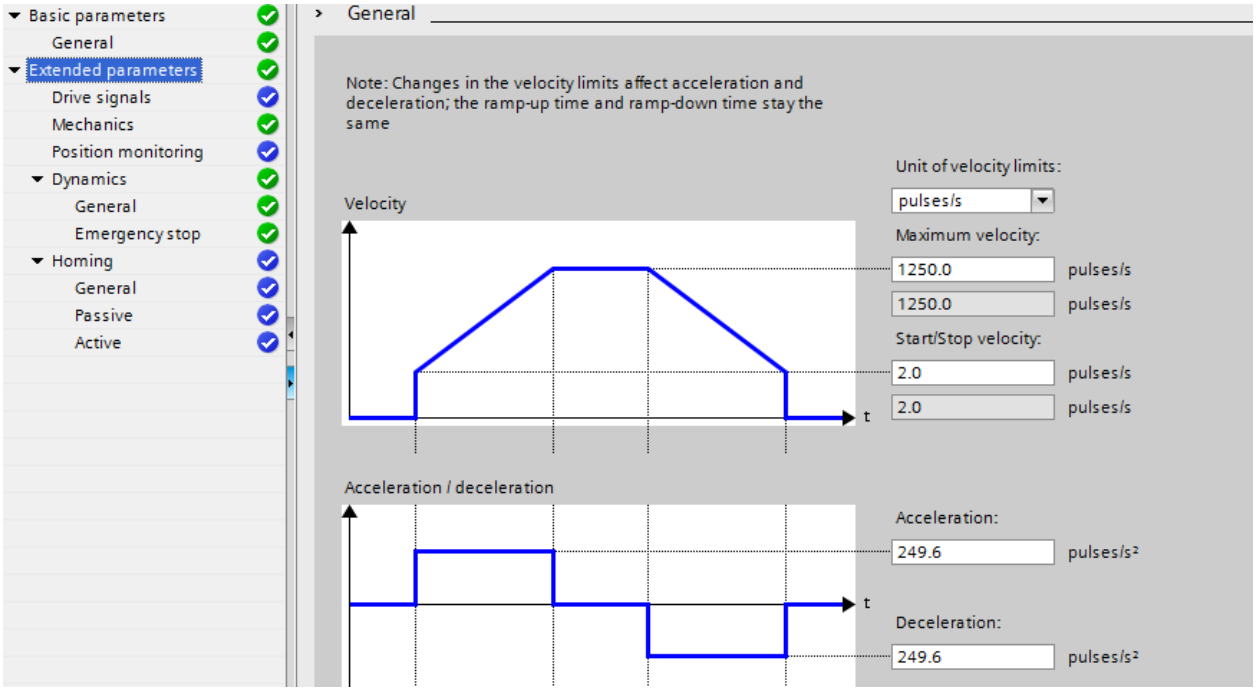


Fig. 17-36

Set Ramp Up/Ramp Down Time

Set Emergency Ramp Time
Determine the Emergency Decel Time for immediate stops:

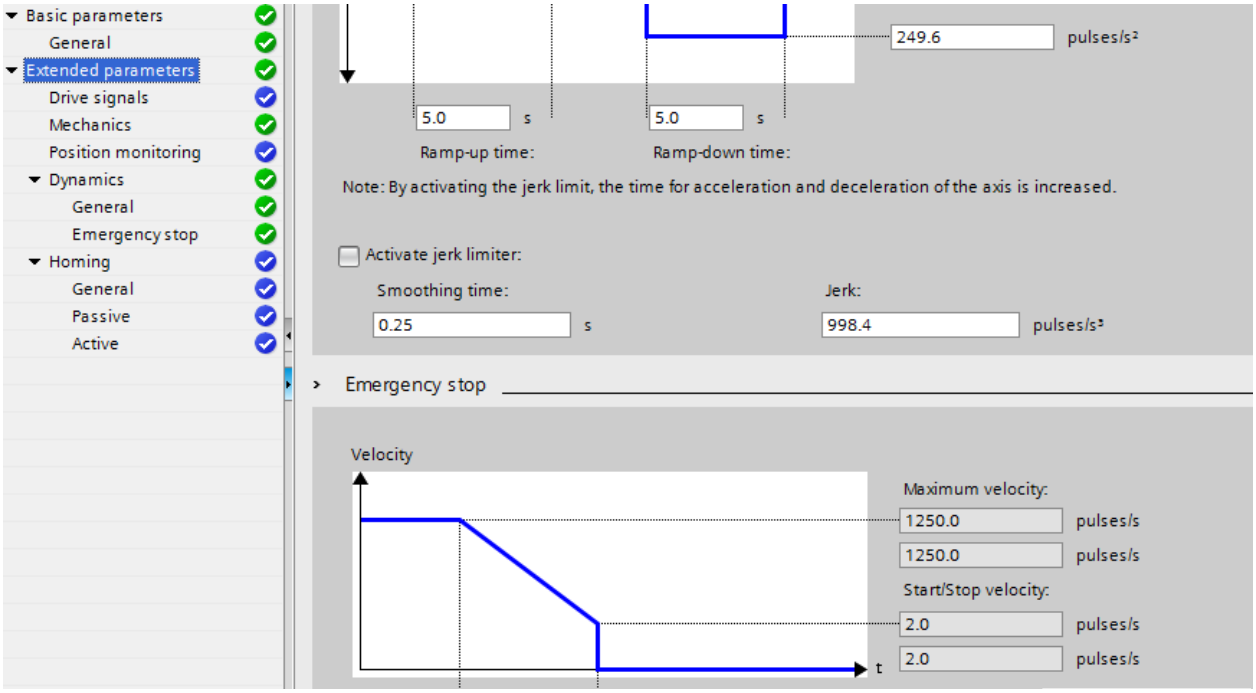


Fig. 17-37

Emergency Stop Deceleration
Homing Configuration (leave open)

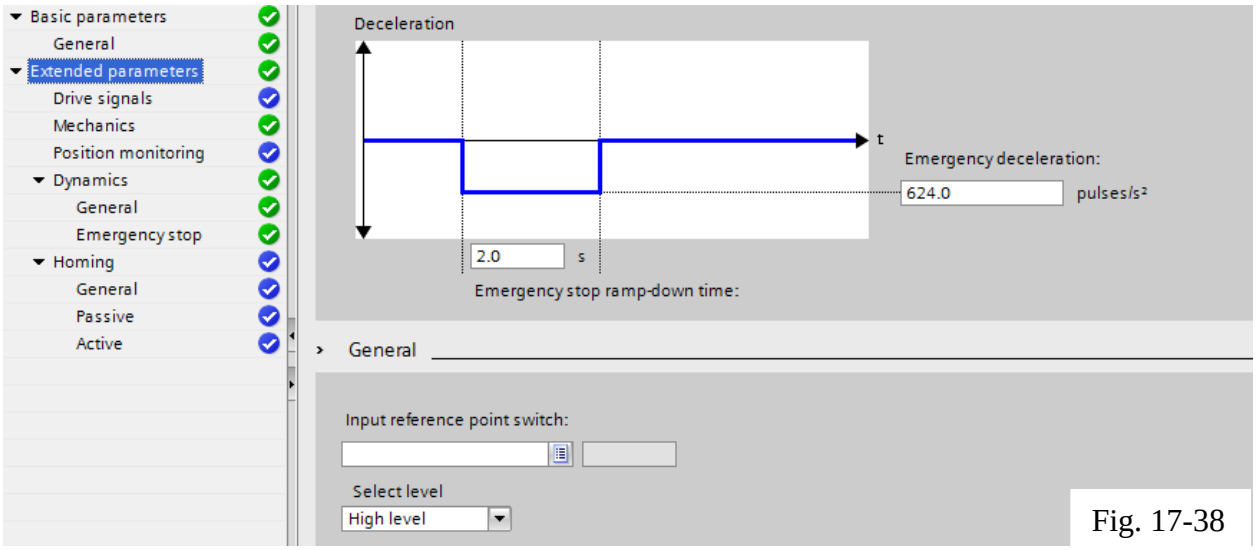


Fig. 17-38

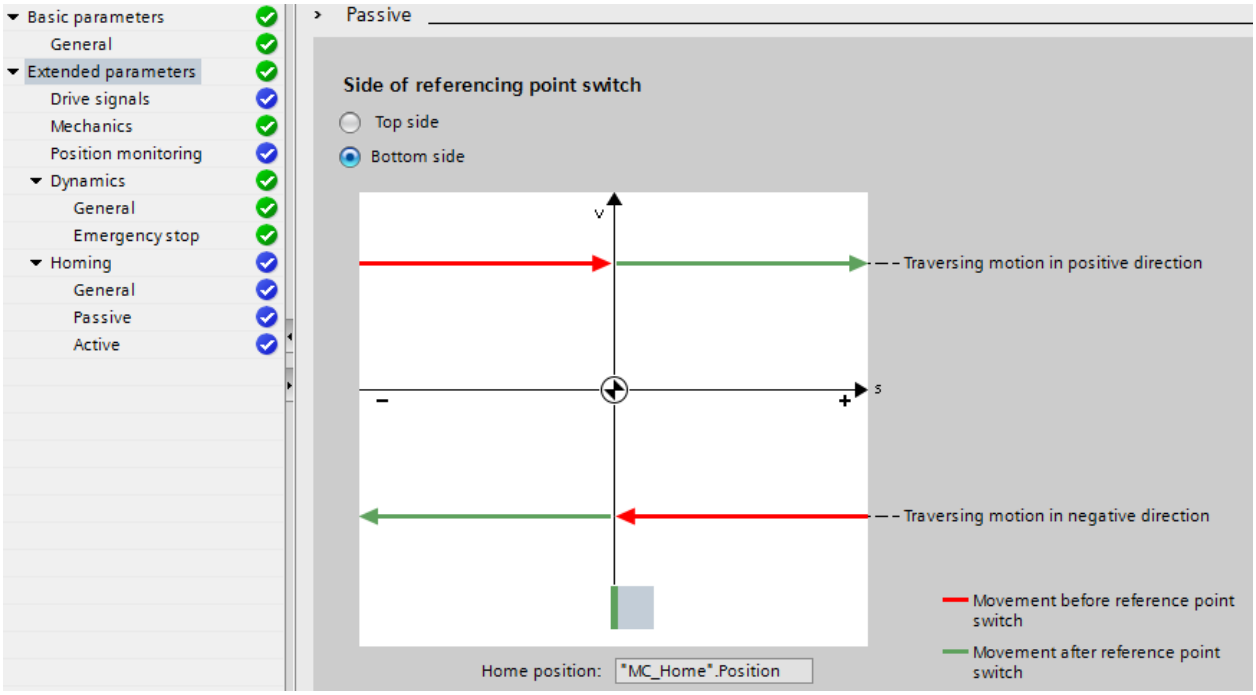


Fig. 17-39

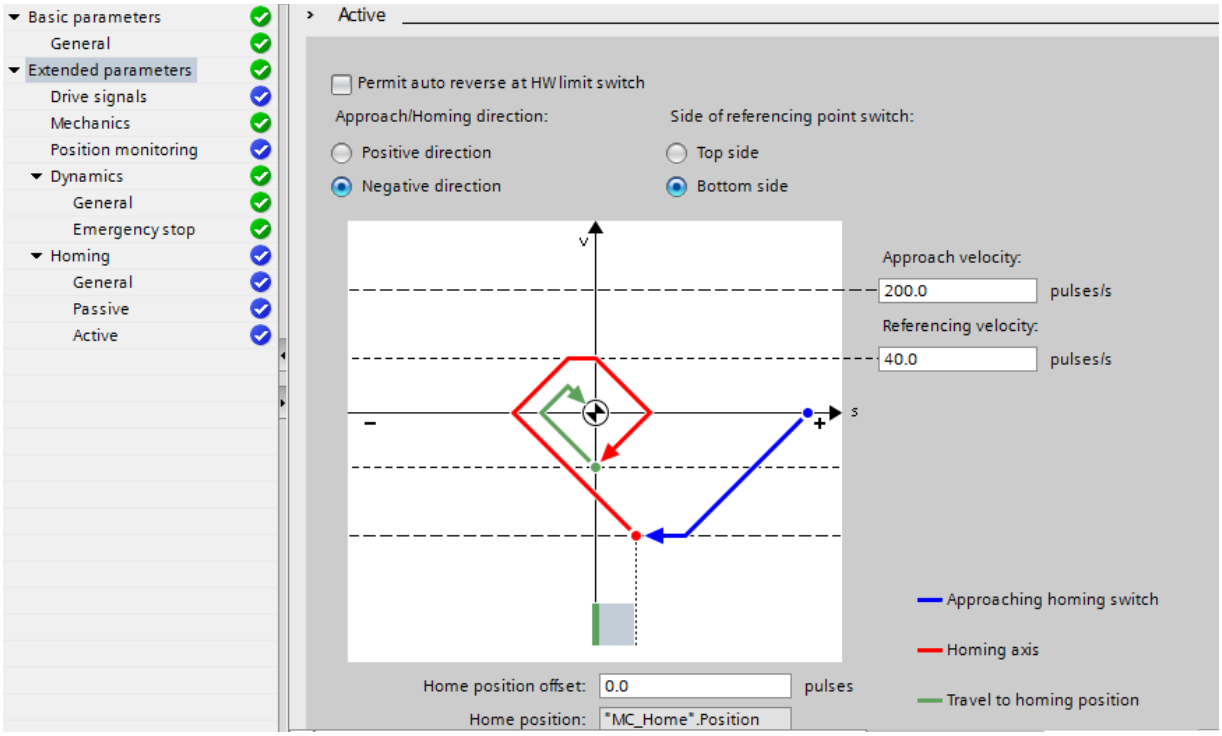


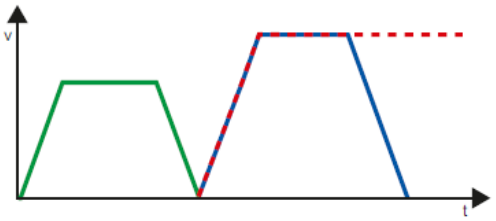
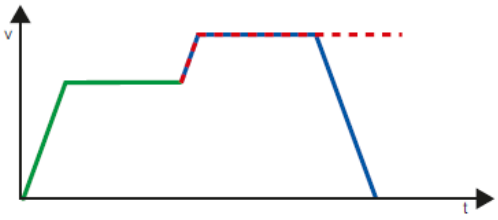
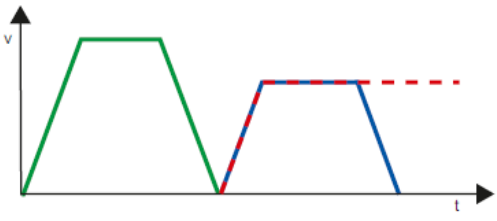
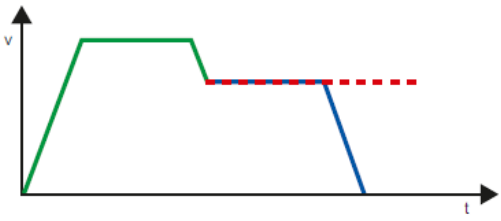
Fig. 17-40

(The Technology Object for Motion configuration is complete.)

Motion transition with preceding velocity jobs

Transition from "Complete command" to "Blend motion"

The charts below show the transition between movements in various different transition modes in the "Next step" column:

Complete job	Blend motion
Transition from lower to higher velocity  A job with high velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	Transition from lower to higher velocity  A job with high velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The second job starts with the new velocity.
Transition from higher to lower velocity  A job with low velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	Transition from higher to lower velocity  A job with low velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The first job starts with the new velocity.

Next begin commissioning:
Manually move the axis to verify operation
1 - Select manual control
2 - Enable axis
3 - Select action (jog)
4 - Set Home Position Offset
5 - Set Accel Rate
6 - Make it go

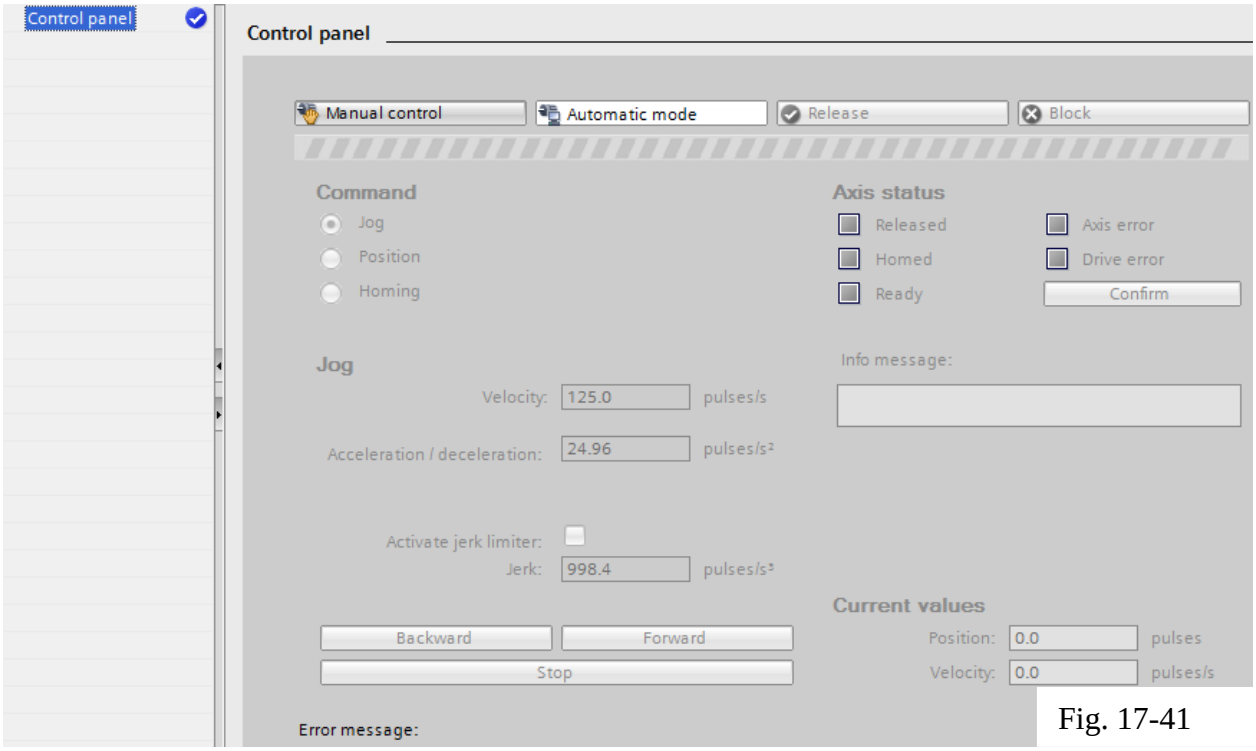


Fig. 17-41

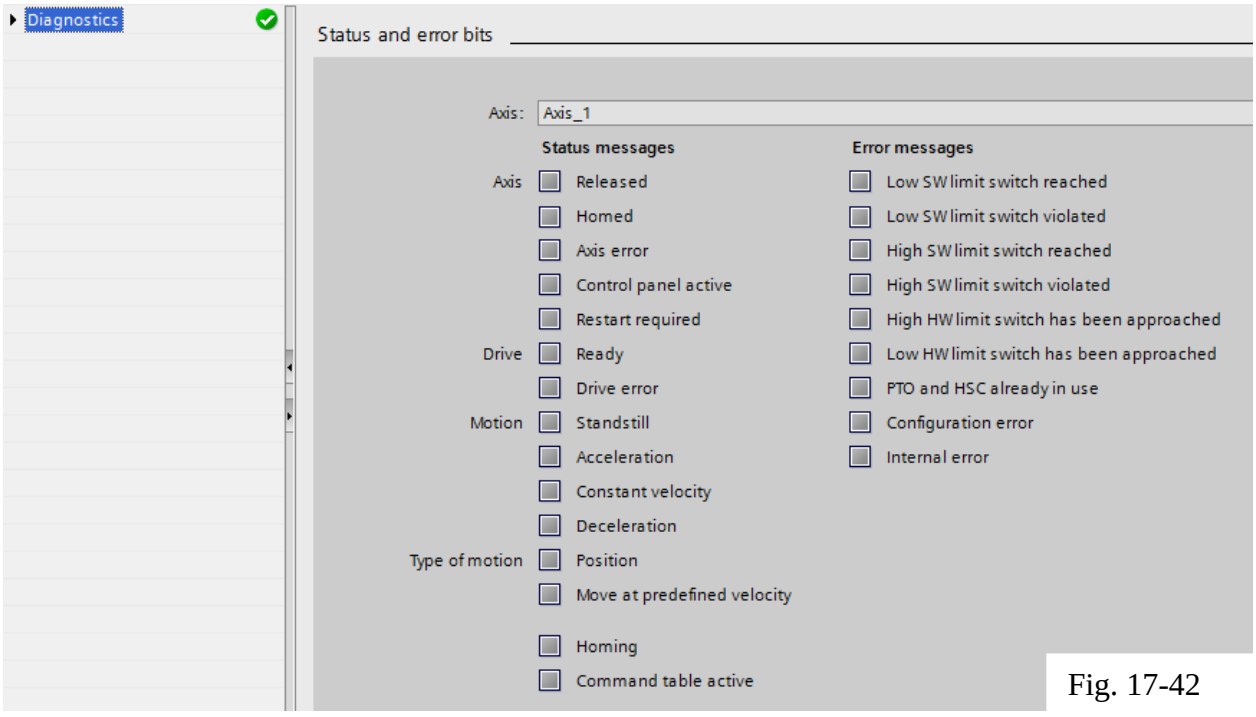
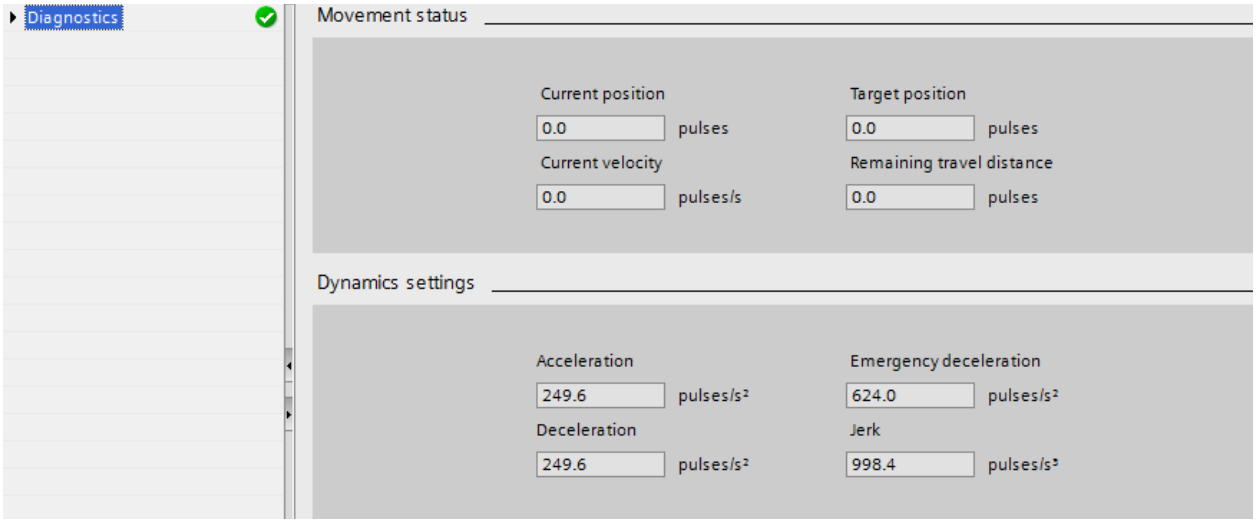
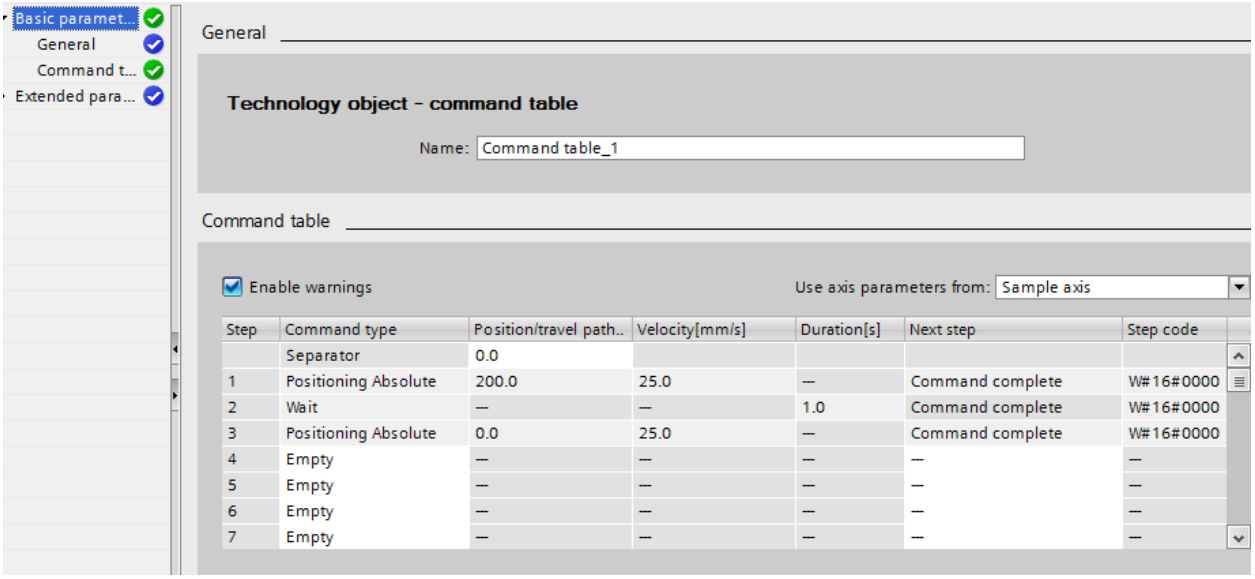


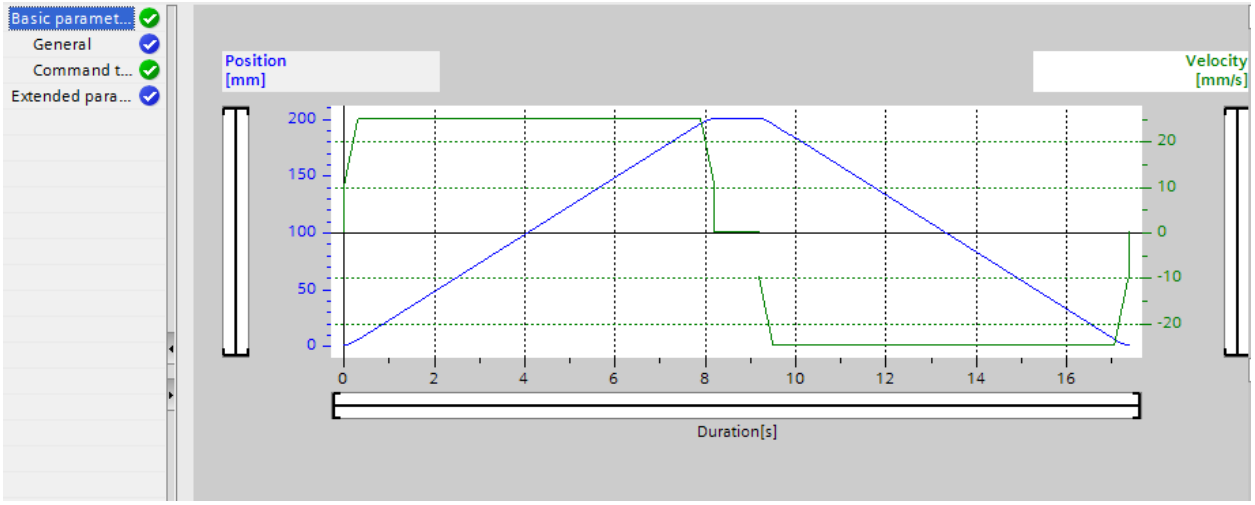
Fig. 17-42



This figure shows Movement Status and Dynamic Settings Fig 17-43



This figure shows a Command Table Fig. 17-44



This figure shows the motion and position of the movements of the Command Table Fig. 17-45

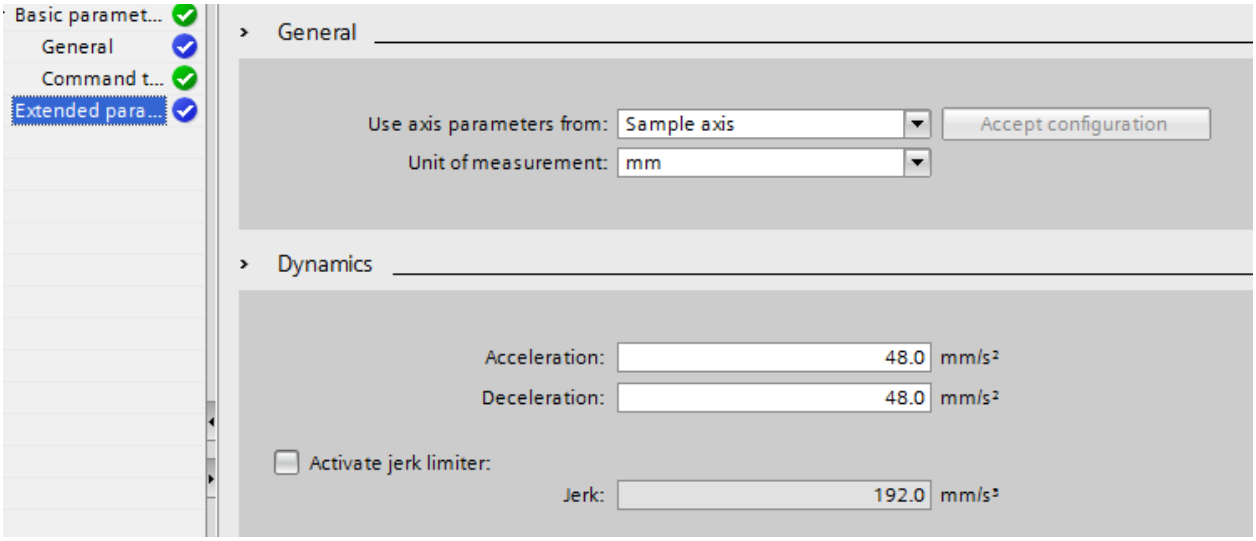


Fig. 17-46

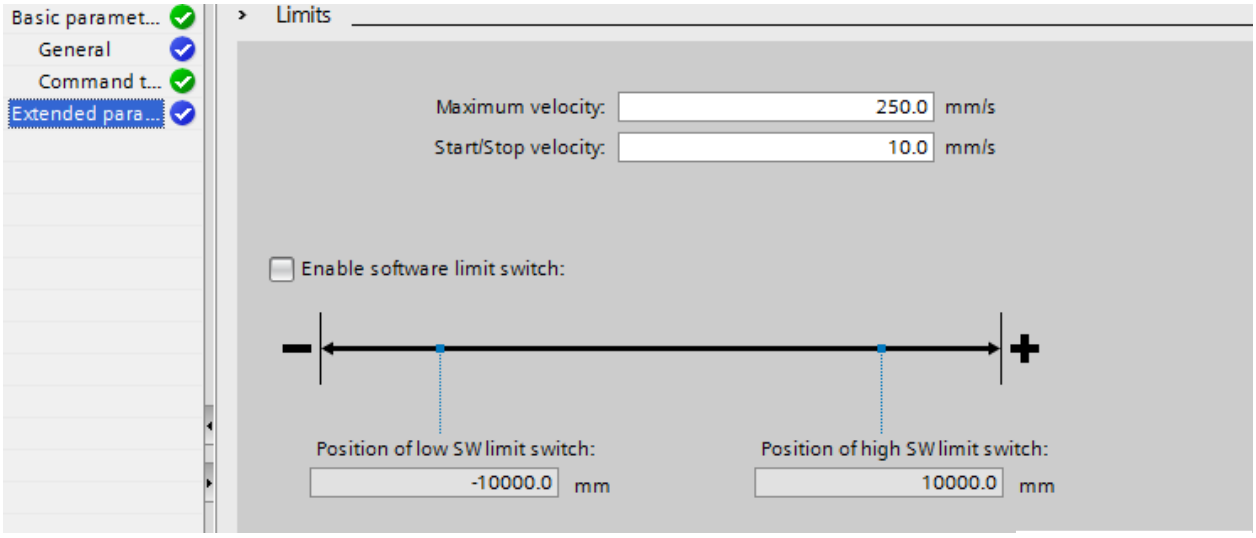


Fig. 17-47

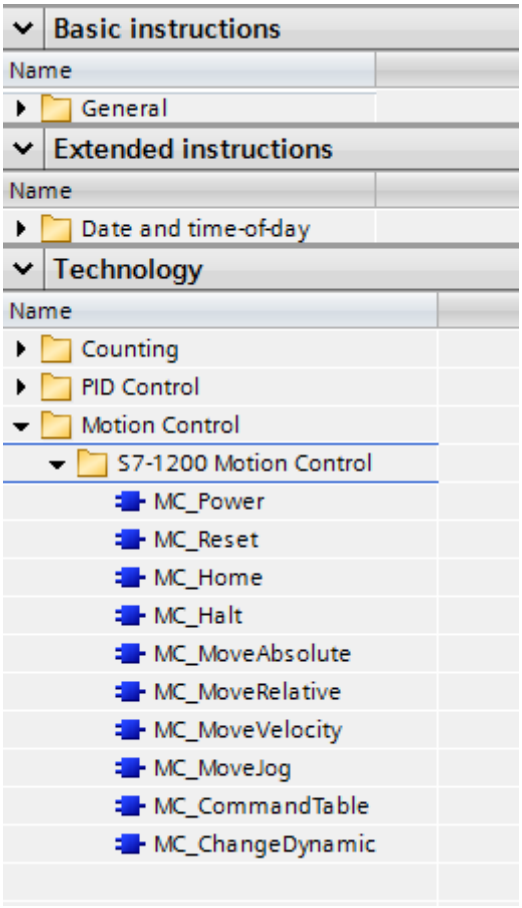


Fig. 17-48
Instructions used in the Motion Application to control the axis

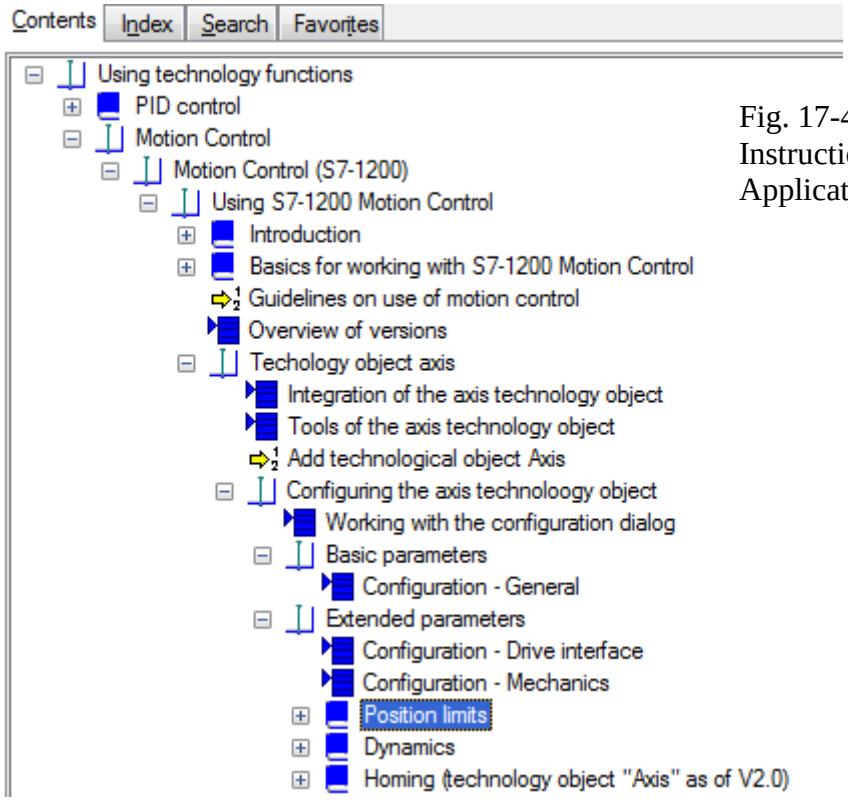


Fig. 17-49a
Instructions used in the Motion Application Help Screens

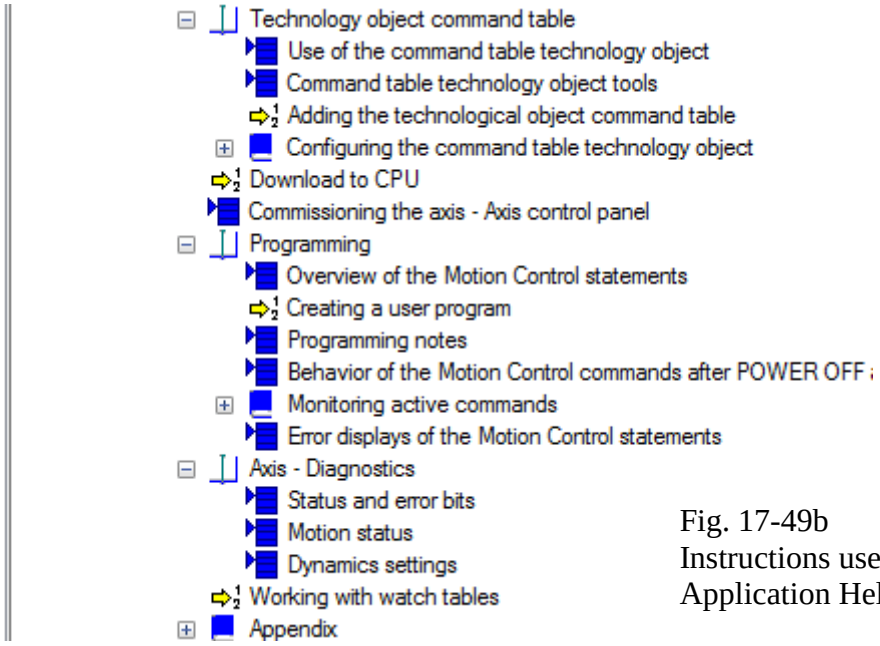


Fig. 17-49b
Instructions used in the Motion Application Help Screens

The following instructions are tied to inputs from the switched inputs directly wired to the PLC. Each input executes a specific action. For instance, I0.0, the first input, executes a drive reset instruction per Fig. 17-50 below

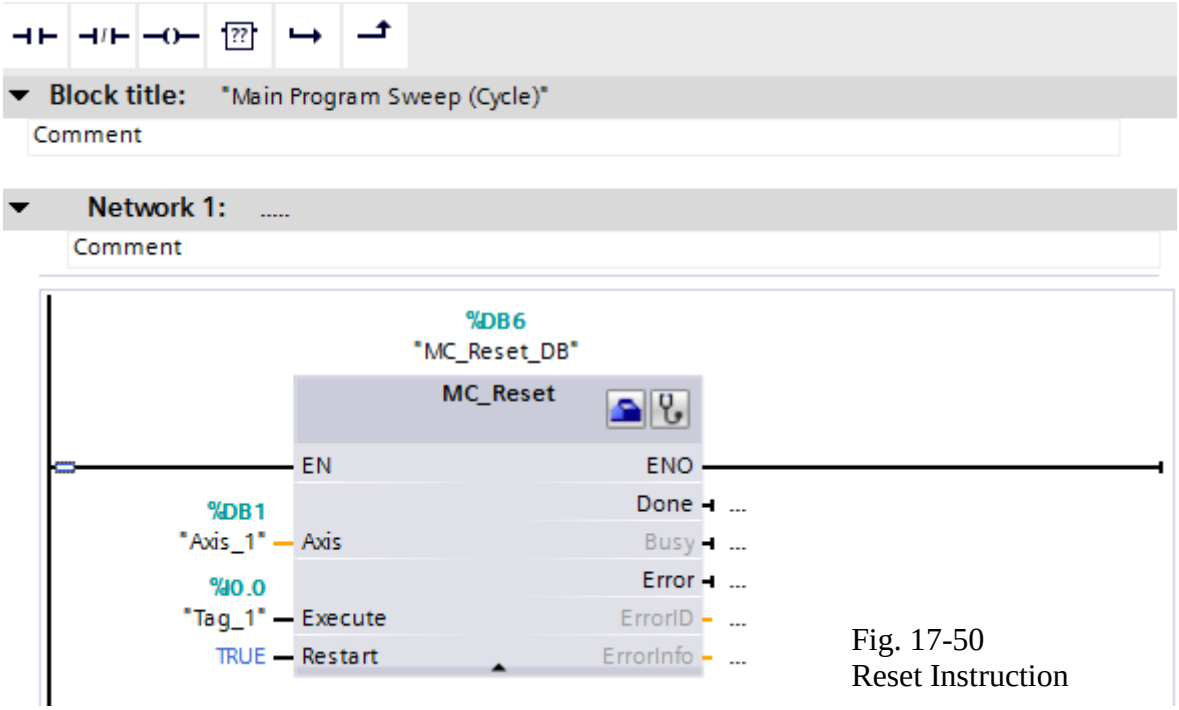


Fig. 17-50
Reset Instruction

The Reset Instruction is used to reset the axis. It is referenced to I0.0 which is the first input on the switch panel on top of the PLC. Use this switch input to reset the axis.

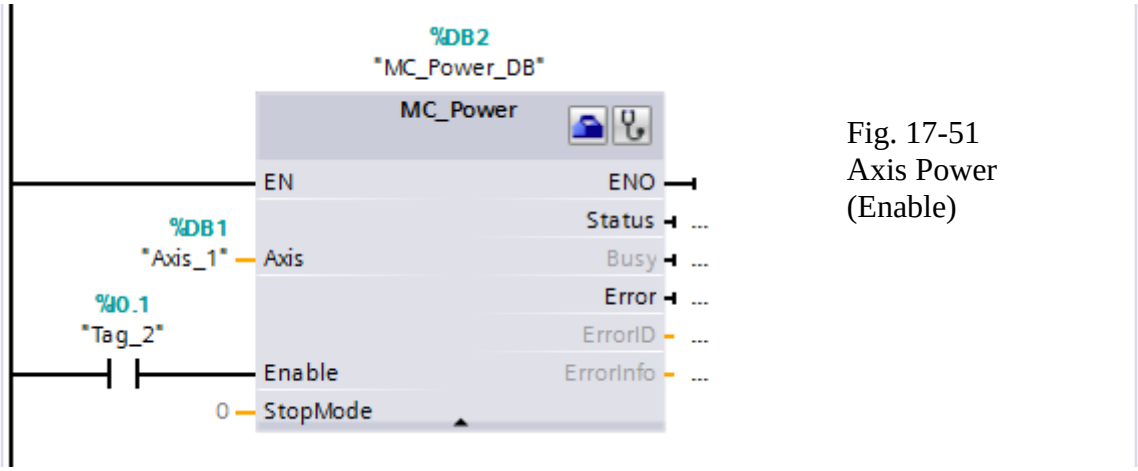


Fig. 17-51
Axis Power
(Enable)

The Power Instruction is used to enable the axis. It is referenced to I0.1 which is the second input on the switch panel on top of the PLC. Use this switch input to enable the axis.

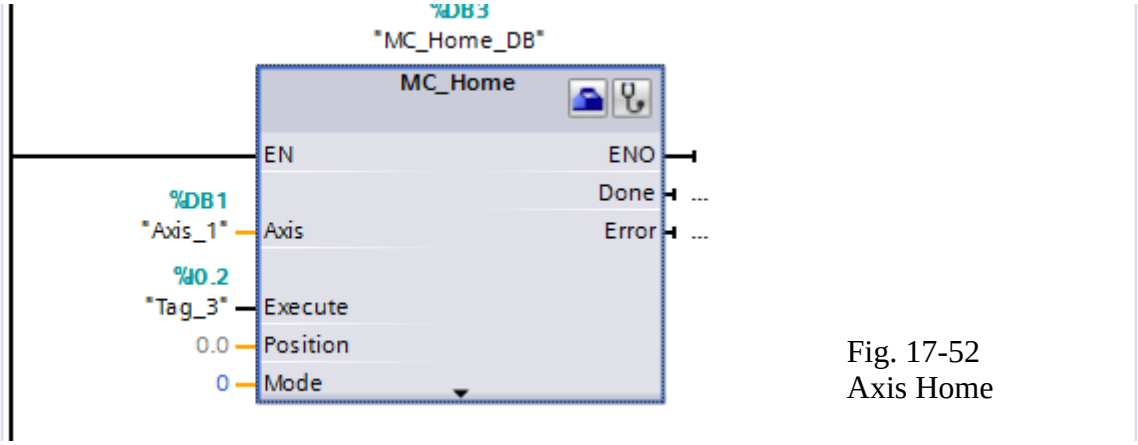


Fig. 17-52
Axis Home

The Home Instruction is used to home the axis. It is referenced to I0.2 which is the third input on the switch panel on top of the PLC. If there are no home limit switches, the present position is used as the home position and absolute moves can be entered following the Home block being executed

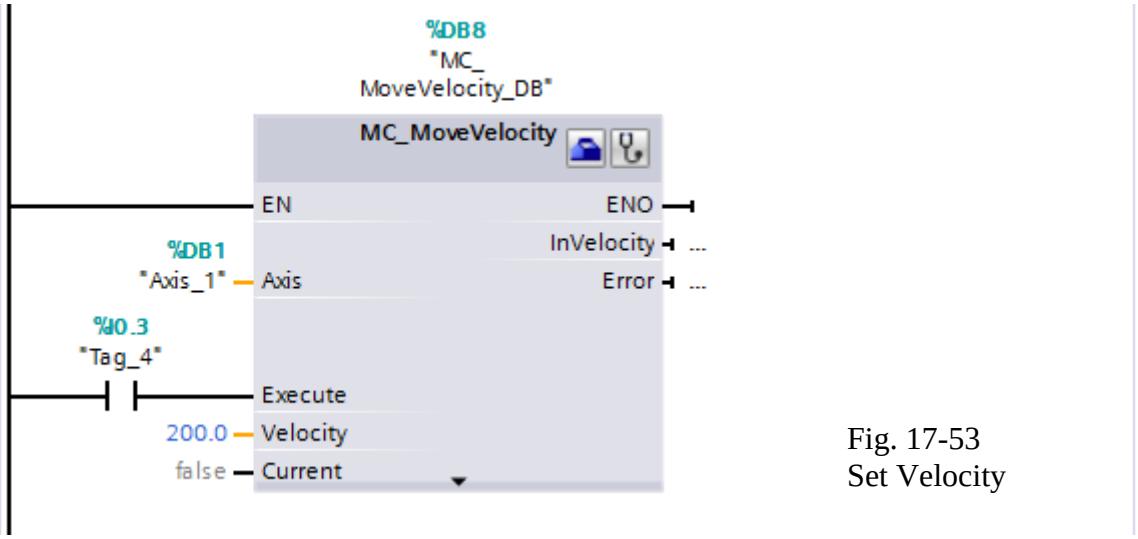


Fig. 17-53
Set Velocity

The Move Velocity Instruction is used to set the velocity of the axis. It is referenced to I0.3.

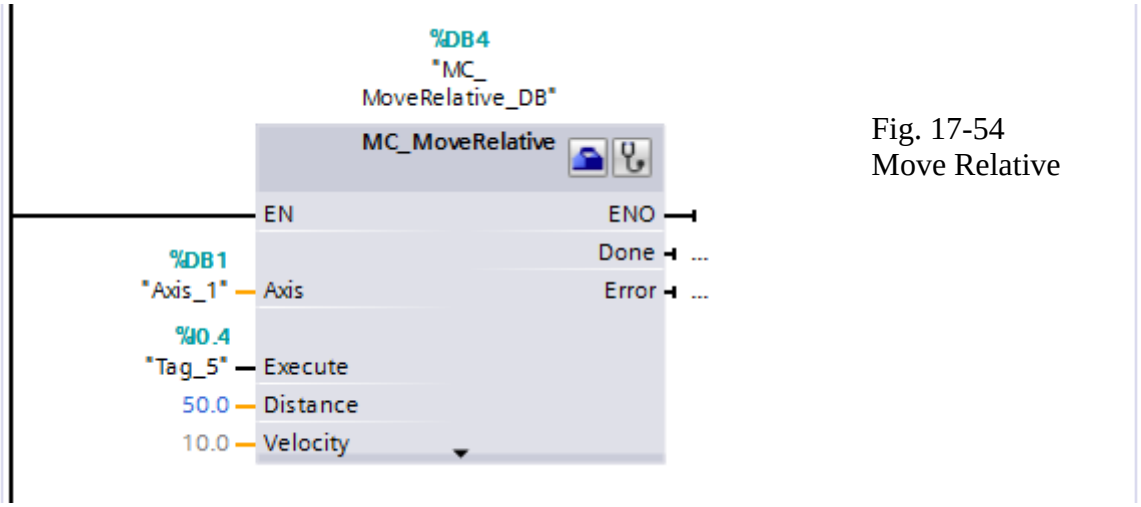


Fig. 17-54
Move Relative

The Move Relative Instruction is used to trigger a relative move of the axis. It is referenced to I0.4.

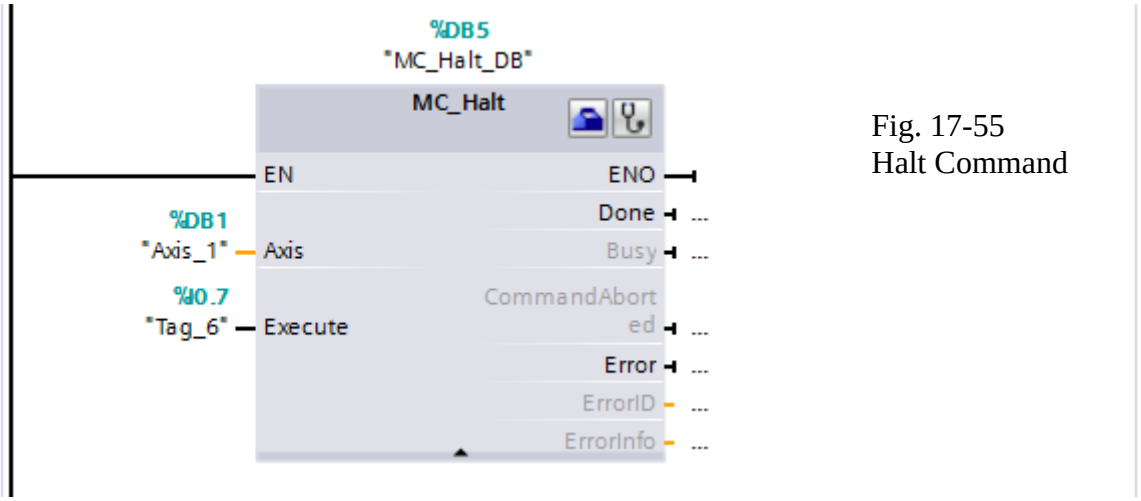


Fig. 17-55
Halt Command

The Halt Instruction is used to halt a move of the axis. It is referenced to I0.7.

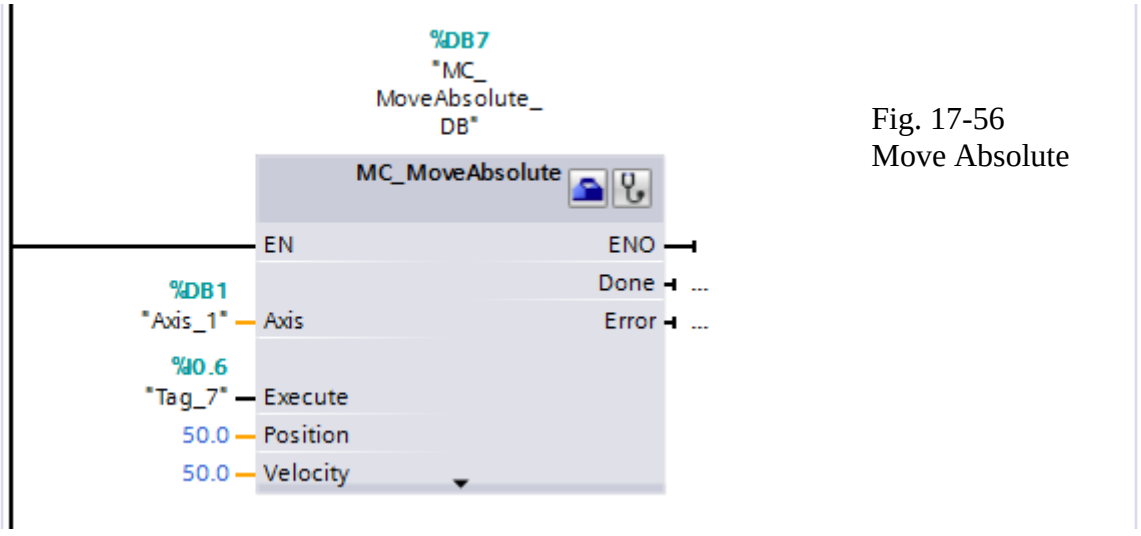


Fig. 17-56
Move Absolute

The Move Absolute Instruction is used to trigger an absolute move of the axis. It is referenced to I0.6. It will move the axis to a position relative to the home instruction.

In addition to the discrete instruction listed above, a command table may be used to store the commands of a sequence of move commands for the Siemens processor. The same command table can be generated for Allen-Bradley although this feature was never completely made to work by the instructor or lab tech in the course to date. It would be a challenge for a student to execute the command sequence for both drives using that feature from the manufacturer's software.

Allen-Bradley's Kinetix 350 Single-axis Servo Drive

The focus changes to A-B and a motion application with servo drives. The information that is condensed here is found to a large part in the User Manual for this product. Information such as parameter lists will be deferred to the manual and not duplicated here.

Some tables will be repeated here for ease of start-up:

Table 30 – Status Display Information

Status Indicator	Description
Hx.xx	Hardware revision. For example H2.00
Fx.xx	Firmware revision. For example F2.06.
dHCP	Ethernet DHCP Configuration: 0 = dHCP is disabled; 1 = dHCP is enabled
IP_1	Allows modification of the first octet of the IP address
IP_2	Allows modification of the second octet of the IP address
IP_3	Allows modification of the third octet of the IP address
IP_4	Allows modification of the fourth octet of the IP address
nEt1	Allows modification of the first octet of the netmask
nEt2	Allows modification of the second octet of the netmask
nEt3	Allows modification of the third octet of the netmask
nEt4	Allows modification of the fourth octet of the netmask
gat1	Allows modification of the first octet of the gateway
gat2	Allows modification of the second octet of the gateway
gat3	Allows modification of the third octet of the gateway
gat4	Allows modification of the fourth octet of the gateway

Table 32 – Module State Status Indicator

Status Indicator	State
Off	Power off
Flash red/green	Drive self-testing
Flashing green	Standby
Solid green	Operational
Flashing red	Major recoverable fault
Solid red	Major unrecoverable fault

Table 33 – Axis State Status Indicator

Status Indicator	State
Off	Off
Flash red/green	Self-test
Off	Initialization – bus not up
Flashing green	Initialization – bus up
Off	Shutdown – bus not up
Flashing amber	Shutdown – bus up
Off	Pre-charge – bus not up
Flashing amber	Start inhibit
Flashing green	Stopped
Solid green	Stopping
	Starting
	Running
	Testing
Flashing red	Aborting
	Major fault
Solid red	Aborting
	Major fault

The axis and the drive define minor fault conditions. While a minor fault does not affect the drive status indicator, it does affect the axis status indicator. When a minor fault condition is detected, a normally solid green status indicator indication changes to alternating red-green-red-green, a normally flashing green status indicator indication changes to alternating red-off-green-off, and a normally flashing amber indications changes to red-off-amber-off.

The drive also defines alarm conditions. When an alarm condition is detected, a normally solid green status indicator indication changes to alternating amber-green-amber-green while a normally flashing green status indicator indication changes to alternating amber-off-green-off.

Table 34 –Network State Status Indicator

Status Indicator	State
Steady off	Not powered, no IP address
Flashing green	No connections
Steady green	Connected
Flashing red	Connection time-out
Steady red	Duplicate IP
Flashing green and red	Self-test

Fig. 17-57
The A-B Kinetix Drive

Fig. 17-58a The L30 Processor is Chosen to Drive the Single Axis Servo

From A-B, "The IP address of the Kinetix 350 drive is composed of four sub-octets that are separated by three dots to conform to the Class C Subnet structure. Each sub-octet can be configured with number between 1 and 254. As shipped from the factory, the default IP address of a drive is 192.168.124.200."

The present IP address can be obtained from the drive's display using the up-down keys and reading the address one sub-octet at a time.

A-B states that the drive can be assigned either using DHCP (dynamic IP address) or statically. The drive must be configured statically for our application. You must check that the IP address is already set or ping the address to check if it is operating. There should be an address label on the drive. If checking the drive, use the up-down arrow keys to locate the DHCP parameter and verify that it is set to 0. If not, set to 0 and cycle power.

When using the file given for the course, the controller is configured and ready to run except for the drive's IP address. Configuration of the drive has been accomplished. The next few pages lead one through the process of defining the drive during the configuration process in preparation for the move command programming to follow.

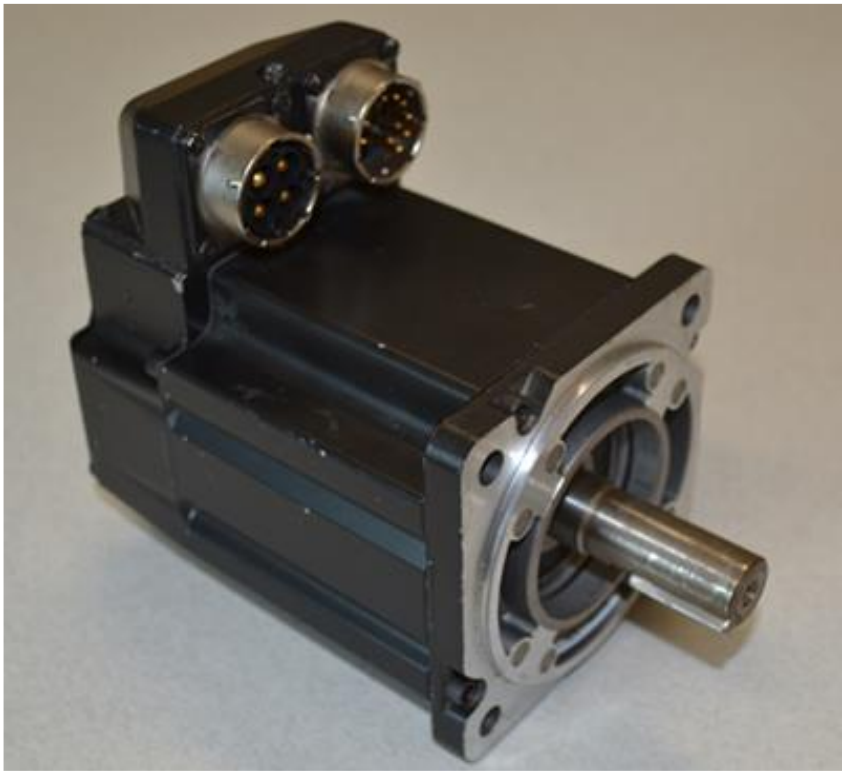


Fig. 17-58b

A-B Servo Motor

Servo Motor with Encoder Shown
This is an Incremental Encoder

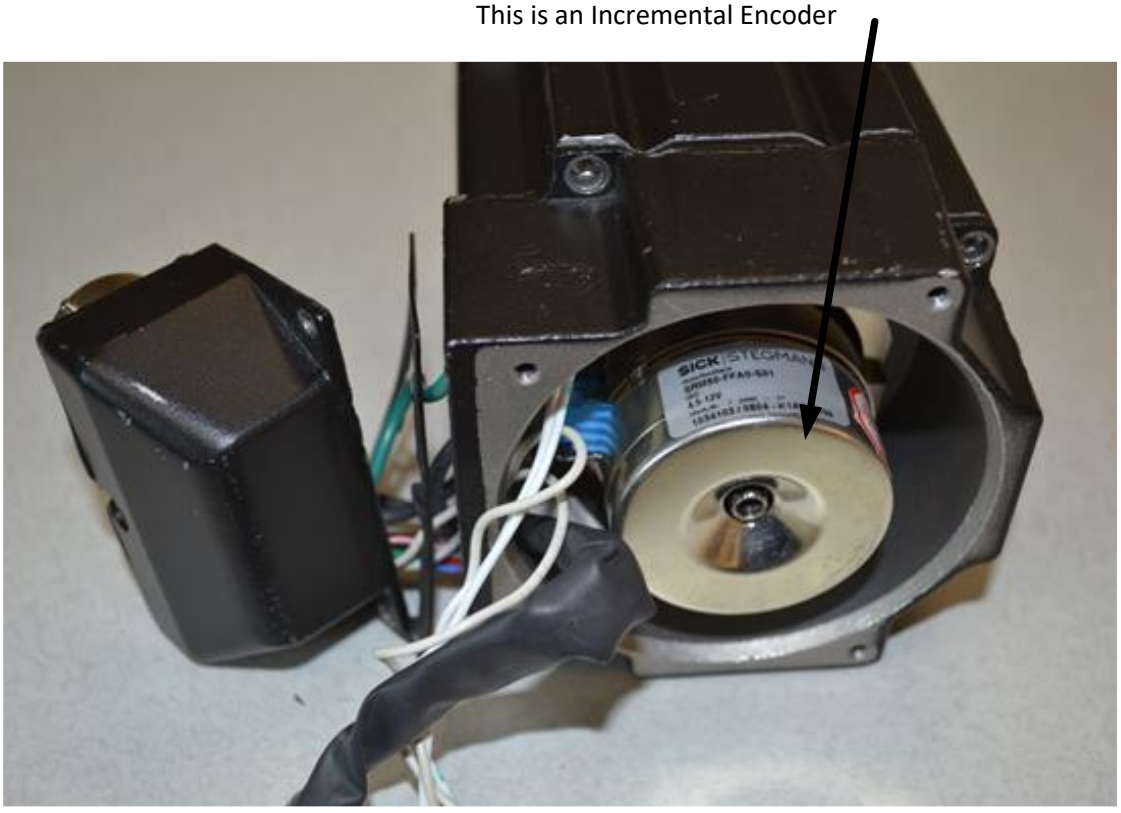


Fig. 17-58c

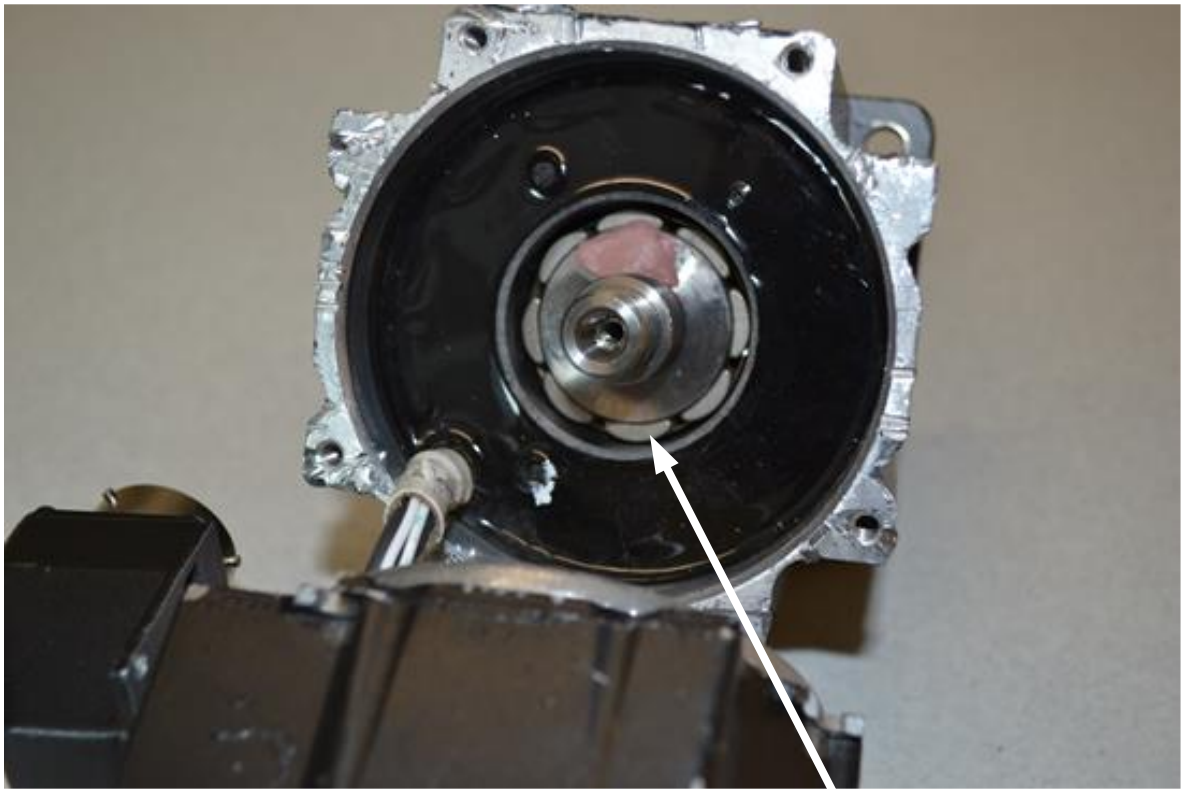


Fig. 17-58d

Servo Motor with rotor shown
Notice the field is electrically wound
and the armature is magnets.

If the controller is not configured, first follow the procedure below:

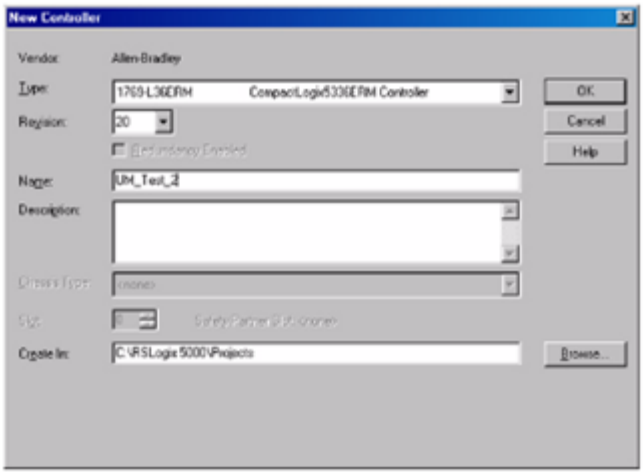


Fig. 17-59

Notice that the controller must be at revision 20 or higher. Under the controller properties dialog box, click the Date/Time tab and enable Time Synchronization.

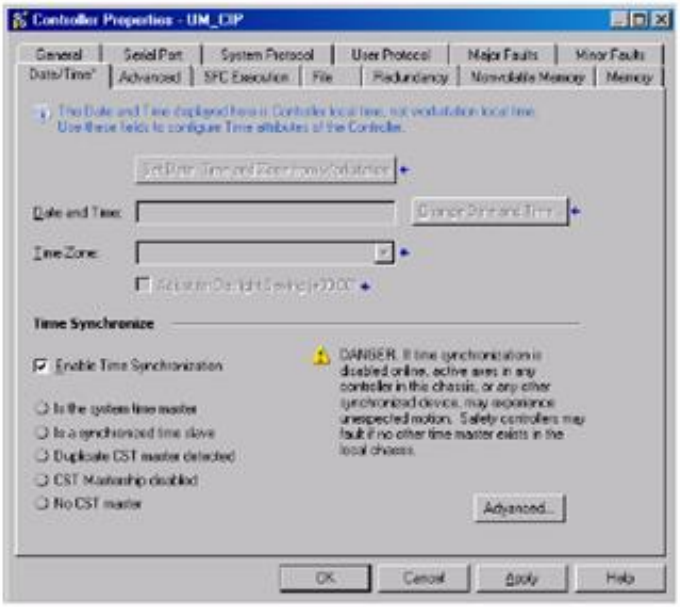


Fig. 17-60

Configure the Kinetix 350 Drive. Right click to create a New Module. Clear the module type filters and check the Drive and Motion categories. Select the appropriate drive.

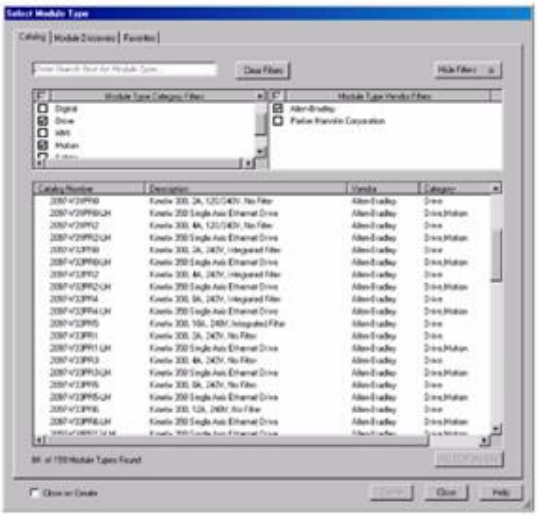


Fig. 17-61

Configure the New Module using the following dialog box. The etnernet address must match the address set for the drive.

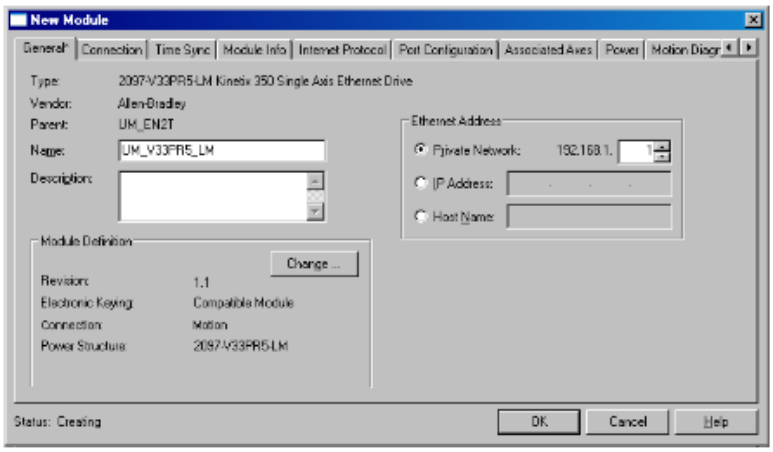


Fig. 17-62

Under Change Module Definition, change the following:

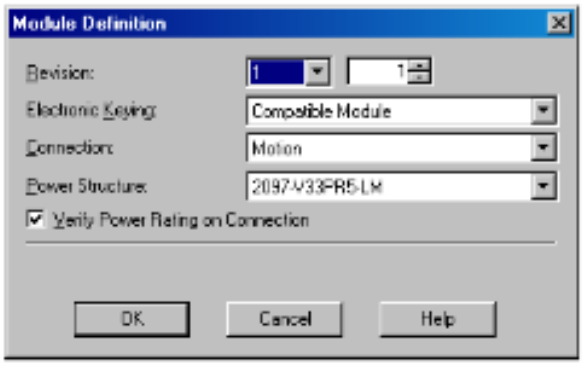


Fig. 17-63

Under the Associated Axes tab, click ‘new axis’ and add information:

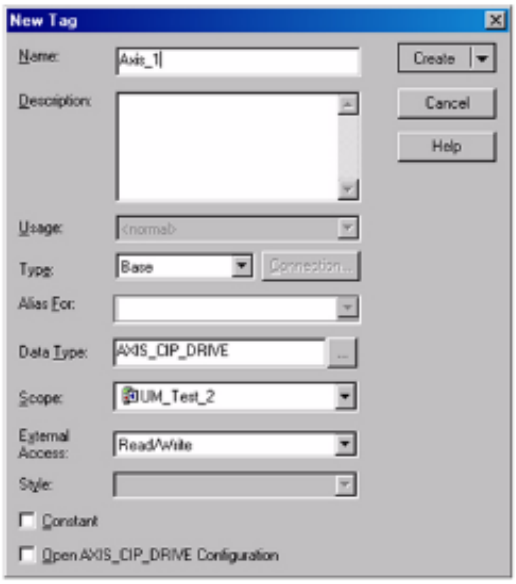


Fig. 17-64

Finish by checking ‘create’. Next configure the Motion Group. In the Controller Organizer, right click Motion Groups and choose New Motion Group. Assign the axis just created to this motion group.

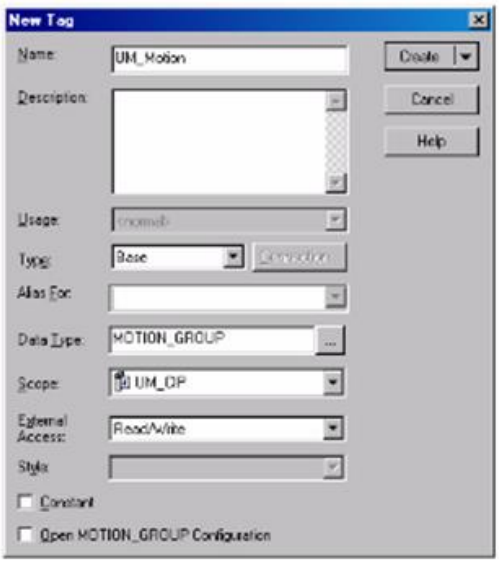


Fig. 17-65

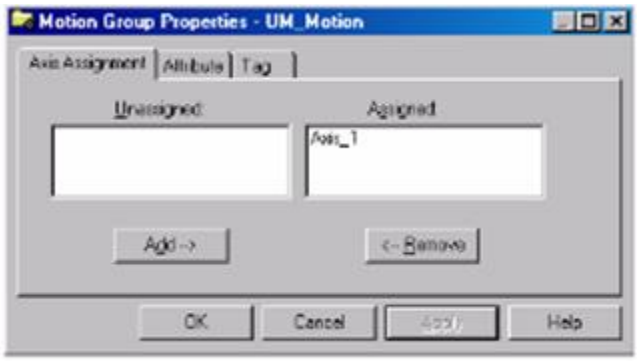


Fig. 17-66

Right click on the axis in the Controller Organizer to change properties of the drive. For the motor:

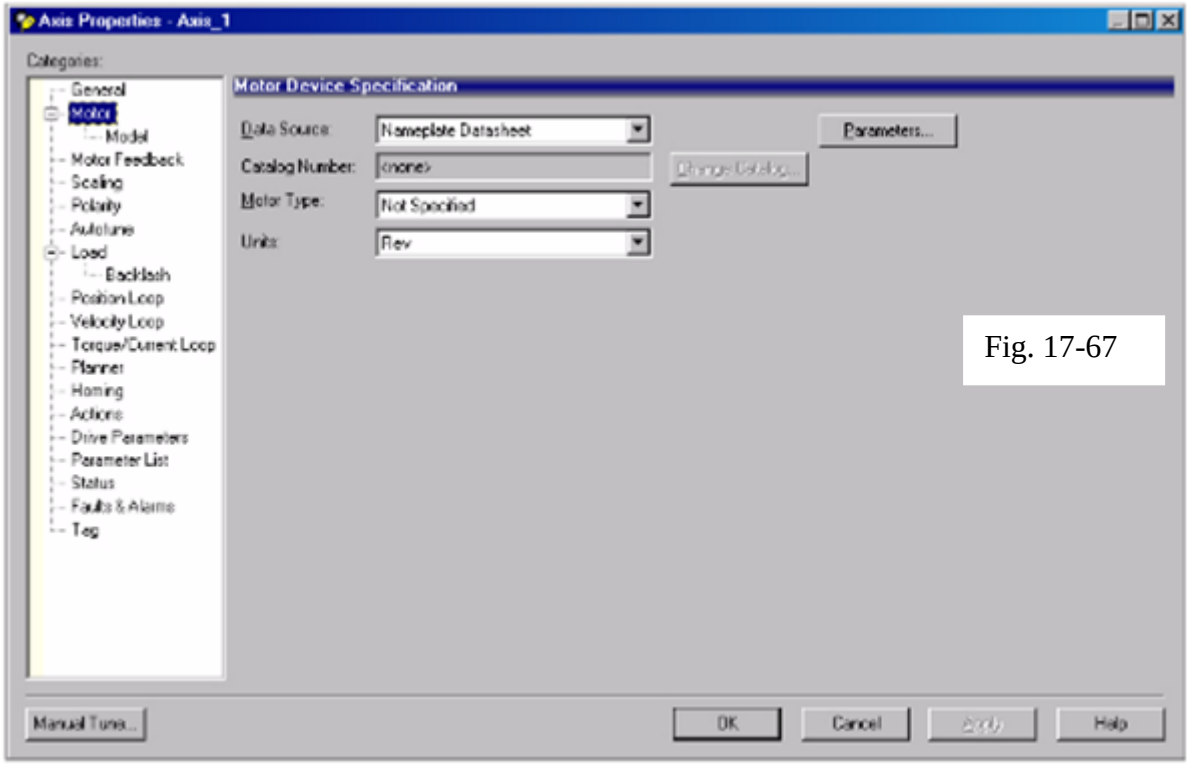


Fig. 17-67

To Configure the Motor Fig. 17-68

Use the scaling and loads appropriate for the application:

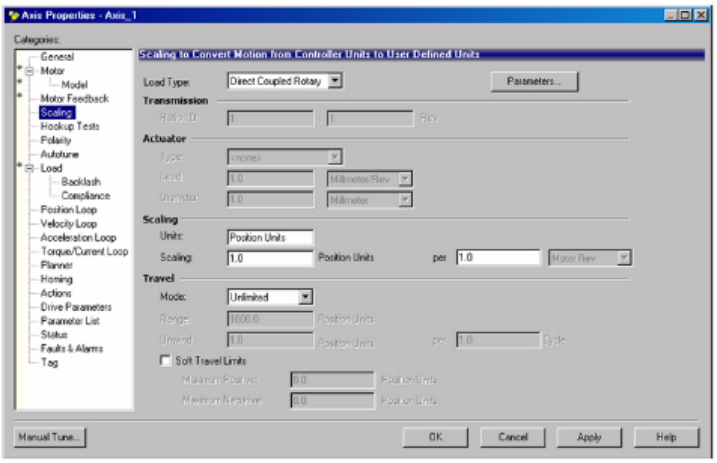


Fig. 17-69

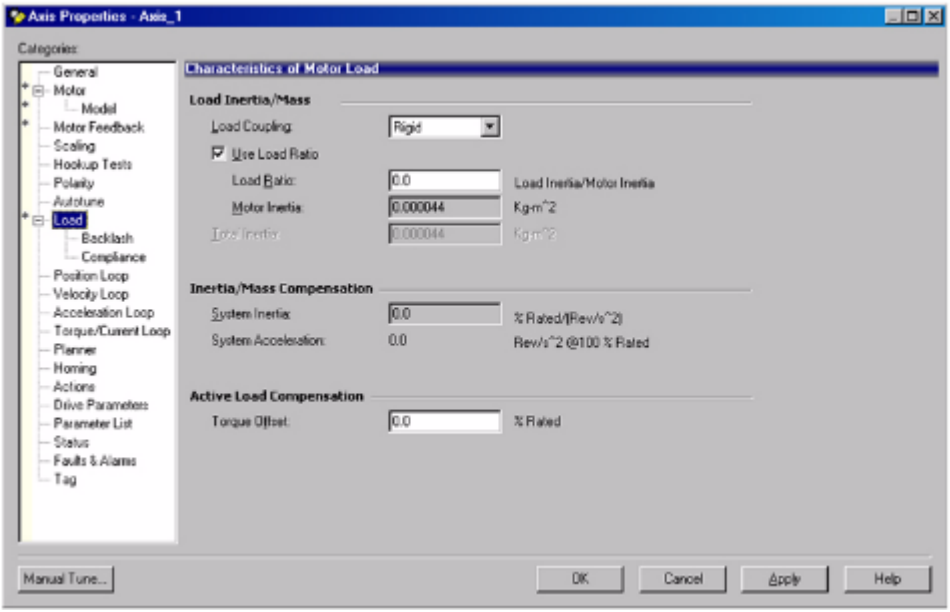


Fig. 17-70

Actions and Parameters:

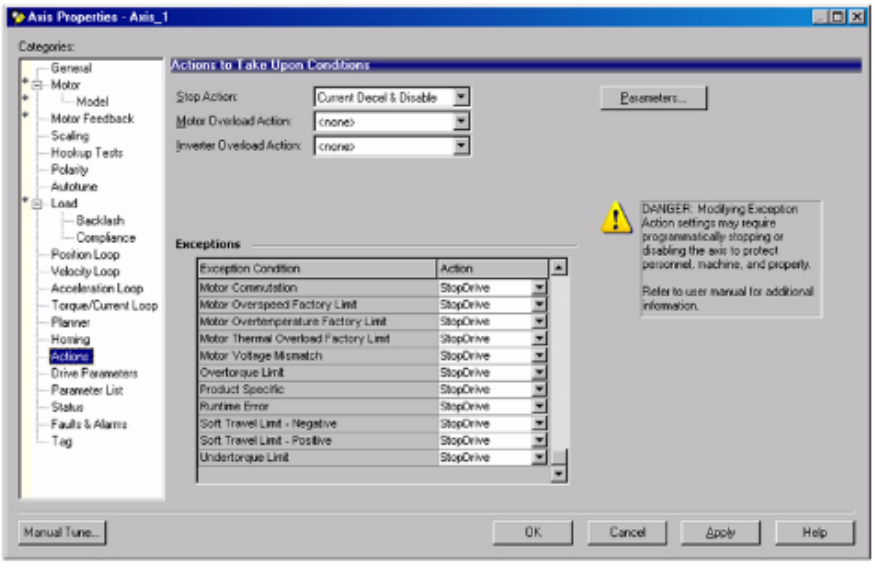


Fig. 17-71

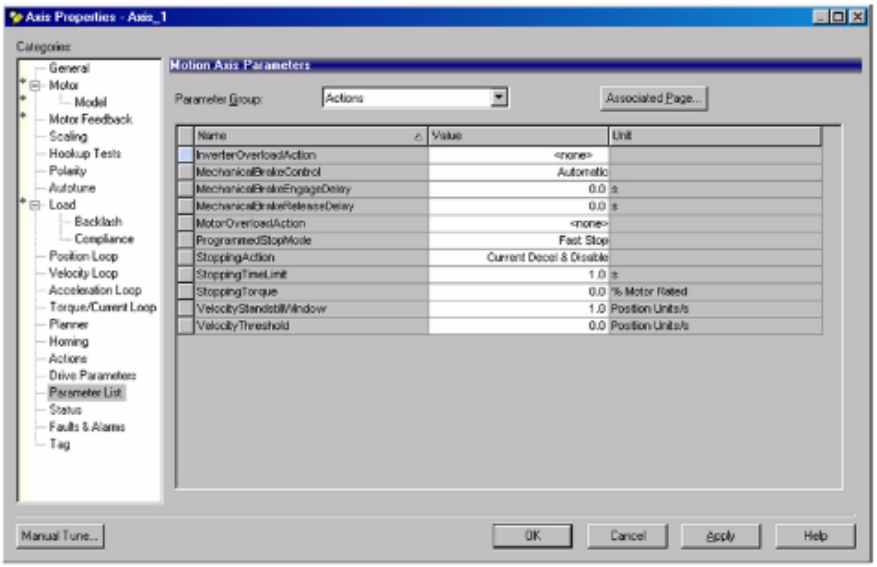


Fig. 17-72

Download the application and test and tune the axes.

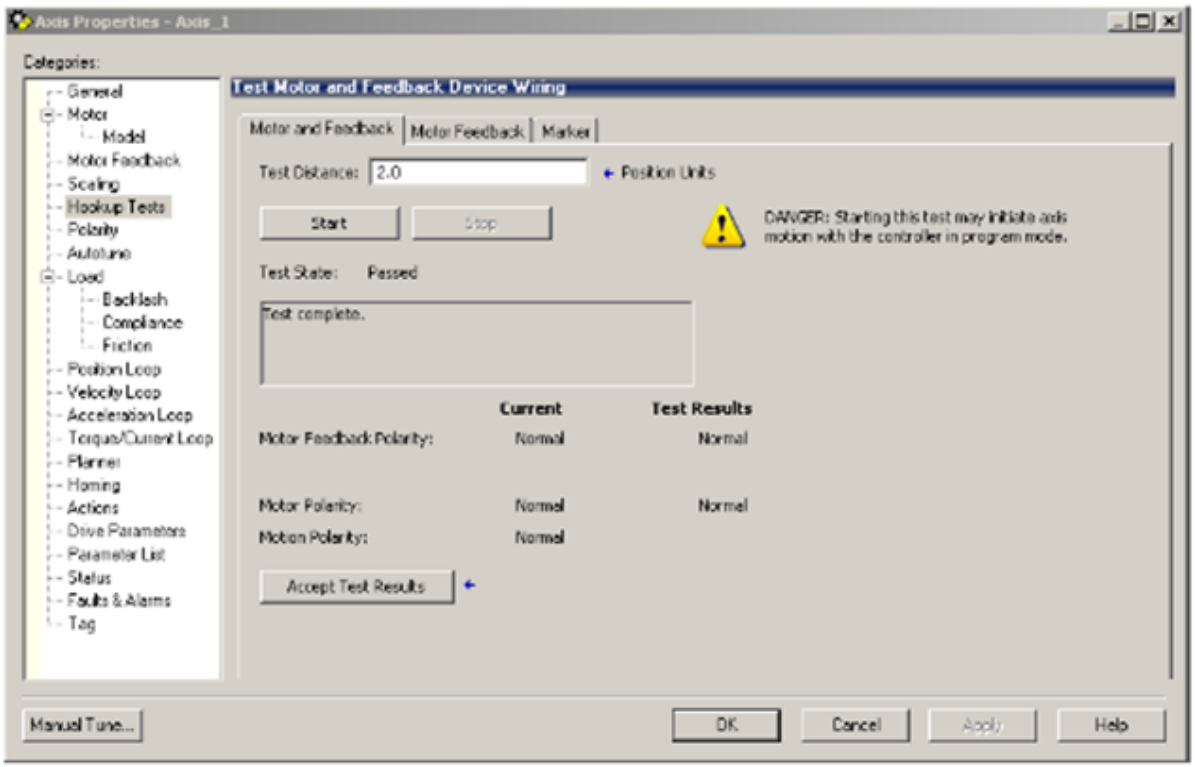
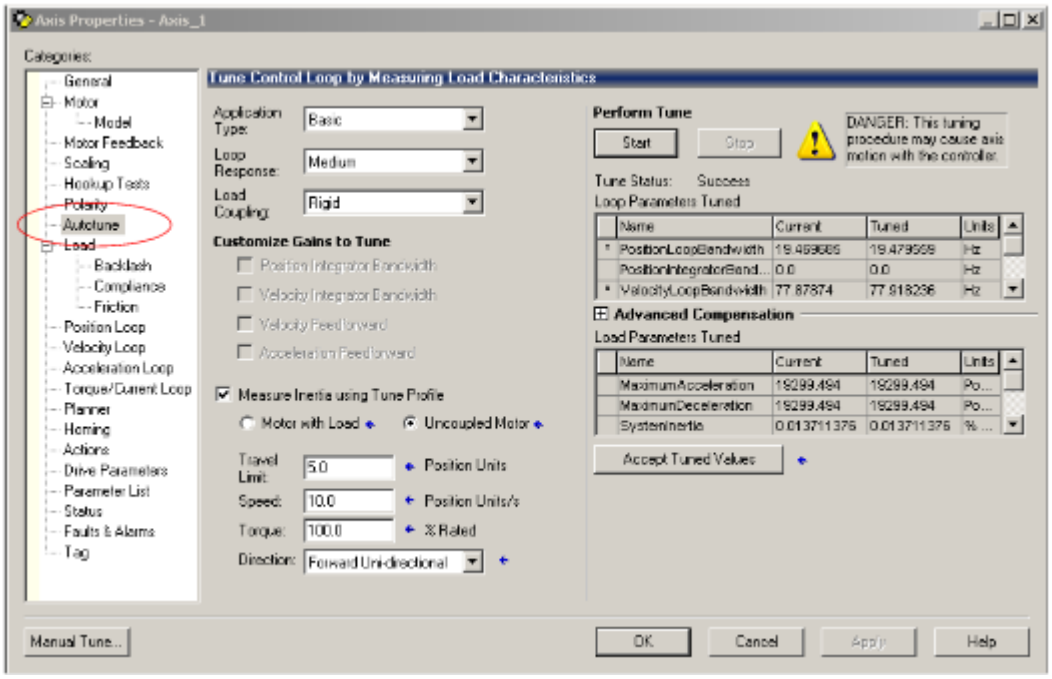
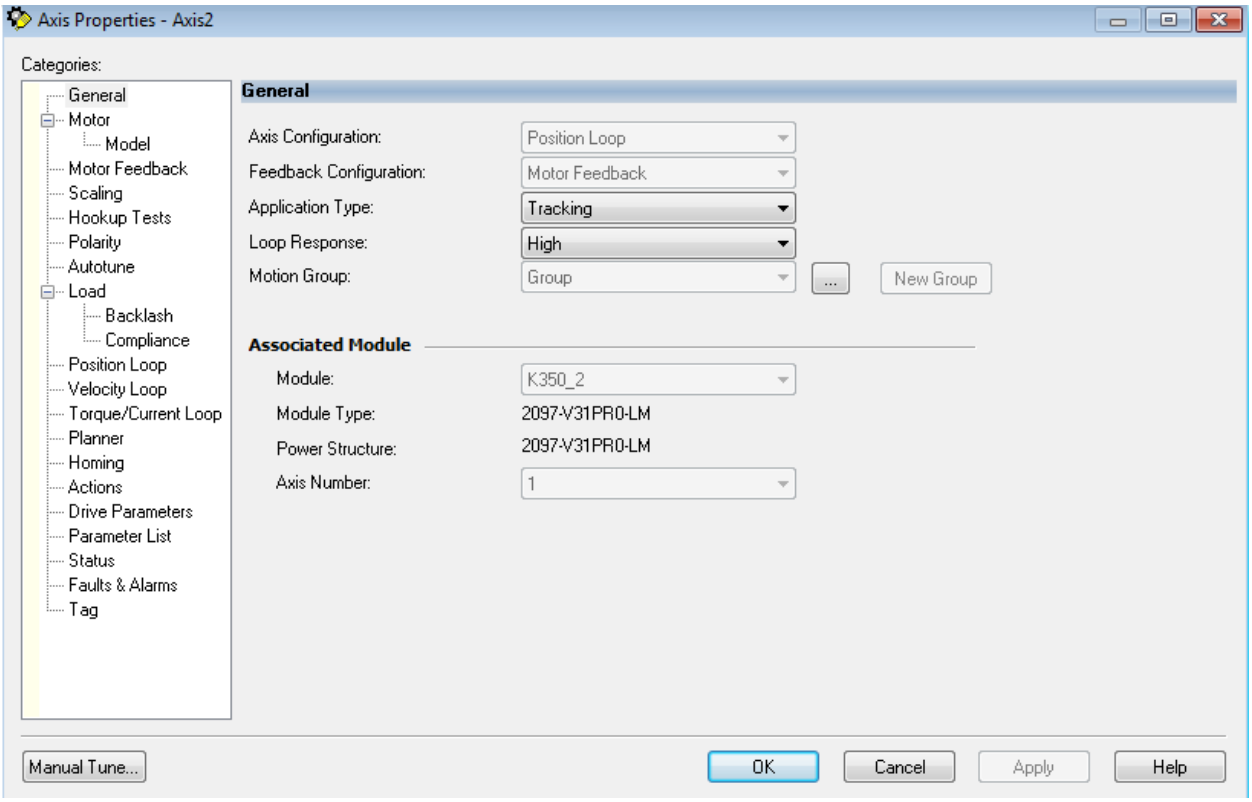


Fig. 17-73

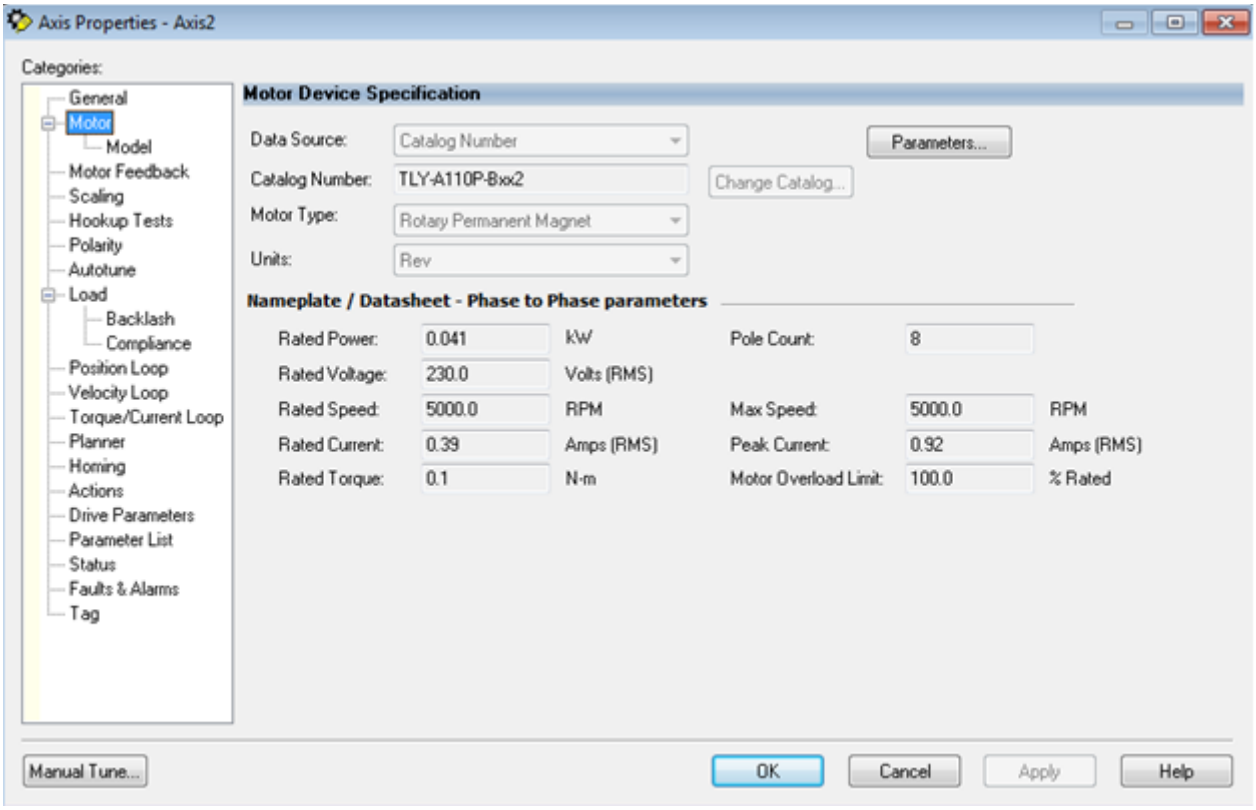


Auto-tune the Drive Fig. 17-74

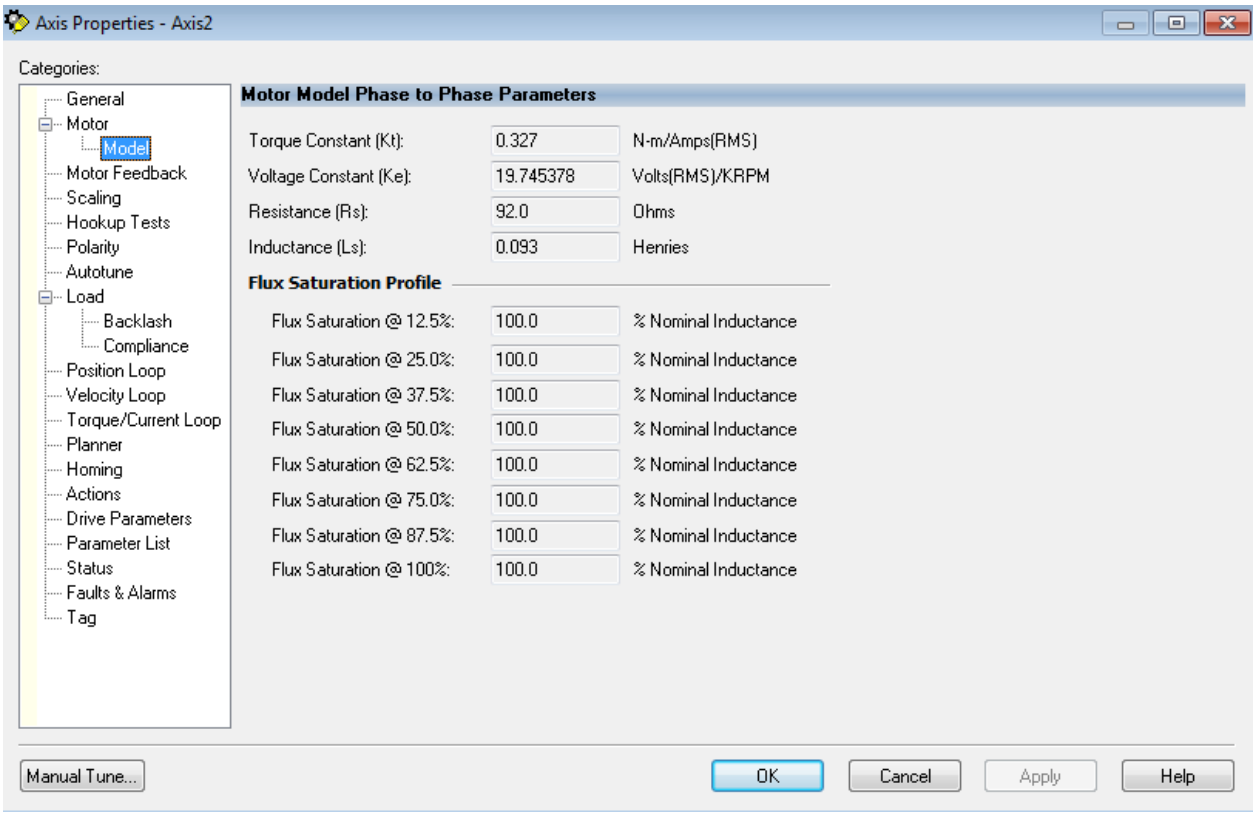
Axis Properties for our application:



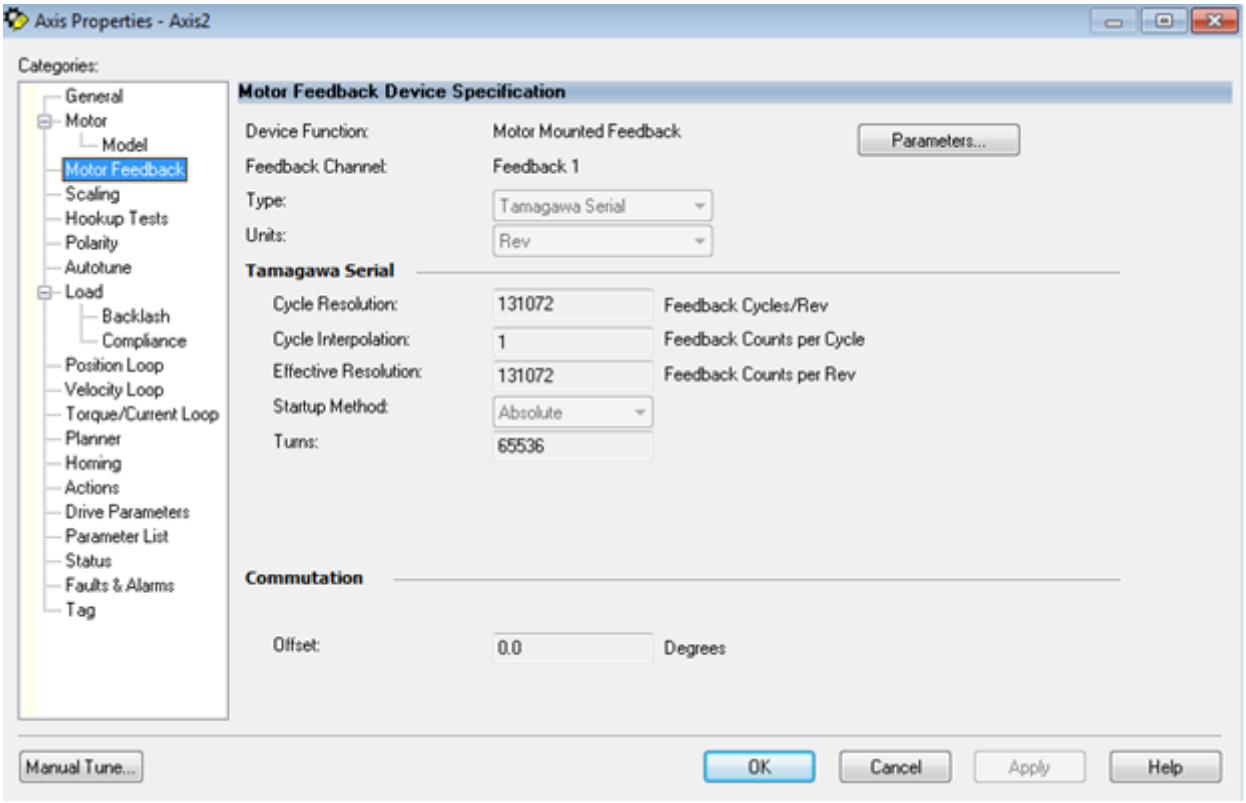
General Axis Properties Fig. 17-75



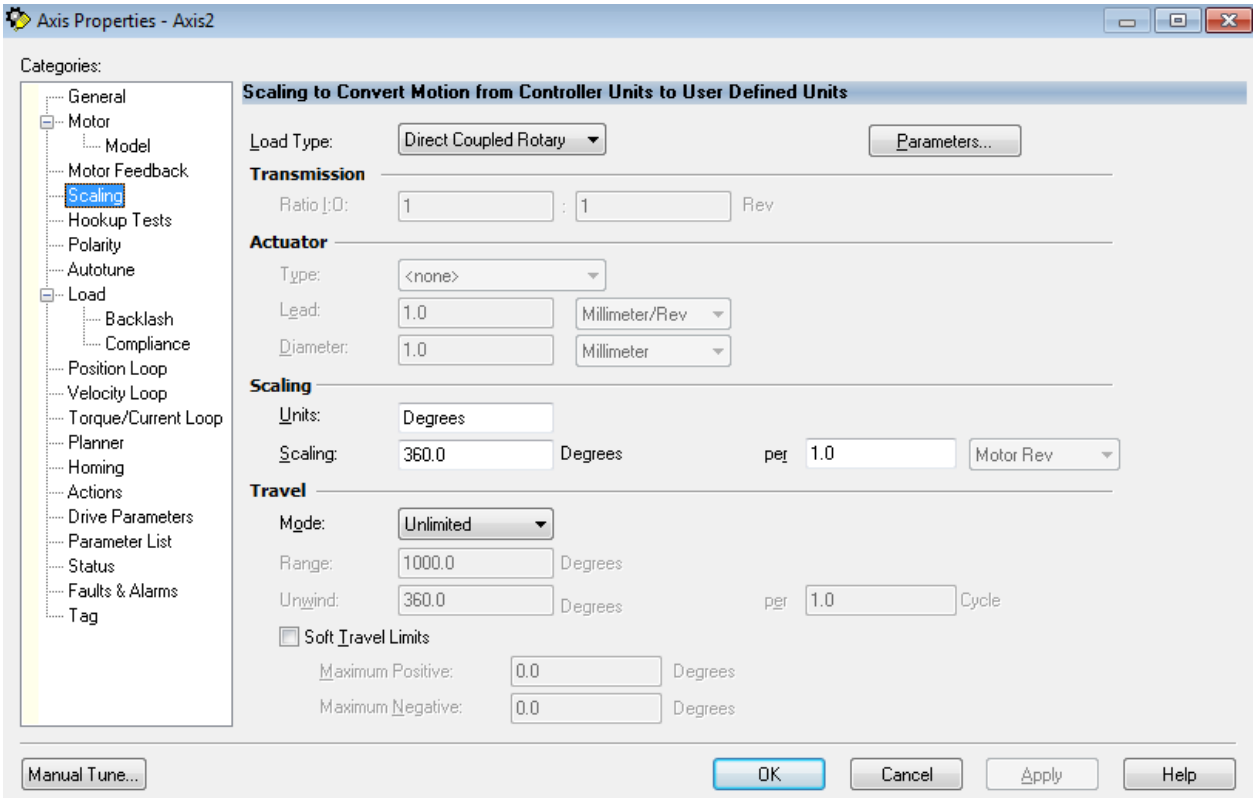
Motor Device Specification Properties Fig. 17-76



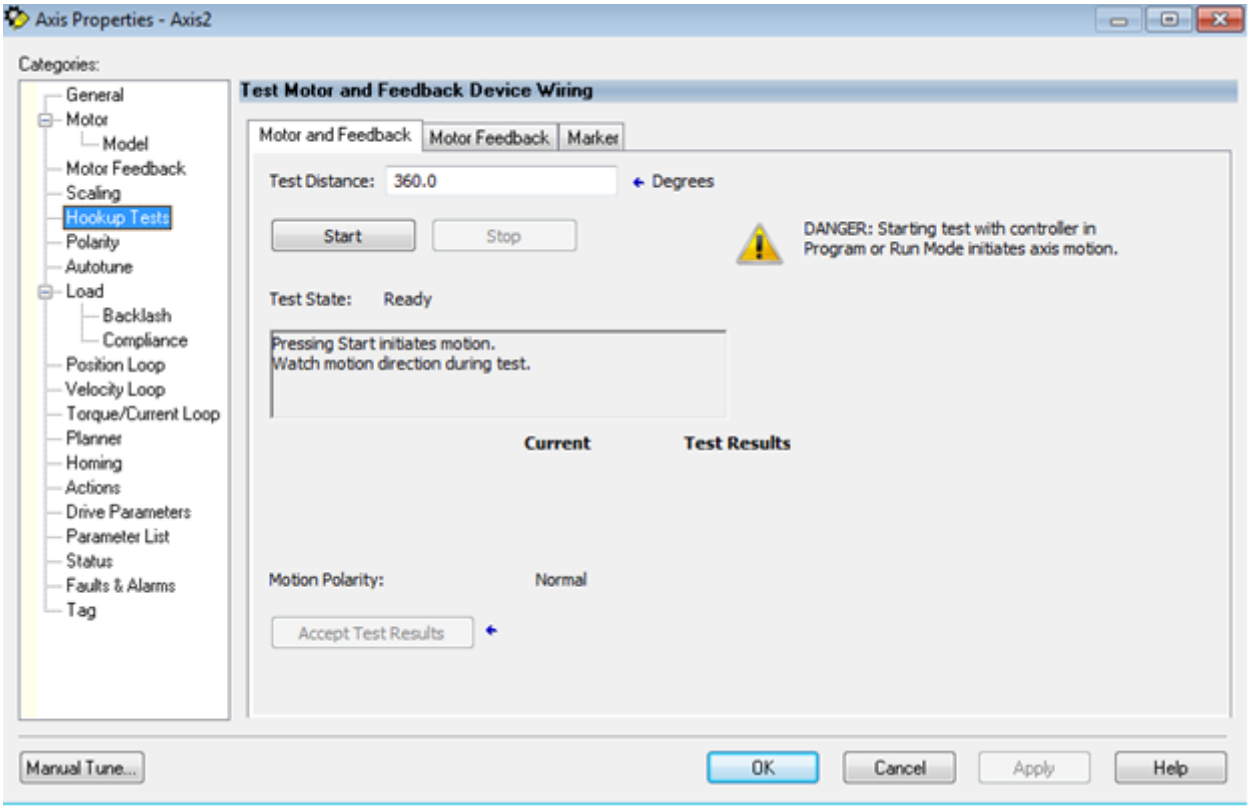
Motor Model Phase to Phase Properties Fig. 17-77



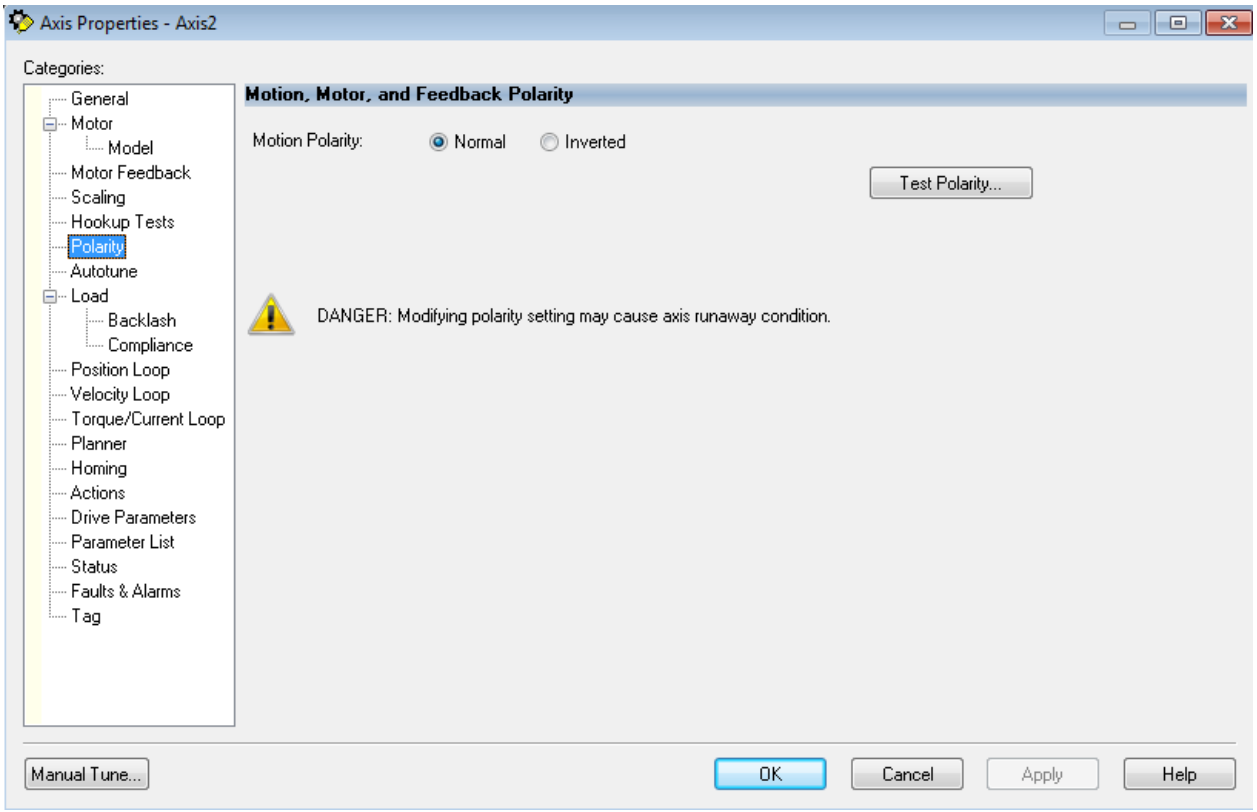
Motor Feedback Device Specification Properties Fig. 17-78



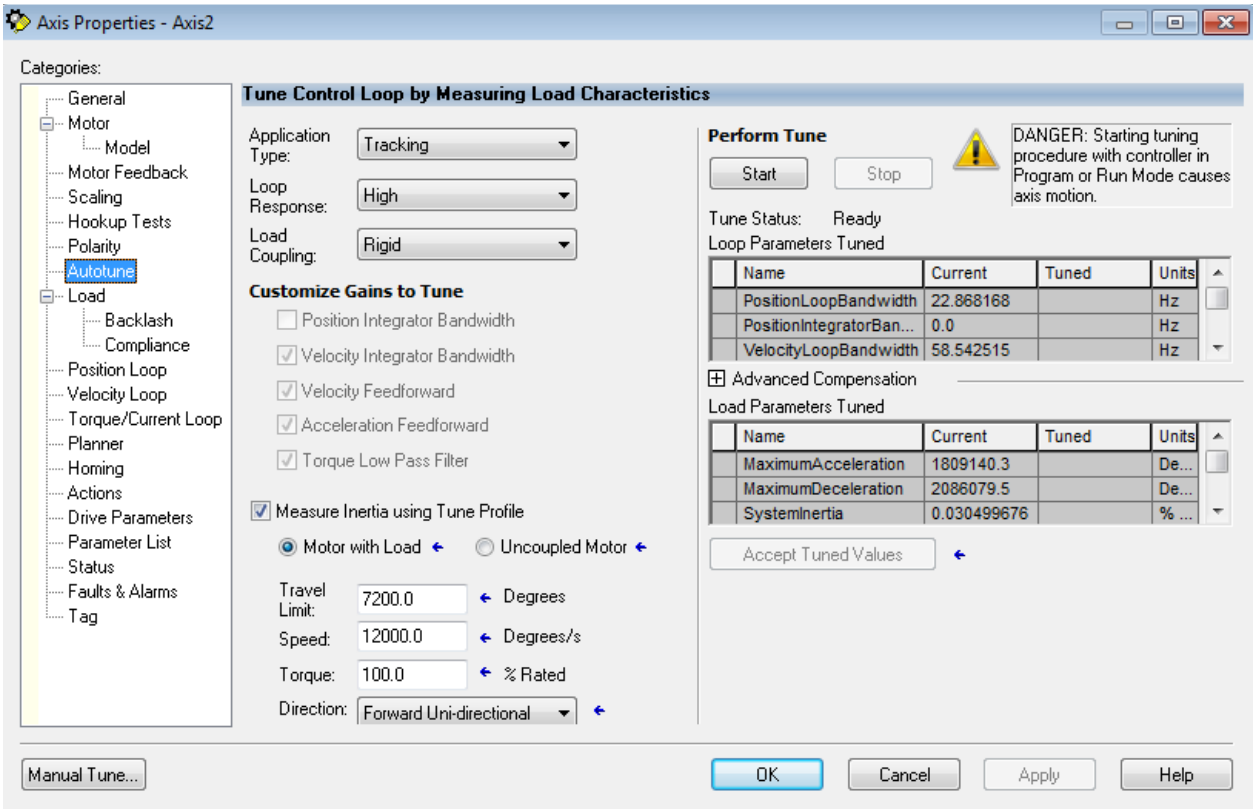
Scaling Parameters Fig. 17-79



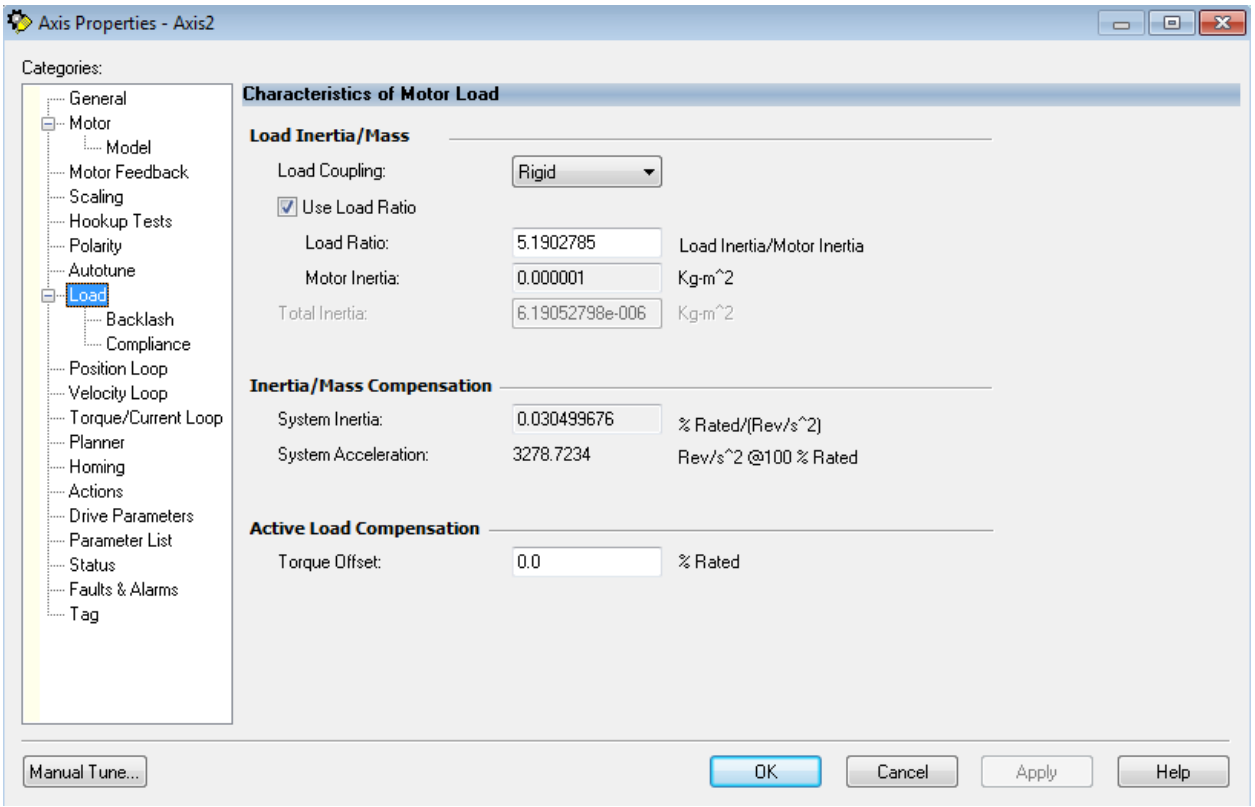
Test Motor Screen Fig. 17-80



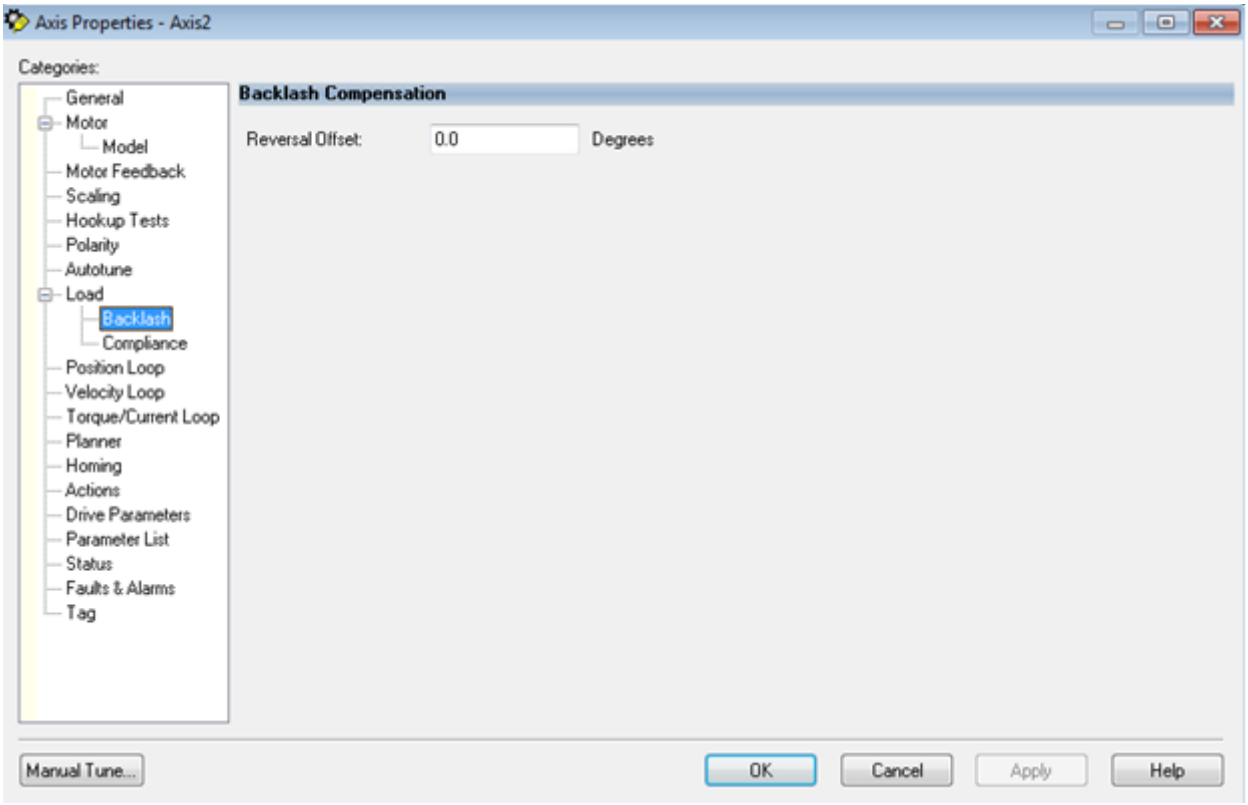
Test Polarity Screen Fig. 17-81



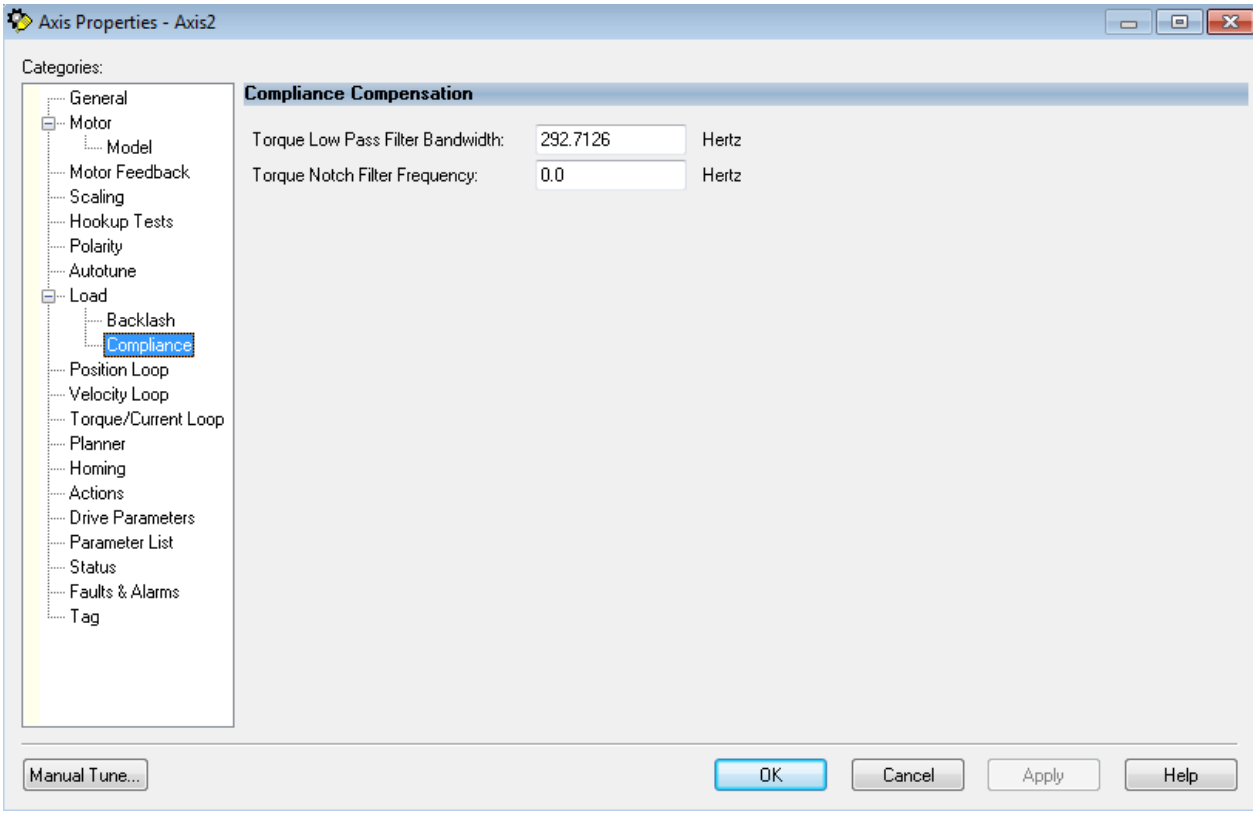
Tuning Parameters Fig. 17-82



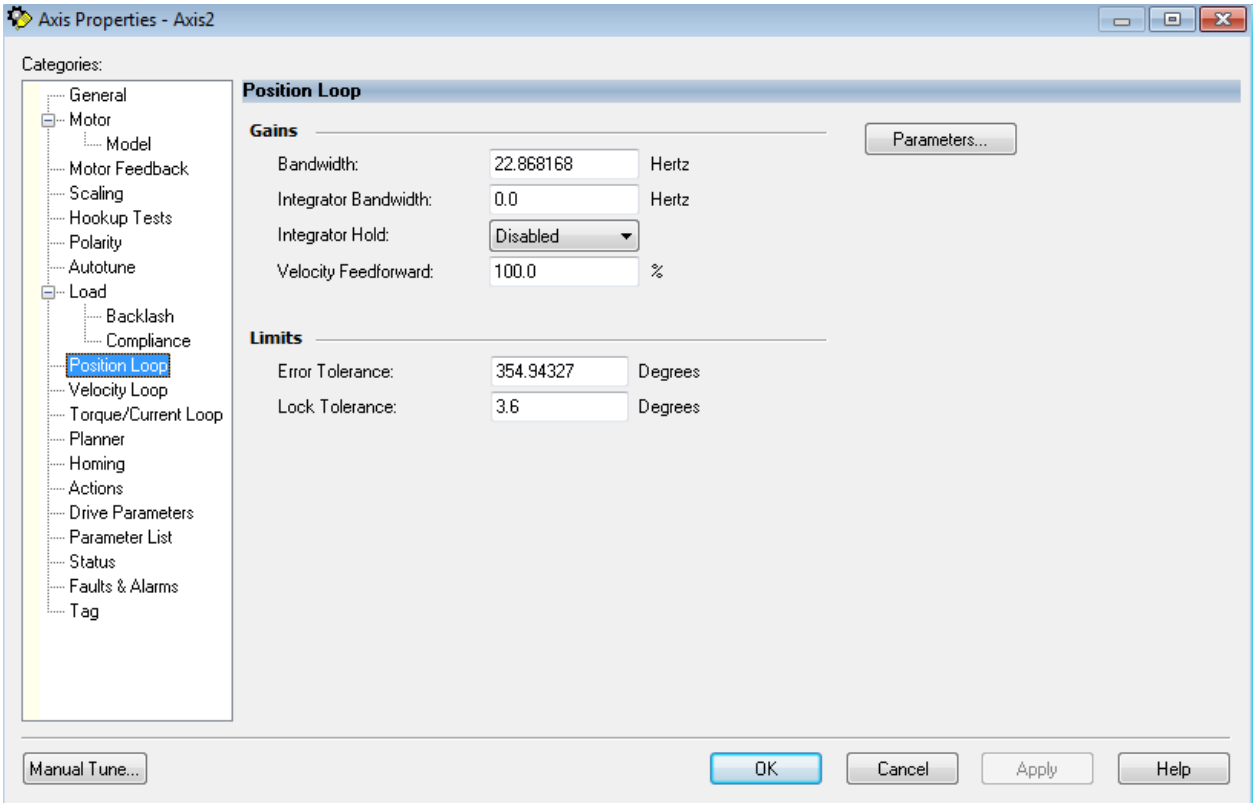
Motor Load Characteristics Fig. 17-83



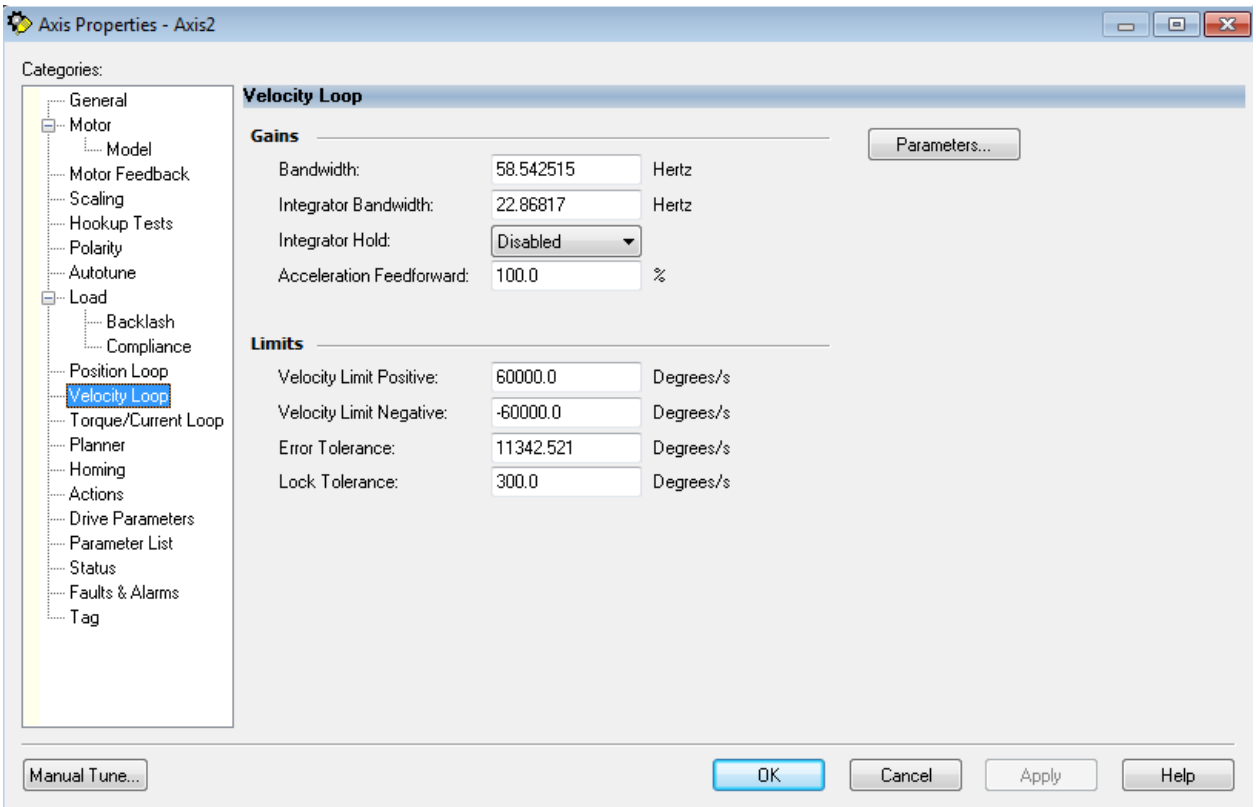
Backlash Compensation Fig. 17-84



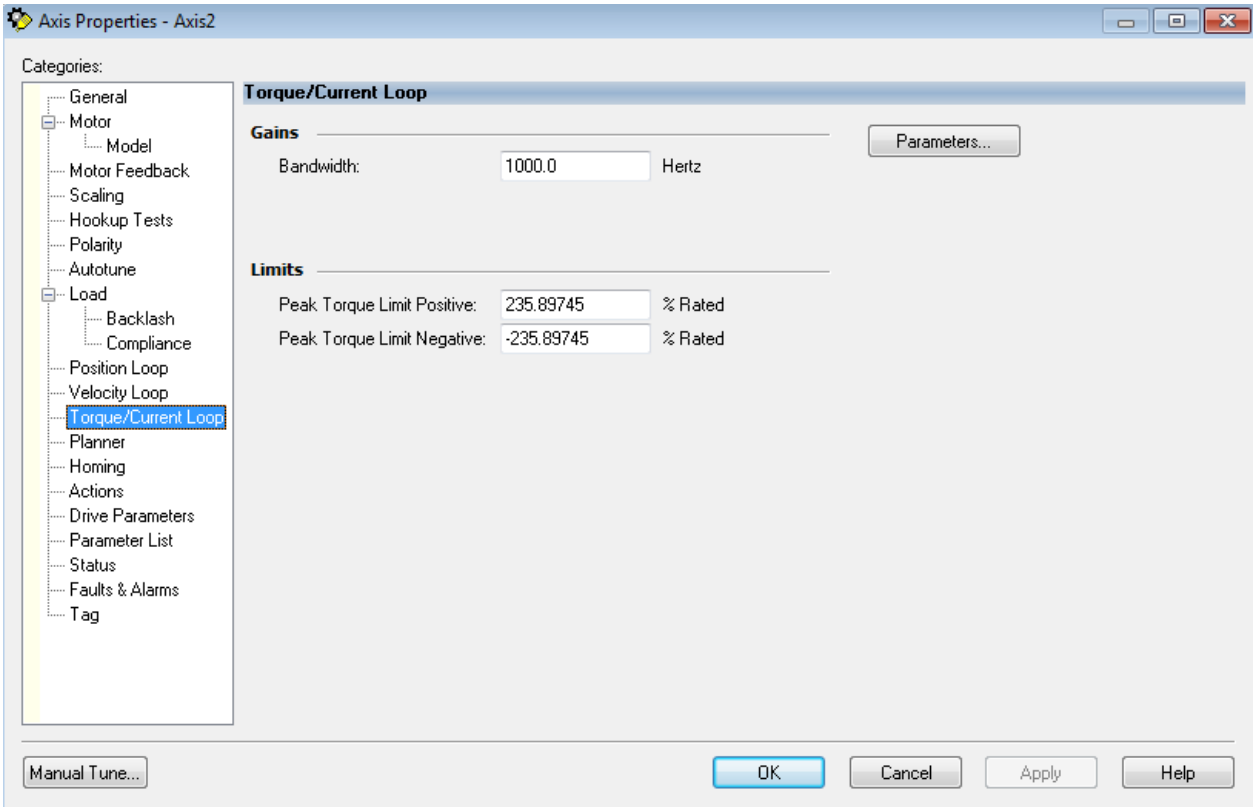
Compliance Compensation Fig. 17-85



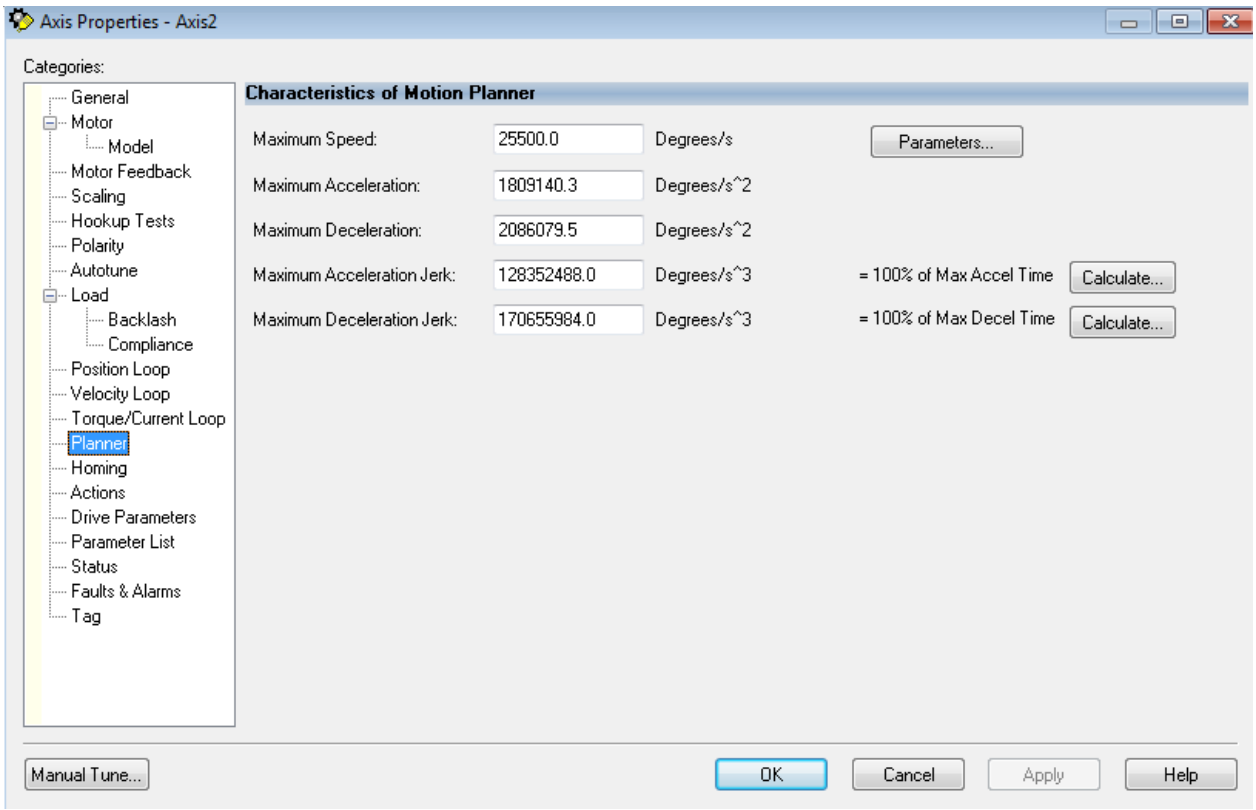
Position Loop Properties Fig. 17-86



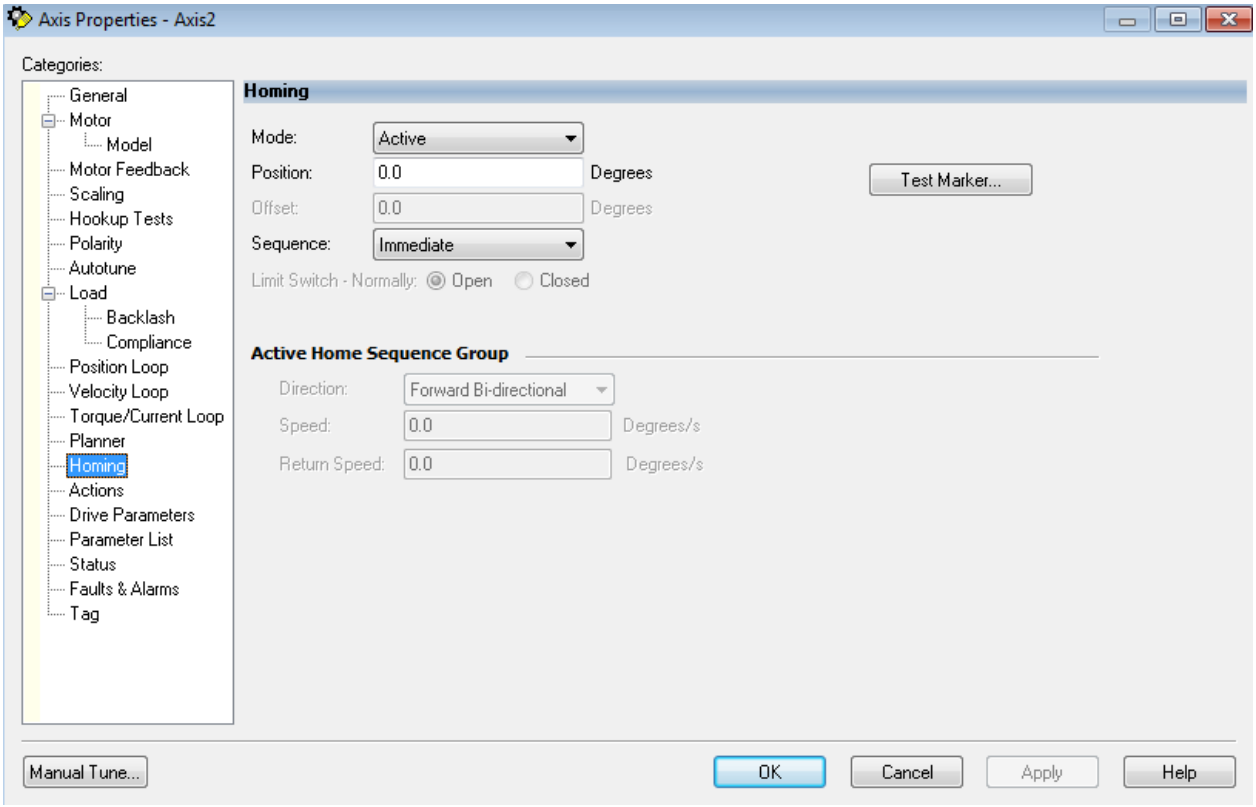
Velocity Loop Properties Fig. 17-87



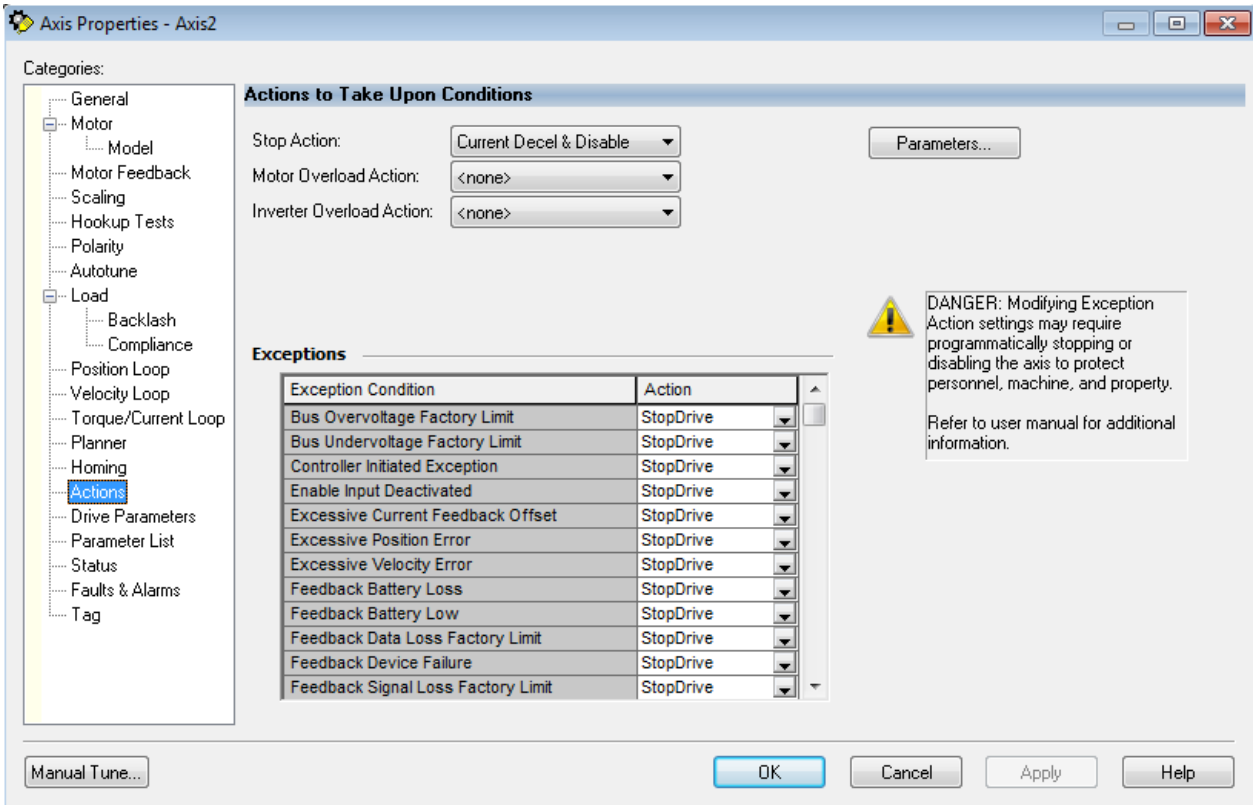
Torque/Current Loop Properties Fig. 17-88



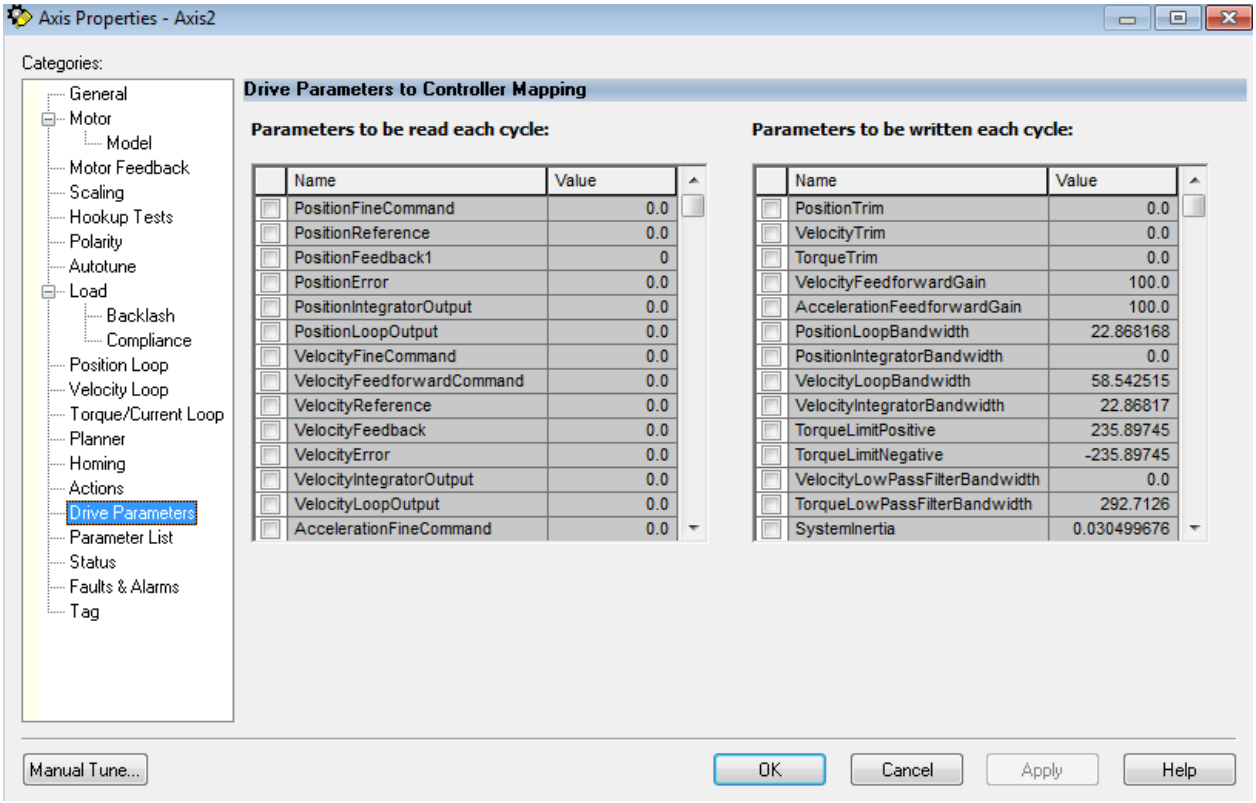
Motion Planner Fig. 17-89



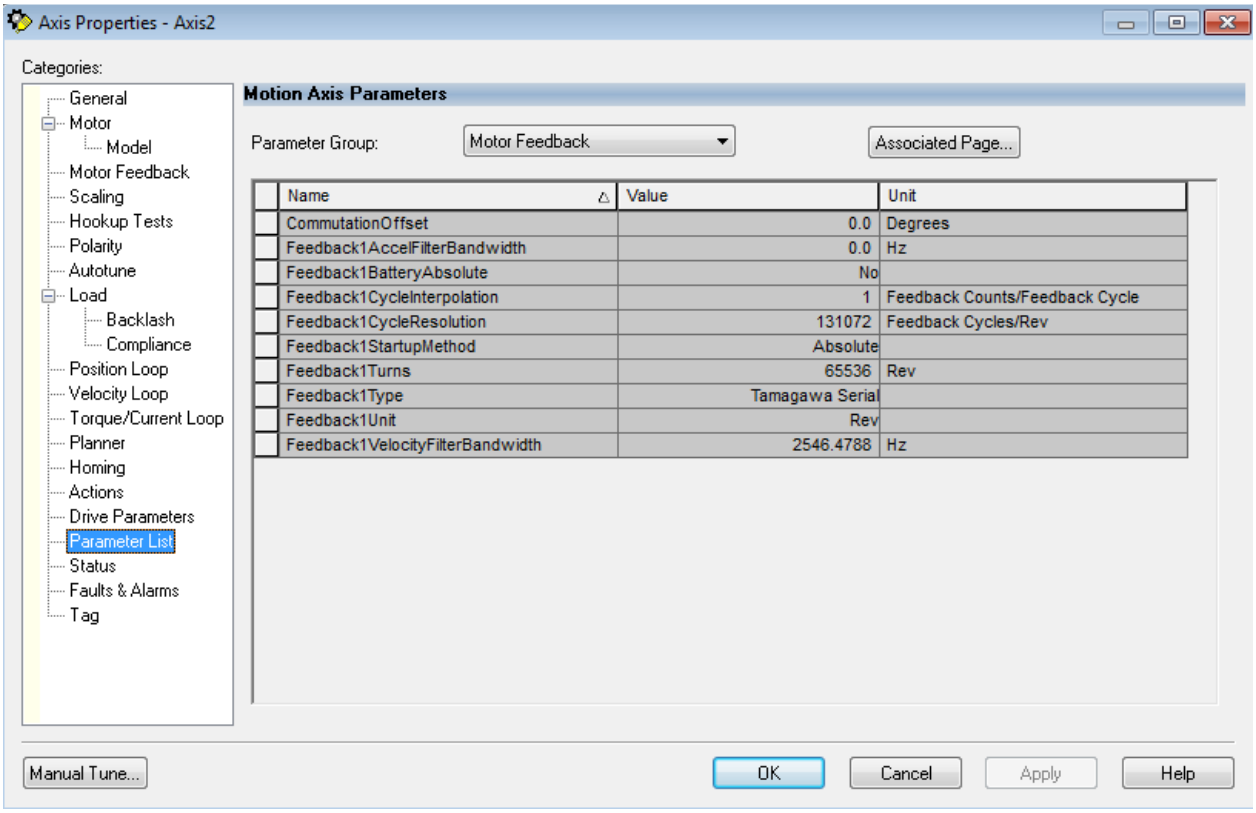
Homing Fig. 17-90



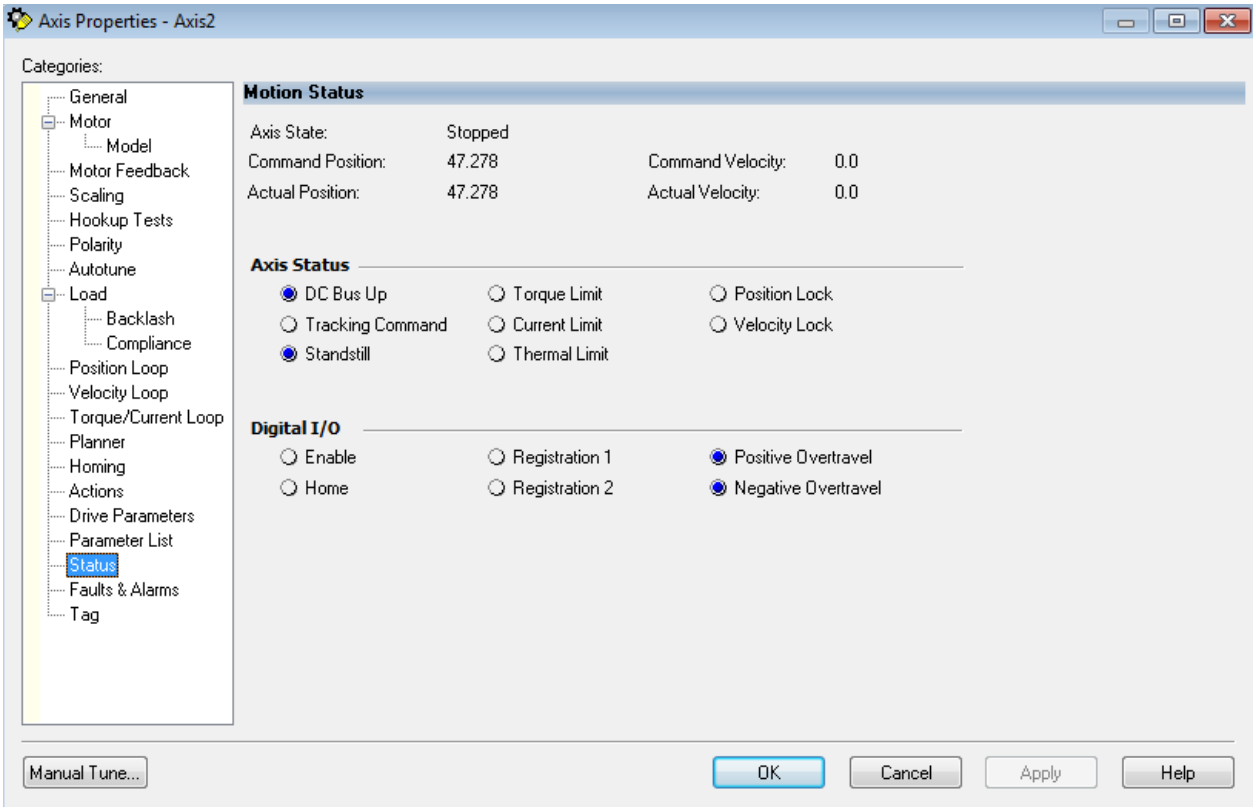
Actions to Take Upon Conditions Fig. 17-91



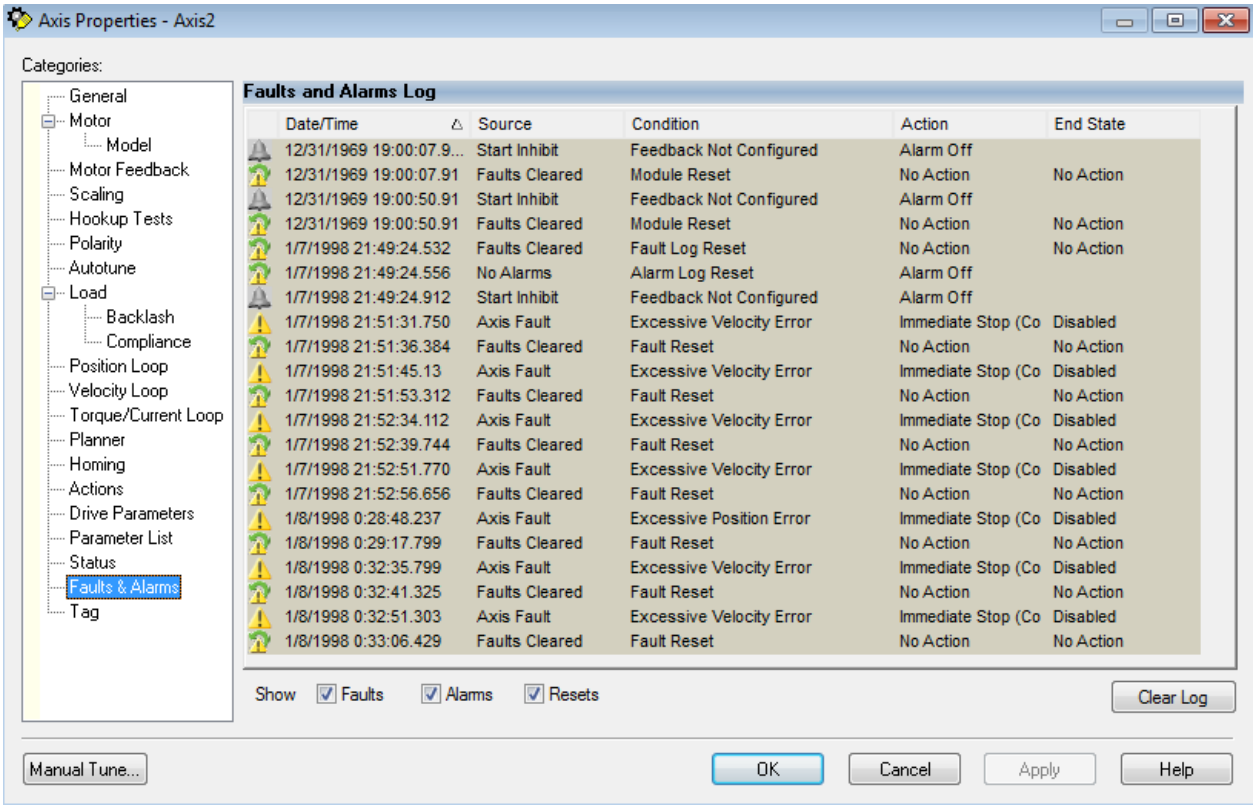
Drive Parameter List Fig. 17-92



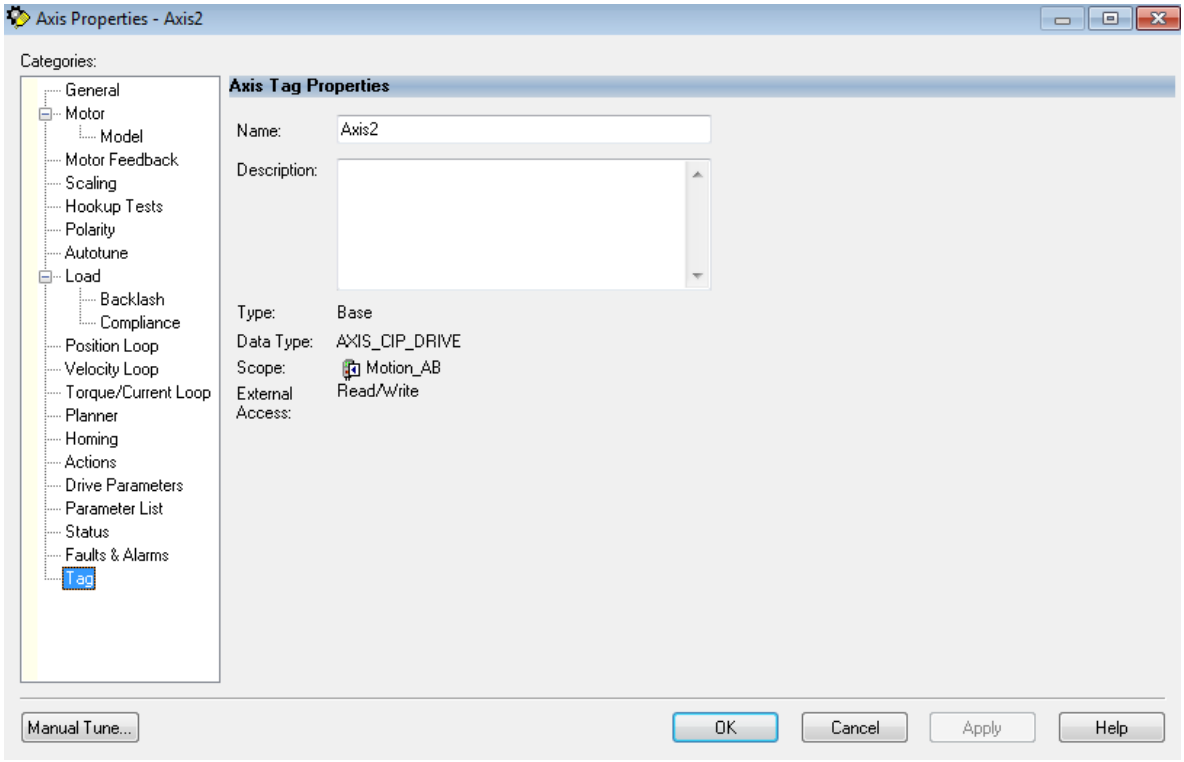
More Drive Parameters Fig. 17-93



Motion Status Screen Fig. 17-94



Faults and Alarm Log Screen Fig. 17-95

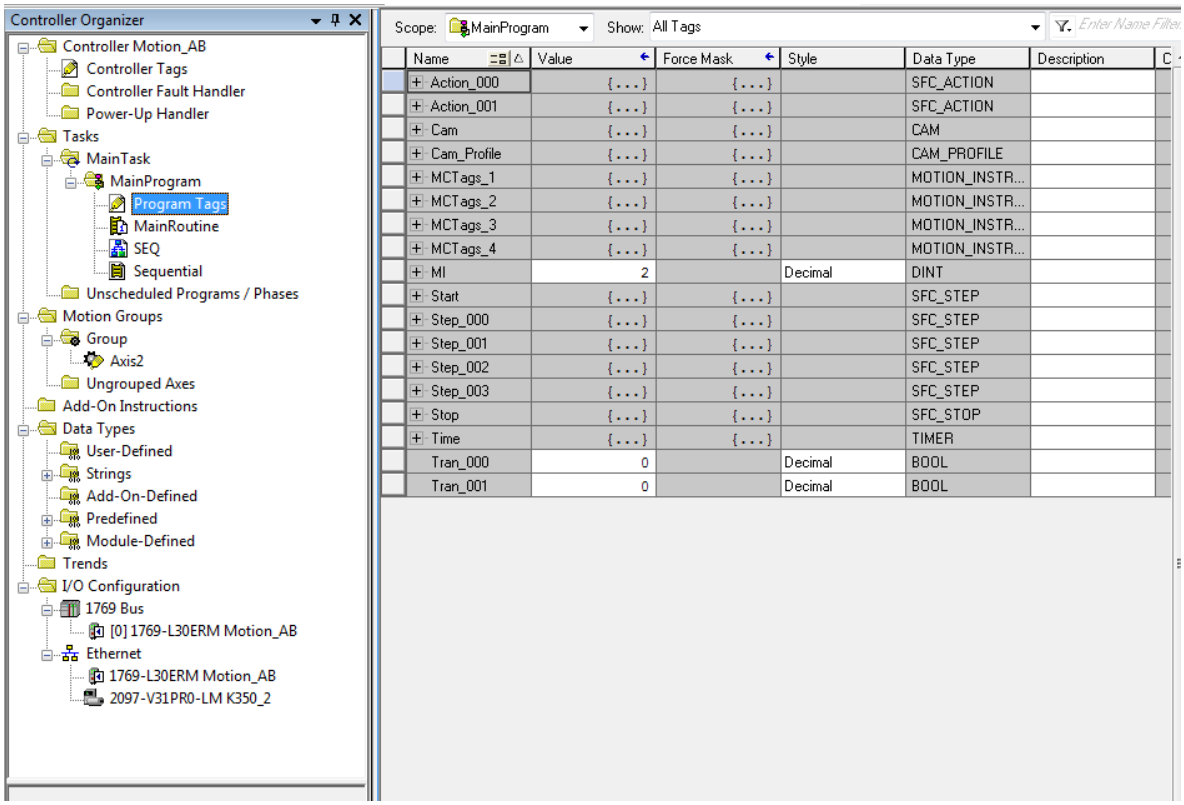


Tag Properties Fig. 17-96

Note:

Under Drive Parameters, Feedback1BatteryAbsolute, set to 'No' or 0. Otherwise, the hardware will generate a fault and you will not be able to run the axis.

Sample Commands for the A-B Application



Setting Up a Sequence Fig. 17-97

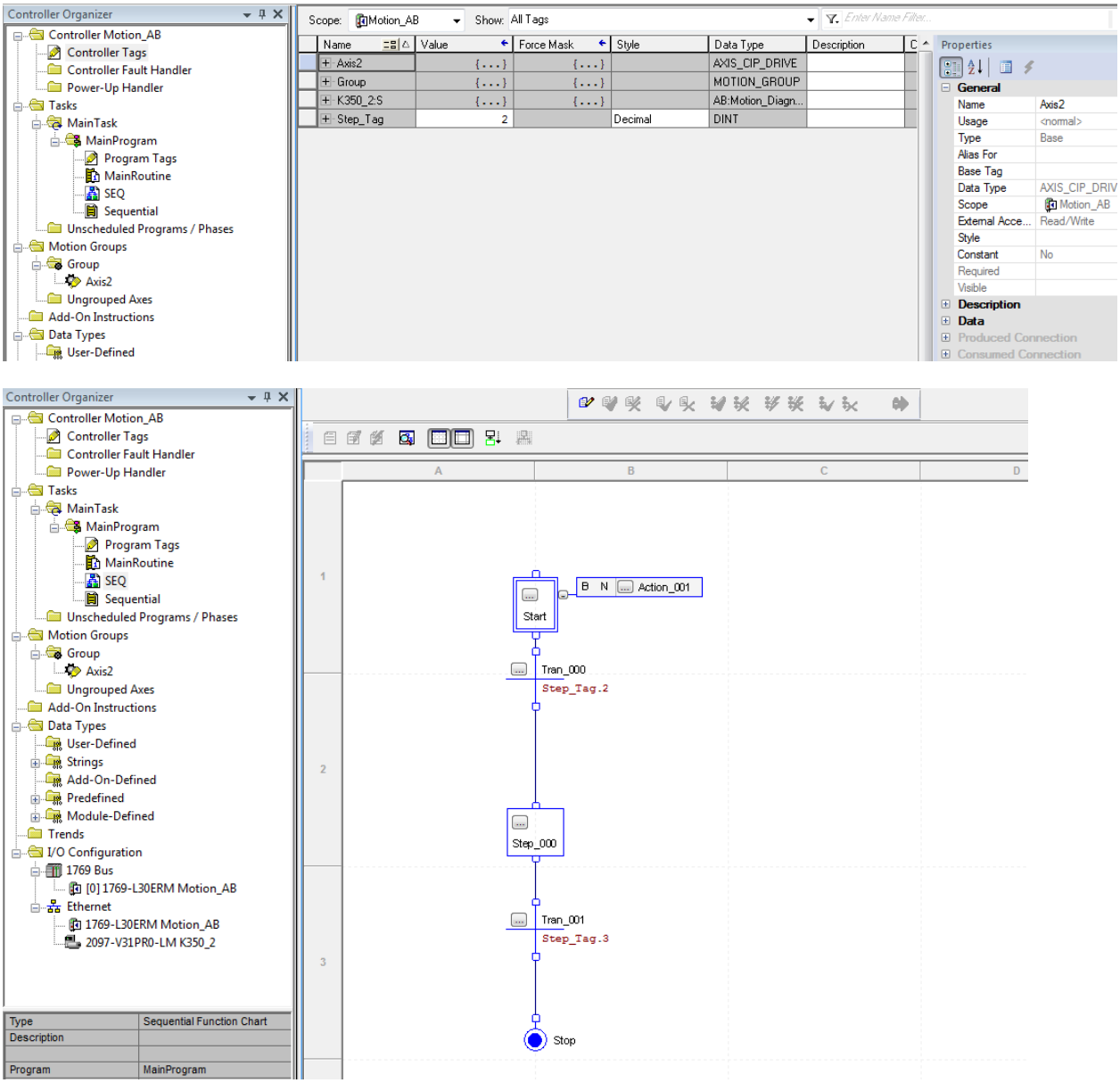


Fig17-98 Example of Sequencer

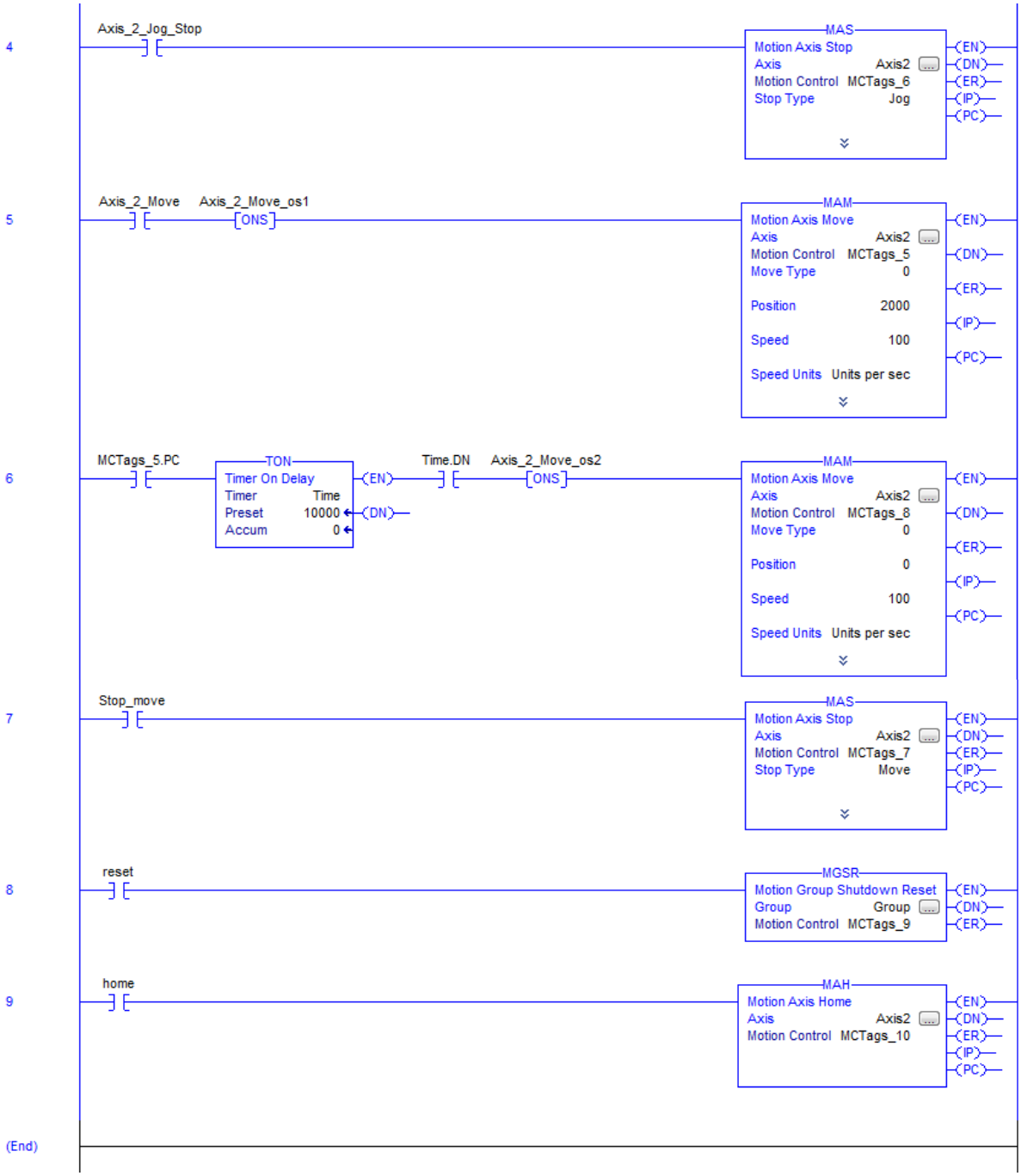


Fig. 17-99 Sample Program for A-B Single Axis Servo

The program given above is a sample program for download. It is to be used to provide a starting point for the single axis lab described at the end of the chapter. An HMI similar to that described for Siemens may also be built which allows the various contacts to be toggled from a screen. To begin, the control bits may be toggled from the Studio 5000 ladder program while in the run mode.

While we mainly only discuss single axis motion, the following description of a vendor's two-axis motion control card is useful to describe the motion command instructions for the next step in moving from single-axis to multiple axis control:

AMCI's 3602 – Synchronization of Two Axes

The 3602 is the second servo/stepper controller for the CompactLogix and MicroLogix 1500 platforms.

2-Axis Stepper (and Servo) Controller For Allen-Bradley CompactLogix

Product Features:

- 2-axis motion controller
- Enhanced motion profiles
- Interpolated motion (circular & linear)
- 200 kHz maximum output frequency
- Open Collector Outputs for large voltage operating range; wire for sinking/sourcing inputs
- Use for Stepper or Servo control

Fig. 17-100

The module offers two independent motion control axes that function in an open-loop configuration. Each axis can be used to drive a stepper or a servo with step/direction input capability. The 3602 can also **synchronize** the two axes, giving you the ability to control linear and circular motions in an XY plane.

The 3602 requires sixteen input words and sixteen output words from the PLC. Power draw from the PLC is 250mA from the +5Vdc supply. Each axis of the 3602 offers a full 32 bit (±2 billion+) motor position register, move lengths of up to 231 (±1 billion+) counts, programmable S-curve acceleration types, five discrete inputs for various control functions and a differential encoder input. The 3602 uses a Rockwell Automation 18-pin Removable Terminal Block for all of its I/O connections.

The 3602 has the following I/O connections for each axis:

- Single ended outputs to servo or stepper driver. Maximum output frequency is 150 KHz.
- Home Input. This single ended input is typically used when defining the home position on the machine.
- CW Limit and CCW Limit Inputs. These single ended inputs are used to define the maximum clockwise and counter-clockwise positions on the machine. If one of these inputs becomes active while traveling in that direction, the 3602 will immediately stop the move.
- Emergency Stop Inputs. If one or both of the CW and CCW Limits are not required, then the input can be configured as an Emergency Stop input. The 3602 will immediately stop the move if an Emergency Stop input becomes active.
- Capture Input. This single ended input can be used to capture the motor position during a move. This is useful in applications where you must capture the position value and the event is too short to be captured by the PLC.
- External Input. A single ended input that can be used to bring moves to a controlled stop or to bring a servo controller's Move Complete output back into the PLC.
- General Purpose Output. This single ended output is controlled through a bit from the PLC.

Front Panel LED's

The 3602 has three LED's to indicate the status of the module and each of its axes. The MODULE LED shows the status of the module while AXIS 1 and AXIS 2 LEDs show you the status of the axis.

Fig. 17-101

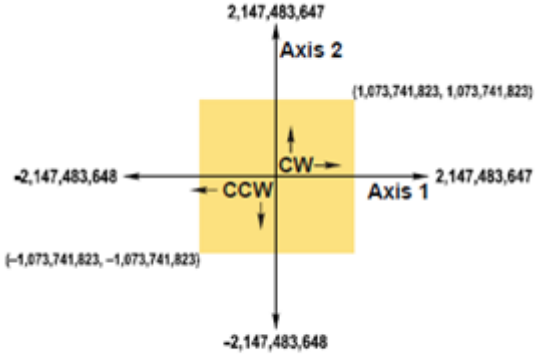


Figure 3.1 X-Y Plane

The X-Y Plane

Its easiest way to explain the interpolated move types is in terms of motion in the X-Y plane defined by the two axes. His plane is shown in the figure above.

- A move that results in increasing counts on the axis will cause CW pulses on the outputs of the axis. Likewise, a move that results in decreasing counts on the axis will cause CCW pulses on the outputs of the axis.
- The +/-2 billion+ counts at the ends of each axis represent the minimum and maximum values of the current position register for the axis, which is a signed 32 bit value. These values are not hard limits. If you are performing a relative move in a CW direction that exceeds 2,147,483,647 counts, the current position value will roll over to its maximum negative value.
- The colored square with limits of +/- 1,073,741,823 counts represents the limits of absolute coordinates, when programming an absolute move, the starting position can be outside this range, but all positions defined by the command must be within these limits. In the case of circular interpolated moves, the move can travel beyond these limits while running.

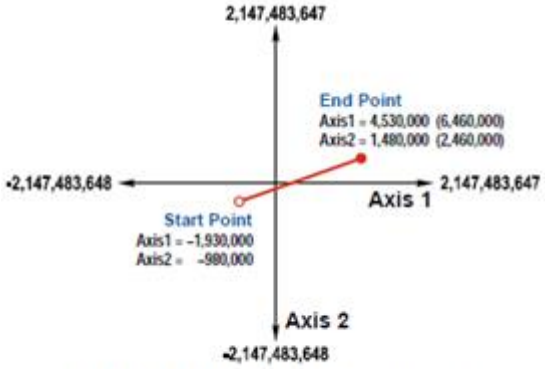


Fig. 17-102

Figure 3.2 Linear Interpolated Move

Linear Interpolated Moves

Conceptually, a linear interpolated move is performed by travelling the shortest distance between two points on the X-Y plane defined by the two axes of the 3602. The start point is the current position defined by the two axes. The end point can be programmed with relative or absolute coordinates. In the figure above, the relative coordinates are shown in parentheses.

Relative Linear Move

When programming a move with relative coordinates, you program the number of steps or offset, you want each axis to travel.

Absolute Linear Move

Absolute coordinates treat the end point as an actual position on the machine. Note that you must set the home position for both axes of the machine before you can run an absolute linear move.

Required Parameters

Five parameters are required to define a linear interpolated move:

- End point X (Axis 1) coordinate (Absolute or Relative)
- End point Y (Axis 2) coordinate (Absolute or Relative)
- Interpolated Target Speed
- Interpolated Acceleration
- Interpolated Deceleration

Note that the target speed, acceleration and deceleration define the vector for the path as a whole, not the individual axes, so they are programmed only once.

Circular Interpolated Moves

Conceptually, a circular interpolated move is performed by traveling between two points in the X-Y plane along an arc of a circle defined within the plane. As with linear interpolated moves, the start point of every move is the current position of the two axes. The other points can be programmed with absolute or relative coordinates.

For non-interpolated, single axis moves, the terms clockwise and counter-clockwise refer to whether or not the motor position value reported back to the PLC will increase or decrease as a

result of the move. For circular interpolation moves, the terms clockwise and counter-clockwise refer to the direction of travel when looking down at the X-Y plane.

There are three methods to specify a circular interpolated move:

Center Point Method

The center point method is shown in figure 3-4. In addition to the end point, this method defines the center point of the circle. The 3602 verifies that the length from the center point to the start point is equal to the length from the center point to the end point before running the profile. Both lengths are radii of the circle and must be equal. Even though these three points completely define the circle, they cannot define the direction of travel along the circle to move from the start point to the end point. Because of this, the 3602 has two commands for use with the center point method. One command causes CW motion along the circle path while the other causes CCW motion. Note that the CW and CCW motion in this case refers to the direction of travel in the X0Y plane. It does not refer to the state of outputs during the move. Depending on the size and location of the circular path, each axis may output both CW and CCW pulses during the move.

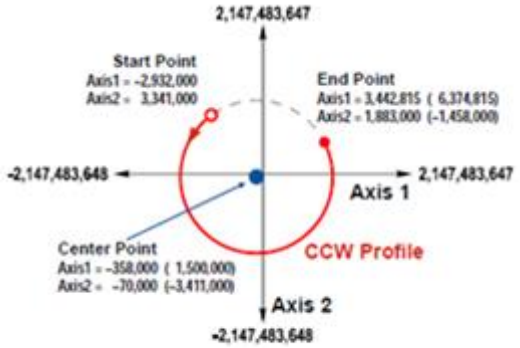


Figure 3.4 Center Point Circular Move

Fig. 17-103

The center point and end point cam be specified with relative or absolute coordinates and the two points must use the same coordinate system. This is the only method that allows you to set the end point equal to the start point and travel along the entire circular path in the X-Y plane. All other methods only allow you to travel over an arc of the defined circle.

Radius Method

The radius method is shown below. In addition to the end point, this method defines radius of the circular path. These three pieces of information actually define two circles in the plane, so an additional piece of information is used to define the move path. This piece of additional information is the sign of the radius value. If the radius is positive, the move will travel the shortest arc between the two points. If the radius is negative, the moves will travel the longest arc between the two points.

In order to determine the direction of travel, clockwise or counter-clockwise, the 3602 has two commands for use with the radius method. One command causes CW motion along the circle path while the other causes CCW motion. Note that CW and CCW motion in this case refers to the direction of travel in the X-Y plane as shown in the figure. It does not refer to the state of the outputs during the move. The bottom half of the figure shows the four available moves based on the sign of the radius value and the move direction.

Path A: CW move, Negative radius
Path B: CW move, Positive radius
Path C: CCW move, Positive radius
Path D: CCW move, Negative radius

The end point can be specified with relative or absolute coordinates.

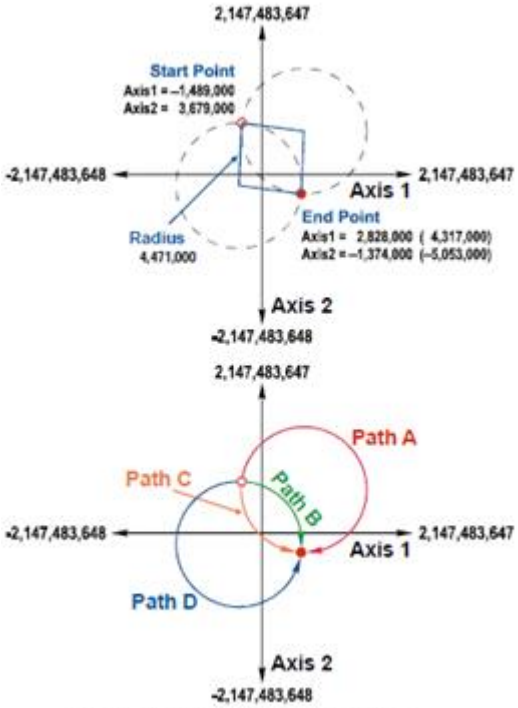


Fig. 17-104

Figure 3.5 Radius Circular Move

Via Point Method

The via point method defines a third point on the circular path that the move will pass through while traveling from the start point to the end point. This method only has one command associated with it because the 3602 can determine the direction of travel for the move with the three given points. The via point and end point can be specified with absolute or relative coordinates and the two points must be use the same coordinate system.

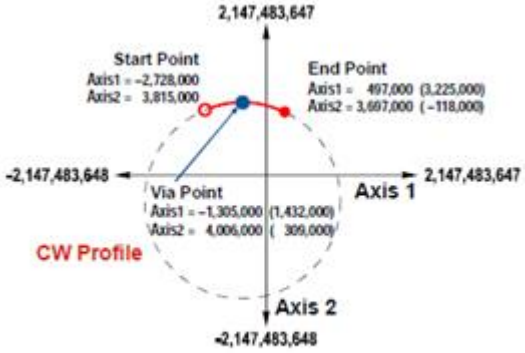


Fig. 17-105

Figure 3.6 Via Point Circular Move

G Code – Synchronization of Many Axes

What would one expect next as a strategy to control multiple axes? G Code is an answer. The example here shows the instructions for cutting an upside A from metal:

Well it is an upside down letter the capital A. Obviously it would be difficult to hand code in a lot of letters and also support scaling but there are a couple of freeware utilities that will produce GCODE from TrueType fonts.

```
%
G00 Z 0.2 (raise bit)
G00 X1 Y1 (move to start)
G01 Z -0.2 F5 (plung bit)
G01 X2 Y3 F11 (draw one leg from top to bottom)
G00 Z 0.2 (raise bit)G00 X1 Y1 (air move)
G01 Z -0.2 F5 (plunge bit)
G01 X0 Y3 F11 (draw one leg top to bottom to top)
G00 Z 0.2 (raise bit)
G00 X 0.5 Y1.8 (air move)
G01 Z -0.2 F5  (plunge)G01 X 1.5 (cross on the A)
G00 Z 0.2 (raise the bit)
%
```

Fanuc Robots

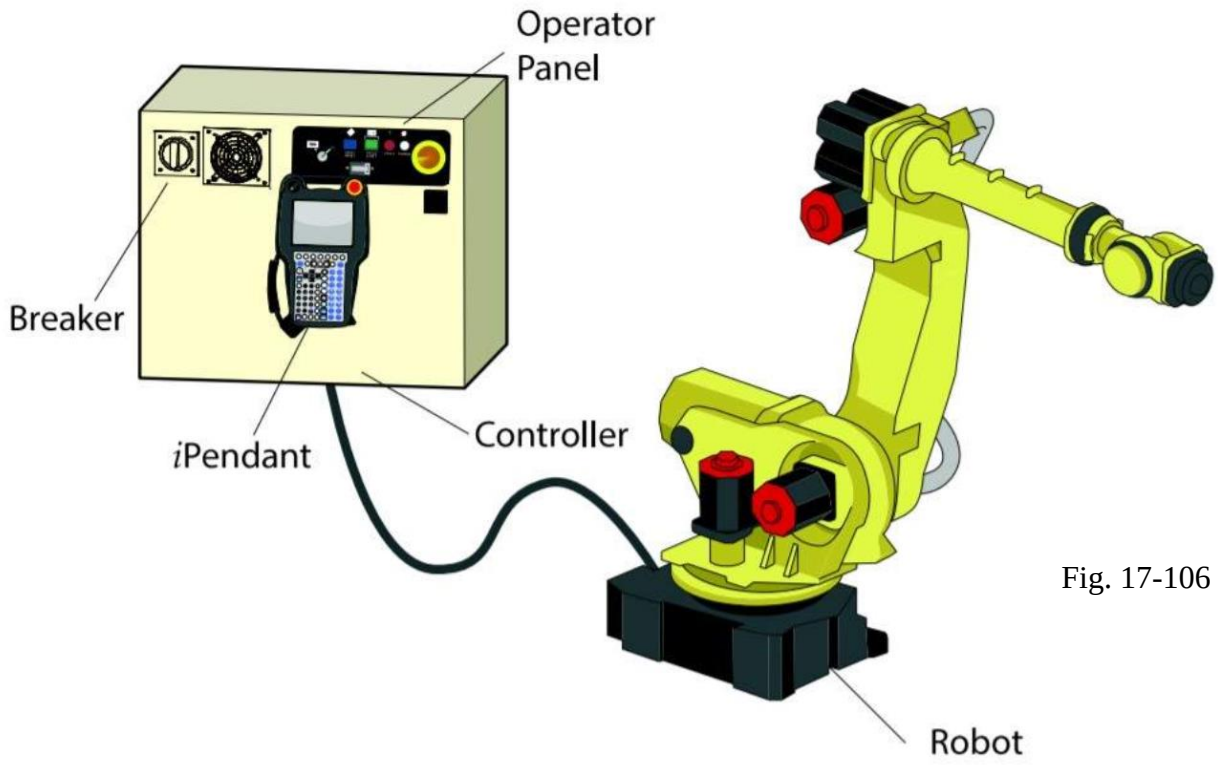


Fig. 17-106

The Fanuc Robot above is a mechanical unit with six servo motors. It is the logical next step with coordinated axis control for automation of a process. The choice between a coordinated axis system using programmable axes from either the single axis or multi-axis methods or with a robot many times is made by others with the electrical programmer an implementer. However, choosing between type of robot or type of electrical drive may be your decision. The robot section is continued in Hybrid Lab Text – Ch. 29.

The following are the motion commands for the Siemens 1200's and 1500's. They are much more expansive than the ones listed earlier:

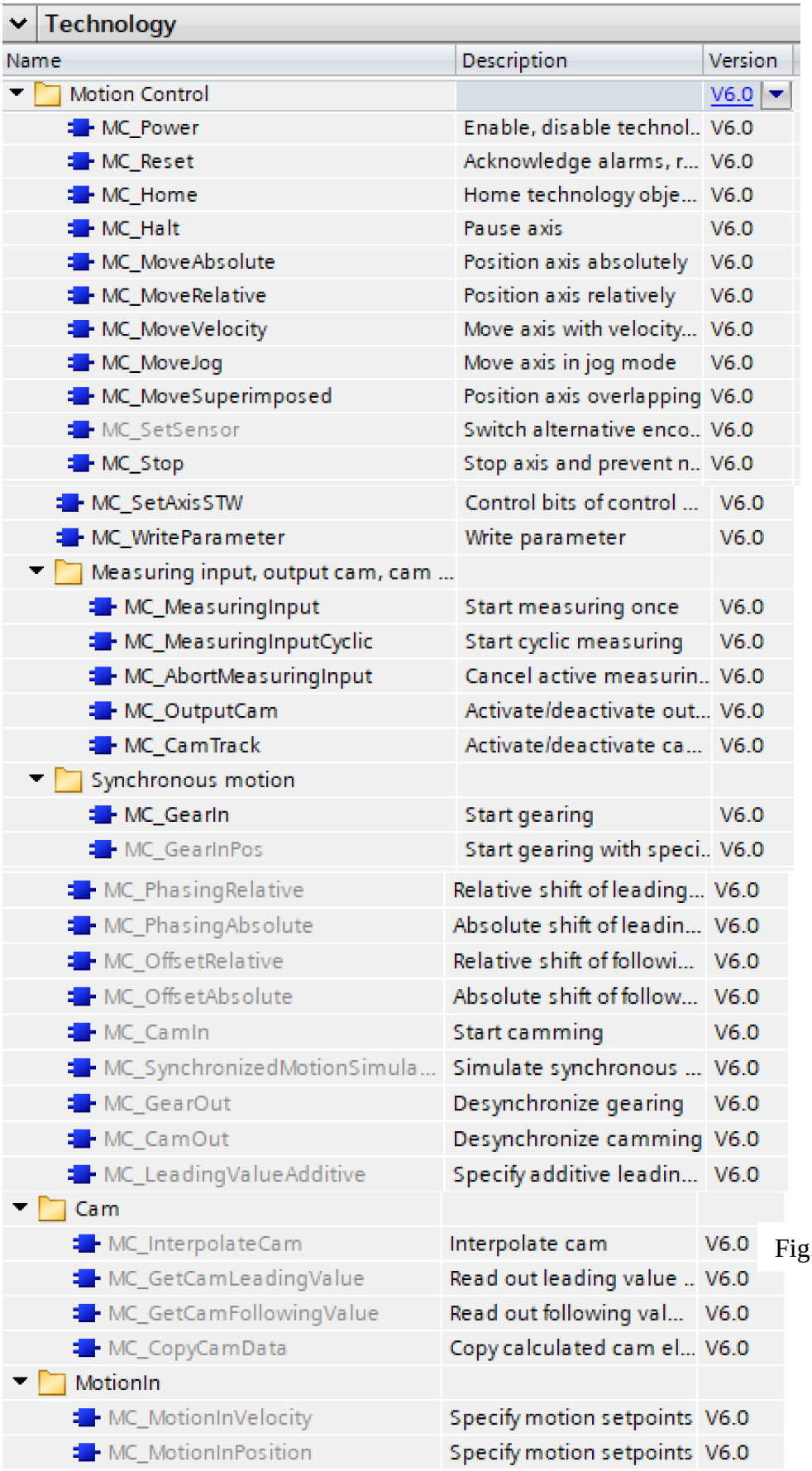
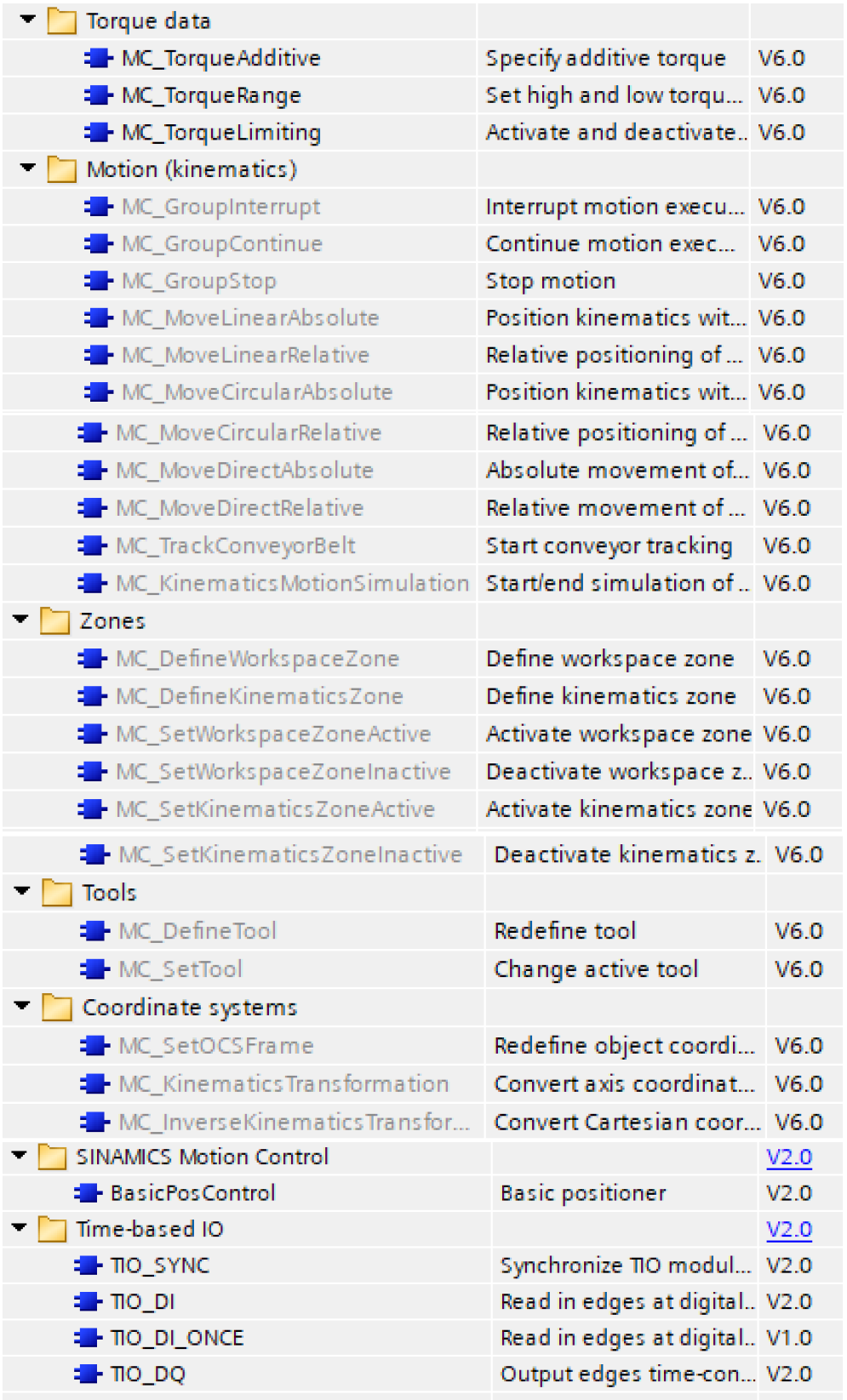


Fig. 17-107



Torque data		
MC_TorqueAdditive	Specify additive torque	V6.0
MC_TorqueRange	Set high and low torqu...	V6.0
MC_TorqueLimiting	Activate and deactivate..	V6.0
Motion (kinematics)		
MC_GroupInterrupt	Interrupt motion execu...	V6.0
MC_GroupContinue	Continue motion exec...	V6.0
MC_GroupStop	Stop motion	V6.0
MC_MoveLinearAbsolute	Position kinematics wit...	V6.0
MC_MoveLinearRelative	Relative positioning of ...	V6.0
MC_MoveCircularAbsolute	Position kinematics wit...	V6.0
MC_MoveCircularRelative	Relative positioning of ...	V6.0
MC_MoveDirectAbsolute	Absolute movement of...	V6.0
MC_MoveDirectRelative	Relative movement of ...	V6.0
MC_TrackConveyorBelt	Start conveyor tracking	V6.0
MC_KinematicsMotionSimulation	Start/end simulation of ..	V6.0
Zones		
MC_DefineWorkspaceZone	Define workspace zone	V6.0
MC_DefineKinematicsZone	Define kinematics zone	V6.0
MC_SetWorkspaceZoneActive	Activate workspace zone	V6.0
MC_SetWorkspaceZoneInactive	Deactivate workspace z..	V6.0
MC_SetKinematicsZoneActive	Activate kinematics zone	V6.0
MC_SetKinematicsZoneInactive	Deactivate kinematics z.	V6.0
Tools		
MC_DefineTool	Redefine tool	V6.0
MC_SetTool	Change active tool	V6.0
Coordinate systems		
MC_SetOCSFrame	Redefine object coordi...	V6.0
MC_KinematicsTransformation	Convert axis coordinat...	V6.0
MC_InverseKinematicsTransfor...	Convert Cartesian coor...	V6.0
SINAMICS Motion Control		V2.0
BasicPosControl	Basic positioner	V2.0
Time-based IO		V2.0
TIO_SYNC	Synchronize TIO modul...	V2.0
TIO_DI	Read in edges at digital..	V2.0
TIO_DI_ONCE	Read in edges at digital..	V1.0
TIO_DQ	Output edges time-con...	V2.0

Fig. 17-108

Summary

In this chapter a very broad subject, motion control, has been introduced and discussed. Some areas are just introduced and then left to the reader to expand upon later.

It is the hope that the experiences of these next two labs will give some measure of experience and give the student a desire to continue the discussion and have a healthy curiosity with respect to motion and motion control applications.

As they say, after writing a program for the shrink wrap "that's a wrap". Or for the grinder, "What a grind". Oh well….

Lab 17.1 Program the Stepper Application using the Siemens PLC, Stepper Module and Siemens HMI to control the dial for a variable number of complete turns either forward (CW) or reverse (CCW) at a various number of speeds.

To design the HMI panel, use the description above Fig. 17-25. The description there describes a simple single-axis machine. As an automatic sequence, use at least two different rotations. One could be of 4 turns followed by a dwell followed by a second 4 turns followed by a dwell followed by a return to home.

Notice the switches on the PLC. They are attached to inputs which set up the motion application. They may be used for all inputs except those specific to the operation of the machine such as the auto-manual, jog forward, jog reverse, halt and resume commands. Other commands such as 'home', 'reset' and 'power' can remain as switches instead of being incorporated into the HMI.

The following state diagram is a partial state diagram for building the assigned program above:

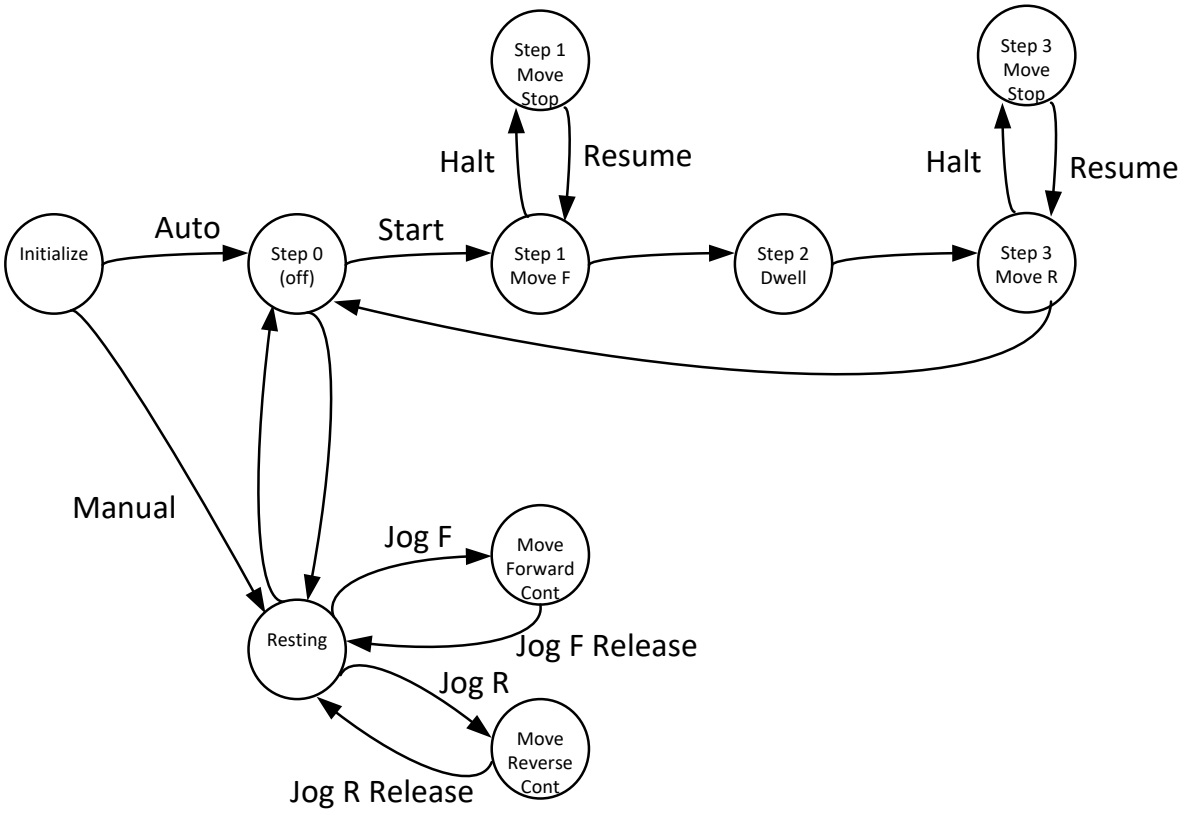
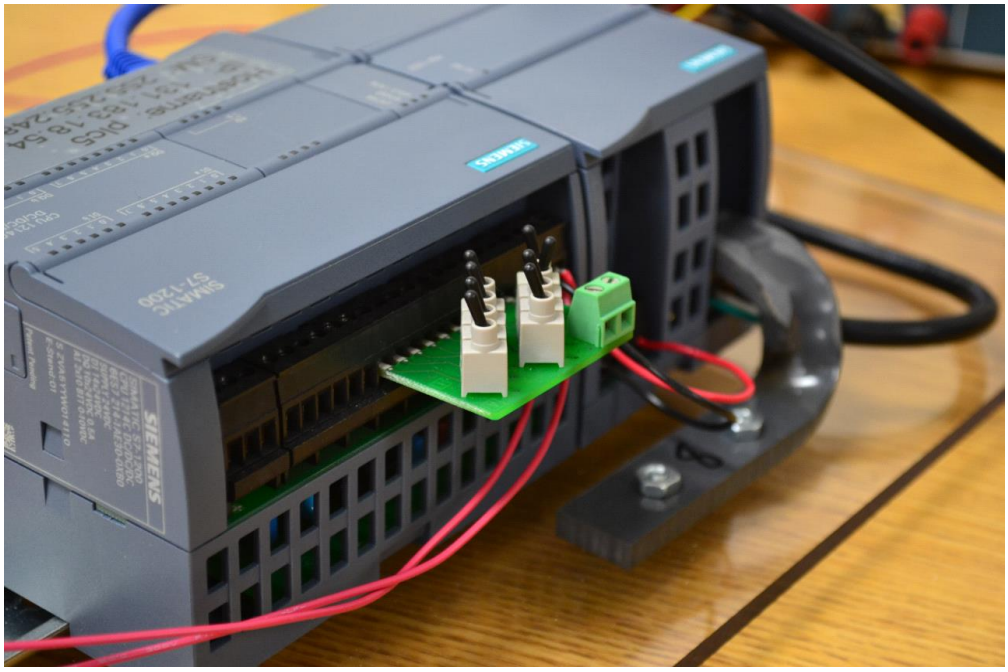


Fig. 17-109 State Diagram of Motion Application



The program given allows the user to toggle the various switches and rotate the motor. Some of the commands will be modified in the actual program. Notice that there is a command table that may be used. It is not usable if the pause and resume are to work properly. Try the toggle switches with the command table and then halt the motion. Then resume the motion. Notice that the motion is reset and starts again. The problem associated with the reset action requires the additional programming. Follow the commands below in this order to begin the Siemens stepper application.

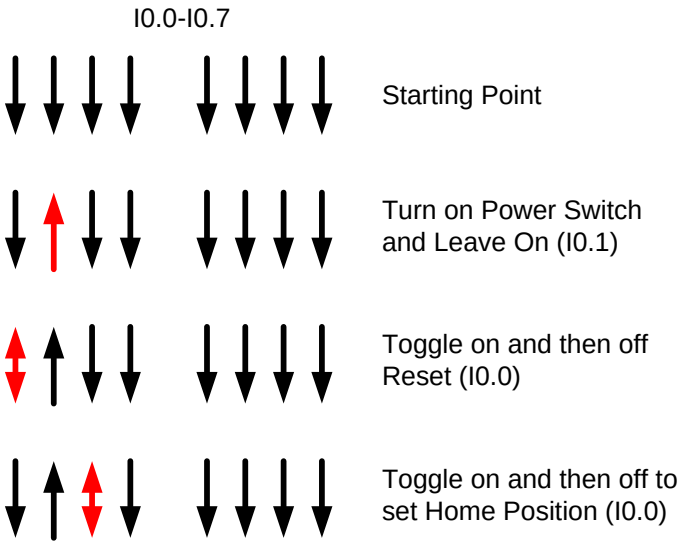
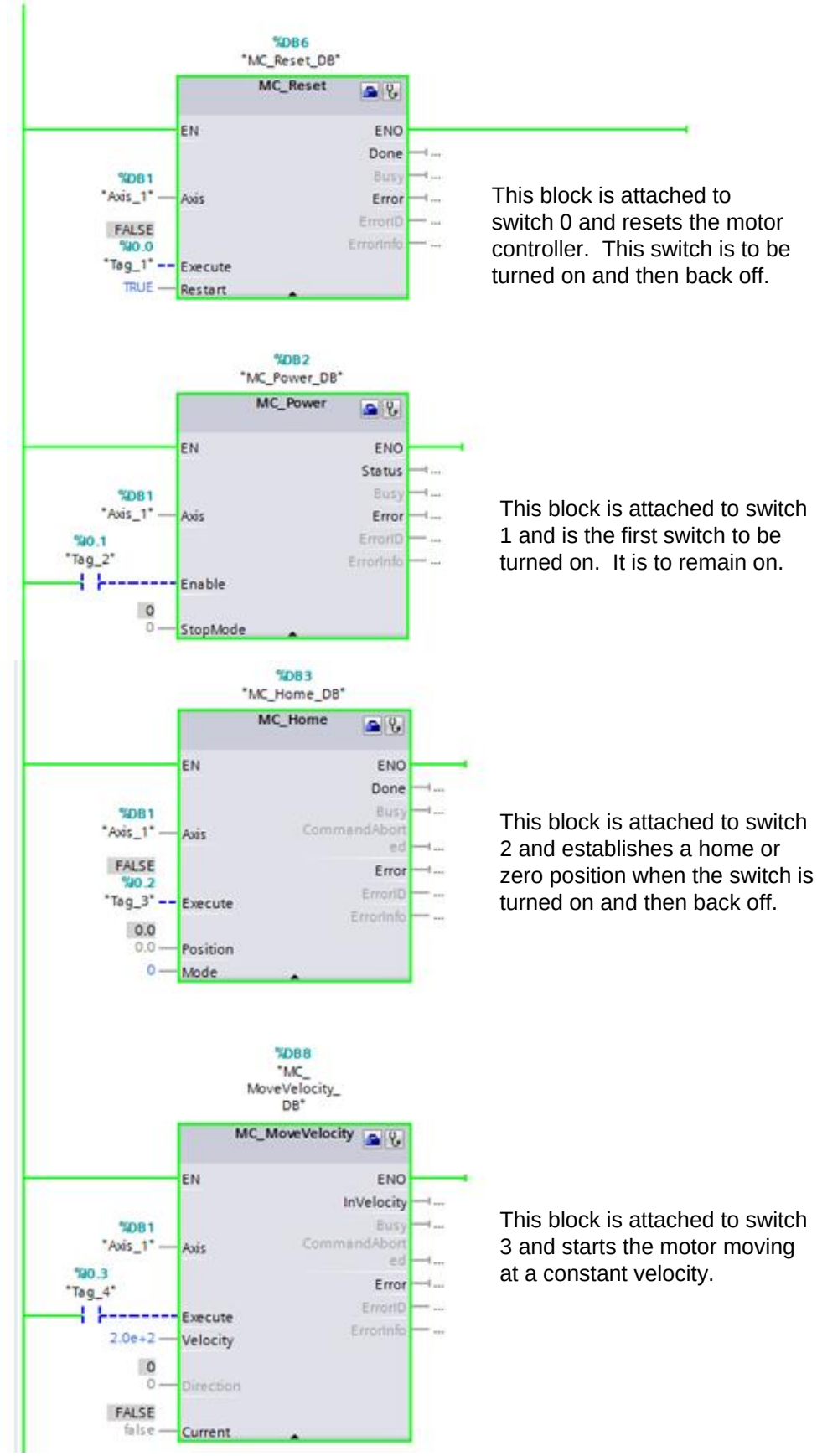


Fig. 17-110 Toggle Switch Settings for Siemens Stepper

The program given here is the program given on the website. It serves as a starting point for the program on the following pages:



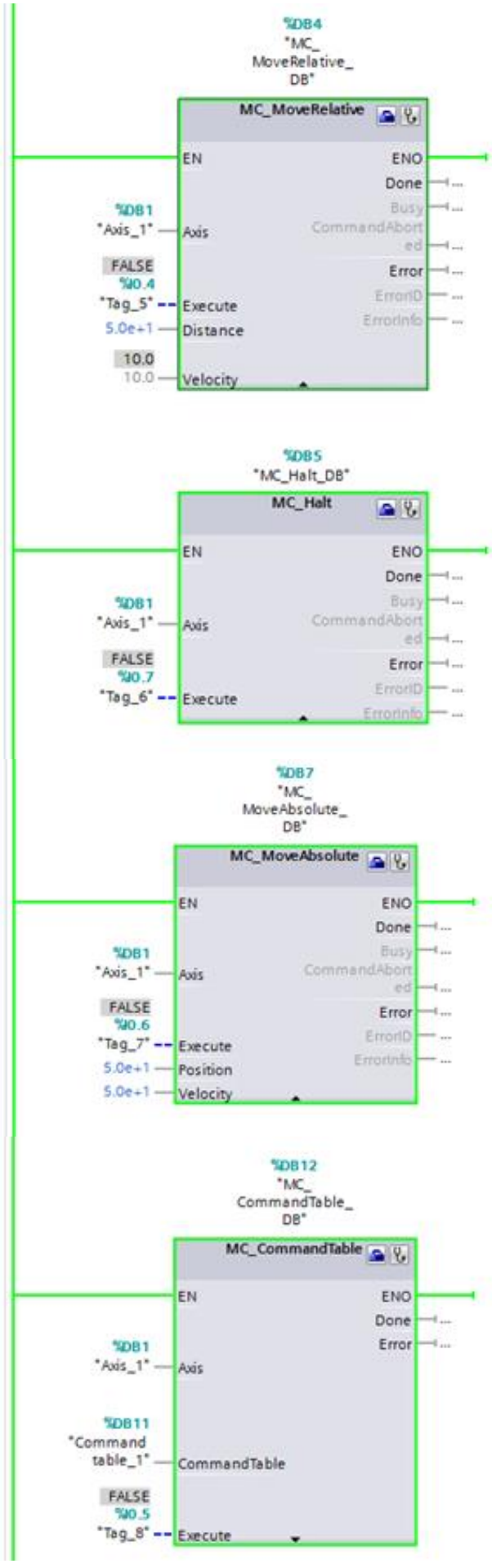
This block is attached to switch 0 and resets the motor controller. This switch is to be turned on and then back off.

This block is attached to switch 1 and is the first switch to be turned on. It is to remain on.

This block is attached to switch 2 and establishes a home or zero position when the switch is turned on and then back off.

This block is attached to switch 3 and starts the motor moving at a constant velocity.

Fig. 17-111



This block is attached to switch 4 and is not used for the lab since absolute moves are needed.

This block is attached to a programmed switch which will halt the motion.

This block is attached to programmed points per the attached program below.

This block is attached to switch 5 but should not be used since the program controls movement, not this Command Table.

Fig. 17-112

The MoveAbsolute Block Defined

MC_MoveAbsolute: Absolute positioning of axes V1...3

Description

The "MC_MoveAbsolute" Motion Control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveAbsolute command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Below is the Move Command from the Command Table Program (not used in this program)

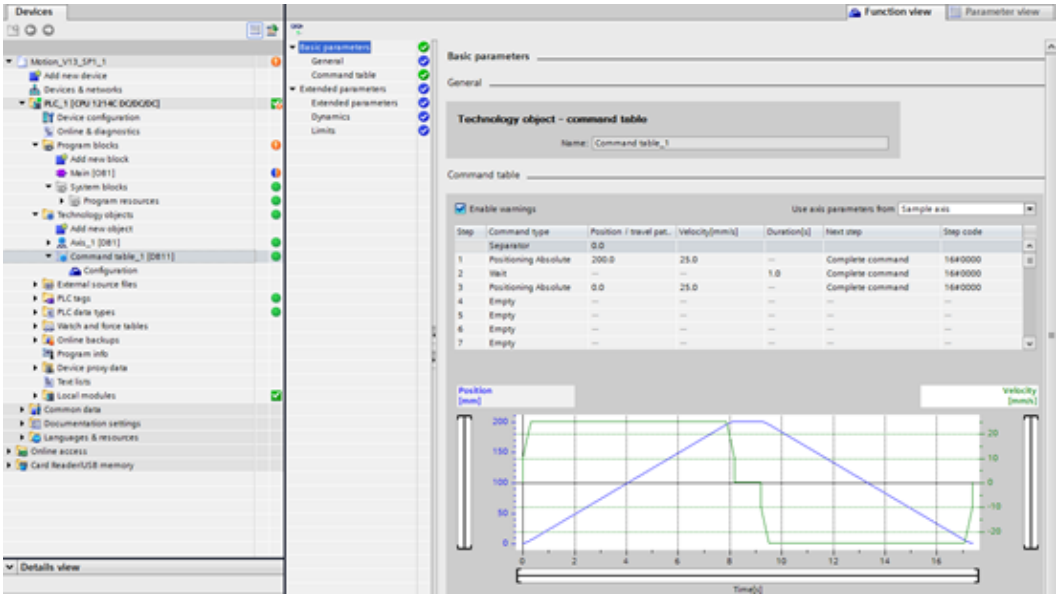


Fig. 17-113

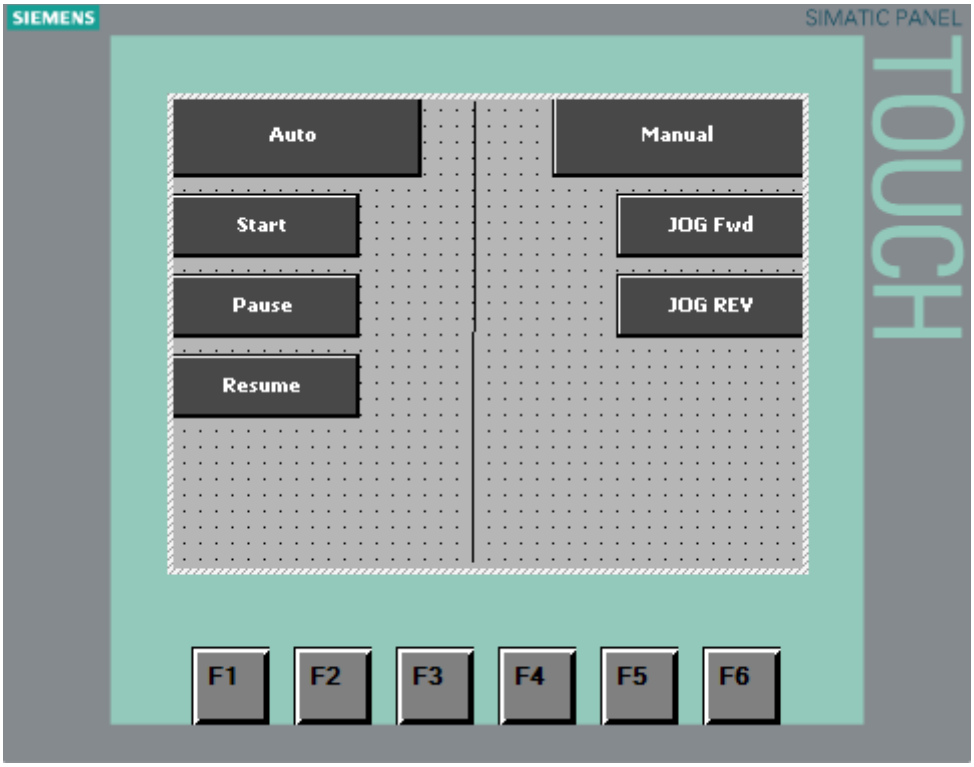


Fig. 17-113

There should be three objects on the screen for auto/manual. There are two buttons and one indicator. The program would be linked to these devices with tags developed in the Ladder Program similar to the diagram below:

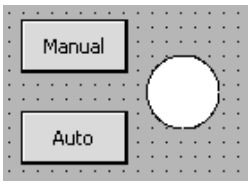
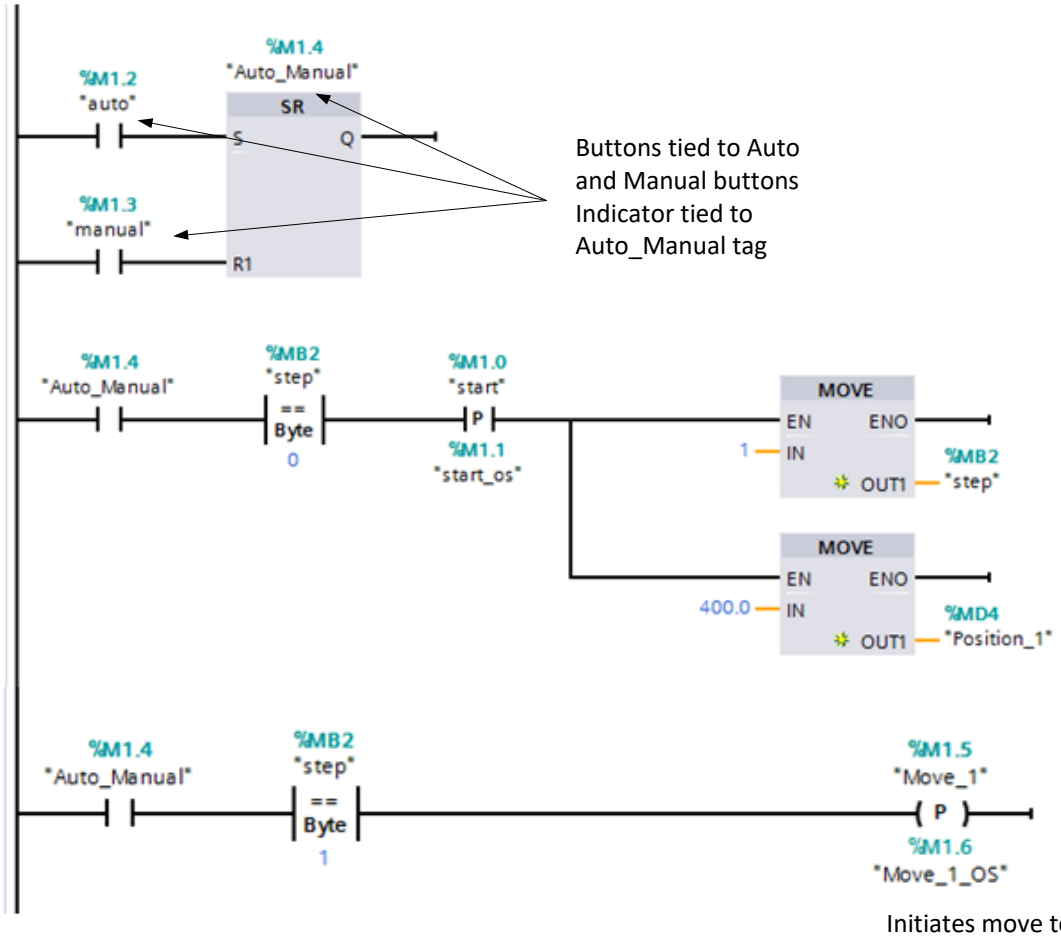


Fig. 17-114

The program per the State Diagram (partial):



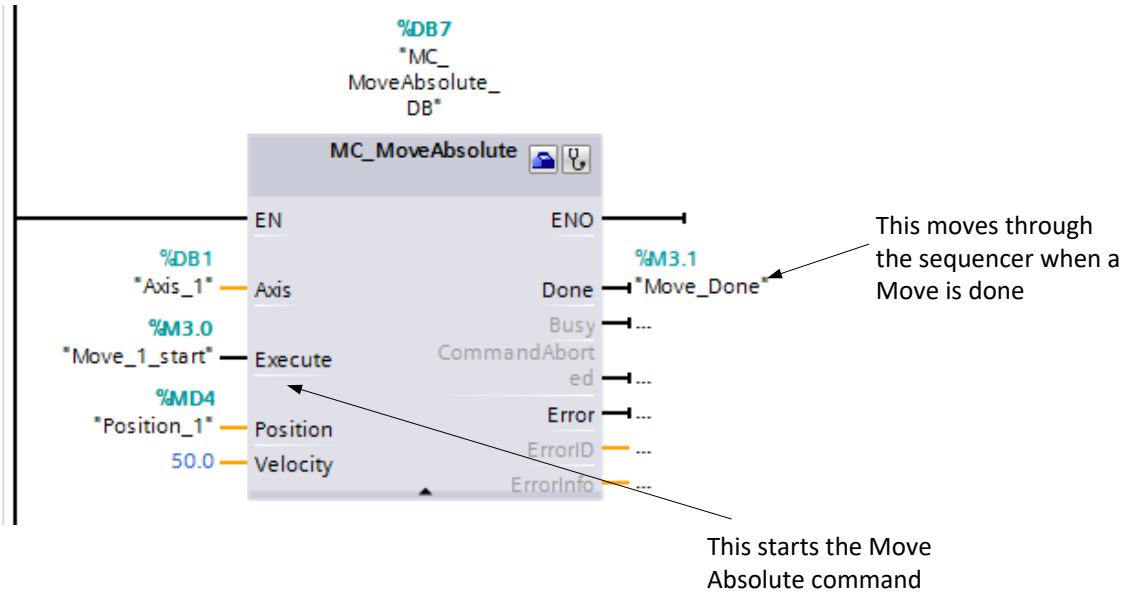


Fig. 17-115

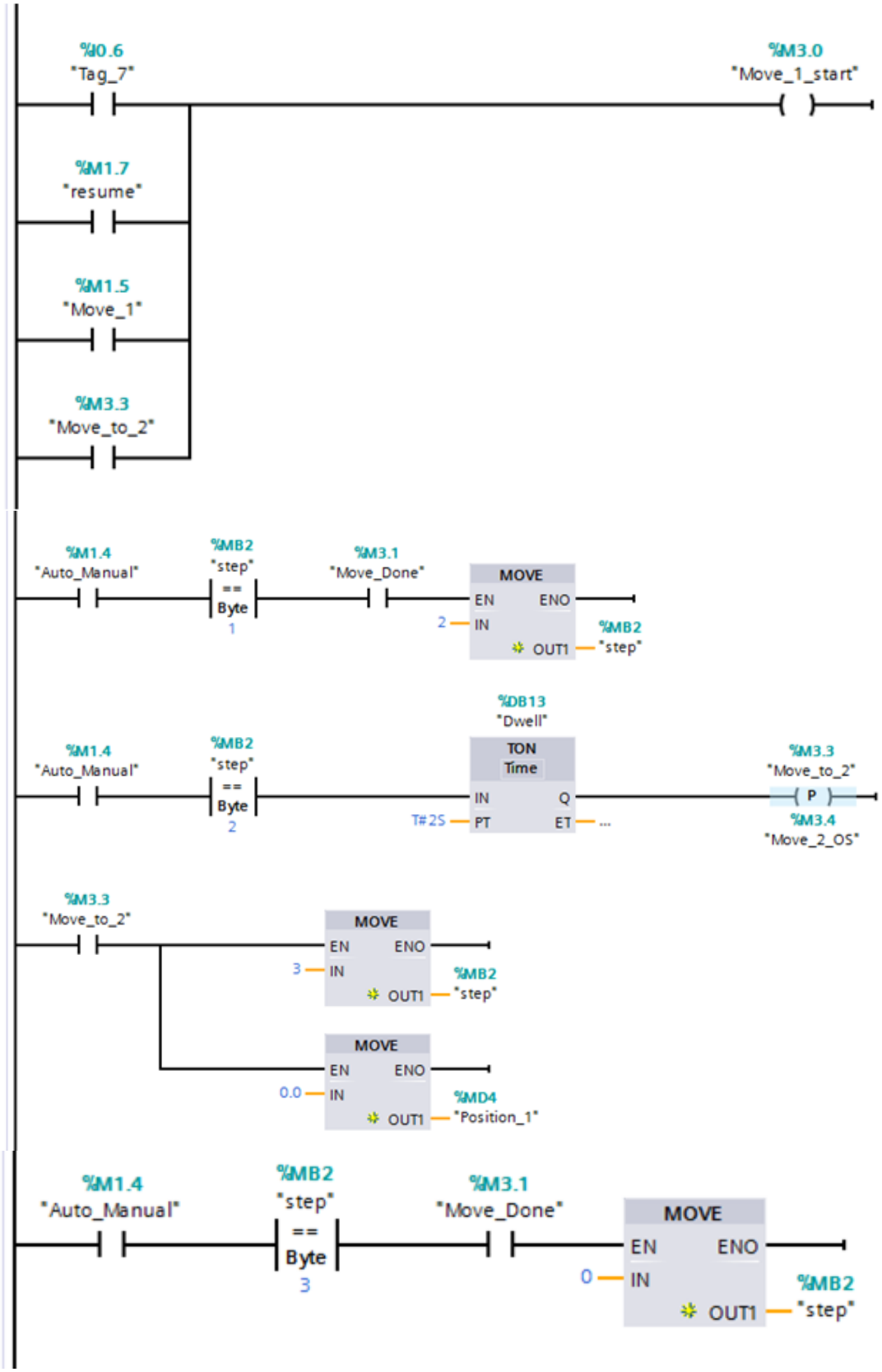

%I0.6
"Tag_7"
%M3.0
"Move_1_start"
%M1.7
"resume"
%M1.5
"Move_1"
%M3.3
"Move_to_2"
%M1.4
"Auto_Manual"
%MB2
"step"
==
Byte
1
%M3.1
"Move_Done"
MOVE
EN
ENO
2
IN
OUT1
%MB2
"step"
%DB13
"Dwell"
TON
Time
IN
Q
T#2S
PT
ET
...
%M3.3
"Move_to_2"
P
%M3.4
"Move_2_OS"
3
0.0
%MD4
"Position_1"
0

Fig. 17-116

For Manual jog forward or reverse, add the following:

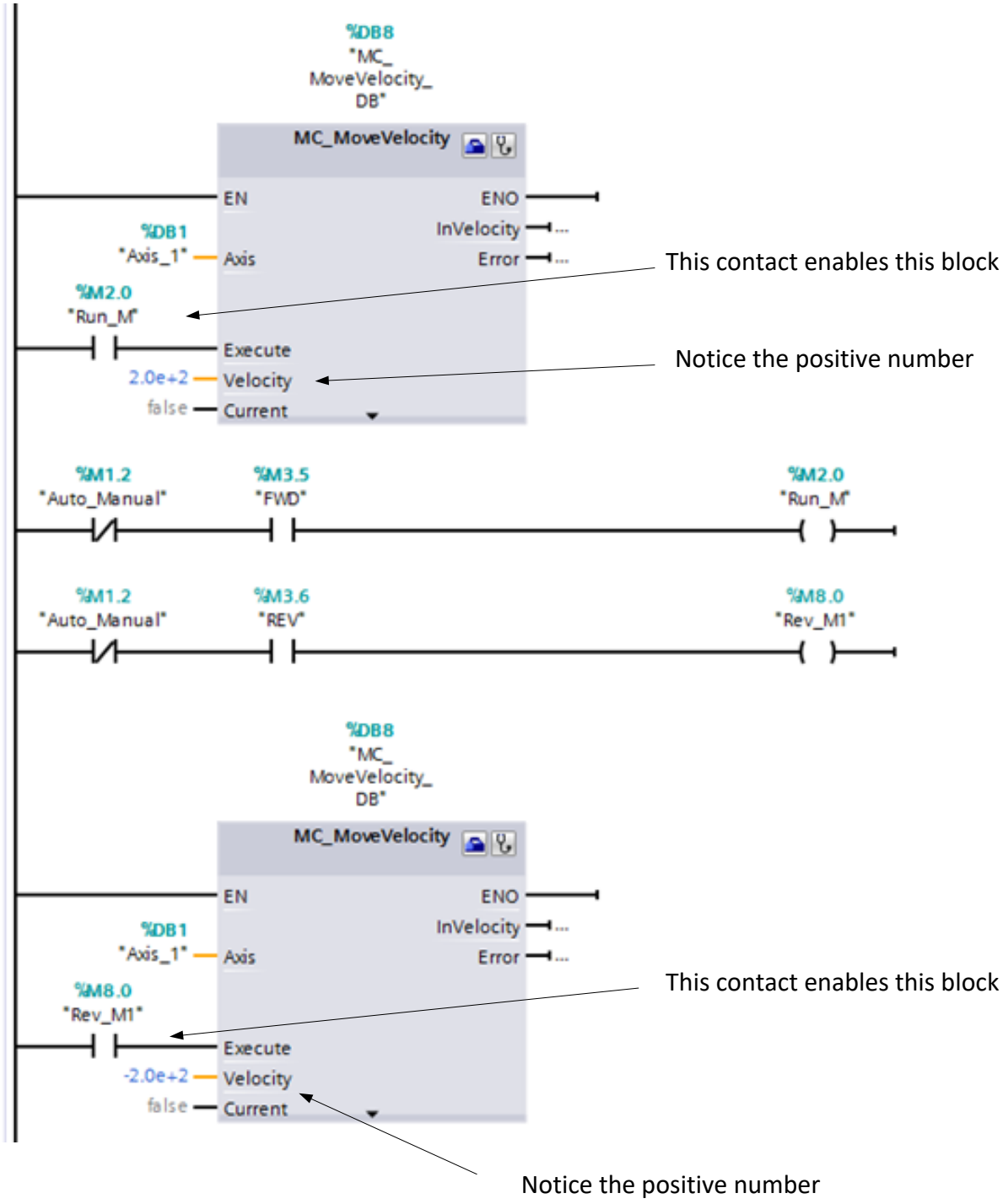


Fig. 17-117

For halting the motor in auto or reverse, add the following:

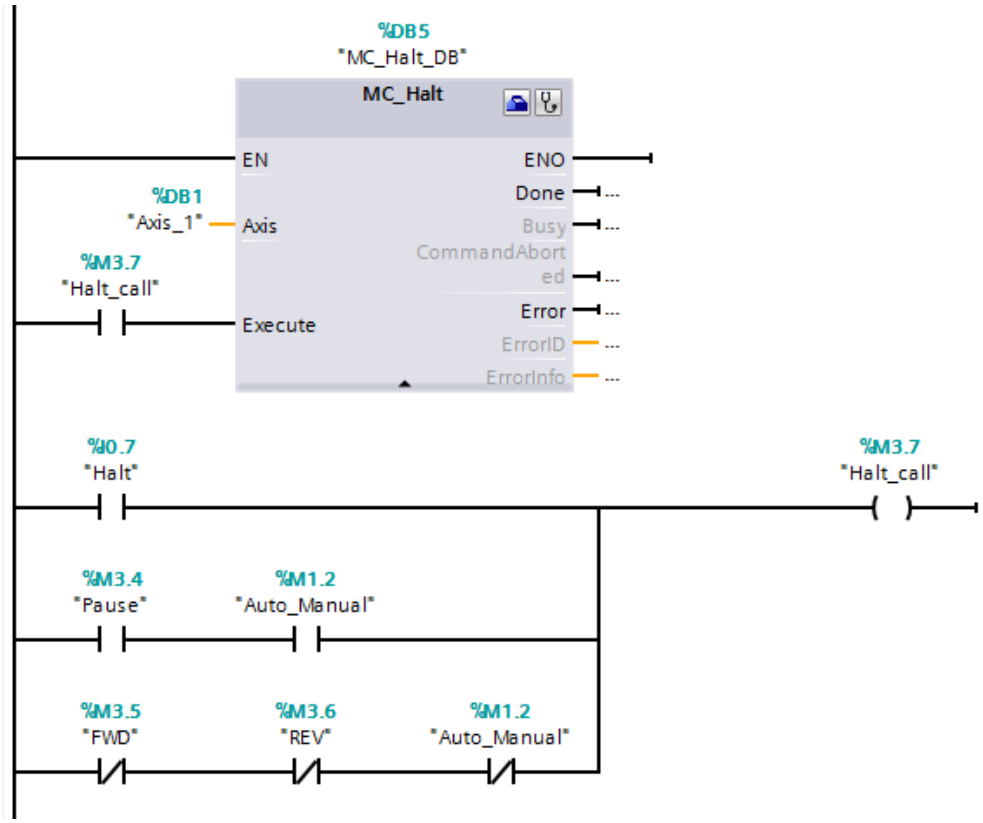


Fig. 17-118

Logic for homing the drive is not added here but should be considered. There are two homing events to be considered. First, homing is considered to establish a base home point. This is shown in the figure below immediately after initialize. Then a re-establishment should be inserted before the automatic operation is to proceed. This is shown after the start command is issued.

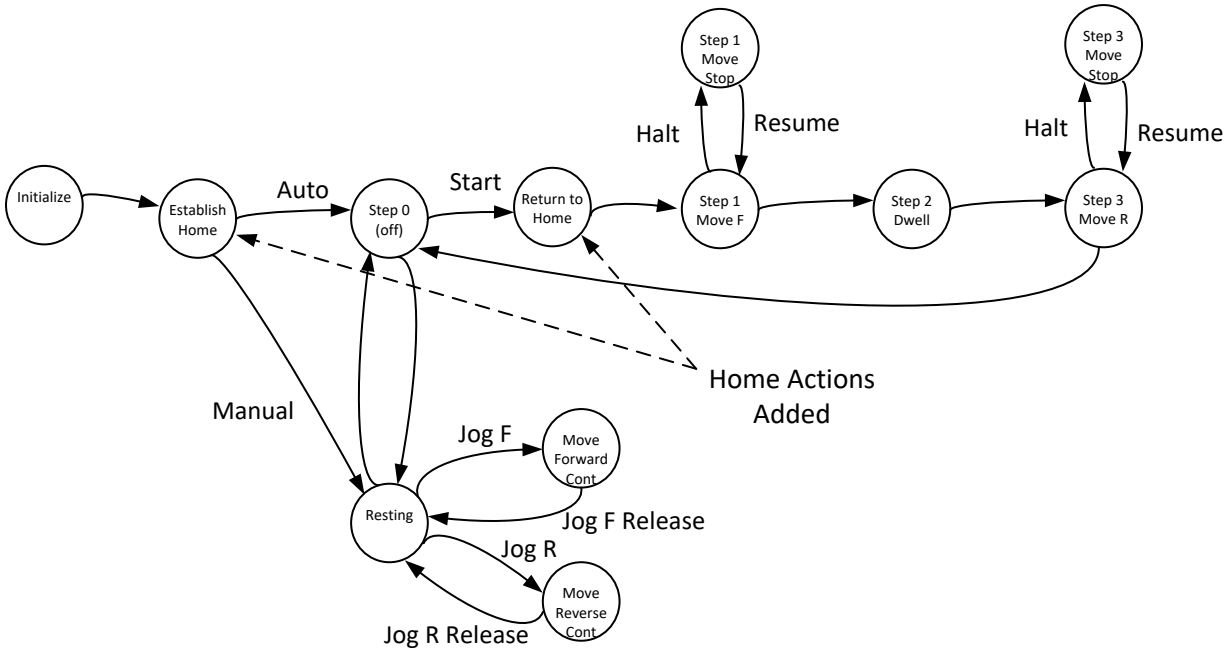


Fig. 17-119

Lab 17.2 Program the Servo Application using the A-B PLC, Kinetix 350 Servo and the HMI software to control the dial for a variable number of complete turns either forward (CW) or reverse (CCW) at a various number of speeds.

Again, design the HMI panel using the description above Fig. 17-25.

Again, design the program similar to Siemens' to allow for an auto and a manual move. In auto, provide for a sequence that may be interrupted with a pause and re-started with a resume button. In manual, provide a jog button for both forward and reverse operation.

Access for the position tag is found in the tag table as: Axis2.ActualPosition

The Auto/Manual Button may be programmed similarly to Siemens or with the following:

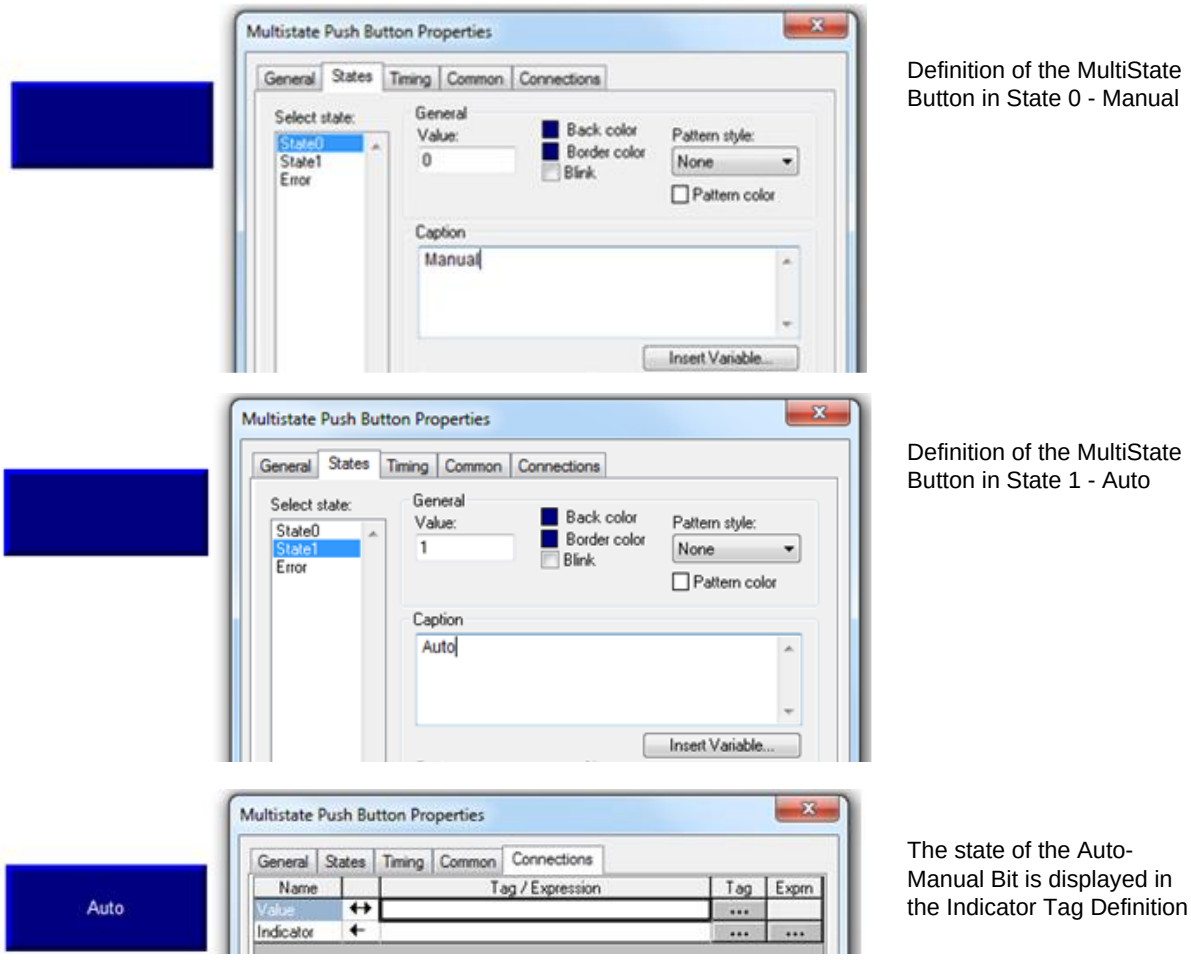


Fig. 17-120

The Motion Axis Move instruction is defined in part by the following table:

Operand	Type	Format	Description		
Axis	AXIS_CIP_DRIVE AXIS_VIRTUAL AXIS_GENERIC AXIS_GENERIC_DRIVE AXIS_SERVO AXIS_SERVO_DRIVE	Tag	Name of the axis. For an Absolute or Incremental Master Offset move, enter the slave axis.		
Motion Control	MOTION_INSTRUCTION	Tag	Control tag for the instruction		
Move type	DINT	Immediate Tag	**To**	**Use This Move Type**	**And Enter**
			Move an axis to an absolute position	Absolute	0
			Move an axis a specified distance from where it is now	Incremental	1
			Move a Rotary axis to an absolute position in the **shortest** direction regardless of its current position	Rotary Shortest Path	2
			Move a Rotary axis to an absolute position in the **positive** direction regardless of its current position	Rotary Positive	3
			Move a Rotary axis to an absolute position in the **negative** direction regardless of its current position	Rotary Negative	4
			Offset the master value of a position cam to an absolute position	Absolute Master Offset	5
			Offset the master value of a position cam by an incremental distance	Incremental Master Offset	6
			See Choose a Move Type for a Rotary Axis below for more information about rotary moves.		
Position	REAL	Immediate Tag	Absolute position or incremental distance for the move		
			For This Move Type	**Enter This Position Value**	
			Absolute	Position to Move to	
			Incremental	Distance to Move	
			Rotary Shortest Path	Position to move to. Enter a positive value that is less than the Position Unwind value.	
			Rotary Positive		
			Rotary Negative		
			Absolute Master Offset	Absolute Offset Position	
			Incremental Master Offset	Incremental Offset Distance	
Speed	REAL	Immediate Tag	Speed to move the axis in Speed Units		

Reference Manual

Allen-Bradley

Logix5000 Controllers Motion Instructions

Catalog Numbers 1756 ControlLogix, 1756 GuardLogix, 1768 CompactLogix, 1789 SoftLogix

Fig. 17-121

Lab 17.3

Demonstrate a simple Fanuc program from the youtube videos or other video you found.

You may want to explore some of the youtube videos from Tim Mehring or Adam Willea.

Lab 17.4

From the more expansive programming statement list and using an appropriate 1500, write a program using some of these advanced programming statements.

Problems

1. Why is the EVAL board necessary in the control of the stepper motor?

2. How does one control the direction of travel for the stepper motor, the servo motor?

3. Describe the process of "homing". Is this process necessary

4. List the homing modes for a Siemens motion controller, for an A-B Servo controller.

5. Show by drawing a string of pulses how the stepper accelerates, decelerates.

6. How does A-B accomplish the task similar to Siemens' Command Table?

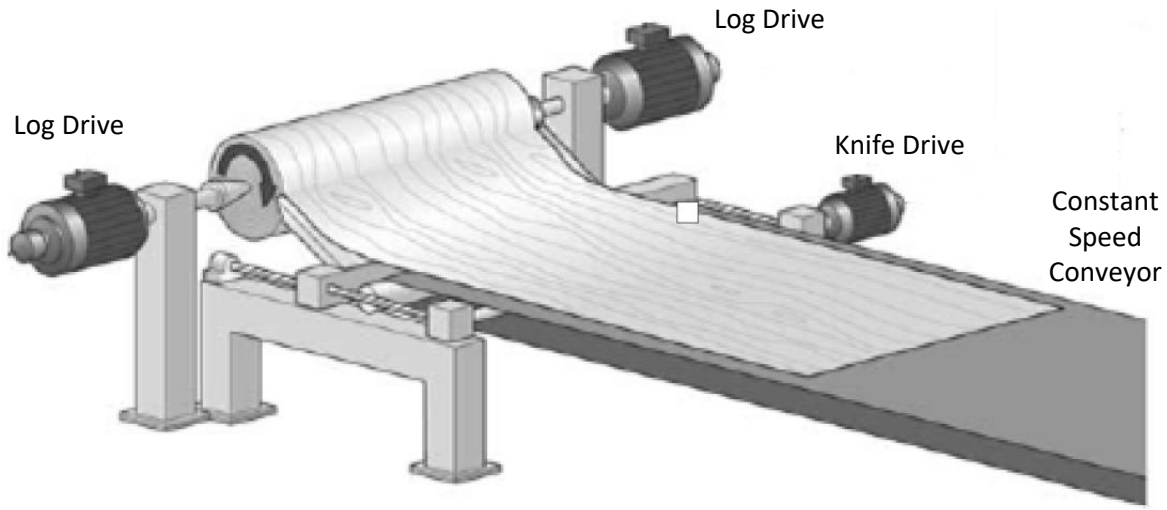
7. What is synchronization and when is it necessary? Determine which of the applications in Figs. 17-13 to 24 require synchronization.

8. A relative move and an absolute move both start from a known position (0). Each moves + 5 and then each moves -5. What is the total movement of each and where do they end relative to each other?

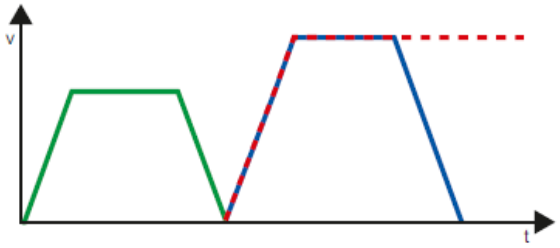
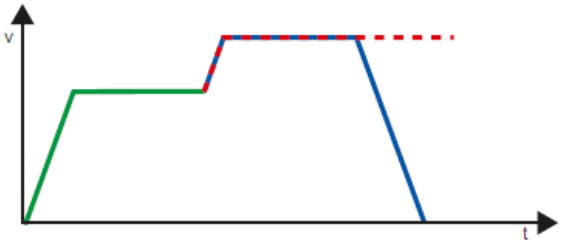
9. What are the main criteria used when determining whether to use a servo or a stepper for a motion application? Describe the advantage for using each for the various criteria.

10. In a motion command string, if the axis is not returned to zero speed before moving to the next position, what is this called?

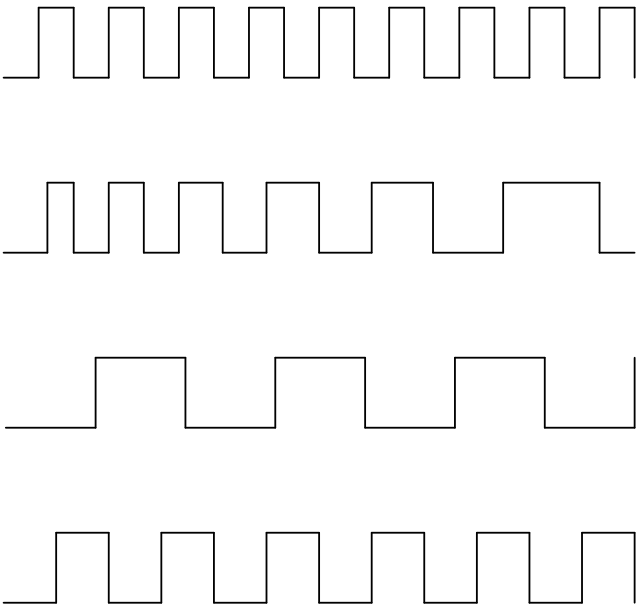
11. Use either the A-B or Siemens motion command (either relative or absolute) to control the feed action of the screw feed controlling the knife shown below. Write a program to provide this function:



12. The motion application at right is different from the one at left. What term is used to describe the profile on the right:

Transition from lower to higher velocity	Transition from lower to higher velocity
	
A job with high velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	A job with high velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The second job starts with the new velocity.

13. Identify the following speed patterns from the pulse trains for the stepper motor. Choose from (slow, fast, moderate, accelerating, decelerating)



14. Use the Siemens motion commands below to control the feed action of the two conveyors shown. Write a program to provide this function. Notice the photo-eye switches which are used to properly align the box with the hat. Assume the drive has been previously reset and is enabled.